# RERC: Accessible Pill Dispenser/Cutter

**BME 400**
**Fall 2007**
**Client: Prof. John Enderle**

**Advisor: Prof. Naomi Chesler**

Team Members:
Joe Ferris
Bryan Fondrie
Ashley Huth
Max Michalski

## Table of Contents

## Abstract

Currently, errors in medication administration and compliance are persistent problems in home medication. There are many medication organization systems on the market that aim to minimize errors; however, they often prove to be inadequate or expensive. Another concern is the rising cost of prescriptions. Pill cutting is becoming a more widely accepted as a method to decrease prescription costs; however, no medication organizer currently incorporates an automated cutting mechanism. Over the course of the year, we created an accessible pill dispenser and cutter capable of administering specified dosages of pills and half pills, which are selected using electronic buttons on the front of our device. We additionally wrote programming which can be implemented into a handheld PC, which can be programmed by a caretaker to input several regimens for different prescription schedules and create a text file, which contains all the information to be implemented into the microcontroller program. We have tested our pill cutting mechanisms and determined that on average 92% of the pill mass is retained after being cut in halves. Future work will involve creation of a program that automatically interfaces the input programming to the microcontroller program.

## I. Background Information

### *Motivation*

Numerous people have a difficult time taking their set dosage of pills on time. Whether it's forgetfulness, other priorities, a decision to omit dosage, or a physical and or emotional handicap (1), people do not regularly adhere to their daily regimens of pill consumption. Adherence is described as the consistency in consumption of physician prescribed medicine at a specified time. In the United States, reported average adherence rates for patients receiving treatment for chronic conditions were between 43%-78% (2). In addition to patients who suffer from chronic illnesses, the elderly population has a difficult time with adherence. Estimates suggest that only 25%-60% of the elderly population has close to perfect adherence. One main reason there is such a low adherence rate is due to the amount of pills that are taken. One study suggests that 25% of the elderly population takes at least three pills a day, and in hospital settings, up to eight pills a day (3). Also, doctors often have a hard time monitoring the adherence of their patients. Methods such as monitoring blood levels for medicine, recording the number of refills, and patient self-reports are often expensive, time-consuming, and unreliable. A pill dispenser that can automatically distribute a dosage of pills at a set period in time could drastically raise the adherence level of patients who have difficulty maintaining a consistent dosage regimen. Furthermore, pill dispensers can be built with the capability to record dosages that were distributed, giving doctors a better estimate of adherence level of patients.

Physical and mental handicaps often prevent people from taking their medicine on time. It is therefore in the best interest of the patient to construct a pill dispenser that is as universal as possible. The goal of the pill dispenser design is to overcome limitations in strength, coordination, sight, and hearing apparent in many users. A pill dispenser that has the capability to cut a pill in half is necessary for patients who do not have the full use of their hands.  Visible and audio alarms are necessary for people suffering from visual or audio impairment and a user-friendly interface complete with a large touch screen is imperative for personal use of all intellectual capacities.  Here is a hypothetical client base that may benefit from a pill dispenser with these capabilities.

John Smith
*Problem*: John has recently been involved in a tragic car accident that left him with an amputated left arm.  He also has very little use in his right hand due to severe trauma to his peripheral sensory neurons.  John must take a variety of pain medications to combat the phantom pain caused by neuromas that formed from injured nerve endings at the stump site which continue to fire action potentials (4).

*Solution*:  John can benefit from a pill dispenser that has the ability to dispense multiple medications for his pain.  Also, John may be instructed to start at reduced level of medication so that his body becomes use to the dosage.  If this is the case, John will not be able to cut his own pills.  A pill dispenser that can automatically cut his pills may prove to be beneficial.  Due to the emotional stress caused by the accident, John may take too many pain-killers at one time, leading to an increased risk of overdosing.  A pill dispenser can help regulate the set amount of dosage administered to John at one time.

Ann Johnson
*Problem*: Ann suffers from a genetic disorder known as Huntington's Disease. Huntington's Disease results from genetically programmed degeneration of neurons in certain areas of the brain. This degeneration causes uncontrolled movements, loss of intellectual faculties, and emotional disturbance (5). Although her condition is mild, the illness occasionally disrupts her coordination and causes tremors (6).  Ann must take medication to suppress the disease.  Side effects of the medication include fatigue and restlessness.

*Solution:* Ann could benefit from a pill dispensing device with an interface that is physically easy to use with large buttons which require little coordination to use.  Also, a simple interface that requires little input from the user may benefit Ann because of her increased restlessness and inability to concentrate for extend periods of time. Furthermore, a device with a loud alarm indicating scheduled dosage times could potentially help if Ann is feeling drowsy.  An automated pill cutter may also assist Ann with obtaining half pill dosages.

Lu Yang
*Problem*: Lu suffers from Anterograde amnesia as a result of a motorcycle accident at age 25.  Anterograde amnesia is a form of amnesia where new events are not transferred to the person's long-term memory.  People who suffer from this illness remember memories of events before the accident but cannot form any new memories.  Lu severely injured his hippocampus in the accident, the long-term memory center in the brain (7).  As a result of the accident, Lu cannot remember to take his daily dosage of medication.  Other than this condition, Lu is in fine health.

*Solution*: Lu could benefit from a pill dispenser that can be set on a monthly schedule to deliver his daily set dosage of pills. With visual and audio reminders, Lu will be informed every day that he does in fact take medication. Furthermore, a pill dispenser that has the ability to contact off-site caregivers once dosages are running out can help caregivers assist in refilling medications for Lu.

Gretchen Ramsel

*Problem*: Gretchen is a 74 year old grandmother that spends the majority of her time house-sitting for her widowed granddaughter's children. Gretchen is lively and in relatively good health. She does, however, take medication for her high blood pressure and limited arthritis in her fingers. She also takes a calcium tablet each day to strengthen her bones. Gretchen lives a hurried life-style which is centered around her grandchildren, and she often forgets to take her necessary medication. She also is farsighted which gives her trouble making out small print.

*Solution*: Gretchen could benefit from a pill dispenser with audible and visual alarms to remind her when to take her set dosage of pills each day. Moreover, Gretchen could benefit from an interface with large fonts and easy-to-press buttons so that she has no problem reading and programming the pill dispenser. With her arthritic fingers, cutting pills may prove to be a challenge. A pill dispenser capable of automatically cutting pills could save Gretchen a lot of pain, frustration, and time.

***Pill Dispensers/Organizers***

Individuals taking multiple medications require clear-cut organization of pills and strict adherence to an appropriate medication schedule. The most basic organization product on the market today is a weekly medication organizer. Weekly organizers are most commonly recognized in the form of *Figure 1* where one plastic container has seven separated compartments for pills (8). In addition to only holding a week's worth of medication, these organizers also require self-loading of pills by the patient and do not provide any alerts to remind users to take their medication. The simplicity of this product allows for a low market price ranging from \$5 - \$60 (8). Prices increase considerably for

Figure 1: Weekly pill organizer

Figure 2: Automatic pill dispenser/organizer

alert systems that monitor a patient's adherence to medication regimens. At the appropriate time, products such as those shown in *Figure 2*, will sound an alarm to alert a user that it is time to take their medication (8). Many of these products are fairly new to the market and despite being more effective than earlier organizers, they still require self-loading of medication and demand prices upwards of $1500.

***Pill Cutting***

Current pill organizers have not integrated the ability to split pills. Pill cutters make up an entirely separate market. There are multiple reasons why pill cutters are utilized. Some drugs are produced in doses that fail to meet patients' medication regimens; therefore, these drugs must be cut in half. Furthermore, increasing prices of medication prescriptions is an issue causing many patients to turn toward pill cutting. Often the same medication of two different sizes has the same co-pay. To avoid the extra cost of paying multiple co-pays for low dose prescriptions, people often opt to cut pills. Most pill cutters are handheld devices, small in size and operable by dexterous persons. *Figures 3 & 4* depict current pill cutters that share similar designs and are priced under $20 (8). Pill cutters generally consist of a holding area for one medication and an area in which the patient positions the pill under a cutting apparatus. The cutting area is usually a surface with a high coefficient of friction, such as rubber, to prevent the pill from slipping. Most cutting mechanisms use a thin razor blade and rely on the user's own strength to raise and lower the blade for cutting. A unique feature to be considered in possible designs is a tapered blade that applies varying pressure on the pill during cutting to alleviate stress concentrations and provide a cleaner cut.

Figures 3 & 4: Conventional pill cutters

Even and accurate cuts are essential requirements for pill splitting devices. Many pills are scored for cutting purposes with a depressed line across the middle. However, the material of pills does not always ensure the pill will separate into two equal halves. Some pills are not made to be cut at all, and therefore should not be subjected to a blade at any time. Another element of pill cutting to be considered is the residue left behind after a cut. If too much is lost to residue, the "halves" may not contain enough medication to be effective. Using a rotating blade, despite possibly making a cleaner cut, is not feasible. Mixing pill residues could also have harmful effects on patients, and separate blades in isolated cutting containers should be used to cut different medications. One blade could be used if a cleaning mechanism was integrated, but any such mechanism should completely remove all residues and not leave behind any trace of cleaning substances.

## II. Design Considerations

### *Problem Statement*

Medication administration regimens cause significant dispensing and adherence issues for many individuals, often compounded by the necessity of slicing pills in half. In order to improve the living conditions of those suffering from these ailments, we aim to build a pill dispenser/cutter that will administer a set dosage of pills. Specifically, it should be able to dispense the same dosage of half, one, or two pills at appropriate time intervals, as specified by the pharmacist. Additionally, the device should automatically alert the client when to take a pill, and not release any pills except during the set dosing period.

### *Essential Design Functionalities*

In order to generate a set of guidelines for the device, Product Design Specifications (PDS) were developed and can be found in the Appendix. The PDS is an outline for the functionality, operation, and design criteria for the device.

A key function of the design is the ability to dispense set doses of medication consisting of a combination of half and full pills based on the programmed schedule. In order to administer half doses, the design will be capable of cutting pills in half mechanically. Additionally, the dispenser will automatically alert users when to take pills and inform personnel offsite if doses have been missed.

Additionally, the automated pill dispensing device should be easy to use by persons with diverse capabilities and safely assist with dispensing a single dosage during the prescribed interval. It should remind users to take their medications, record what medications have already been dispensed, provide multi-modal indicators of current status, and only dispense the pills within the specified time windows each day. The device should also alert offsite medical personnel if a dose is missed.

***Accessibility***
Accessibility is a key design constraint considered to ensure our device can reach a diverse population of medical patients, particularly older patients and users with disabilities or activity limitations. Regardless of the disability it is essential that all users are able to perceive the information presented by the product, successfully operate the device, understand the device and its outputs, and be able to navigate the product (4). Moreover to achieve accessibility, the design should strictly adhere to ADA specifications and follow these recommendations (9):

*To accommodate users who may be blind or have visual impairments:*
Recommendations

a. Sans Serif font should be used
b. Text should be larger and have broad lettering
c. Both upper and lower case lettering should be used
d. Sufficient spacing between letters, words, and lines of text should be provided
e. Screens should have high contrast and resolution
f. Supplementary auditory information should be provided
g. Glare on screen should be minimized
h. Tactile indicators should further aid those with visual impairments

Users may be blind or have uncorrectable limitations including loss of visual field or temporary visual impairments such as low lighting or an obstructed line of sight (10). Thus, optimizing the visual components of the device is essential for increasing clarity and visibility for those with visual impairments. It is also recommended that the device is not overly dependent on visual information. Instead, redundancies either auditory or tactile indicators should be incorporated to provide the user the same information as displayed in text.

*To accommodate users who may be deaf or hard of hearing:*
Recommendations

a. Audio outputs should be have incremental controls
b. Auditory default volume should be reasonably loud
c. Redundant visual and tactile information for auditory content should be provided

Users may be deaf or hard of hearing. The most common approach for addressing this issue is incorporating redundant visual and tactile information for audio content. Alarms for the device could provide an adjustable audible tone, a flashing visual display, and a vibrating mechanism to insure the user is successfully alerted (10).

*To accommodate users who may have fine motor impairments, limited reach, strength, or control:*
Recommendations

a. Device should be operable with one hand
b. Device should be operable with either right or left hand
c. Force required to activate controls should be minimized
d. Controls should require general motion, rather than precise motions

e. Users should not easily be able to accidentally activate functions

Users may have motor disabilities that can affect their interactions with a device. These disabilities could include decreased fine motor controls, decreased grip strength, inability to exert force, and inability to perform two-handed tasks (9). The design of the device should minimize the motor demands of the device and allow for alternative voice activated controls.

*To accommodate users who may have cognitive or memory impairments:*
Recommendations

a. Prompts should be provided for user
b. Language should be simple
c. Redundant labels should clearly communicate function
d. User should be able to recover from errors
e. User should be able to easily navigate the produce

Users may have cognitive or memory impairments; thus, alarms and alerts should be incorporated so that minimal memory is required.

***Universal Design Principles***

Universal design concepts are broader than accessible design concepts, as they aim to improve the design for everyone, not just people with disabilities. Thus the aims of universal design concepts are slightly different. Listed below are main universal design principles that have been outlined by Connell, et Al. and are applicable to our design.

*Equitable and Flexible Use*

The design should provide both equitable and flexible use**.** The design should be able to provide the same means to users of all abilities and be able to accommodate individual's preferences and abilities. Thus the design could consist of alternative operational methods to satisfy a wide range of individuals, both disabled and not.

*Simple and Intuitive*

The design should also be simple and intuitive in multiple ways. A user of any experience, knowledge, or ability should find the design easy to understand. In our design this would consist of including effective prompting and feedback during information input and machine use.

*Perceptible Information*

The design should present information in a perceptible manner. All necessary information would be effectively communicated to the user through the design. In our design following pill dispensing alters should be redundantly presented in a clear manner. This would include both audio and visual alerts.

*Tolerance for Error*

The design should not tolerate error. The design should incorporate fail safe features, provide alerts for hazards or errors, and isolate hazardous elements. The mechanism to cut pills should not be accessible by a regular user. The design should also incorporate sensors to provide multiple checks that the device is functioning properly. In the case of an error, alerts should inform the user immediately and offer alternatives to resolve the problem.

*Low Physical Effort*
The design should be used efficiently and comfortably by the user. Since the pill dispenser is automated it will naturally address this design consideration.

*Size and Space for Approach and Use*
The design should be of an appropriate size for all users. Moreover there should be sufficient space provided for reach and manipulation regardless of the user's body size, posture, or mobility. In particular, the site from which pills will be retrieved should be adequately designed to accommodate all potential users.

By addressing many of these design criteria, a universal and accessible device can be produced for a wide user base.

## III. Alternative Designs

### ***Manual Measuring Device***

*Overview*
The manual measuring device (*Figure 5*) is set up to gauge the dimensions of each medication in order to generate a dispensing device that properly aligns pills for cutting and dispensing. Before loading the entire prescription into the pill holding funnel, the first pill is set onto the measuring platform. The platform mounting panels are then adjusted by the user to touch each edge of the pill. These mounting panels are mechanically linked to the dispensing tube which would adjust so that pills are only capable of falling in the desired vertical orientation. Additionally, the height of each pill is known from loading on the measuring platform which adjusts the height of the cutting blade to half of the total pill height.

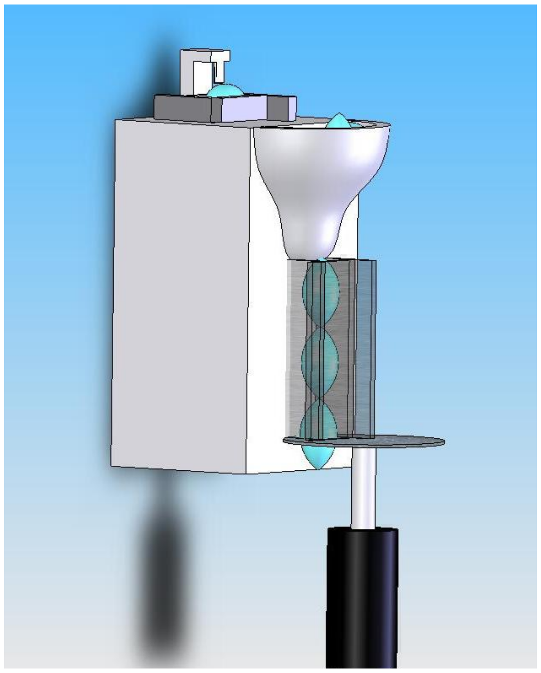


**Figure 5: Manual measurement design**

*Pros*
Manual measurement allows for precise cutting of a large variety of pills due to the initial pill height measurement. Additionally, this device would not require a large amount of counter space in home usage due to the vertical orientation of each unit.

*Cons*
Although the device can be adjusted to multiple pill shapes and sizes it is labor intensive for users and requires understanding of how the measuring platform is used which is not completely

intuitive. Furthermore, it would be difficult to prevent pills from jamming within the tight fitting dispensing tube which could prevent pills from be dispensed.

***PEZ® Pill Dispenser Device***

*Overview*

Pills of known size can be loaded and dispensed similar to the mechanism of a PEZ® dispenser (*Figure 6*). Instead of needing a variety of pill dispensing holders for the various sizes and shapes of pills, this universal holding mechanism was designed. *Figure 7* is a schematic of the pill holder looking longitudinally down the pill holding shaft. The pill **(1)** is first placed up against the back stop **(2)**. The blocking arms **(3)** extend the full length of holding shaft can and can be adjusted in the directions shown by the arrows. Once the pill is fitted, the arms then lock into place at their respective positions. Various sizes of pills can use the same holders instead of making separate holders for each different pill. Pills are packed into the holding shaft, which is then placed into one of four loading docks in the pill dispensing console. At the end of each loading dock, a small spring would be used to advance one pill at a time.

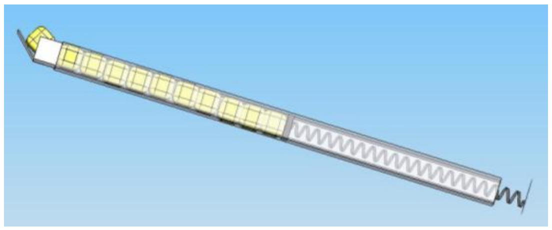


**Figure 6: PEZ dispenser design**

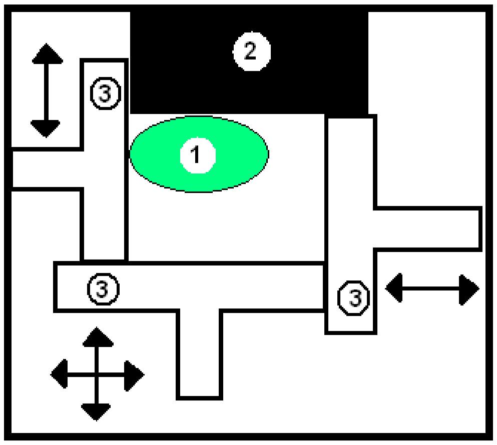


**Figure 7: Longitudinal view of PEZ dispenser design**

*Pros*

The PEZ® pill dispenser design could effectively distribute one pill at a time which is an important component of the overall design. This deign also makes it possible to add multiple PEZ® pill dispenser shafts for different kinds of pills. Each holder acts independently from the other holders, so the whole pill dispensing device can be set up for multiple PEZ® pill dispenser holding shafts. Also, with the use of arm blockers, the PEZ® pill dispenser is able to accommodate any size pill that needs to be dispensed. This is an important attribute to the overall design because it eliminates the need for custom-designed holders for each individual pill type.

*Cons*

One problem is the overall complexity of the design. The mechanical arm blockers will all need to be machined and then placed on tracks for movement. Furthermore, the pushing mechanism that advances the pill will some need to accommodate different pill sizes. A cutting mechanism is not incorporated into the PEZ® pill dispensing design at this time. Finally, this design involves extensive interaction between the care-giver and the device. Individually loading pills may take a considerable amount of time. If our device is too complicated, it may require extra training for the care-giver to use our design.

***Previous BME 400 Dispensing Design***

*Overview*

A previous BME 400 design team utilized a toothpick-type dispensing mechanism. This device uses a rotating pill drum that has been cut on one edge to the shape of the pill being dispensed. The pill drum is mounted to a stepper motor which is rotated below the outlet of a pill funneling component designed to position pills to drop into the pill drum as shown in *Figure 8*. Multiple interchangeable pill drums were created to facilitate the dispensing of a variety of pill shapes and sizes (11).

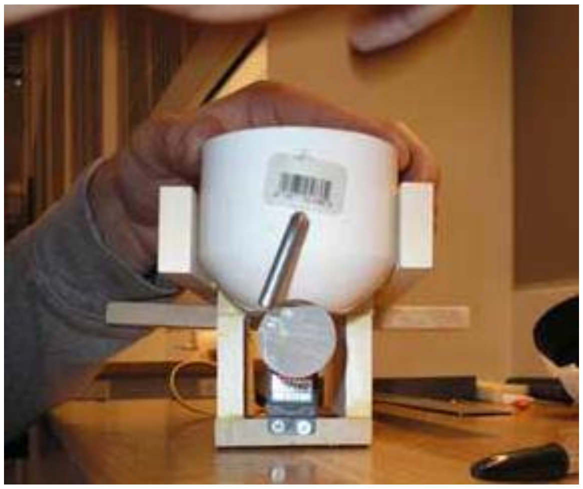


**Figure 8: Previous BME 400 design**

*Pros*

A clear advantage of a toothpick-mechanism is the testing performed by the previous group that demonstrated individual pills are regularly dispensed within three rotations of the pill drum. Additionally, the interchangeable pill drums increased the universality of the design to work with numerous types of medication. Finally, the computer code for rotating the drum can be easily written and executed using a BASIC STAMP 2 microcontroller.

*Cons*

While the previous design team's dispensing mechanism has been shown to adequately dispense pills, there is no component for pill cutting that is a major component of our design specifications. Additionally, use of interchangeable pill drums requires the user to remove and replace the drum onto the motor that may lead to problems with alignment. Since there is no consideration for pill cutting, the design lacks concern for pill residue after cutting and mechanisms for cleaning or replacing cutting blades to prevent contamination.

***Evaluation***

While each design has advantages and it is difficult to predict which would generate the best end prototype to meet our design criteria, our final design is the continuation with modification to the previous BME 400 project. Each device presented many mechanical challenges which must be overcome but the pill drum mechanism has been shown to successfully dispense individual pills (11). Interchangeable inserts for each pill size are more feasible and intuitive for inexperienced users to successfully load medication as opposed to measuring each medication and setting up each module for a different pill. With a proven mechanism to dispense individual pills, increased attention can be directed towards a mechanism for pill cutting which is an important component of the design criteria not addressed by the previous pill drum design.

## IV. Final Design: UW-Pill Cutter

While the main basis of our design still centers on a rotating drum to catch and dispense pills, with a solenoid to drive a blade to cut pills, many changes have been made. In the following sections appropriate changes will be explained in depth and justified.

### *Mixing Mechanism*

The mixing mechanism has been created to easily facilitate catching pills by the pill drum. The pill drum rotates into the internal volume of a funnel which is used for pill storage. The funnel has been designed to match the curvature of the pill drum so that it protrudes into the funnel which assists in catching pills as it rotates below them. Additionally, a continuous servo motor is used to rotate our vertical stirring mechanism within the pill funnel. The stirring mechanism consists of four bundles of soft brush bristles, which are oriented in a cross shaped pattern and spin directly above the pill drum.

### *Pill Drum*

The pill drum is composed of a Delrin rod which has been modified extensively to accommodate many facets of our design. Delrin rod was chosen due to its inherent ability to serve as dry lubricant, which prevents friction while rotating within the casing and easily dispenses pills after catching them. Each end was precisely machined to accommodate the addition of stainless steel bearings which also prevent friction during rotation. Further, the bearings support the pill drum during pill cutting which exerts a strong force on the center of the pill drum. A small slit was machined about the center line of the pill drum that prevents the cutting blade from striking the pill drum which could cause sticking and dulling of the blade. Additional modifications were made to the pill drum which allowed for sensors to be embedded, inserts to be added, as well as wires drawn through the center of the pill drum and out through the center of a bearing.

### *Inserts*

Accommodating pills of different sizes and shapes was a main design criterion. Each insert features a different pill cutout and was split down the middle so that the blade could easily pass through. The bottom of the insert was milled so that the buttons could be easily attached to the bottom of each insert. Inserts use a snap button mechanism to become securely fastened to the pill drum for a temporary period of time. A tunnel was also cut through the x axis of the inserts so that light from the IR sensors can pass through and determine if a pill has been caught.

### *Motors and Mounting*

Standard and continuous servo motors are used to control various physical mechanisms of the final prototype including pill mixing, rotation of the pill drum, and opening and closing of the half pill holder trap door. Both continuous and standard servo motors have the ability to rotate clockwise and counterclockwise. Their rotation is controlled by 1.5 micro-second current pulses that are sent to the motor’s signal line across a wide range of bandwidths. The technical and physical specifications of both motors are listed in *Figure 9*. The I/O pin on the microcontroller controls the motion of the servo motors by the

| | **Futabay Standard Servo** | **Futaby Continuous Servo** |
|---|---|---|
| **Power** | 6 VDC max | 6 VDC max |
| **Speed** | 0 to 180 degrees, 1.5 second average | 60 rpm |
| **Weight** | 45.0 grams/ 1.59 oz | 45.0 grams/ 1.59 oz |
| **Torque** | 3.40 kg-cm/ 47 oz-in | 3.40 kg-cm/ 47 oz-in |
| **Size (mm)** | 40.5x20.0x38.0 | 40.5x20.0x38.1 |
| **Size (in)** | 1.60x0.79x1.50 | 1.60x0.79x1.51 |

**Figure 9: Servo motor specifications**

PULSOUT command in PBASIC coding language. The three variables following the PULSOUT command are the microcontroller pin number, the number of pulses being sent, and the bandwidth of the signal over which each pulse is sent. Standard and continuous servo motors are directed differently by this code.

Standard servo motors receive the short pulses of current and rotate to a specific position within an 180˚ range. Futaba's standard servo motors can operate on bandwidths in the range of 250 – 1250 which correlates to positions around a half circle. Bandwidths outside the specified range can do damage to the motor and decrease its useful life. The number of pulses sent to the standard servo motor controls how long the motor will hold its position. These intrinsic properties of the standard servo motor make it perfect for controlling the precise rotation of the pill drum for pill sensing, cutting, and whole and half pill delivery, as well as, for control of the opening and closing of the half pill holder's trap door.

Continuous servo motors are capable of continuous 360˚ rotation and react differently to the previously explained PULSOUT command. The bandwidth variable no longer controls the position of the servo motor, but instead determines the speed by which it rotates. A bandwidth of 750 defines an approximate starting point that correlates to a speed of 0 rpm. Adjusting the bandwidth up from 750 increases the speed clockwise and adjusting the bandwidth down from 750 increases the speed counterclockwise. The number of pulses sent to the motor controls the amount of time over which these rotations will last. Our prototype uses a continuous servo motor to mix the pills clockwise and counterclockwise at varying speeds so that one pill will fall into the pill inserts to begin the pill delivery process.

***Sensors embedded in pill drum***
Once a pill has landed in the correct orientation of the pill inserts, it is necessary to communicate this information to the microcontroller. In order to alert the program that a pill was caught, a pair of infrared sensors was imbedded in the pill drum. The infrared sensors were bought from Radioshack, and they include a light emitting infrared diode and a phototransistor receiver. The infrared LED emits a 940 nm electromagnetic wave between 40-150 mA with a voltage between 1.3-20 VDC. The phototransistor receives this wave between 20-150 mA with a voltage between 1.3-20 VDC. We connected this sensor pair in the circuit shown in *Figure 10.*

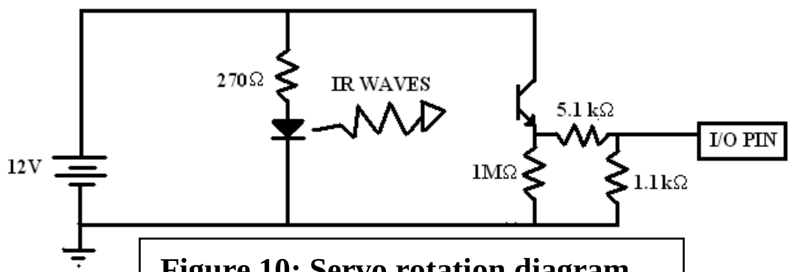


**Figure 10: Servo rotation diagram**

The microcontroller can read inputs on a simple high/low voltage scale between 0-5 V with a threshold of 1.5 V. This means that if a signal between 1.5V to 5V is being sent to the I/O pin of the microcontroller, the PBASIC language will interpret this as a 1. In the same way, if the output voltage is between 0-1.5V, the microcontroller interprets this as a 0. When there is no pill in the inserts, the signal sent to the I/O pin of our microcontroller reads the number 1, or high. When a pill has fallen in to the pill inserts, the pathway between the emitter and the detector is broken and the microcontroller reads this broken signal as 0, or low. Having these sensors allow us to place checks in our program that allow us to retry catching a pill or signal error messages when a pill has not been caught by the pill inserts.

***Solenoid***
A non-rotating razor blade should be used for cutting pills in half. Lining up the blade with the center of pills is easiest when the motion of the blade stays in a constant linear plane. A linear push solenoid moves straight in one direction when excited by an electrical current. The solenoid is a "modified electromagnet" that consists of two major parts, a central core or armature and a coil of wire that surrounds the armature (11). Sending current through the wire coil creates a magnetic field with strength directly related to the number of wire coils. More wire coils create a stronger magnetic field. The magnetic field generates force that pushes the central core up through the solenoid. The distance the armature travels is referred to as the solenoid's stroke and generally ranges from 0.5" to 1.5". Once the wire coil is excited by the current and the armature pushes forward in a linear motion, it will remain at its excited position until the current source is turned off. Switching off the current allows the solenoid's armature to fall back down to its original position which is normally regulated by a spring mechanism.

Sealed solenoids are enclosed in durable casing to protect the wire coils from the external environment which extends the life of the solenoid. With a maximum stroke of 0.8" the solenoid is able to generate varying amounts of force depending on the duty cycle and stroke length. The range of these forces are listed below. The solenoid requires a 12 volt DC source that can supply over 7 amperes of current to the wire coil and create a strong enough magnetic field to drive the armature. Exciting the solenoid using the Parallax Basic Stamp 2 microcontroller requires the use of an industrial relay. Relays are switches that can handle large amounts of current running through an internal coil. The relay switch controls whether a circuit is complete or not and is activated or switched on by very low voltages and currents. Since the microcontroller is only a 5 volt source, a 5 volt relay must be used. The current needed to excite the relay's internal coil is 40 milliamps. One I/O pin on the microcontroller can source a maximum of 50 milliamps, so only one pin must be used to provide sufficient current to the relay. Running the solenoid from the microcontroller allows for simple on/off control. Exciting the appropriate pin excites the relay creating a closed circuit and activating the solenoid's linear stroke. Conversely, turning the same pin off switches the relay again, making the solenoid circuit incomplete and allowing the armature to return to its resting position.

There are two limiting factors when considering the power of this low profile solenoid. The first of these factors is the aforementioned stroke of the solenoid. As the stroke of the solenoid increases, the maximum power of the solenoid decreases in linear proportion. With this in mind, the stroke length used in the prototype is 0.25 in. The cycle by which the solenoid is energized and de-energized is referred to as the duty cycle and is the second limiting factor of the solenoid's strength. The list below shows the possible duty cycles of the solenoid and the correlated maximum power that can be generated at various stroke lengths. Four duty cycles exist for the solenoid distinguished by the approximate amount of time that the solenoid will remain on during one excitation. The percentage which is associated with each duty cycle is calculated by the following equation:

Duty Cycle = [*On time* / (*On time* + *Off Time*)] * 100%

*On Time* is the amount of time the solenoid stays energized during one excitation and *Off Time* is the amount of time the solenoid stays de-energized between excitations. In order to make an accurate and complete cut our solenoid will have an *On Time* of approximately three seconds and depending upon the needs of individual users varying values for the *Off Time*. However, it is reasonable to assume users will require only one cut of a pill at any specific dosing period and so the *Off Time* will be considerably longer than the *On Time*. If multiple pills need to be cut at one dosing period the amount of time between cuts should be over 27 seconds to maximize the power output of the solenoid. This *Off Time* value combined with our *On Time* value of 3 seconds, yields a Duty Cycle of 10% or as described below a Pulse Duty Cycle.

**C = Continuous (100%) Duty Cycle, Maximum On-Time = Infinite**
**Approximate Input Power = 20 Watts**
Force at 0.05" (1.3 mm) stroke: 720 Oz (20412 gr)
Force at 0.25" (6.4 mm) stroke: 144 Oz (4082 gr)
Force at 0.50" (12.7 mm) stroke: 48 Oz (1361 gr)

**I = Intermittent (50%) Duty Cycle, Maximum On-Time = 230 Seconds**
**Approximate Input Power = 40 Watts**
Force at 0.05" (1.3 mm) stroke: 960 Oz (27216 gr)
Force at 0.25" (6.4 mm) stroke: 288 Oz (8165 gr)
Force at 0.50" (12.7 mm) stroke: 80 Oz (2268 gr)
Force at 0.80" (20.3 mm) stroke: 11 Oz (312 gr)

**L = Long Pulse (25%) Duty Cycle, Maximum On-Time = 90 Seconds**
**Approximate Input Power = 80 Watts**
Force at 0.05" (1.3 mm) stroke: 1200 Oz (34020 gr)
Force at 0.25" (6.4 mm) stroke: 530 Oz (15026 gr)
Force at 0.50" (12.7 mm) stroke: 170 Oz (4820 gr)
Force at 0.80" (20.3 mm) stroke: 30 Oz (851 gr)

**P = Pulse (10%) Duty Cycle, Maximum On-Time = 30 Seconds**
**Approximate Input Power = 200 Watts**
Force at 0.05" (1.3 mm) stroke: 1730 Oz (49046 gr)
Force at 0.25" (6.4 mm) stroke: 1040 Oz (29484 gr)
Force at 0.50" (12.7 mm) stroke: 416 Oz (11794 gr)
Force at 0.80" (20.3 mm) stroke: 96 Oz (2722 gr)

***Blade holder system***
The blade holder system was created to allow precise linear motion of the blade without physical attachment to the solenoid. This would allow a single solenoid coupled with a rotational motor to be used by multiple pill drum modules. The solenoid must be separate from the blade in order to prevent cross contamination of several pill types. The blade holder is a modified #3 X-ACTO® blade handle and #11 knife. The X-ACTO® blade handle allows easy changing of blades after one has become dull. The plastic grip of the blade handle was removed and the metal blade handle was mounted into a ¾” x ¾” x 2” plastic guide. The guide is contained within a plastic housing that only allows linear motion as well as safely prevents any access to the blade without removing two set screws for interchanging blades. Also contained within the plastic housing is a set of two springs which retract the blade system from the pill drum after the cut has been made and the solenoid is relaxed. Two springs were used so that initially the resistance to motion is a long spring with a small stiffness coefficient followed by a short stiff spring which provides enough force to remove the blade from the pill drum and prevent sticking.

***Pill Drum Casing***
The pill drum sits in a PVC casing shown in *Figure 11*. The pill drum casing consists of a half-cube with a cylindrical area bored out of it in which the pill drum rests. The cylindrical area where the pill drum rests is a smooth surface to reduce friction. An area of the cube has been cut out for the insertion of the cutting blade. There is also a hole cut in the side of the casing to allow for wires from the sensors in the pill drum to be drawn out. Finally, two holes are notched out of the inside of the cylindrical bore, one at 75 degrees from the top (half pill notch) and the other at 180 degrees from the top (half/full pill notch). As the pill inserts in the pill drum spin past the notches, the pills are able to

fall into either the half pill holding container or the half/full pill collecting dish. The other half of the cube is integrated with the pill funnel. The top half rests on two pins that protrude from the bottom half of the cube.

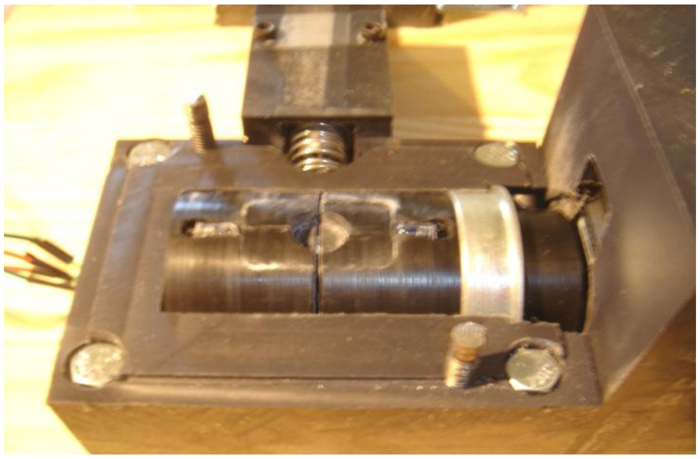


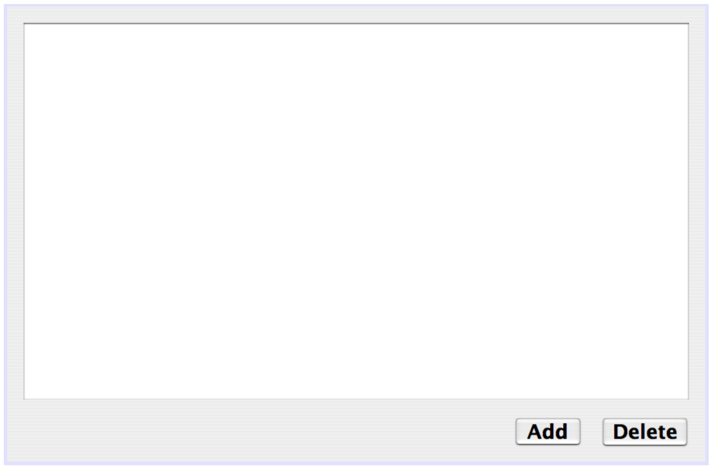
**Figure 11: Pill drum and casing**

***Half pill holding container***

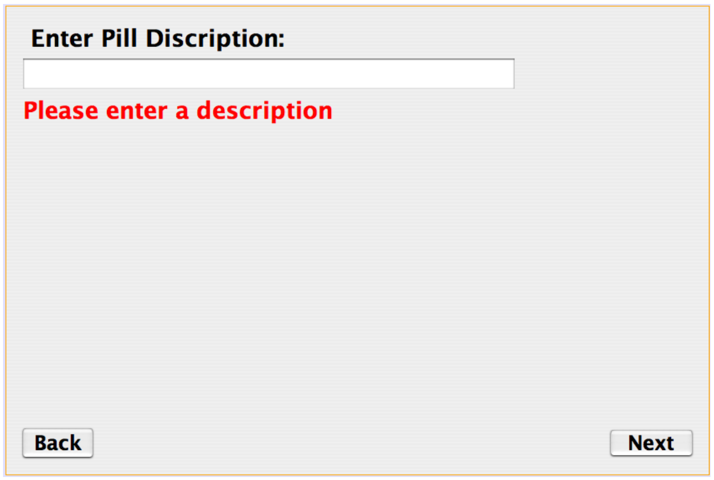
After a pill is cut in half, one of the halves needs to be separated and stored. When the pill drum spins past the half pill notch in the pill drum casing, a half pill falls into a pill holding container. This container consists of a box with a downward slanting bottom. When the pill is dropped into the container, the slanted bottom of the box directs the pill to a trough where the bottom meets a side trap door. Across the trough is a set of infrared sensors (the same kind used in the pill drum). When the pill comes to rest between the trap door and the bottom, the IR pathway is broken and the sensor reads low. The next time a half a pill needs to be delivered for a dosage, the program checks the half pill sensor. If the sensor reads low, the program knows that there is a half pill in the container that can be delivered. The trapdoor works on a hinge with a standard servo motor (the same kind used to rotate the pill drum). Once the program reads low from the IR sensor, it sets in motion a servo motor that cranks a wire connected to the trap door, raising it, and dispensing the half pill into the final collection dish.

***Interface***

Our user interface program was created to be simple for care providers to operate. This program requires the user to go through a series of questions to insert pertinent information about the patient's pill regimen. All information input is stored to be later called by the program which operates the mechanics of the device. Information included is times per day to dispense the pill, each time of day to dispense the pills, and number of pills that need to be dispensed at each time. Moreover, information is added about the number pills that were inserted into the device by the care provider. As pills are dispensed the number of pills in the device can be determined using the input values and subsequently a care provider can be alarmed that more pills need to be inserted when they are running low.

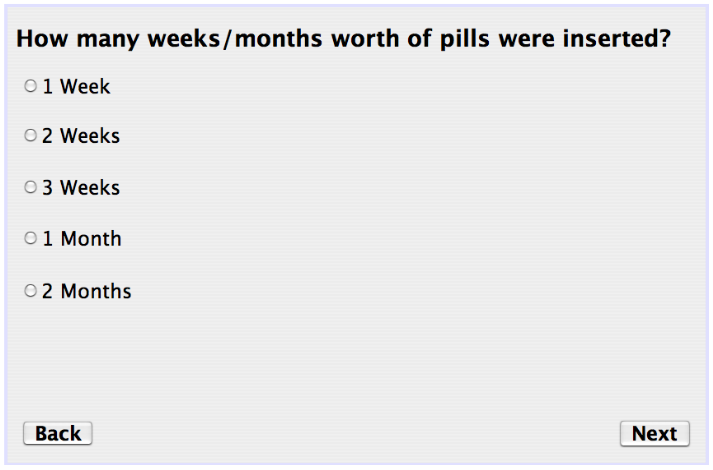


Depicted above is the main programming screen that will contain a list of pills and their respective input parameters after data has been entered. Also from this page you can choose to add a new pill or delete an existing entry.


Enter Pill Discription:

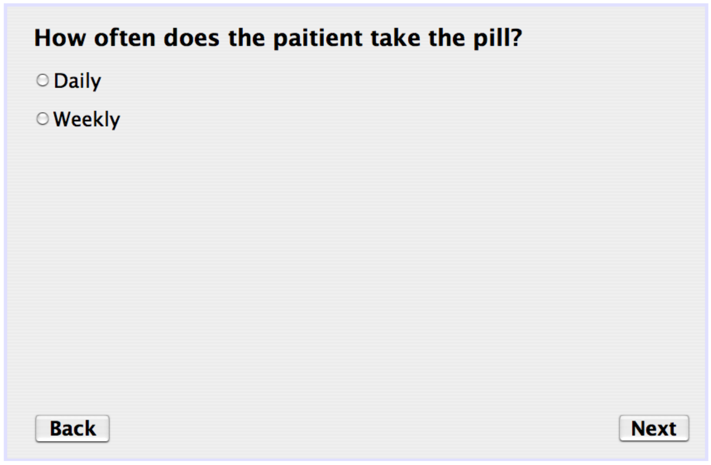
Please enter a description

If add is chosen the user will then be asked to enter a simple description of the type of pill they are adding. Input suggestions include type and size of pill (ex. 500mg Vitamin C).


How many weeks/months worth of pills were inserted?

- ○ 1 Week
- ○ 2 Weeks
- ○ 3 Weeks
- ○ 1 Month
- ○ 2 Months

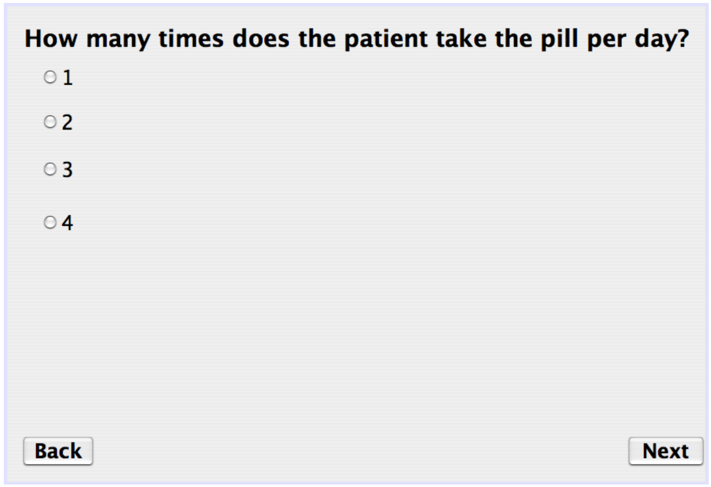
Following the user will identify the quantity of pills that they have inserted into the pill funnel. This entry will later be used to calculate the number of pills that are left in the funnel after each dosage has been dispensed.


How often does the paitient take the pill?

○ Daily

○ Weekly

Since some pills are only intended to be taken once a week, our device provides the user with the options to dispense a pill daily or just once a week. Depending on their selection the program will provide different question prompts for *Track 1* (daily) and *Track 2* (weekly).


How many times does the patient take the pill per day?

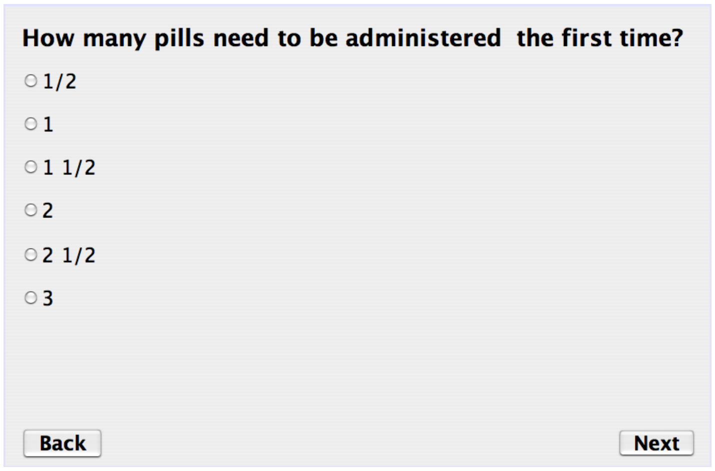
○ 1

○ 2

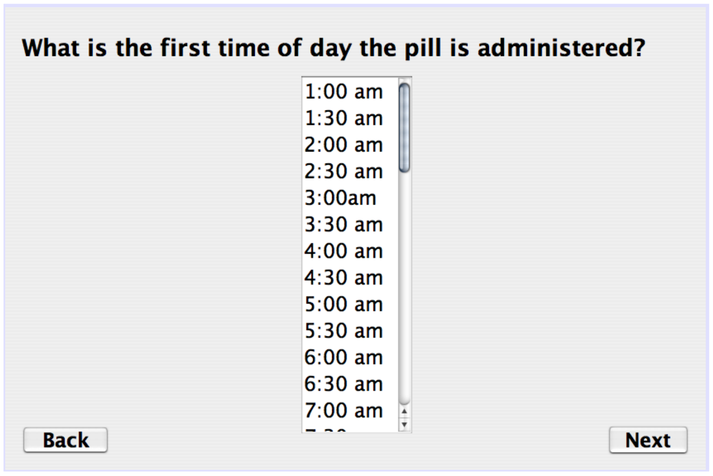
○ 3

○ 4

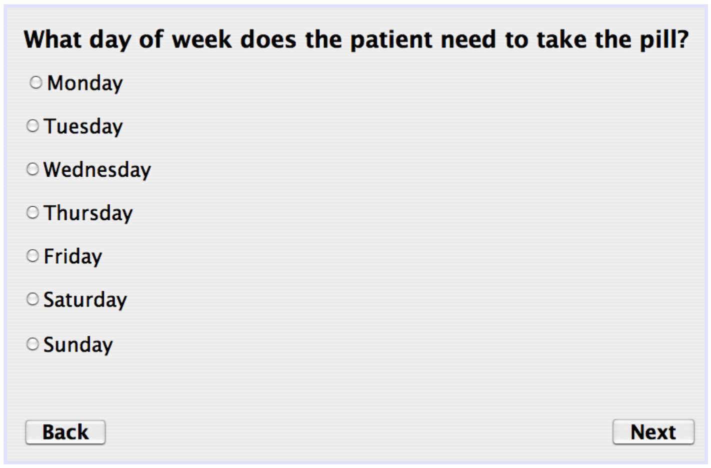
Pills are frequently taken more than once per day. Thus, our program allows the user to select how many times a particular pill should be administered daily. Following this selection the following two screens will appear to allow the user to indicate when pills should be dispensed during the day and how many pills should be dispensed at each time.


How many pills need to be administered  the first time?

- ○ 1/2
- ○ 1
- ○ 1 1/2
- ○ 2
- ○ 2 1/2
- ○ 3


What is the first time of day the pill is administered?

1:00 am
1:30 am
2:00 am
2:30 am
3:00am
3:30 am
4:00 am
4:30 am
5:00 am
5:30 am
6:00 am
6:30 am
7:00 am


What day of week does the patient need to take the pill?

- ○ Monday
- ○ Tuesday
- ○ Wednesday
- ○ Thursday
- ○ Friday
- ○ Saturday
- ○ Sunday

If *Track 2* is selected the first prompt will determine the day of the week that the pill needs to be taken, followed by the time of day and number of pills screens as seen in *Track 1*.

The final screen that will be presented to the user will be a confirmation screen that will display the user's selections. At this point if selections are incorrect they will be able to select the back button and modify their selections.

***Basic Stamp 2 Microcontroller***
The physical components of our prototype are all directed by the Parallax Basic Stamp 2 Microcontroller (BS2). The BS2 is the intelligence of the prototype design and is programmable using its unique version of BASIC code: PBASIC. Code written on a PC in the Basic Stamp Editor program can be transferred to the BS2 by means of a USB serial port that sits on the BS2's project board. Once transferred, a single program can be stored in the BS2's Electrically Erasable Programmable Read Only Memory (EEPROM). This feature allows users to turn the BS2 on and off and be able to run the saved program without having to reconnect the USB and reload the data. The BS2 features 16 Input/Output (I/O) pins capable of controlling signals to peripheral physical elements and exchanging or inputting data from those elements back to the microcontroller. Each I/O pin on the BS2 controls a single physical element of our prototype: sensors, motors, speakers, and buttons. The project board of the BS2 contains a 5 VDC regulator that allows connection of a 12 VDC power source to the BS2, of which only 5VDC is actually supplied. A voltage of this magnitude only allows minimal amounts of current to be sourced and sunk by each BS2 I/O pin. The maximum current that one pin can source to or sink from a peripheral element is in the range of 25 – 50 mA. The voltage required for activating just one pin (threshold voltage) is 1.5 V. Code of the final PBASIC program stored on our BS2 is provided with annotations is located in the appendix.

***Prototype Production Costs***
*Listed below are the costs of components used to produce our working prototype.*

| **Item** | **Company** | **Quantity** | **Unit Price** | **Cost** |
|---|---|---|---|---|
| **IR Sensors and Detectors** | **RadioShack** | **2** | **$3.49** | **$6.98** |
| **Piezo Buzzer** | **RadioShack** | **1** | **$11.49** | **$11.49** |
| **110 Watt AC-DC Power Adapter** | **Ituner Network Corp** | **1** | **$35.95** | **$35.95** |
| **5 Volt REED Relay** | **RadioShack** | **5** | **$2.99** | **$14.95** |
| **Wood** | **Home Depot** | **2** | **$10.49** | **$20.98** |
| **Microcontroller Startup Kit** | **Parallax** | **1** | **$99.95** | **$99.95** |
| **Razor Blade Set** | **Ace Hardware** | **1** | **$20.49** | **$20.49** |
| **Standard Servo Motor** | **Parallax** | **2** | **$12.95** | **$25.90** |
| **Plastic Scraps** | **Laird Plastics** | **1** | **$50.00** | **$50.00** |
| **Solenoid** | **Electromechanic** | **1** | **$105.49** | **$105.49** |
| **Misc Electrical Materials** | **RadioShack** | **1** | **$30.00** | **$30.00** |
| **Misc Hardware** | **Ace Hardware** | **1** | **$20.00** | **$20.00** |
| | | | **TOTAL** | **$442.18** |

## *Testing and Validation*

In order to measure the success of our pill cutting technique we focused on two different measurements. The first measurement we took was the average mass of half pills. This measurement was taken to determine the amount of variability inherent to the cut. As seen in *Figure 12* half pills varied about 5 mg either way, demonstrating that pills were not cut exactly in half. The second measurement that was taken was the total mass of the pill after the cut. This measurement was compared to the mass of the pill prior to cutting to demonstrate the mass lost following cutting. As seen in *Figure 13*, there was some loss in mass, however this is typical for hand held pill cutters as well. It should also be noted that this numbers is specific to the 500 mg Vitamin C pills that were cut, a pill with a different composition or thickness may respond differently when cut.

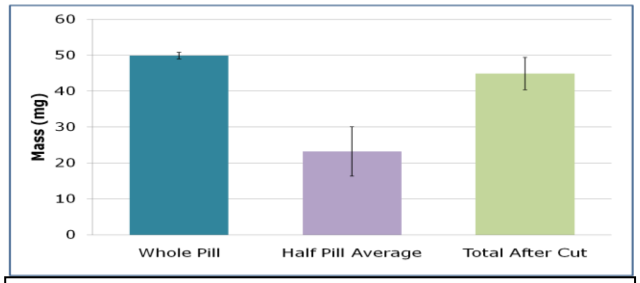


**Figure 12: Half pill mass testing**

Another experiment that was performed involved the accuracy of dispensing a set amount of pills in a row. A total of twenty different trials were performed, ten for the full pill and ten for the half pill. At the start of each trial, six full pills were loaded into the pill funnel. The program was run and observations were made as to whether a pill has been dispensed. The experimental results can be seen in *Figure 13*. In theory, one would expect a total of six full pills delivered per trial for the full pill, and a total of twelve half pills delivered per trial for the half pill. In actuality, this was not the case. There was some error involved with the experiment, which we deemed Type I and Type II errors. Type I error involved the failure of a full pill to be caught in the pill inserts after three consecutive tries. Type II error refers to the failure of our device to dispense a pill/half pill after three consecutive tries. From the experimental data, it can readily be seen that full pill dispensing accuracy was perfect for 70% of its trials and 60% perfect for the half pills. Type I errors were due to either the pill being awkwardly positioned in the pill inserts, or the mixing mechanism unable to guide a pill into the inserts. Whereas, Type II errors were due to frictional complications after the pill had been cut. In our program, however, we have anticipated the likelihood of such errors and placed alarms to indicate that one of these types of errors had occurred.

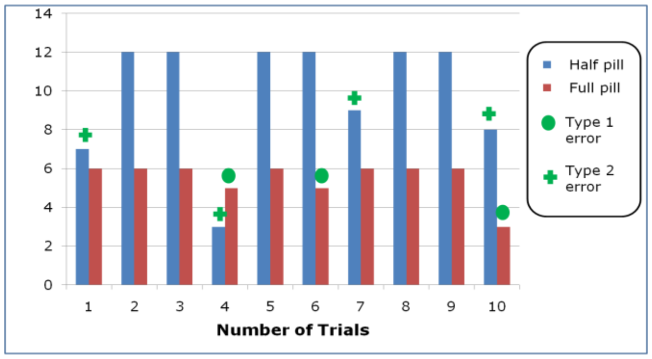


**Figure 13: Validation Testing**

## V. Future Work

Much needs to be done to successfully address all of the design constraints. We need to create a more compact set up which contains at least four different pill dispenser modules. Once the four different modules are put together, we will need to interface the input programming with the microcontroller program. Following construction of our final prototype consisting of four modules tests that are focused on the user should be conducted to ensure that our device is accessible to a variety of individuals. To perform this type of assessment, an IRB would be required and enlisting a sample population that is representative of our projected user base would be necessary. To obtain an IRB we would need to create a protocol that informs our participants and maximizes their safety. Both quantitative data on the dispenser's performance and qualitative data focused on overall accessibility of the dispenser could provide insight into both the benefits and shortcomings of our design.

## VI. References

(1) Osterberg, Lars, and Terrence Blaschke. "Adherence to Medication" *Drug Therapy* 353: 487-497.

(2) Smith, D., Compliance Packaging: A Patient Education Tool, American Pharmacy, Vol. NS29, No 2 February 1989.

(3) Salzman, C. "Medication Compliance in the Elderly." *J Clin Psychiatry* 56 (1995): 18-22.

(4) 4 Ramachandran, V. S. & Hirstein, William (2008), "The Perception of Phantom Limbs: The D. O. Hebb Lecture", *Brain* **121** (1): 1603-1630.

(5) http://www.ninds.nih.gov/disorders/huntington/huntington.html.

(6) http://www.mayoclinic.com/health/huntingtons-disease.

(7) http://neurology.health-cares.net/anterograde-amnesia.php.

(8) "E-pill Medication Reminders". <http://www.epill.com>. 2004.

(9) "Americans with Disabilities Act Homepage". <http://www.ada.gov/>. Oct 2, 2007.

(10) "Accessibility". <http://www.ahrq.gov/accessibility.htm>. Oct 4, 2007.

(11)www.theamdd.com.

(12) Personal Interview with Ken Oneill (UW-Pharmacy) October 25, 2007.

## VII. APPENDIX

***PDS***

May 11, 2008

## Product Design Specifications

**Title:** Accessible Pill Dispensing/Cutting Device

**Team:**
Max Michalski- Team Leader
Ashley Huth- Communicator
Joseph Ferris- BWIG
Bryan Fondrie- BSAC

**Function:** Dispensing set doses of medication from half to double doses of pills based on the programmed schedule. In order to administer half doses, the design will be capable of cutting pills in half mechanically. Additionally, the dispenser will automatically alert patients when to take pills and inform personnel offsite if doses have been missed.

**Client requirements:** The automated device should be easy to use by clients with diverse capabilities and safely assist with dispensing a single dosage during the prescribed interval. The prototype should be able to dispense any of 1/2, 1 or 2 pills at a time and be able to cut pills in half if required for 1/2 pill dosage. It should remind users to take their medications, record what medications have already been dispensed, provide multi-modal indicators of current status, and only dispense the pills within the specified time windows each day. The device should alert someone offsite if a dose is missed. The prototype can be larger than a final product for demonstration purposes.

**Design requirements:**
**1. Physical and Operational Characteristics**

*a. Performance requirements*
Device must be capable of accurately dispensing set doses at designated time. The device will only dispense at given time intervals and inhibit patient access to medication at non-designated times. The design must incorporate a cutting device which halves a variety of pill shapes and sizes. The device must also promptly inform medical personnel when doses have been missed by the patient.

*b. Safety*
The mechanical pill cutter must accurately cut pills in half so as to administer correct doses of medication. Additionally, the cutting device must be contained within the support casing.

c. *Accuracy and Reliability*
The dispenser must administer the appropriate dosage of medication at the programmed time interval.

d. *Life in Service*
Multiple years and the dispenser can be reprogrammable for a different medication regimen.

e. *Shelf Life*
The mechanical pill cutter must safely contain pills for the duration of the prescription.

f. *Operating Environment*
This device could be used in a variety of settings including, but not limited to, homes, hospitals, and nursing homes.

g. *Size*
The device should be of minimal size; however the final prototype may be larger for demonstration purposes. Moreover, the device may be scaleable to handle both large and small medication regimens and pill bottle sizes.

h. *Material*
FDA approved plastic materials, which are easily sterilized to allow for repeated usage.

i. *Aesthetics, Appearance, and Finish*
The device should be aesthetically pleasing.

**2. Production Characteristics**

a. *Quantity*
One large-scale working prototype.

b. *Target Product Cost*
The total cost of the project may be no more than $2000 but minimal cost is desired to allow access for patients of all economic classes.

**3. Miscellaneous**

a. *Standards and Specification*
Must be FDA approved in order to put into service.

b. *Customer*
Individuals that have numerous medications, or individuals who have trouble complying with their recommended medication regimen.

c. *Patient-related concerns*
Dispensing the appropriate dosage of medication at the scheduled time and not allowing patient to access medication at non-scheduled times.

d. *Competition*
Other devices are on the market that addresses medication regimen compliance by reminding the individual to take their pills.

***Interview with pharmacist***

Ken O'neill of the UW-Pharmacy department, several questions were asked about the relative size and frequency of pills that are commonly cut (12). The answers to the questions are provided below.

*What is the most common smallest pill to cut?*

The smallest pill to effectively cut would be either Atenolol 25mg or 25 mg hydrothroclorizide. The size of these pills is similar to over-the-counter Claritin

*What is the largest pill that would typically be dispensed?*

Potassium Chloride pill. Similar in size to Centrum.

*What are your thoughts on the pill cutters that are currently used?*

Currently, pill cutters often crush pills instead of splitting which is a limitation. However, using a guillotine approach, less product is lost because the blade does not saw through but merely splits pills into two halves.

*How common is it for individuals to be prescribed a larger pill to cut instead of single smaller pills?*

This can depend on the marketing of the pill, if an effective higher dose has been approved and the patient has run out, it may not be cost effective to produce a smaller pill. However a physician may wish to prescribe a lower dosage and thus the individual has no choice but to cut the pill. People who are more cost conscious also often split pills.

From the interview, a range of sizes of pills can be roughly estimated. Using these values loosely, we can design our pill cutter to effectively position and cut the necessary sizes of pills. It is still necessary, however, to consider all sizes of pills that customers may take in order to create a pill dispenser capable of selecting any size pill for delivery.

## ***Final BASIC Stamp program***

```
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ Variables ]----------------------------------------------------
idx             VAR     Byte                    ' loop counter
PSENSE          VAR     Byte
PSENSE2         VAR     Byte
MIXER           VAR     Byte
sensor          VAR     Byte
PILLS           VAR     Byte
ALARM           VAR     Byte
ALARM2          VAR     Byte
BUTN            VAR     Byte

' -----[ Program Code ]-------------------------------------------------

Main:



PILLS = 20
DO WHILE PILLS > 0

BUTN = 0
```

```
DO WHILE BUTN = 0          '---------Start button loop until 1 button is
pushed-----

IF (IN4 = 1) THEN
  BUTN = 4
  HIGH 8

  GOSUB Half_Pill_Program                   'half pill program runs once


ELSEIF (IN5 = 1) THEN

  BUTN = 5
  HIGH 9

  GOSUB Whole_Pill_Program                  'whole pill program runs once


ELSEIF (IN6 = 1) THEN

  BUTN = 6

  HIGH 10
  GOSUB Whole_Pill_Program
  HIGH 10
  GOSUB Half_Pill_Program                   'whole pill program runs once,
followed by half pill program


ELSEIF (IN7 = 1) THEN

  BUTN = 7
  HIGH 11

  GOSUB Whole_Pill_Program
  HIGH 11
  GOSUB Whole_Pill_Program


ELSE

ENDIF

LOOP

LOOP



END

'--------Whole Pill Program Sub-routine

Whole_Pill_Program:

FOR idx = 1 TO 100
  PULSOUT 13, 250
  PAUSE 20
NEXT

PSENSE = 1
ALARM = 0
DO WHILE PSENSE = 1
```

```
FOR MIXER = 1 TO 150
  PULSOUT 12, 1000
  PAUSE 20
NEXT

FOR MIXER = 1 TO 150
  PULSOUT 12, 500
  PAUSE 20
NEXT

PAUSE 2000

FOR idx = 1 TO 50
  PULSOUT 13, 500
  PAUSE 20
NEXT
PAUSE 1000
  IF (IN3 = 1) THEN
  ALARM = ALARM + 1
    IF (ALARM = 4) THEN
        PAUSE 500
        LOW 9
        PAUSE 500
        HIGH 9
        PAUSE 500
        LOW 9
        PAUSE 500
        HIGH 9
        FREQOUT 0, 2000, 4000
        LOW 9
        PAUSE 500
        HIGH 9
        PAUSE 500
      LOW 9
      LOW 11
      PAUSE 2000
      FOR idx = 1 TO 50
        PULSOUT 13, 250                  'Back to start, Type 1 Error
        PAUSE 20
      NEXT
      DEBUG  "NO PILL WAS CAUGHT", CR
END                                       'Program Ends
    ELSE
      PAUSE 1000
      FOR idx = 1 TO 50
        PULSOUT 13, 250                   'Back to top for more mixing (Less
than three attempts)
        PAUSE 20
      NEXT
    ENDIF
  ELSE
    PSENSE = PSENSE - 1
  ENDIF
LOOP

PAUSE 1000


FOR idx = 1 TO 100
  PULSOUT 13, 1250
  PAUSE 20                        'Pill drum rotates down to 180 degrees to
dispense whole pill
```

```
NEXT

PSENSE2 = 1
ALARM2 = 0
DO WHILE PSENSE2 = 1
IF (IN3 = 0) THEN
  PAUSE 1000
  ALARM2 = ALARM2 + 1
  IF (ALARM2 = 3) THEN                         'Type 2 Error:  Pill is Stuck
      FOR idx = 1 TO 2
        PAUSE 500
        LOW 9
        PAUSE 500
        HIGH 9
        PAUSE 500
        LOW 9
        PAUSE 500
        HIGH 9
        FREQOUT 0, 2000, 4000
        LOW 9
        PAUSE 500
        HIGH 9
        PAUSE 500
      NEXT
      LOW 9
      LOW 11

    PAUSE 2000
    FOR idx = 1 TO 50
      PULSOUT 13, 250
      PAUSE 20
    NEXT
    DEBUG "CHECK PILL DRUM, YOUR PILL MUST BE STUCK.", CR
END                                                    'Program Ends
  ELSE
    PAUSE 2000
    FOR idx = 1 TO 30
      PULSOUT 13, 1200                             'Rotate back UP to try again
      PAUSE 20
    NEXT
    PAUSE 1000
    FOR idx = 1 TO 30
      PULSOUT 13, 1250                             'Rotate down to try and
dispense
      PAUSE 20
    NEXT
  ENDIF
ELSE
    PSENSE2 = PSENSE2 - 1
    PAUSE 500
ENDIF
LOOP

DEBUG "Pill has been dispensed", CR

PAUSE 1000

FOR idx = 1 TO 150
  PULSOUT 13, 250
  PAUSE 20
NEXT

PAUSE 2000
```

```
LOW 9
LOW 11
PAUSE 1000
HIGH 9
HIGH 11
PAUSE 1000
LOW 9
LOW 11
PAUSE 1000
HIGH 9
HIGH 11
PAUSE 1000
FREQOUT 0, 2000, 4000                      'ENDING TONE ROTATE BACK TO TOP
PAUSE 1000
LOW 9
LOW 11

PILLS = PILLS - 1

RETURN


'''''''''''''''''''STARTING HALF PILL
PROCEDURE'''''''''''''''''''''''''''''''''''''''''''



Half_Pill_Program:

FOR idx = 1 TO 100
  PULSOUT 13, 250
  PAUSE 20
NEXT

PAUSE 1000

IF (IN1 = 1)THEN            '--------Checks sensor in Half pill holder---------
-------------
PSENSE = 1
ALARM = 0
DO WHILE PSENSE = 1

FOR MIXER = 1 TO 150
  PULSOUT 12, 1000
  PAUSE 20
NEXT

FOR MIXER = 1 TO 150
  PULSOUT 12, 500
  PAUSE 20
NEXT


PAUSE 2000

FOR idx = 1 TO 50
  PULSOUT 13, 500
  PAUSE 20
NEXT
PAUSE 1000
  IF (IN3 = 1) THEN
  ALARM = ALARM + 1
```

```
    IF (ALARM = 4) THEN
      FOR idx = 1 TO 2
        PAUSE 1000
        LOW 8
        PAUSE 500
        HIGH 8
        PAUSE 500
        LOW 8
        PAUSE 500
        HIGH 8
        FREQOUT 0, 2000, 4000
        LOW 8
        PAUSE 500
        HIGH 8
        PAUSE 500
        LOW 8
        PAUSE 500
        HIGH 8
        PAUSE 500
      NEXT
      LOW 8
      LOW 10
      PAUSE 2000
      FOR idx = 1 TO 50
        PULSOUT 13, 250                  'Back to start, Type 1 Error
        PAUSE 20
      NEXT
      DEBUG  "NO PILL WAS CAUGHT", CR
END                                       'Program Ends
    ELSE
      PAUSE 1000
      FOR idx = 1 TO 50
        PULSOUT 13, 250                   'Back to top for more mixing (Less
than three attempts)
        PAUSE 20
      NEXT
    ENDIF
  ELSE
    PSENSE = PSENSE - 1
ENDIF
LOOP

PAUSE 1000

FOR idx = 1 TO 50
  PULSOUT 13, 650                      'Rotating down to 90 degrees for cutting
  PAUSE 20
NEXT

PAUSE 3000

HIGH 2                                 'Activate Solenoid for cutting
PAUSE 2500
LOW 2                                  'Deactivate solenoid for cutting
PAUSE 2000


FOR idx = 1 TO 30
  PULSOUT 13, 900                     'Rotating down to deliver first half pill
  PAUSE 20
NEXT

PAUSE 2000
```

```
PSENSE2 = 1
ALARM2 = 0
DO WHILE PSENSE2 = 1
PAUSE 1000
IF (IN1 = 1) THEN
  PAUSE 1000
  ALARM2 = ALARM2 + 1
  IF (ALARM2 = 3) THEN                         'Type 2 Error:  Pill is Stuck
    FOR idx = 1 TO 2
        PAUSE 1000
        LOW 8
        PAUSE 1000
        HIGH 8
        PAUSE 1000
        LOW 8
        PAUSE 1000
        HIGH 8
        FREQOUT 0, 2000, 4000
        LOW 8
        PAUSE 1000
        HIGH 8
        PAUSE 1000
        LOW 8
        PAUSE 1000
        HIGH 8
        PAUSE 1000
    NEXT
    LOW 8
    LOW 10
    PAUSE 2000
    FOR idx = 1 TO 150
      PULSOUT 13, 250                       'Rotate back up to starting
position
      PAUSE 20
    NEXT
END                                                  'Program Ends
  ELSE
    PAUSE 1000
    FOR idx = 1 TO 30
      PULSOUT 13, 860                             'Rotate back up to try again
      PAUSE 20
    NEXT
    PAUSE 1000
    FOR idx = 1 TO 30
      PULSOUT 13, 900                             'Rotate down to try and
dispense first half pill
      PAUSE 20
    NEXT
    PAUSE 1000
    FOR idx = 1 TO 30
      PULSOUT 13, 860                             'Rotate back up to try again
      PAUSE 20
    NEXT
    PAUSE 1000
    FOR idx = 1 TO 30
      PULSOUT 13, 900                             'Rotate down to try and
dispense first half pill
      PAUSE 20
    NEXT
    PAUSE 2000
  ENDIF
ELSE
```

```
    PSENSE2 = PSENSE2 - 1
    PAUSE 500
ENDIF
LOOP

PAUSE 1000
FOR idx = 1 TO 100
  PULSOUT 13, 1250
  PAUSE 20                           'Pill drum rotates down to 180 degrees to
dispense other half pill
NEXT

PSENSE2 = 1
ALARM2 = 0
DO WHILE PSENSE2 = 1
IF (IN3 = 0) THEN
  PAUSE 1000
  ALARM2 = ALARM2 + 1
  IF (ALARM2 = 3) THEN                         'Type 2 Error:  Pill is Stuck
    FOR idx = 1 TO 2
        PAUSE 1000
        LOW 8
        PAUSE 500
        HIGH 8
        PAUSE 500
        LOW 8
        PAUSE 500
        HIGH 8
        FREQOUT 0, 2000, 4000
        LOW 8
        PAUSE 500
        HIGH 8
        PAUSE 500
        LOW 8
        PAUSE 500
        HIGH 8
        PAUSE 500
    NEXT
    LOW 8
    LOW 10
    PAUSE 2000
    FOR idx = 1 TO 150
      PULSOUT 13, 250
      PAUSE 20
    NEXT
END
                                                    'Program Ends
ELSE
    PAUSE 1000
    FOR idx = 1 TO 30
      PULSOUT 13, 1200                             'Rotate back UP to try again
      PAUSE 20
    NEXT
    PAUSE 1000
    FOR idx = 1 TO 30
      PULSOUT 13, 1250                             'Rotate down to try and
dispense
      PAUSE 20
    NEXT
  ENDIF
ELSE
    PSENSE2 = PSENSE2 - 1
    PAUSE 500
```

```
ENDIF
LOOP

DEBUG "Half Pill has been dispensed", CR

PAUSE 1000

FOR idx = 1 TO 150
  PULSOUT 13, 250
  PAUSE 20
NEXT

PAUSE 2000

LOW 8
LOW 10
PAUSE 1000
HIGH 8
HIGH 10
PAUSE 1000
LOW 8
LOW 10
PAUSE 1000
HIGH 8
HIGH 10
PAUSE 1000
FREQOUT 0, 2000, 4000                      'ENDING TONE ROTATE BACK TO TOP
PAUSE 1000
LOW 8
LOW 10


PILLS = PILLS - 1

ELSE                 '---------------------Half pill sensor senses pill and
half pill will be delivered---------


FOR idx = 1 TO 100
  PULSOUT 14, 750
  PAUSE 20
NEXT

PAUSE 2000

FOR idx = 1 TO 100
  PULSOUT 14, 250
  PAUSE 20
NEXT

DEBUG "Half Pill has been dispensed", CR

PAUSE 2000

FOR idx = 1 TO 150
  PULSOUT 13, 250
  PAUSE 20
NEXT

PAUSE 2000


LOW 8
LOW 10
PAUSE 1000
HIGH 8
HIGH 10
PAUSE 1000
LOW 8
LOW 10
PAUSE 1000
HIGH 8
HIGH 10
PAUSE 1000
FREQOUT 0, 2000, 4000                        'ENDING TONE ROTATE BACK TO TOP
PAUSE 1000
LOW 8
LOW 10




ENDIF


RETURN   '------------RETURN TO MAIN PROGRAM LOOP???????????????----------




'PIN 0 = Alarm
'PIN 1 = Half Pill Sensor
'PIN 2 = Solenoid
'PIN 3 = Pill Drum Sensor
'PIN 4 = 1/2 Pill Button
'PIN 5 = 1 Pill Button
'PIN 6 = 1 and 1/2 Pills Button
'PIN 7 = 2 Pills Button
'PIN 8 = 1/2 Pill Light
'PIN 9 = 1 Pill Light
'PIN 10 = 1 and 1/2 Pills Light
'PIN 11 = 2 Pills Light
'PIN 12 = Mixing Servo Motor
'PIN 13 = Pill Drum Servo Motor
'PIN 14 = Half Pill Holder/Trap Door Servo Motor
```

## *Final interface program*

**Pill.Java**

```
package scheduler;

import java.util.Date;

/**
 *
 * @author huth
 */
public class Pill {

    public int id;
    public String Description = "";
    public double quantity = 7.0;
    public String DailyOrWeekly = "Daily";
    public int DayOfWeek = 1;
    public int TimesPerDay = 0;
    public String TimeOfDay1 = "";
    public String TimeOfDay2 = "";
    public String TimeOfDay3 = "";
    public String TimeOfDay4 = "";
    public double NumberOfPills1 = 0.0;
    public double NumberOfPills2 = 0.0;
    public double NumberOfPills3 = 0.0;
    public double NumberOfPills4 = 0.0;
    public double Total_Pills_Inserted = 0.0;
//    public Date dispenseDtm;

    public Pill () {}

    public void setID (int ai_ID) { id = ai_ID; }

    public int getID(){ return id; }

    public void setTotalPillsInserted (double a_total_pills_inserted) { Total_Pills_Inserted = a_total_pills_inserted;}

    public double getTotalPillsInserted() { return Total_Pills_Inserted; }

    public void setDesc (String as_description) { Description = as_description; }

    public String getDesc () { return Description; }

    public int GetTimesPerDay () { return TimesPerDay; }

    public void SetTimesPerDay(int al_TimesPerDay) { TimesPerDay = al_TimesPerDay; }

    public void setQuantity (double as_quantity) { quantity = as_quantity; }

    public double getQuantity () { return quantity; }

    public void SetTimeOfDay1 (String adt_timeofday) { TimeOfDay1 = adt_timeofday; }

public String GetTimeOfDay1 () { return TimeOfDay1; }

    public void SetTimeOfDay2 (String adt_timeofday) { TimeOfDay2 = adt_timeofday; }

    public String GetTimeOfDay2 () { return TimeOfDay2; }

    public void SetTimeOfDay3 (String adt_timeofday) { TimeOfDay3 = adt_timeofday; }

    public String GetTimeOfDay3 () { return TimeOfDay3; }

    public void SetTimeOfDay4 (String adt_timeofday) { TimeOfDay4 = adt_timeofday; }

    public String GetTimeOfDay4 () { return TimeOfDay4; }

    public void SetNumberOfPills1 (double ad_numberOfPills) { NumberOfPills1 = ad_numberOfPills; }

    public double GetNumberOfPills1 () { return NumberOfPills1; }

    public void SetNumberOfPills2 (double ad_numberOfPills) { NumberOfPills2 = ad_numberOfPills; }

    public double GetNumberOfPills2 () { return NumberOfPills2; }

    public void SetNumberOfPills3 (double ad_numberOfPills) { NumberOfPills3 = ad_numberOfPills; }

    public double GetNumberOfPills3 () { return NumberOfPills3; }
    public void SetNumberOfPills4 (double ad_numberOfPills) { NumberOfPills4 = ad_numberOfPills; }

    public double GetNumberOfPills4 () { return NumberOfPills4; }

    public void SetDayOfWeek (int a_dayOfWk) { DayOfWeek = a_dayOfWk; }

    public double GetDayOfWeek () { return DayOfWeek; }

    public void SetDailyOrWeekly (String a_dailyOrWeekly) { DailyOrWeekly = a_dailyOrWeekly; }

    public String GetDailyOrWeekly () { return DailyOrWeekly; }

   // public void setDispenseDtm (Date as_dispenseDtm) { dispenseDtm = dispenseDtm; }

    //public Date getDispenseDtm () { return dispenseDtm; }

    public String toString() { return ""; }
}
```

**Window.Java**

```
package scheduler;

/**
 *
 * @author huth
 */
public class Window extends javax.swing.JFrame {
    protected Pill pill = null;
    protected int timesPerDayCnt;
    protected int windowNum;


    public void setPill(Pill a_pill) { pill = a_pill; }

    public Pill getPill() { return pill; }

    public void setTimesPerDayCnt(int a_timesPerDayCnt) { timesPerDayCnt = a_timesPerDayCnt; }

    public int gettimesPerDayCnt() { return timesPerDayCnt; }

    public void setWindowNum (int a_WindowNum) { windowNum = a_WindowNum;  }

    public int getWindowNum() { return windowNum; }

    }
```

**WindowManager.Java**

```
package scheduler;

import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFrame;

/**
 *
 * @author huth
 */
public class WindowManager {

    public static void switchWindow (Window currentWindow, int al_destinationWindow)
    {
       int destinationWindow = al_destinationWindow;
       int timesPerDayCnt = currentWindow.gettimesPerDayCnt();
       Pill pill = currentWindow.getPill();
       Window window = WindowManager.getWindow(destinationWindow, pill);


       if (destinationWindow == 7)
       {
          timesPerDayCnt++;
          if(timesPerDayCnt < pill.GetTimesPerDay())
          {
             window = WindowManager.getWindow(5, pill);
          }
       }
       else if (destinationWindow == 4)
       {
          if (timesPerDayCnt > 0 && currentWindow.windowNum!=3)
          {
             timesPerDayCnt--;
             window = WindowManager.getWindow(6, pill);
          }
       }

       window.setTimesPerDayCnt(timesPerDayCnt);



       window.setVisible(true);
       currentWindow.setVisible(false);
    }

    public static void switchWindow (int al_destinationWindow)
    {
       int destinationWindow = al_destinationWindow;
       Window window = WindowManager.getWindow(destinationWindow, new Pill());
       window.setVisible(true);
    }

    private static Window getWindow (int al_destinationWindow, Pill a_pill)
    {
 Window window = null;
    try {
          int destinationWindow = al_destinationWindow;
             switch (destinationWindow) {
             case 1:
                window = new PillMaintenanceWindow();
                break;
             case 2:
                window = new PillsInserted(a_pill);
                break;
             case 3:
                window = new DailyWeekly(a_pill);
                break;
             case 4:
                window = new TimePerDay(a_pill);
                break;
             case 5:
                window = new TimesOfDay(a_pill);
                break;
             case 6:
                window = new NumberOfPills(a_pill);
                break;
            case 7:
                window = new PillAddConfirm(a_pill);
                break;
            case 8:
                window = new DayOfWeek(a_pill);
                break;
            case 9:
                window = new Description(a_pill);
                break;
        }
        window.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        window.setLocationRelativeTo(null);
        window.pack();
    }
    catch (ClassNotFoundException ex) {
        Logger.getLogger(WindowManager.class.getName()).log(Level.SEVERE, null, ex);
    }
    return window;
  }
}
```

**PillMaintenence.Window**

```
package scheduler;

import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author  huth
 */
public class PillMaintenanceWindow extends Window {
    private JTable jTable2;
    private ArrayList<Pill> pills;
    private XmlDAO xmlDAO;
    private BspDAO bspDAO;
    public int destinationWindow = 1;

    /** Creates new form PillMaintenanceWindow */
    public PillMaintenanceWindow() throws ClassNotFoundException {
        //try {
            this.setWindowNum(1);
            bspDAO = new BspDAO("Pill.bsp");
            xmlDAO = new XmlDAO("Pill.xml");
            pills = xmlDAO.read();
            jTable2 = xmlDAO.fill(pills);
            initComponents();
            //Pill newPill = new Pill();
            //newPill.setID(3);
            //newPill.setDesc("Pill3");
            //newPill.setQuantity(1.5);
            //newPill.setDispenseDtm(new Date());
            //pills.add(newPill);
            //xmlDAO.write(pills);
            // jTable1.set
      //  } catch (FileNotFoundException ex) {
```

```
    //     Logger.getLogger(PillMaintenanceWindow.class.getName()).log(Level.SEVERE, null, ex);
    // }
     // jTable1 = xmlDAO.fill(pills);

  }

  /** This method is called from within the constructor to
   * initialize the form.
   * WARNING: Do NOT modify this code. The content of this method is
   * always regenerated by the Form Editor.
   */
  // <editor-fold defaultstate="collapsed" desc="Generated Code">
  private void initComponents() {

      jScrollPane1 = new javax.swing.JScrollPane();
      jTable1 = new javax.swing.JTable();
      jButton1 = new javax.swing.JButton();
      jButton3 = new javax.swing.JButton();

      setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

      jTable1.setFont(new java.awt.Font("Lucida Grande", 0, 24));
      jTable1.setModel(jTable2.getModel());
      jTable1.setAutoResizeMode(javax.swing.JTable.AUTO_RESIZE_ALL_COLUMNS);
      jTable1.setFocusable(false);
      jTable1.setRowHeight(24);
      jTable1.setSelectionBackground(new java.awt.Color(241, 241, 255));
      jScrollPane1.setViewportView(jTable1);

      jButton1.setFont(new java.awt.Font("Lucida Grande", 1, 24));
      jButton1.setText("Add");
      jButton1.addActionListener(new java.awt.event.ActionListener() {
          public void actionPerformed(java.awt.event.ActionEvent evt) {
              jButton1ActionPerformed(evt);
          }
      });

      jButton3.setFont(new java.awt.Font("Lucida Grande", 1, 24));
      jButton3.setText("Delete");
      jButton3.addActionListener(new java.awt.event.ActionListener() {
          public void actionPerformed(java.awt.event.ActionEvent evt) {
              jButton3ActionPerformed(evt);
          }
      });

      org.jdesktop.layout.GroupLayout layout = new org.jdesktop.layout.GroupLayout(getContentPane());
      getContentPane().setLayout(layout);
      layout.setHorizontalGroup(
          layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
          .add(layout.createSequentialGroup()
              .addContainerGap()
              .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                  .add(jScrollPane1, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 810, Short.MAX_VALUE)
                  .add(org.jdesktop.layout.GroupLayout.TRAILING, layout.createSequentialGroup()
                      .add(jButton1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 83,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                      .add(23, 23, 23)
                      .add(jButton3, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 107,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
              .addContainerGap())
      );
      layout.setVerticalGroup(
          layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
          .add(org.jdesktop.layout.GroupLayout.TRAILING, layout.createSequentialGroup()
              .addContainerGap()
              .add(jScrollPane1, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 459, Short.MAX_VALUE)
              .add(18, 18, 18)
              .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                  .add(jButton1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 40,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                  .add(jButton3, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 41,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
              .addContainerGap())
      );

      java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
      setBounds((screenSize.width-850)/2, (screenSize.height-577)/2, 850, 577);
  }// </editor-fold>

  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
     //1. Create the frame.
     //TimesPerDay.main(args)
     //PillType.main(null);
     //this.setVisible(false);
      pill = new Pill();
      WindowManager.switchWindow(this, 9);
      // destinationWindow = 2;
      //frame = new TimesPerDay("FrameDemo");

//2. Optional: What happens when the frame closes?
//frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

//3. Create components and put them in the frame.
//...create emptyLabel...
//frame.getContentPane().add(emptyLabel, BorderLayout.CENTER);

//4. Size the frame.
//frame.pack();

//5. Show it.
//frame.setVisible(true);

  }

  private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
      try {
          int selectedRow = jTable1.getSelectedRow();

          if (jTable1.getRowCount() > 0)
          {
              ((DefaultTableModel)jTable1.getModel()).removeRow(selectedRow);
              pills.remove(selectedRow - 1);
              xmlDAO.write(pills);
              bspDAO.write(pills);
          }
      } catch (FileNotFoundException ex) {
          Logger.getLogger(PillMaintenanceWindow.class.getName()).log(Level.SEVERE, null, ex);
      }
  }

  /**
   * @param args the command line arguments
   */
  public static void main(String args[]) {
      java.awt.EventQueue.invokeLater(new Runnable() {
          public void run() {
              try {
                  new PillMaintenanceWindow().setVisible(true);
              } catch (ClassNotFoundException ex) {
                  Logger.getLogger(PillMaintenanceWindow.class.getName()).log(Level.SEVERE, null, ex);
              }
          }
      });

      //XmlDAO xmlDAO = new XmlDAO("Pill.xml");
      //ArrayList<Pill> pills = xmlDAO.read();
      //jTable1 = xmlDAO.fill(pills);
  }

  // Variables declaration - do not modify
  private javax.swing.JButton jButton1;
  private javax.swing.JButton jButton3;
  private javax.swing.JScrollPane jScrollPane1;
  private javax.swing.JTable jTable1;
  // End of variables declaration

}
```

**Description.Java**

```
package scheduler;

/**
 *
 * @author  huth
 */
public class Description extends Window {

    /** Creates new form Description */
    public Description() {
        initComponents();
    }

    public Description(Pill a_pill) {
        this.setWindowNum(9);
        pill = a_pill;
        initComponents();
        errorMessage.setVisible(false);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jTextField1 = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        errorMessage = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        addWindowFocusListener(new java.awt.event.WindowFocusListener() {
            public void windowGainedFocus(java.awt.event.WindowEvent evt) {
                formWindowGainedFocus(evt);
            }
            public void windowLostFocus(java.awt.event.WindowEvent evt) {
            }
        });

        jTextField1.setFont(new java.awt.Font("Lucida Grande", 0, 24));
        jTextField1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField1ActionPerformed(evt);
            }
        });

        jLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 28));
        jLabel1.setText("Enter Pill Discription:");

        jButton1.setFont(new java.awt.Font("Lucida Grande", 1, 24));
        jButton1.setText("Next");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setFont(new java.awt.Font("Lucida Grande", 1, 24));
        jButton2.setText("Back");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        errorMessage.setFont(new java.awt.Font("Lucida Grande", 1, 28));
        errorMessage.setForeground(new java.awt.Color(255, 0, 0));
        errorMessage.setText("Please enter a description");

        org.jdesktop.layout.GroupLayout layout = new org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(layout.createSequentialGroup()
                        .addContainerGap()
                        .add(jButton2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 89,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 613, Short.MAX_VALUE)
                        .add(jButton1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 103,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                    .add(layout.createSequentialGroup()
                        .add(29, 29, 29)
                        .add(jLabel1))
                    .add(layout.createSequentialGroup()
                        .addContainerGap()
                        .add(errorMessage))
                    .add(layout.createSequentialGroup()
                        .addContainerGap()
                        .add(jTextField1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 587,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .addContainerGap()
                .add(jLabel1)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                .add(jTextField1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                .add(errorMessage)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 358, Short.MAX_VALUE)
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                    .add(jButton2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 43,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(jButton1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 39,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .addContainerGap())
        );

        java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-839)/2, (screenSize.height-580)/2, 839, 580);
    }// </editor-fold>

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        pill.setDesc("");

        WindowManager.switchWindow(this, 1);
    }

    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        if (jTextField1.getText().trim().length() == 0)
        {
            errorMessage.setVisible(true);
        }
        else
        {
            errorMessage.setVisible(false);

            pill.setDesc(jTextField1.getText());

            WindowManager.switchWindow(this, 2);
        }
    }

    private void formWindowGainedFocus(java.awt.event.WindowEvent evt) {
        jTextField1.setText(pill.getDesc());
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Description().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JLabel errorMessage;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JTextField jTextField1;
    // End of variables declaration

}
```

**PillsInserted.Java**

```
package scheduler;

/**
 *
 * @author  huth
 */
public class PillsInserted extends Window {

    /** Creates new form PillsInserted */
    public PillsInserted() {
        initComponents();
    }

    public PillsInserted(Pill a_pill) {
        this.setWindowNum(2);
        pill = a_pill;
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        buttonGroup1 = new javax.swing.ButtonGroup();
        jLabel1 = new javax.swing.JLabel();
        jRadioButton1 = new javax.swing.JRadioButton();
        jRadioButton2 = new javax.swing.JRadioButton();
        jRadioButton3 = new javax.swing.JRadioButton();
        jRadioButton4 = new javax.swing.JRadioButton();
        jRadioButton5 = new javax.swing.JRadioButton();
        jButton2 = new javax.swing.JButton();
        jButton1 = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        addWindowFocusListener(new java.awt.event.WindowFocusListener() {
            public void windowGainedFocus(java.awt.event.WindowEvent evt) {
                formWindowGainedFocus(evt);
            }
            public void windowLostFocus(java.awt.event.WindowEvent evt) {
            }
        });

        jLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 28));
        jLabel1.setText("How many weeks/months worth of pills were inserted?");
```

```
buttonGroup1.add(jRadioButton1);
jRadioButton1.setFont(new java.awt.Font("Lucida Grande", 0, 24));
jRadioButton1.setText("1 Week");
jRadioButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton1ActionPerformed(evt);
    }
});

buttonGroup1.add(jRadioButton2);
jRadioButton2.setFont(new java.awt.Font("Lucida Grande", 0, 24));
jRadioButton2.setText("2 Weeks");
jRadioButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton2ActionPerformed(evt);
    }
});

buttonGroup1.add(jRadioButton3);
jRadioButton3.setFont(new java.awt.Font("Lucida Grande", 0, 24));
jRadioButton3.setText("3 Weeks");
jRadioButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton3ActionPerformed(evt);
    }
});

buttonGroup1.add(jRadioButton4);
jRadioButton4.setFont(new java.awt.Font("Lucida Grande", 0, 24));
jRadioButton4.setText("1 Month");
jRadioButton4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton4ActionPerformed(evt);
    }
});

buttonGroup1.add(jRadioButton5);
jRadioButton5.setFont(new java.awt.Font("Lucida Grande", 0, 24));
jRadioButton5.setText("2 Months");
jRadioButton5.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton5ActionPerformed(evt);
    }
});

jButton2.setFont(new java.awt.Font("Lucida Grande", 1, 24));
jButton2.setText("Back");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

jButton1.setFont(new java.awt.Font("Lucida Grande", 1, 24));
jButton1.setText("Next");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});
```

```
        jLabel2.setFont(new java.awt.Font("Lucida Grande", 1, 24));

        org.jdesktop.layout.GroupLayout layout = new org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .addContainerGap()
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(layout.createSequentialGroup()
                        .add(jRadioButton3, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 814, Short.MAX_VALUE)
                        .addContainerGap())
                    .add(layout.createSequentialGroup()
                        .add(jRadioButton1)
                        .addContainerGap(720, Short.MAX_VALUE))
                    .add(org.jdesktop.layout.GroupLayout.TRAILING, layout.createSequentialGroup()
                        .add(jLabel2)
                        .add(297, 297, 297))
                    .add(layout.createSequentialGroup()
                        .add(jRadioButton2)
                        .addContainerGap(708, Short.MAX_VALUE))
                    .add(layout.createSequentialGroup()
                        .add(jRadioButton4, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 769, Short.MAX_VALUE)
                        .add(61, 61, 61))
                    .add(layout.createSequentialGroup()
                        .add(jRadioButton5, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 769, Short.MAX_VALUE)
                        .add(61, 61, 61))))
            .add(layout.createSequentialGroup()
                .add(10, 10, 10)
                .add(jLabel1)
                .addContainerGap(42, Short.MAX_VALUE))
            .add(layout.createSequentialGroup()
                .addContainerGap()
                .add(jButton2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 88,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 636, Short.MAX_VALUE)
                .add(jButton1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 89,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .addContainerGap()
                .add(jLabel1)
                .add(24, 24, 24)
                .add(jRadioButton1)
                .add(26, 26, 26)
                .add(jRadioButton2)
                .add(29, 29, 29)
                .add(jRadioButton3)
                .add(28, 28, 28)
                .add(jRadioButton4)
                .add(30, 30, 30)
                .add(jRadioButton5)
                .add(93, 93, 93)
                .add(jLabel2)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 44, Short.MAX_VALUE)
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                    .add(jButton2)
                    .add(jButton1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 39,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
```

```
            .addContainerGap())
    );

    java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
    setBounds((screenSize.width-847)/2, (screenSize.height-576)/2, 847, 576);
}// </editor-fold>

private void jRadioButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jRadioButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jRadioButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jRadioButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    pill.setQuantity(7.0);

    WindowManager.switchWindow(this, 9);
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    if (jRadioButton1.isSelected())
    {
        pill.setQuantity(7.0);
    }
    else if (jRadioButton2.isSelected())
    {
        pill.setQuantity(14.0);
    }
    else if (jRadioButton3.isSelected())
    {
        pill.setQuantity(21.0);
    }
    else if (jRadioButton4.isSelected())
    {
        pill.setQuantity(30.0);
    }
    else if (jRadioButton5.isSelected())
    {
        pill.setQuantity(60.0);
    }

    WindowManager.switchWindow(this, 3);
}

private void formWindowGainedFocus(java.awt.event.WindowEvent evt) {
    if (pill.getQuantity() == 7.0)
    {
        jRadioButton1.setSelected(true);
    }
    else if (pill.getQuantity() == 14.0)
    {
        jRadioButton2.setSelected(true);
        }
        else if (pill.getQuantity() == 21.0)
        {
            jRadioButton3.setSelected(true);
        }
        else if (pill.getQuantity() == 30.0)
        {
            jRadioButton4.setSelected(true);
        }
        else if (pill.getQuantity() == 60.0)
        {
            jRadioButton5.setSelected(true);
        }
    }

    private void jRadioButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new PillsInserted().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JRadioButton jRadioButton1;
    private javax.swing.JRadioButton jRadioButton2;
    private javax.swing.JRadioButton jRadioButton3;
    private javax.swing.JRadioButton jRadioButton4;
    private javax.swing.JRadioButton jRadioButton5;
    // End of variables declaration

}
```

**DailyWeekly.Java**

```
package scheduler;

/**
 *
 * @author  huth
 */
public class DailyWeekly extends Window {

    /** Creates new form DailyWeekly */
    public DailyWeekly() {
        initComponents();
    }

    public DailyWeekly(Pill a_pill) {
        this.setWindowNum(3);
        pill = a_pill;
    initComponents();
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    buttonGroup1 = new javax.swing.ButtonGroup();
    buttonGroup2 = new javax.swing.ButtonGroup();
    buttonGroup3 = new javax.swing.ButtonGroup();
    jLabel1 = new javax.swing.JLabel();
    rbWeekly = new javax.swing.JRadioButton();
    jButton3 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();
    rbDaily = new javax.swing.JRadioButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    addWindowFocusListener(new java.awt.event.WindowFocusListener() {
        public void windowGainedFocus(java.awt.event.WindowEvent evt) {
            formWindowGainedFocus(evt);
        }
        public void windowLostFocus(java.awt.event.WindowEvent evt) {
        }
    });

    jLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 28));
    jLabel1.setText("How often does the paitient take the pill?");

    buttonGroup1.add(rbWeekly);
    rbWeekly.setFont(new java.awt.Font("Lucida Grande", 0, 24));
    rbWeekly.setText("Weekly");
    rbWeekly.setAutoscrolls(true);
    rbWeekly.setInheritsPopupMenu(true);
    rbWeekly.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            rbWeeklyActionPerformed(evt);
        }
    });

    jButton3.setFont(new java.awt.Font("Lucida Grande", 1, 24));
    jButton3.setText("Back");
    jButton3.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton3ActionPerformed(evt);
        }
    });

    jButton2.setFont(new java.awt.Font("Lucida Grande", 1, 24));
    jButton2.setText("Next");
    jButton2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton2ActionPerformed(evt);
        }
    });

    buttonGroup1.add(rbDaily);
    rbDaily.setFont(new java.awt.Font("Lucida Grande", 0, 24));
    rbDaily.setText("Daily");
```

```
        org.jdesktop.layout.GroupLayout layout = new org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .add(32, 32, 32)
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(rbDaily)
                    .add(jLabel1)
                    .add(layout.createSequentialGroup()
                        .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                            .add(rbWeekly)
                            .add(jButton3, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 94,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 620, Short.MAX_VALUE)
                        .add(jButton2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 89,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .addContainerGap()
                .add(jLabel1)
                .add(18, 18, 18)
                .add(rbDaily)
                .add(13, 13, 13)
                .add(rbWeekly)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 340, Short.MAX_VALUE)
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                    .add(jButton2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 40,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(jButton3, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 41,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .addContainerGap())
        );

        java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-844)/2, (screenSize.height-573)/2, 844, 573);
    }// </editor-fold>

    private void rbWeeklyActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        if (rbDaily.isSelected()){
            pill.SetDailyOrWeekly("Daily");
            WindowManager.switchWindow(this, 4);
        }
        else{
            pill.SetDailyOrWeekly("Weekly");
            WindowManager.switchWindow(this, 8);
        }
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        pill.SetDailyOrWeekly("Daily");
        WindowManager.switchWindow(this, 2);
    }
    private void formWindowGainedFocus(java.awt.event.WindowEvent evt) {
        // TODO add your handling code here:
        if (pill.GetDailyOrWeekly().equals("Daily"))
        {
            rbDaily.setSelected(true);
        }
        else if (pill.GetDailyOrWeekly().equals("Weekly"))
        {
            rbWeekly.setSelected(true);
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new DailyWeekly().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.ButtonGroup buttonGroup2;
    private javax.swing.ButtonGroup buttonGroup3;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JLabel jLabel1;
    protected javax.swing.JRadioButton rbDaily;
    protected javax.swing.JRadioButton rbWeekly;
    // End of variables declaration

}
```

**TimesPerDay.Java**

```
package scheduler;

/**
 *
 * @author  huth
 */
public class TimePerDay extends Window {

    /** Creates new form TimePerDay */
    public TimePerDay() {
        initComponents();
    }

    public TimePerDay(Pill a_pill) {
        this.setWindowNum(4);
        pill = a_pill;
        initComponents();

         switch (pill.GetTimesPerDay())
        {
            case 0:
                rbTime1.setSelected(true);
                break;
            case 1:
                rbTime1.setSelected(true);
                break;
            case 2:
                rbTime2.setSelected(true);
                break;
            case 3:
                rbTime3.setSelected(true);
                break;
            case 4:
                rbTime4.setSelected(true);
                break;

        }
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        buttonGroup1 = new javax.swing.ButtonGroup();
        buttonGroup2 = new javax.swing.ButtonGroup();
        jLabel1 = new javax.swing.JLabel();
        rbTime1 = new javax.swing.JRadioButton();
        rbTime2 = new javax.swing.JRadioButton();
        rbTime3 = new javax.swing.JRadioButton();
        rbTime4 = new javax.swing.JRadioButton();
        jButton5 = new javax.swing.JButton();
        jButton1 = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 28));
        jLabel1.setText("How many times does the patient take the pill per day?");

        buttonGroup1.add(rbTime1);
        rbTime1.setFont(new java.awt.Font("Lucida Grande", 0, 24));
        rbTime1.setText("1");

        buttonGroup1.add(rbTime2);
        rbTime2.setFont(new java.awt.Font("Lucida Grande", 0, 24));
        rbTime2.setText("2");

        buttonGroup1.add(rbTime3);
        rbTime3.setFont(new java.awt.Font("Lucida Grande", 0, 24));
        rbTime3.setText("3");
        rbTime3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                rbTime3ActionPerformed(evt);
            }
        });

        buttonGroup1.add(rbTime4);
        rbTime4.setFont(new java.awt.Font("Lucida Grande", 0, 24));
        rbTime4.setText("4");
        rbTime4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                rbTime4ActionPerformed(evt);
            }
        });
```

```
        jButton5.setFont(new java.awt.Font("Lucida Grande", 1, 24));
        jButton5.setText("Back");
        jButton5.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton5ActionPerformed(evt);
            }
        });

        jButton1.setFont(new java.awt.Font("Lucida Grande", 1, 24));
        jButton1.setText("Next");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jLabel2.setFont(new java.awt.Font("Lucida Grande", 1, 24));

        org.jdesktop.layout.GroupLayout layout = new org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(layout.createSequentialGroup()
                        .add(40, 40, 40)
                        .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                            .add(layout.createSequentialGroup()
                                .add(817, 817, 817)
                                .add(jLabel2))
                            .add(rbTime4)
                            .add(rbTime3)
                            .add(rbTime1)
                            .add(rbTime2)))
                    .add(layout.createSequentialGroup()
                        .addContainerGap()
                        .add(jLabel1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 809,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .add(org.jdesktop.layout.GroupLayout.TRAILING, layout.createSequentialGroup()
                .addContainerGap()
                .add(jButton5, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 87,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 611, Short.MAX_VALUE)
                .add(jButton1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 93,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(49, 49, 49))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .add(20, 20, 20)
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(jLabel2)
                    .add(jLabel1))
                .add(18, 18, 18)
                .add(rbTime1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 22,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(29, 29, 29)
                .add(rbTime2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 26,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
```

```
            .add(26, 26, 26)
            .add(rbTime3)
            .add(29, 29, 29)
            .add(rbTime4)
            .add(233, 233, 233)
            .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
              .add(jButton1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 41,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
              .add(jButton5, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 41, Short.MAX_VALUE))
            .add(25, 25, 25))
    );

    java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
    setBounds((screenSize.width-828)/2, (screenSize.height-585)/2, 828, 585);
  }// </editor-fold>

  private void rbTime3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
  }

  private void rbTime4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
  }

  private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    pill.SetTimesPerDay(0);
    WindowManager.switchWindow(this, 3);
  }

  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    if (rbTime1.isSelected())
    {
      pill.SetTimesPerDay(1);
    }
    else if (rbTime2.isSelected())
    {
      pill.SetTimesPerDay(2);
    }
    else if (rbTime3.isSelected())
    {
      pill.SetTimesPerDay(3);
    }
    else if (rbTime4.isSelected())
    {
      pill.SetTimesPerDay(4);
    }

    WindowManager.switchWindow(this, 5);
    // if (rbTime1.isCursorSet())
    //{
    //     pill.SetTimesPerDay(1);
    //} else if (rbTime2.isCursorSet())
    //{
    //     pill.SetTimesPerDay(2);
    //} else if (rbTime3.isCursorSet())
    //{    pill.SetTimesPerDay(3);
    //} else
    //{    pill.SetTimesPerDay(4);
    //WindowManager.switchWindow(this, number);


    // if (rbTime1.isCursorSet())
```

```
        //{
        //      pill.SetTimesPerDay(1);
        //} else if (rbTime2.isCursorSet())
        //{
        //      pill.SetTimesPerDay(2);
        //} else if (rbTime3.isCursorSet())
        //{    pill.SetTimesPerDay(3);
        //} else
        //{    pill.SetTimesPerDay(4);
        //WindowManager.switchWindow(this, number);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new TimePerDay().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.ButtonGroup buttonGroup2;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton5;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JRadioButton rbTime1;
    private javax.swing.JRadioButton rbTime2;
    private javax.swing.JRadioButton rbTime3;
    private javax.swing.JRadioButton rbTime4;
    // End of variables declaration
```

**TimeOfDay.Java**

```
package scheduler;

/**
 *
 * @author  huth
 */
public class TimesOfDay extends Window {

    /** Creates new form TimesOfDay */
    public TimesOfDay() {
        initComponents();
    }

    public TimesOfDay(Pill a_pill) {
        this.setWindowNum(5);
        pill = a_pill;
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        lbTimeOfDay = new javax.swing.JList();
        jButton2 = new javax.swing.JButton();
        jButton1 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        addWindowStateListener(new java.awt.event.WindowStateListener() {
            public void windowStateChanged(java.awt.event.WindowEvent evt) {
                formWindowStateChanged(evt);
            }
        });
        addWindowFocusListener(new java.awt.event.WindowFocusListener() {
            public void windowGainedFocus(java.awt.event.WindowEvent evt) {
                formWindowGainedFocus(evt);
            }
            public void windowLostFocus(java.awt.event.WindowEvent evt) {
            }
        });

        jLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 28));
        jLabel1.setText("What is the first time of day the pill is administered?");

        lbTimeOfDay.setFont(new java.awt.Font("Lucida Grande", 0, 24));
        lbTimeOfDay.setModel(new javax.swing.AbstractListModel() {
            String[] strings = { "1:00 am", "1:30 am", "2:00 am", "2:30 am", "3:00am", "3:30 am", "4:00 am", "4:30 am",
"5:00 am", "5:30 am", "6:00 am", "6:30 am", "7:00 am", "7:30 am", "8:00 am", "8:30 am", "9:00 am", "9:30 am",
"10:00 am", "10:30 am", "11:00 am", "11:30 am", "12:00 pm", "12:30 pm", "1:00 pm", "1:30 pm", "2:00 pm", "2:30
pm", "3:00 pm", "3:30 pm", "4:00 pm", "4:30 pm", "5:00 pm", "5:30 pm", "6:00 pm", "6:30 pm", "7:00 pm", "7:30
pm", "8:00 pm", "8:30 pm", "9:00 pm", "9:30 pm", "10:00 pm", "10:30 pm", "11:00 pm", "11:30 pm" };
            public int getSize() { return strings.length; }
            public Object getElementAt(int i) { return strings[i]; }
        });
        jScrollPane1.setViewportView(lbTimeOfDay);

        jButton2.setFont(new java.awt.Font("Lucida Grande", 1, 24));
        jButton2.setText("Back");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        jButton1.setFont(new java.awt.Font("Lucida Grande", 1, 24));
        jButton1.setText("Next");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        org.jdesktop.layout.GroupLayout layout = new org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .addContainerGap()
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(jLabel1, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 807, Short.MAX_VALUE)
                    .add(layout.createSequentialGroup()
                    .add(jButton2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 106,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(232, 232, 232)
                    .add(jScrollPane1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 236, Short.MAX_VALUE)
                    .add(jButton1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 99,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.TRAILING, layout.createSequentialGroup()
                .addContainerGap(31, Short.MAX_VALUE)
                .add(jLabel1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 34,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(18, 18, 18)
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
                    .add(layout.createSequentialGroup()
                        .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                            .add(jButton1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 40,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                            .add(jButton2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 38,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                        .addContainerGap())
                    .add(layout.createSequentialGroup()
                        .add(jScrollPane1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 430,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .add(44, 44, 44))))
        );

        java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-847)/2, (screenSize.height-579)/2, 847, 579);
    }// </editor-fold>

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        switch (timesPerDayCnt)
        {
            case 0:
                pill.TimeOfDay1 = "";
                break;

            case 1:
                pill.TimeOfDay2 = "";
                break;

            case 2:
                pill.TimeOfDay3 = "";
                break;

            case 3:
                pill.TimeOfDay4 = "";
                break;
        }

        if (pill.GetDailyOrWeekly().equals("D"))
        {
            WindowManager.switchWindow(this, 4);
        }
        else
        {
            WindowManager.switchWindow(this, 8);
        }
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        String timeOfDay = lbTimeOfDay.getSelectedValue().toString();
        switch (timesPerDayCnt)
        {
            case 0:
                pill.SetTimeOfDay1(timeOfDay);
                break;
            case 1:
                pill.SetTimeOfDay2(timeOfDay);
                break;
            case 2:
                pill.SetTimeOfDay3(timeOfDay);
                break;
            case 3:
                pill.SetTimeOfDay4(timeOfDay);
                break;
        }
        //if ((Pill.NbrPills = 1 and !Pill.Time1.equals(null)) or
        //    (Pill.NbrPills = 2 and !Pill.Time2.equals(null)
       WindowManager.switchWindow(this, 6);
    }

    private void formWindowStateChanged(java.awt.event.WindowEvent evt) {
        // TODO add your handling code here:
    }

    private void formWindowGainedFocus(java.awt.event.WindowEvent evt) {
        // TODO add your handling code here:
        String TimeNum = "first";
        String timeOfDay = "1:00 am";
        switch (timesPerDayCnt)
        {
            case 0:
                timeOfDay = (pill.TimeOfDay1 == "" ? timeOfDay : pill.TimeOfDay1);
                TimeNum = "first";
                break;

            case 1:
                timeOfDay = (pill.TimeOfDay2 == "" ? timeOfDay : pill.TimeOfDay2);
                TimeNum = "second";
                break;

            case 2:
                timeOfDay = (pill.TimeOfDay3 == "" ? timeOfDay : pill.TimeOfDay3);
                TimeNum = "third";
                break;

            case 3:
                timeOfDay = (pill.TimeOfDay4 == "" ? timeOfDay : pill.TimeOfDay4);
                TimeNum = "forth";
                break;

        }
        jLabel1.setText("What is the " + TimeNum + " time of day the pill is administered?");
        lbTimeOfDay.setSelectedValue(timeOfDay, true);
    }

    /**
```

```
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new TimesOfDay().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JList lbTimeOfDay;
    // End of variables declaration

}
```

**NumberOfPills.Java**

```
package scheduler;

/**
 *
 * @author  huth
 */
public class NumberOfPills extends Window {

    /** Creates new form NumberOfPills */
    public NumberOfPills() {
        initComponents();
    }

    public NumberOfPills(Pill a_pill) {
        this.setWindowNum(6);
        pill = a_pill;
        initComponents();

              //switch()

        //switch(pill.TimeOfDay)
    }
            //String pillNbr;
                //if Pill.Time1.equals(null)
        //{
        //      pillNbr = "first";
        //} else if Pill.Time2.equals(null)
        //{
        //      pillNbr = "second";
        //} else if Pill.Time3.equals(null)
        //{     pillNbr = "third";
        //} else
        //{     pillNbr = "fourth":
        //}
        //  jLabel1 = "How many pills need to be administered the " + pillNbr + " time?";

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
```

```
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    buttonGroup1 = new javax.swing.ButtonGroup();
    jScrollPane1 = new javax.swing.JScrollPane();
    jTable1 = new javax.swing.JTable();
    buttonGroup2 = new javax.swing.ButtonGroup();
    jLabel1 = new javax.swing.JLabel();
    rbPill5 = new javax.swing.JRadioButton();
    rbPill10 = new javax.swing.JRadioButton();
    rbPill15 = new javax.swing.JRadioButton();
    rbPill20 = new javax.swing.JRadioButton();
    rbPill25 = new javax.swing.JRadioButton();
    rbPill30 = new javax.swing.JRadioButton();
    jButton1 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();
    jLabel2 = new javax.swing.JLabel();

    jTable1.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {
            {null, null, null, null},
            {null, null, null, null},
            {null, null, null, null},
            {null, null, null, null}
        },
        new String [] {
            "Title 1", "Title 2", "Title 3", "Title 4"
        }
    ));
    jScrollPane1.setViewportView(jTable1);

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    addWindowFocusListener(new java.awt.event.WindowFocusListener() {
        public void windowGainedFocus(java.awt.event.WindowEvent evt) {
            formWindowGainedFocus(evt);
        }
        public void windowLostFocus(java.awt.event.WindowEvent evt) {
        }
    });

    jLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 28));
    jLabel1.setText("How many pills need to be administered ");

    buttonGroup1.add(rbPill5);
    rbPill5.setFont(new java.awt.Font("Lucida Grande", 0, 24));
    rbPill5.setText("1/2");
    rbPill5.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            rbPill5ActionPerformed(evt);
        }
    });

    buttonGroup1.add(rbPill10);
    rbPill10.setFont(new java.awt.Font("Lucida Grande", 0, 24));
    rbPill10.setText("1");
    rbPill10.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            rbPill10ActionPerformed(evt);
        }
    });

    buttonGroup1.add(rbPill15);
```

```
        rbPill15.setFont(new java.awt.Font("Lucida Grande", 0, 24));
        rbPill15.setText("1 1/2");
        rbPill15.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                rbPill15ActionPerformed(evt);
            }
        });

        buttonGroup1.add(rbPill20);
        rbPill20.setFont(new java.awt.Font("Lucida Grande", 0, 24));
        rbPill20.setText("2");

        buttonGroup1.add(rbPill25);
        rbPill25.setFont(new java.awt.Font("Lucida Grande", 0, 24));
        rbPill25.setText("2 1/2");
        rbPill25.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                rbPill25ActionPerformed(evt);
            }
        });

        buttonGroup1.add(rbPill30);
        rbPill30.setFont(new java.awt.Font("Lucida Grande", 0, 24));
        rbPill30.setText("3");

        jButton1.setFont(new java.awt.Font("Lucida Grande", 1, 24));
        jButton1.setText("Back");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setFont(new java.awt.Font("Lucida Grande", 1, 24));
        jButton2.setText("Next");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        jLabel2.setFont(new java.awt.Font("Lucida Grande", 1, 28));
        jLabel2.setText("the first time?");

        org.jdesktop.layout.GroupLayout layout = new org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .addContainerGap()
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(layout.createSequentialGroup()
                        .add(jButton1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 98,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 620, Short.MAX_VALUE)
                        .add(jButton2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 94,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                    .add(layout.createSequentialGroup()
                        .add(jLabel1)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                        .add(jLabel2))
                    .add(rbPill15)
                    .add(rbPill5)
                    .add(rbPill10)
                    .add(rbPill30)
                    .add(rbPill25)
                    .add(rbPill20))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .addContainerGap()
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                    .add(jLabel1)
                    .add(jLabel2))
                .add(18, 18, 18)
                .add(rbPill5)
                .add(18, 18, 18)
                .add(rbPill10)
                .add(18, 18, 18)
                .add(rbPill15)
                .add(18, 18, 18)
                .add(rbPill20)
                .add(20, 20, 20)
                .add(rbPill25)
                .add(18, 18, 18)
                .add(rbPill30)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 138, Short.MAX_VALUE)
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                    .add(jButton1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 41,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(jButton2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 38,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .add(8, 8, 8))
        );

        java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-849)/2, (screenSize.height-577)/2, 849, 577);
      }// </editor-fold>

      private void rbPill5ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
}

      private void rbPill10ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
}

      private void rbPill25ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
}

      private void rbPill15ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
}

      private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

         double numberOfPills = 0.0;
         if (rbPill5.isSelected())
         {
            numberOfPills = 0.5;
         }
```

```
   else if (rbPill10.isSelected())
   {
      numberOfPills = 1.0;
   }
   else if (rbPill15.isSelected())
   {
      numberOfPills = 1.5;
   }
   else if (rbPill20.isSelected())
   {
      numberOfPills = 2.0;
   }
   else if (rbPill25.isSelected())
   {
      numberOfPills = 2.5;
   }
   else if (rbPill30.isSelected())
   {
       numberOfPills = 3.0;
   }

   switch (timesPerDayCnt)
   {
      case 0:
         pill.SetNumberOfPills1(numberOfPills);
         break;

      case 1:
         pill.SetNumberOfPills2(numberOfPills);
         break;

      case 2:
         pill.SetNumberOfPills3(numberOfPills);
         break;

      case 3:
         pill.SetNumberOfPills4(numberOfPills);
         break;
   }

    WindowManager.switchWindow(this, 7);
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
  switch (timesPerDayCnt)
    {
       case 0:
          pill.NumberOfPills1 = 0.0;

       case 1:
          pill.NumberOfPills2 = 0.0;

       case 2:
          pill.NumberOfPills3 = 0.0;

       case 3:
          pill.NumberOfPills4 = 0.0;
    }

    WindowManager.switchWindow(this, 5);
}
```

```
    private void formWindowGainedFocus(java.awt.event.WindowEvent evt) {
        // TODO add your handling code here:
          double numberOfPills = 0.5;
        String TimeNum = "first";
        switch (timesPerDayCnt)
        {
            case 0:
                numberOfPills = (pill.NumberOfPills1 == 0.0 ? numberOfPills : pill.NumberOfPills1);
                TimeNum = "first";
                break;

            case 1:
                numberOfPills = (pill.NumberOfPills2 == 0.0 ? numberOfPills : pill.NumberOfPills2);
                TimeNum = "second";
                break;

            case 2:
                numberOfPills = (pill.NumberOfPills3 == 0.0 ? numberOfPills : pill.NumberOfPills3);
                TimeNum = "thrid";
                break;

            case 3:
                numberOfPills = (pill.NumberOfPills4 == 0.0 ? numberOfPills : pill.NumberOfPills4);
                TimeNum = "forth";
                break;
        }

        if (numberOfPills == 0.5)
        {
            rbPill5.setSelected(true);
        }
        else if (numberOfPills == 1.0)
        {
            rbPill10.setSelected(true);
        }
        else if (numberOfPills == 1.5)
        {
            rbPill15.setSelected(true);
        }
        else if (numberOfPills == 2.0)
        {
            rbPill20.setSelected(true);
        }
        else if (numberOfPills == 2.5)
        {
            rbPill25.setSelected(true);
        }
        else if (numberOfPills == 3.0)
        {
            rbPill30.setSelected(true);
        }
jLabel2.setText("the " + TimeNum + " time?");
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new NumberOfPills().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.ButtonGroup buttonGroup2;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable jTable1;
    private javax.swing.JRadioButton rbPill10;
    private javax.swing.JRadioButton rbPill15;
    private javax.swing.JRadioButton rbPill20;
    private javax.swing.JRadioButton rbPill25;
    private javax.swing.JRadioButton rbPill30;
    private javax.swing.JRadioButton rbPill5;
    // End of variables declaration

}
```

**PillAddConfirm.Java**

```
package scheduler;


import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.awt.*;
import java.applet.*;
/**
 *
 * @author  huth
 */
public class PillAddConfirm extends Window {

    /** Creates new form PillAddConfirm */
    public PillAddConfirm() {
        initComponents();
    }

 public PillAddConfirm(Pill a_pill) {
        this.setWindowNum(7);
        pill = a_pill;
        initComponents();
    }
    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        Description = new javax.swing.JLabel();
        Frequency = new javax.swing.JLabel();
        TimesPerDayA = new javax.swing.JLabel();
        Time1 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel11 = new javax.swing.JLabel();
jLabel13 = new javax.swing.JLabel();
NumPill1 = new javax.swing.JLabel();
NumPill2 = new javax.swing.JLabel();
NumPill3 = new javax.swing.JLabel();
NumPill4 = new javax.swing.JLabel();
Frequency1 = new javax.swing.JLabel();
jButton2 = new javax.swing.JButton();
jButton1 = new javax.swing.JButton();
jLabel1 = new javax.swing.JLabel();
jLabel2 = new javax.swing.JLabel();
jLabel3 = new javax.swing.JLabel();
jLabel4 = new javax.swing.JLabel();
jLabel5 = new javax.swing.JLabel();
jLabel7 = new javax.swing.JLabel();
jLabel8 = new javax.swing.JLabel();
jLabel9 = new javax.swing.JLabel();
jLabel10 = new javax.swing.JLabel();
jLabel12 = new javax.swing.JLabel();
jLabel14 = new javax.swing.JLabel();
jLabel15 = new javax.swing.JLabel();
jLabel16 = new javax.swing.JLabel();
JLab3 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
addWindowFocusListener(new java.awt.event.WindowFocusListener() {
  public void windowGainedFocus(java.awt.event.WindowEvent evt) {
    formWindowGainedFocus(evt);
  }
  public void windowLostFocus(java.awt.event.WindowEvent evt) {
  }
});

Description.setFont(new java.awt.Font("Lucida Grande", 1, 24));
Description.setText("Description:");

Frequency.setFont(new java.awt.Font("Lucida Grande", 1, 24));
Frequency.setText("Frequency:");

TimesPerDayA.setFont(new java.awt.Font("Lucida Grande", 1, 24));
TimesPerDayA.setText("Times Per Day:");

Time1.setFont(new java.awt.Font("Lucida Grande", 1, 24));
Time1.setText("1) Time:");

jLabel6.setFont(new java.awt.Font("Lucida Grande", 1, 24));
jLabel6.setText("2) Time:");

jLabel11.setFont(new java.awt.Font("Lucida Grande", 1, 24));
jLabel11.setText("3) Time:");

jLabel13.setFont(new java.awt.Font("Lucida Grande", 1, 24));
jLabel13.setText("4) Time:");

NumPill1.setFont(new java.awt.Font("Lucida Grande", 1, 24));
NumPill1.setText("Number of Pills:");
NumPill1.setHorizontalTextPosition(javax.swing.SwingConstants.RIGHT);

NumPill2.setFont(new java.awt.Font("Lucida Grande", 1, 24));
NumPill2.setText("Number of Pills:");
NumPill2.setHorizontalTextPosition(javax.swing.SwingConstants.RIGHT);

NumPill3.setFont(new java.awt.Font("Lucida Grande", 1, 24));
```

```
NumPill3.setText("Number of Pills:");
NumPill3.setHorizontalTextPosition(javax.swing.SwingConstants.RIGHT);

NumPill4.setFont(new java.awt.Font("Lucida Grande", 1, 24));
NumPill4.setText("Number of Pills:");
NumPill4.setHorizontalTextPosition(javax.swing.SwingConstants.RIGHT);

Frequency1.setFont(new java.awt.Font("Lucida Grande", 1, 24));
Frequency1.setText("Total Pills Inserted: ");

jButton2.setFont(new java.awt.Font("Lucida Grande", 1, 24));
jButton2.setText("Back");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

jButton1.setFont(new java.awt.Font("Lucida Grande", 1, 24));
jButton1.setText("Done");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jLabel1.setFont(new java.awt.Font("Lucida Grande", 0, 24));

jLabel2.setFont(new java.awt.Font("Lucida Grande", 0, 24));

jLabel4.setFont(new java.awt.Font("Lucida Grande", 0, 24));

jLabel5.setFont(new java.awt.Font("Lucida Grande", 0, 24));

jLabel7.setFont(new java.awt.Font("Lucida Grande", 0, 24));

jLabel9.setFont(new java.awt.Font("Lucida Grande", 0, 24));

jLabel10.setFont(new java.awt.Font("Lucida Grande", 0, 24));

jLabel12.setFont(new java.awt.Font("Lucida Grande", 0, 24));

jLabel14.setFont(new java.awt.Font("Lucida Grande", 0, 24));

jLabel15.setFont(new java.awt.Font("Lucida Grande", 0, 24));

jLabel16.setFont(new java.awt.Font("Lucida Grande", 0, 24));

JLab3.setFont(new java.awt.Font("Lucida Grande", 0, 24));

org.jdesktop.layout.GroupLayout layout = new org.jdesktop.layout.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
    .add(layout.createSequentialGroup()
        .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .addContainerGap()
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(layout.createSequentialGroup()
                        .add(Description)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.UNRELATED)
                        .add(jLabel1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 112,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                      .add(layout.createSequentialGroup()
                        .add(Frequency)
                        .add(15, 15, 15)
                        .add(jLabel2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 130,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.UNRELATED)
                        .add(jLabel3))
                      .add(layout.createSequentialGroup()
                        .add(TimesPerDayA)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                        .add(JLab3, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 112,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
                    .add(408, 408, 408)
                    .add(jLabel8))
                  .add(layout.createSequentialGroup()
                    .add(35, 35, 35)
                    .add(Frequency1)
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                    .add(jLabel15, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 71,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
                .add(121, 121, 121))
              .add(layout.createSequentialGroup()
                .add(76, 76, 76)
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                  .add(layout.createSequentialGroup()
                    .add(Time1)
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                    .add(jLabel4, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 155, Short.MAX_VALUE))
                  .add(org.jdesktop.layout.GroupLayout.TRAILING, layout.createSequentialGroup()
                    .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
                      .add(org.jdesktop.layout.GroupLayout.LEADING, layout.createSequentialGroup()
                        .add(jLabel6)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                        .add(jLabel7, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 147, Short.MAX_VALUE))
                      .add(org.jdesktop.layout.GroupLayout.LEADING, layout.createSequentialGroup()
                        .add(jLabel11)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                        .add(jLabel9, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 147, Short.MAX_VALUE))
                      .add(org.jdesktop.layout.GroupLayout.LEADING, layout.createSequentialGroup()
                        .add(jLabel13)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                        .add(jLabel12, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 147, Short.MAX_VALUE)))
                    .add(8, 8, 8)))
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING, false)
                  .add(org.jdesktop.layout.GroupLayout.TRAILING, NumPill4,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                  .add(org.jdesktop.layout.GroupLayout.TRAILING, NumPill2,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                  .add(org.jdesktop.layout.GroupLayout.TRAILING, NumPill1,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                  .add(NumPill3))
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
                  .add(jLabel14, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 160, Short.MAX_VALUE)
                  .add(org.jdesktop.layout.GroupLayout.LEADING, jLabel10,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 160, Short.MAX_VALUE)
```

```
                .add(jLabel16, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 160, Short.MAX_VALUE)
                .add(org.jdesktop.layout.GroupLayout.LEADING, jLabel5,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 160, Short.MAX_VALUE))
            .add(140, 140, 140))
        .add(layout.createSequentialGroup()
            .addContainerGap()
            .add(jButton2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 85,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 567, Short.MAX_VALUE)
            .add(jButton1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 123,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .add(62, 62, 62))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
        .add(layout.createSequentialGroup()
            .addContainerGap()
            .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                .add(Description)
                .add(jLabel1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 32,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
            .add(18, 18, 18)
            .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                .add(layout.createSequentialGroup()
                    .add(8, 8, 8)
                    .add(jLabel3, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 16,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .add(layout.createSequentialGroup()
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.UNRELATED)
                    .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                        .add(Frequency)
                        .add(jLabel2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 27,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))))
            .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                .add(layout.createSequentialGroup()
                    .add(34, 34, 34)
                    .add(jLabel8, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 16,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .add(layout.createSequentialGroup()
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.UNRELATED)
                    .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                        .add(TimesPerDayA)
                        .add(JLab3, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 26,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))))
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
            .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                .add(Time1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 16,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(NumPill1)
                .add(jLabel4, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 31,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(jLabel5, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 28,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
            .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                .add(jLabel6)
                .add(NumPill2)
                .add(jLabel7, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 23,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(jLabel16, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 29,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
            .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
              .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                .add(jLabel11)
                .add(NumPill3)
                .add(jLabel9, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 32,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
              .add(jLabel10, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 26,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
            .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
              .add(jLabel13)
              .add(NumPill4)
              .add(jLabel12, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 29,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
              .add(jLabel14, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 25,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 101, Short.MAX_VALUE)
            .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
              .add(org.jdesktop.layout.GroupLayout.TRAILING, layout.createSequentialGroup()
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                  .add(Frequency1)
                  .add(jLabel15, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 33,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .add(102, 102, 102))
              .add(org.jdesktop.layout.GroupLayout.TRAILING, layout.createSequentialGroup()
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                  .add(jButton2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 43,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                  .add(jButton1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 41,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .addContainerGap())))
    );

    java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
    setBounds((screenSize.width-854)/2, (screenSize.height-563)/2, 854, 563);
  }// </editor-fold>

  private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    pill.setTotalPillsInserted(0);
    WindowManager.switchWindow(this, 6);
  }

  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
      XmlDAO xmlDAO = new XmlDAO("Pill.xml");
      ArrayList<Pill> pills = xmlDAO.read();
      pill.id = pills.size() + 1;
      pills.add(pill);
      xmlDAO.write(pills);

      BspDAO bspDAO = new BspDAO("Pill.bsp");
      bspDAO.write(pills);

      pill = null;

    } catch (FileNotFoundException ex) {
      Logger.getLogger(PillAddConfirmation.class.getName()).log(Level.SEVERE, null, ex);
    }

    WindowManager.switchWindow(this, 1);
  }
```

```
private void formWindowGainedFocus(java.awt.event.WindowEvent evt) {
    jLabel1.setText(pill.getDesc());
    jLabel2.setText(pill.GetDailyOrWeekly());
    int timeperdayA = pill.GetTimesPerDay();
    JLab3.setText(String.valueOf(timeperdayA));
    jLabel4.setText(pill.GetTimeOfDay1().equals("") ? "N/A" : pill.GetTimeOfDay1());
    double numPill1 = pill.GetNumberOfPills1();
    jLabel5.setText(numPill1 == 0 ? "N/A" : String.valueOf(numPill1));
    jLabel7.setText(pill.GetTimeOfDay2().equals("") ? "N/A" : pill.GetTimeOfDay2());
    double numPill2 = pill.GetNumberOfPills2();
    jLabel16.setText((numPill2 == 0 ? "N/A" : String.valueOf(numPill2)));
    jLabel9.setText(pill.GetTimeOfDay3().equals("") ? "N/A" : pill.GetTimeOfDay3());
    double numPill3 = pill.GetNumberOfPills3();
    jLabel10.setText(numPill3 == 0 ? "N/A" : String.valueOf(numPill3));
    jLabel12.setText((pill.GetTimeOfDay4().equals("") ? "N/A" : pill.GetTimeOfDay4()));
    double numPill4 = pill.GetNumberOfPills4();
    jLabel14.setText(numPill4 == 0 ? "N/A" : String.valueOf(numPill4));
    double Pillsinserted = pill.getQuantity()*(numPill1+numPill2);
    pill.setTotalPillsInserted(Pillsinserted);
    jLabel15.setText(String.valueOf(Pillsinserted));
    // TODO add your handling code here:
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new PillAddConfirm().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JLabel Description;
private javax.swing.JLabel Frequency;
private javax.swing.JLabel Frequency1;
private javax.swing.JLabel JLab3;
private javax.swing.JLabel NumPill1;
private javax.swing.JLabel NumPill2;
private javax.swing.JLabel NumPill3;
private javax.swing.JLabel NumPill4;
private javax.swing.JLabel Time1;
private javax.swing.JLabel TimesPerDayA;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel13;
private javax.swing.JLabel jLabel14;
private javax.swing.JLabel jLabel15;
private javax.swing.JLabel jLabel16;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
```