Software Engineering Course Project

Virtual Spectrophotometer

Project designed by Ivan Marsic

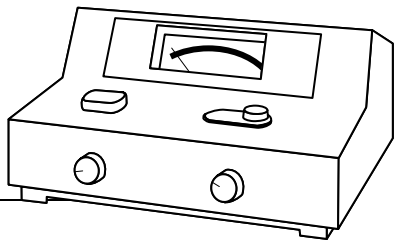
Department of Electrical and Computer Engineering

Rutgers University

Project website: http://www.ece.rutgers.edu/~marsic/books/SE/projects/

Product Vision: *This project will develop a virtual spectrophotometer instrument that will allow biology students to learn how to measure concentration of various substances and will facilitate student grading for the course instructor.*

This project is intended to be done by a team of 4-6 undergraduate students during an academic semester, in conjunction with lectures and other class activities. Other related projects and a software engineering textbook are available for download at this website:
http://www.ece.rutgers.edu/~marsic/books/SE/



1 Project Description

The purpose of this project is to develop a virtual spectrophotometer instrument using which students will learn to operate an actual spectrophotometer. This virtual lab simulates lab exercises that are performed by students who are taking general or upper level biology courses. The spectrophotometer is a machine used to measure the concentration of a substance in a solution by passing light of a specified wavelength through it. A photocell at one end receives the transmitted light and an analog meter displays the percent transmittance of light received by the photocell. Many substances absorb light and transmit light of specific wavelengths within the ultraviolet

(200 – 400 nm), visible (400 – 700 nm), and near-infrared (700 – 1000 nm) regions of the electromagnetic spectrum. According to Beer's law, the amount of light absorbed by a medium is proportional to the concentration of the absorbing material or solute present. In order to use this instrument, the correct wavelength of light must be chosen so that the light is absorbed by the substance contained in a test tube when it tries to pass through the solution (the more substance there is, the less light will pass through and vice versa). The main purpose of the lab is to familiarize students with the spectrophotometer and its use. The students first calibrate the instrument using a reference solution of known concentration. The concentration of unknown samples is calculated according to the transmittance value of the reference solution.

In a typical measurement of the absorbance of light, a beam of light is passed through a transparent tube containing the absorbing material. The intensity I of light reaching a detector is compared to the intensity Io reaching the detector in the absence of the absorbing material. The relationship between the absorbance A, the concentration c of the absorbing substance, and the path length ℓ of the light through the substance is given by the Beer-Lambert law:

$$A = e\,\ell\,c$$

where e, called the molar absorption, is a fundamental property of the absorbing material. The key point is that the absorbance (defined as log [Io/I]) is proportional to concentration.

In reflecting on the usefulness of the spectrophotometer simulation, the students enjoyed the individual and repetitive practice they could engage in. They envisioned that this experience would help them with their lab practical exams as well as fine-tune their skills in operating the spectrophotometer in the actual lab. Overall, the students found the virtual labs to be an interface where they could learn and practice in spite of making errors. They also acknowledged that the simulations were realistic and helped in demonstrating certain processes that were not easily represented in the actual lab.

1.1 Customer Statement of Work

1.2 Spectrophotometer Description

Figure 1 shows the spectrophotometer instrument and its parts.

1. Power switch to turn the instrument ON or OFF
2. Test tubes (or, "cuvettes"), contained in a rack; each cuvette contains a different solution
3. Sample holder (or, "chamber") for holding the test tube to measure the concentration of its substance
4. Dark dial for calibrating the zero illumination
5. Light dial

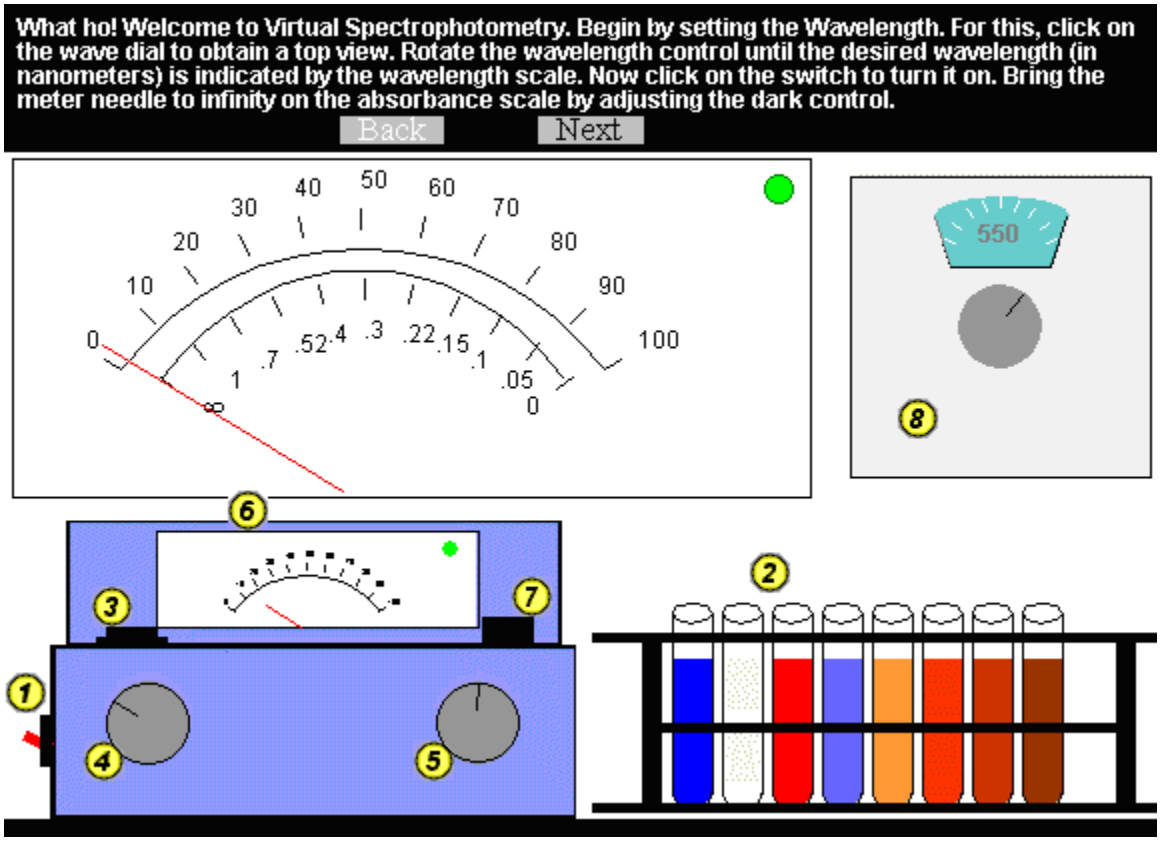


Figure 1: Screen snapshot of the spectrophotometry virtual laboratory. ① on/off switch, ② rack with test tubes with solutions, ③ sample holder for the test tube, ④ zero control dial, ⑤ light control dial, ⑥ meter with needle and pilot light (magnified view shown on top), ⑦ wavelength control dial for setting the color of the illumination light, ⑧ magnified view of the wavelength dial.

6. Absorbance meter with two scales (transmittance and absorbance), a needle, and a pilot lamp to indicate when the instrument is powered. We assume an *analog* spectrophotometer with a needle pointer instead of a digital/numeric readout.
7. Wavelength dial for setting the color of the illumination light

There are some other parts of a physical spectrophotometer that will not be simulated in the virtual instrument, because their operation will not be explicitly visible to the user:

- A *light source* shines the selected wavelength light from the spectrophotometer through the sample
- A focusing device that transmits an intense straight beam of light
- A monochromator to separate the beam of light into its component wavelengths
- A photoelectric detector

If in the future our customer will wish to teach the students how the physical process of spectrophotometry works, we could simulate these parts and the mechanism separately.

1.3 System Requirements

Based on the product vision give at the beginning of this document, we derive the following two functional requirements:

REQ1: The system shall allow measuring the concentration of an unknown solution.

REQ2: The system shall facilitate student grading for the course instructor.

The system requirements can be reformulated as *user stories*, if the developer prefers the agile development terminology:

UST1: As a user, I will be able to measure the concentration of an unknown solution.

UST2: As a course instructor, I will be able faster to grade the student performance. Initially, the system will collect and summarize the information about student's lab performance. In the future, we may consider completely automated grading.

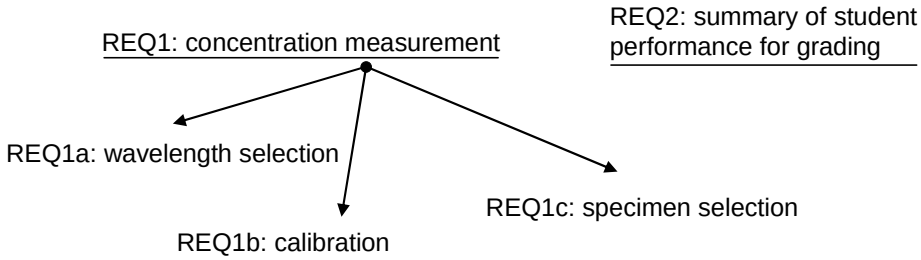
After learning more about the problem domain and the functioning of a physical spectrophotometer, we realize that we need to elaborate the above two high-level requirements with more detailed ones:

REQ1a: The system should allow the user to select the wavelength of the transmitted light.

REQ1b: The system should allow the user to calibrate the instrument using a "blank" solution.

REQ1c: The system should have a rack with several different specimens (solutions) and allow the user to insert the specimen for which the concentration will be measured.

The hierarchical structure of the system functional requirements can be summarized as:



In addition, the developer will need to find if there are any non-functional requirements, and if the customer has any preferences about the on-screen appearance.

1.4 User Interaction

There are *five* interactive parts of a real spectrophotometer:

1. Power switch can be toggled between two positions
2. Lid of the sample holder or chamber can be opened or closed
   (when the lid is opened, a test tube can be inserted into or removed from the chamber)
3. Three dial knobs can be rotated between a minimum and maximum angle:
   a. zero control dial
   b. light control dial
   c. wavelength control dial

In real world, the user may interact in any order with these parts and so the virtual spectrophotometer should allow an arbitrary order of interaction. However, most of random interactions will produce meaningless results. Only certain interaction sequences will produce desired results, as described next.

This section defines a successful usage scenario (the so-called “standard operating procedure”) that should be supported by the virtual spectrophotometer. This usage scenario will be part of a “use case” for the virtual spectrophotometer.

The steps for operating the spectrophotometer are as follows:

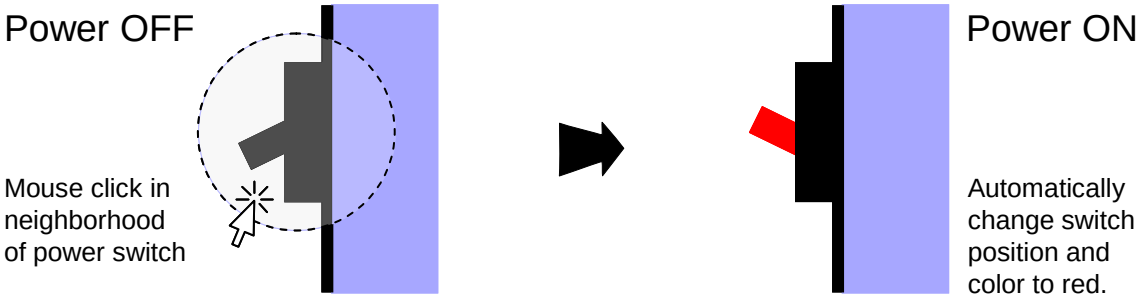
1. Turn on the power (Section 1.4.1)
2. Adjust the zero (or “dark”) control (Section 1.4.2)
3. Select wavelength (Section 1.4.31.4.5
4. Blank adjust (Section 1.4.4)
5. Read specimen (Section 1.4.5)

An important feature in our labs is their “non-linear” nature. When students perform these virtual labs, they are likely to make errors that they might notice only in the subsequent steps (see more discussion in Section 1.5). Thus, in order to give an opportunity to the user to correct the mistakes, we provide a Back button. If the Back button is not implemented, students who made an error would be required to start the process all over again, which can be frustrating.

The above steps are described in detail next.

1.4.1 Operating the Power Switch

Initially, the instrument is turned off and the power switch (indicated with ① in Figure 1), is shown in the OFF configuration, as in the figure below on the left side. Note that the pilot light, which is part of the meter scale ⑥, will be shown in white color (i.e., it will be invisible). When the user clicks the mouse within a given neighborhood around the switch, the switch will automatically change the configuration to the ON position and its handle will change the color to red. At the same time, the pilot light will glow in green color, as shown in Figure 1.



The power switch operates in a “toggle” mode, so that when clicked again, it will go into the “off” position and the instrument will turn off.

One issue to consider is that a real physical spectrophotometer requires at least 10-30 minutes warm-up time (depends on the model?). If we ignore this fact and just make the instrument operational immediately, the student will not learn an important aspect of spectrophotometer operation. On the other hand, it is annoying for the user to wait idle for 10 minutes needlessly, given that the virtual spectrophotometer can operate immediately. A compromise solution may be

to generate an animation that counts down time quickly, much faster than real time, so that 10 seconds countdown simulates a 10 minutes warm-up time.

What if the user turns off the instrument while a test tube is still inside the sample chamber? Alternative solutions include:

- Silently reset the "current state" of the instrument (including returning the test tube to the rack with test tubes with solutions, marked with ② in Figure 1) and turn the instrument off; next time the user powers on the instrument, start from the initial state.
- Silently store the state, so the next time the user power up the instrument, the test-tube in the sample holder will be restored, and the user can continue from that point.
- Issue a warning and not allow powering down the instrument until the test tube is removed from the sample holder and returned to the rack with test tubes with solutions.

Given that our main goal is to simulate the real-world spectrophotometer and in real world it is possible to power down the machine while a tube is still in the chamber, we decide that that the second option best simulates the real world.

Another issue is whether to allow the user to open the chamber lid and insert a tube while the machine is powered down. Again, our guide is the real world, where such scenario is possible. Therefore, we will allow the user to insert a cuvette in the chamber even if the power switch is OFF. Of course, while the power switch is OFF the measurement scale will show nothing (the needle will remain motionless) regardless of user interactions.

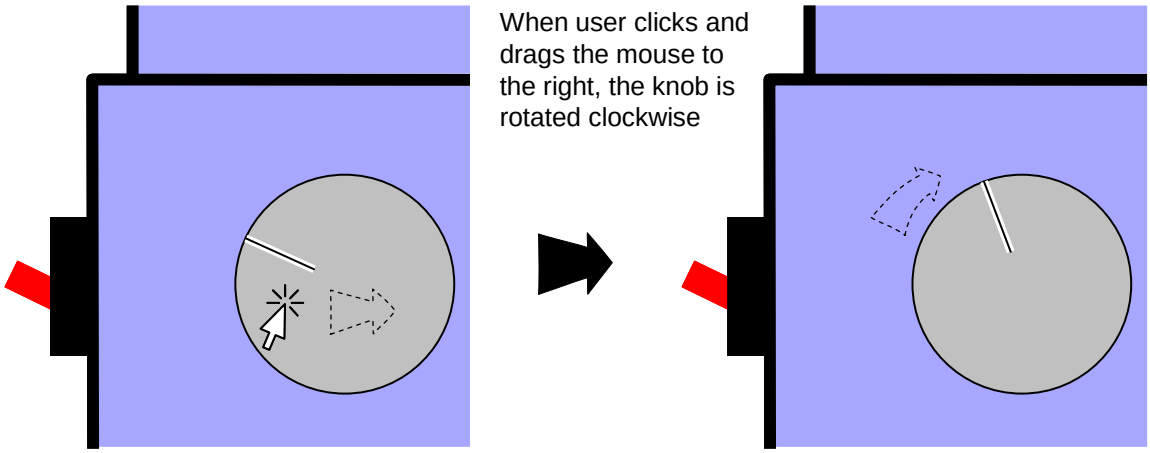
Generally, the developers should select the option they feel is the best in the context of the given problem and state their arguments for selecting that solution.

1.4.2 Spectrophotometer Calibration: Adjusting the Zero Control Dial

With no cuvette in the chamber, a shutter cuts off all light from passing though the cuvette chamber. Under this condition therefore, the machine may be adjusted to read infinite absorbance (zero % transmittance) by rotating zero adjust knob (front left dial marked with ④ in Figure 1). *This knob must not be touched during the rest of the measurement procedure, to keep the instrument properly calibrated*.

Zero control dial (also called "dark control") is used to set the meter reading to an infinite absorbance on the left side of the meter scale. It is used with a very dark specimen where no light passes through, or no specimen at all in the chamber.

The zero control dial (indicated with ④ in Figure 1) is displayed as a circle, which represents the knob, and a line, which represents the reference mark. A dial knob can be rotated to set the desired value. If the user clicked within the dial circle area and dragged the mouse to the right, then the dial knob should be rotated in the clockwise direction. Similarly, if the user dragged the mouse to the left, then the dial knob should be rotated in the counterclockwise direction. A more sophisticated solution of detection the mouse rotational movement is left to the developer to specify and implement, if desired.



When the dial is rotated, this causes the spectrophotometer to redo the measurement. The "measuring" behavior performs the calculations based on the density of the solution contained in the test tube and the wavelength of the shone light. As a result, the Absorbance Meter's needle will display the measure wavelength.

Note that there is no notion of "calibration state", so the instrument does *not* "know" when the user properly completed the calibration. Even if the user has not adjusted the instrument to read infinite absorbance and continues on the next step, the spectrophotometer is not aware that this is "wrong." For the same reason, the instrument will not give any indication that the user correctly adjusted it to read infinite absorbance, except that the needle will point to the infinite value on the left side of the scale (Figure 1).

1.4.3 Wavelength Selection

In order to use this instrument, the correct wavelength of light source must be chosen so that the light being shone is absorbed by the substance contained in a test tube when it tries to pass through the solution.

Rotate the wavelength control (indicated with ⑦ in Figure 1) until the desired wavelength (in nanometers) is indicated by the wavelength scale (magnified view ⑧ in Figure 1).

1.4.4 Blank Adjustment

Blank cuvette = The sample used to calibrate the zero end of the absorbance scale.

1. Fill the B (blank) cuvette with the solvent used to dissolve specimen (often distilled water). Polish to clean, insert into the cuvette chamber, aligning mark to front. Close chamber cover.

2. Rotate blank adjust (or, "light control") knob to adjust absorbance to read zero. This is the front right knob, marked by ⑤ in Figure 1.

3. Remove blank cuvette, place in plastic test tube rack.

1.4.5 Measuring a Specimen Concentration

The concentration of an unknown solution can be determined by comparing the absorbance reading of the unknown to the standard curve.

Transmittance = The fraction of light in the original beam that passes through the sample and reaches the detector.

Read each sample (standard and unknowns) directly off the absorbance scale.

Remember that absorbance is proportional to concentration. If the sample does not contain many molecules that absorb light, there will be little color and the measured absorbance will be low. On the other hand, if the sample contains more molecules that absorb light, there will be more color and the measured absorbance will be higher.

1.4.6 Other Interactions

Additional interactions with the spectrophotometer include manipulating the lid of the test-tube chamber (or, sample holder). In real world, any time the sample holder's lid is opened or closed, the needle of the meter will move to show different values. Similarly, the virtual spectrophotometer should simulate this "measuring" behavior and redo the measurement and show the new value.

1.5 Pedagogical Aspects of the Virtual Laboratory

If this system is implemented so that the user is strictly guided through the process, step-by-step without a chance of making mistakes, then the system may not have much pedagogical value and may be boring to use. Because this is an instructional system, it should allow the user to demonstrate the mastery of the course material. Therefore, we should allow the student to make mistakes during the instrument operation. The user should be allowed to follow "blind alleys" until he or she realizes that they made a mistake and need to backtrack to an earlier point and continue the lab execution in the right direction. For example, after the spectrophotometer is calibrated in step 2 of operation (Section 1.4.2), we stated that this knob must not be touched during the rest of the measurement procedure. If the student by mistake touched this dial, we have two options:

(a) Pop up a dialog box to issue a warning that the dark dial should not be touched during the rest of the measurement procedure
(b) Silently allow the student continue with the rest of the measurement procedure, which will produce wrong results given that now the instrument is not calibrated

Option (b) may be better, because the actual spectrophotometer does not issue warnings, and the student would learn to rely on something that does not exist in real world. Of course, if the system usage is made too difficult, then the user may lose interest in using it. It is important to strike the right balance between a rigid linear process and a complex web of pathways and "dead alleys."

For example, we mentioned in Section 1.4 that we provide a Back button. Although this feature does not exist on real-world spectrophotometers, by their nature they allow more flexibility in operation, so one may argues that a Back button would not have a negative pedagogical impact.

One option is to ask the user initially to set the "difficulty level" before they start the laboratory exercise. Another option is to keep track of user's progress and if the system somehow can detect that the user is not making progress and is becoming frustrated, then offer help and guide the user out of the "blind alley" where the user is currently stuck in.

For example, in Section 1.4.2, we mentioned that the real-world instrument does *not* have notion of "calibration state" and it does *not* "know" whether or not the user properly completed the calibration. With the *virtual* spectrophotometer we can implement different levels of difficulty, by introducing the notion of "calibration state." Then, at the "beginner's level," the system will issue warnings if the user tried to proceed with the subsequent steps before the instrument is correctly adjusted it to read infinite absorbance.

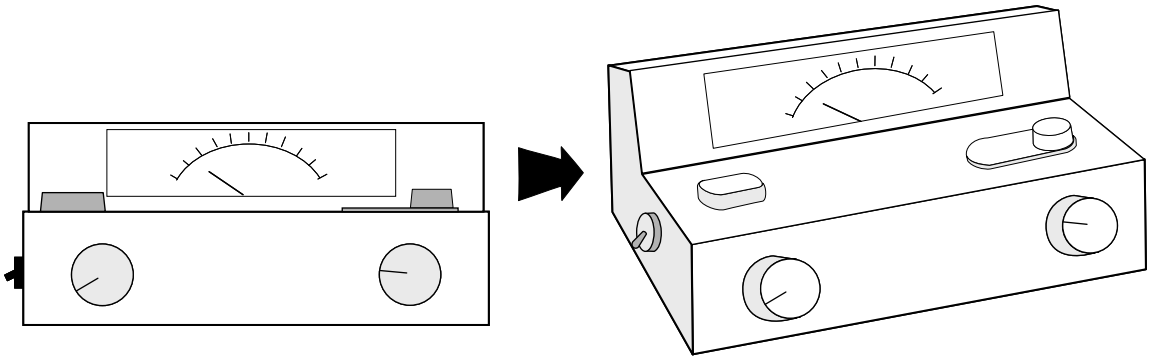
1.5.1 User Interaction Assumptions and Intuitions

Here are some assumptions that need to be tested to design a user-friendly and intuitive user interface:

1. Students will know where the on/off switch is located (labeled as ① in Figure 1).
   [Note: *To simplify the user interface design, the switch is located in a different place in Figure 1 than in the actual instrument*.]
2. Students will know which dial is the wavelength dial (labeled as ⑦ in Figure 1).
   [Note: *As there are three dials, there is a possibility of confusion, crying out for tool tips to help the user distinguish the dials*.]
3. Students will know which tube in the rack (labeled as ② in Figure 1) contains the transparent solution.
   [Note: *It helps if all the other test tubes have rather distinct colors*.]
4. Students will know that they have to set the wavelength (using the wavelength dial, labeled as ⑦ in Figure 1) before calibration.
5. Students will know that the top view of the spectrophotometer exists (by default, only the front view is shown, and the top view is available upon request).
6. Students will know how to obtain the top view of the spectrophotometer.
7. Students will know how to make the top view disappear.
   [Note: *Students may not bother trying to make it disappear if it did not cause hindrance*.]
8. Students will know that they have switched ON the spectrophotometer (by observing the green oval in the measurement scale, known as the "pilot light").
9. Students will know that they cannot use the spectrophotometer when they switch it OFF.
   [Note: *We observed that most students did try to operate the spectrophotometer without switching it ON; as they saw that nothing was happening, they read the instructions to see that it has to be turned ON first*.]
10. Students will differentiate between the transmittance and absorbance scales—see the absorbance meter, labeled as ⑥ in Figure 1, with a magnified view shown on top)—notice that there are two scales with different series of numbers.
    [Note: *Students might not remember which scale they used—top or bottom—and report an incorrect result*.]
11. Students will recognize the correct reading on the correct scale.
    [Note: *As one scale is a logarithmic scale, it is the only one with an "infinity" reading; so this should be intuitive*.]
12. Students will remember the adverse effect of sample holder's lid not being closed.
    [Note: *Though they may know that the reading is wrong, they may not be not sure how different it will be (higher or lower) with a correct reading*.]

13. Students will notice the Back button getting automatically activated when backtracking becomes possible.
14. Students will click on the Back button when they are instructed to by the text instructions.

The developer should test these assumptions or naive users and check if the assumptions hold. If they do not hold, the developer will have to develop user interface aids to help the user recognize the available functionality and use it properly. One option for tackling the issues #5 and #6 is to visualize the spectrophotometer in a side view (instead of the front view shown in Figure 1), such as:



However, although this solution increases the user's awareness of the top panel, it may still be necessary to bring up a separate top view (similar to the magnified view ⑧ in Figure 1) when the user wishes to adjust the wavelength dial, because it may be difficult to visualize and view the dial movement and the wavelength scale in the side view.

The developer may need to provide a tool tip that tells the students where to click on the dial and rotate it.

Ideally, the developer should verify the student's readings with the actual readings given by the spectrophotometer to ensure a more meaningful test of correct operation and use of the virtual spectrophotometer.

1.6 Student Grading

An important aspect of a virtual biology laboratory is grading the student performance, stated as REQ2 in Section 1.3. Ideally, the system should grade the student performance automatically.

In absence of an ideal (automated) solution, the developer needs to solve the problem of what kind of trace should be captured to facilitate the instructor's grading. Some issues to consider include:

- Each student should be identified ($\Rightarrow$ System Requirement: Login), so that his or her performance can be recorded and tracked.
- Student's performance should be tracked over multiple sessions and progress (or lack thereof) should be detected and reported to the instructor.
- During every run, each student's action should be recorded to capture a trace for performance grading purposes.
- The session traces should be summarized instead of presenting them unprocessed to the instructor, because step-by-step replaying would be very time-consuming and laborious.

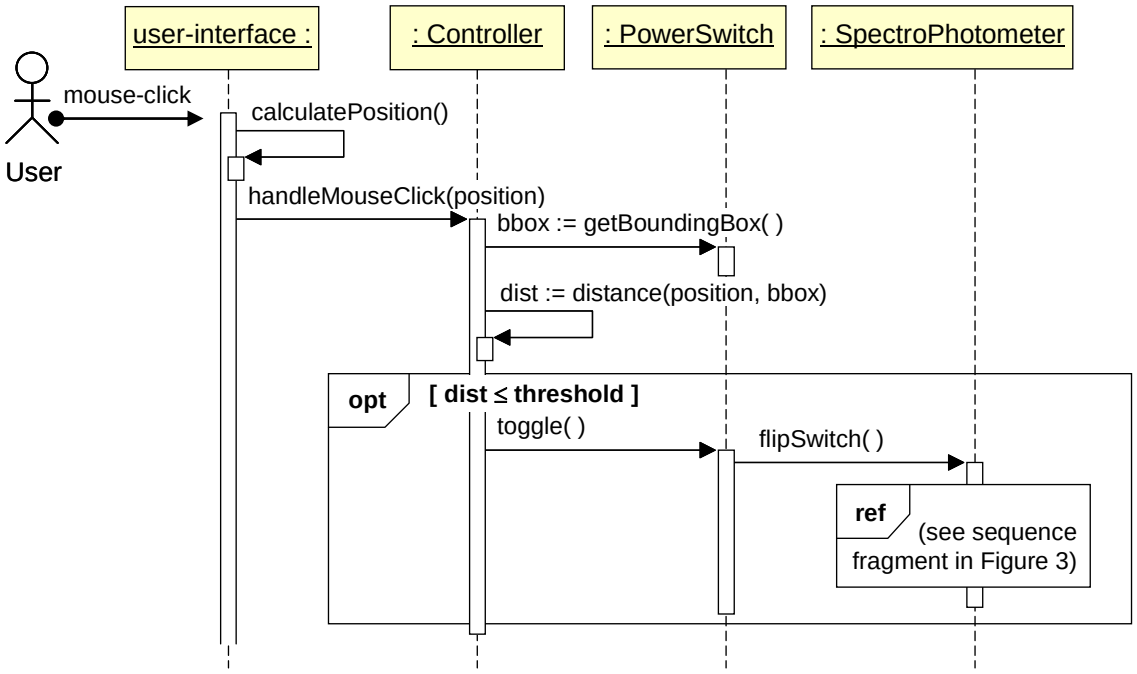


Figure 2: Interaction diagram for operating the spectrophotometer power switch. Note that this diagram refers to a fragment shown in Figure 3.

The developer should consider carefully the above issues and formulate smart solutions.

Some very sophisticated mathematical models can be constructed to summarize the student performance during the lab exercise or across successive sessions. We need to decide on what basis is the grading done, what parameters are required and how the grade will be calculated. For example, one parameter to consider in grading is the "difficulty level" discussed in Section 1.5. We need to have a general idea of where and what the instructor can specify as the criteria for performance summarization and grading. Even the user/student will need to know this information before he starts running the lab. Otherwise, the student will not understand what criteria will be used for grading his performance.

2 Spectrophotometer Interaction Diagrams

This section follows the structure of Section 1.4 that describes the types of user interaction with the spectrophotometer.

2.1 Operating the Power Switch

This section describes the class interactions that implement the behavior described in Section 1.4.1. The interaction with the system starts by user clicking the mouse in the neighborhood of the power switch. The system detects a mouse click, measures its distance from the center of the power switch, and if it is less than a given threshold, it activates the switch. When the switch

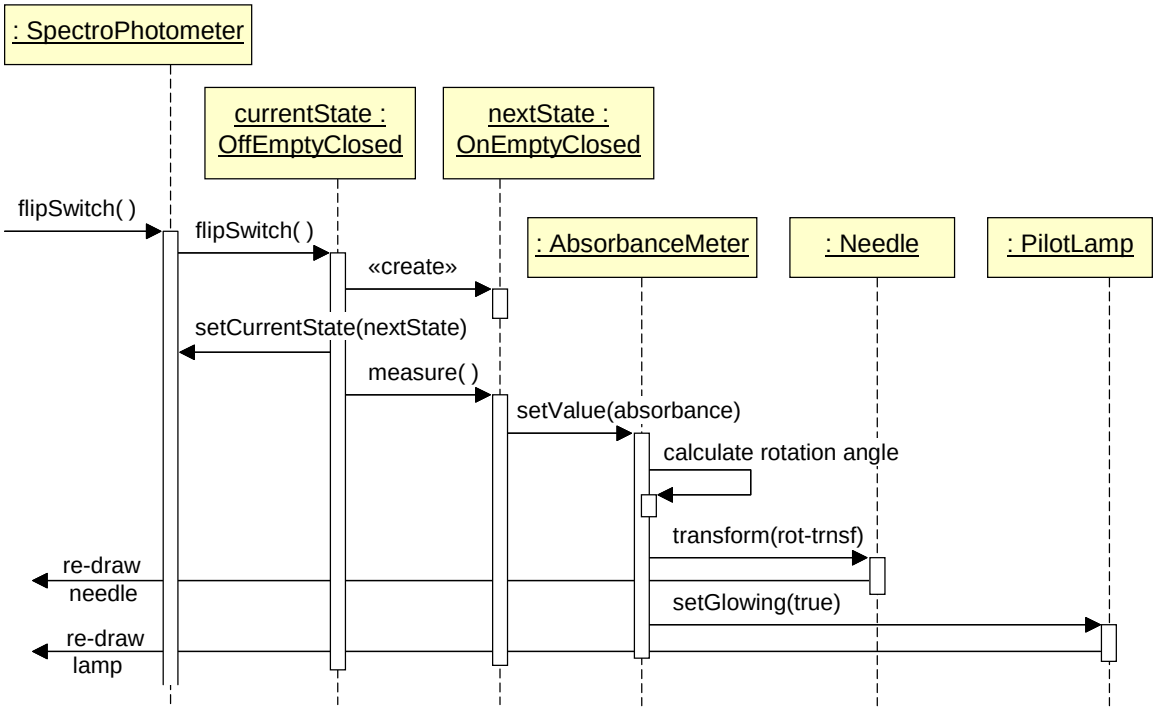


Figure 3: Interaction sequence fragment for the event `flipSwitch()`. (This diagram is referred to from Figure 3.)

object is activated, it in turn activates the pilot lamp to indicate that the instrument is "on", and enables interaction with other parts of the spectrophotometer.

Figure 2 shows the UML sequence diagram that represents the above steps for operating the power switch. The process starts with the user clicking the mouse. The "user interface" component calculates the mouse position within the laboratory window. The implementation of this component depends on the programming language used for developing the virtual spectrophotometer. For example, if Java is used, then in the Swing library we will need to listen for "mouse clicked" events within the lab window. Next, the Controller checks whether the click position is within a threshold distance from the switch center (determined as the center of the switch's bounding box). If yes, Power Switch object is asked to toggle its state. If its state was previously OFF, now it will become ON. As a result, the pilot lamp should be activated to glow green.

Note that the diagram in Figure 2 refers to a fragment shown in Figure 3. We consider the operation of the power switch as an event and define different states of the system. In Figure 3 we assume that the spectrophotometer's initial state is OffEmptyClosed (described in Section 4.1.3). More details are provided in Section 4.

2.2 Adjusting the Zero Control Dial

This section describes the class interactions that implement the behavior described in Section 1.4.2. When the zero dial is rotated, this causes the spectrophotometer to redo the measurement. The "measuring" behavior performs the calculations based on the density of the solution

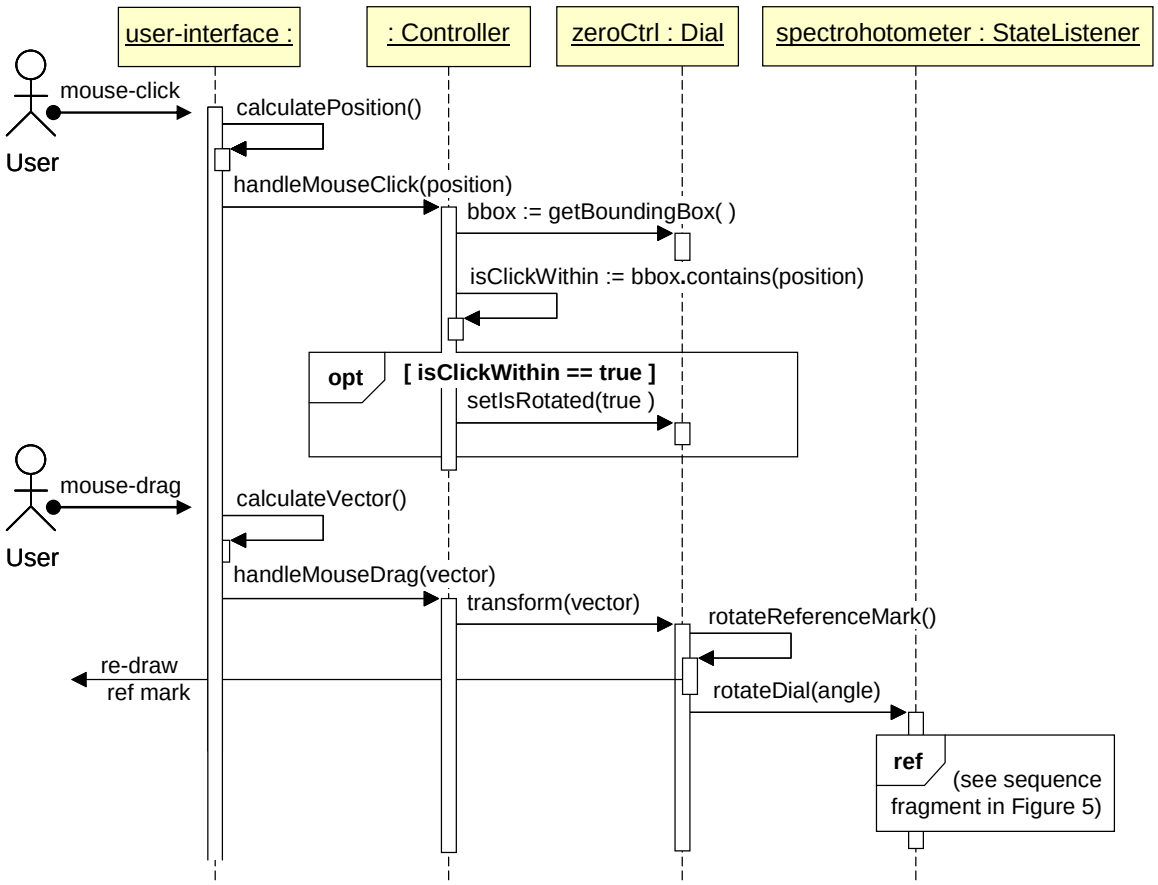


Figure 4: Interaction diagram for adjusting the zero control dial. Note that this diagram refers to a fragment shown in Figure 5.

contained in the test tube and the light wavelength and sends a command to the AbsorbanceMeter needle to display the wavelength.

Figure 4 shows the sequence diagram of object interactions. The process starts with the user clicking in the dial area and dragging the mouse cursor in the desired direction. The system detects if the click is within the dial's area and if so, sets the dial attribute `isRotated` to `true`. This is necessary so that the dial object knows that the subsequent mouse-drag events should be properly processed.

When a mouse-drag event arrives, the dial object reacts by notifying all of its listener objects that it has been transformed, i.e., rotated. When the mouse is dragged, we calculate the vector by which the mouse cursor changed its location (see method `calculateVector()` in Figure 4. Next, the Controller calculates the rotation angle from this vector and calls a rotation transformation on the Dial object. The dial first rotates the reference mark line to show its new rotation angle. Next, it notifies other objects ("listeners") to perform the measurement behavior.

The listener objects will be contained in the `stateListeners` attribute. The zero-dial listeners include (see Figure 4):

- Object that performs the measurement calculations, based on the density of the solution contained in the test tube and the light wavelength. We assume that the SpectroPhotometer object will perform this calculation and therefore its method

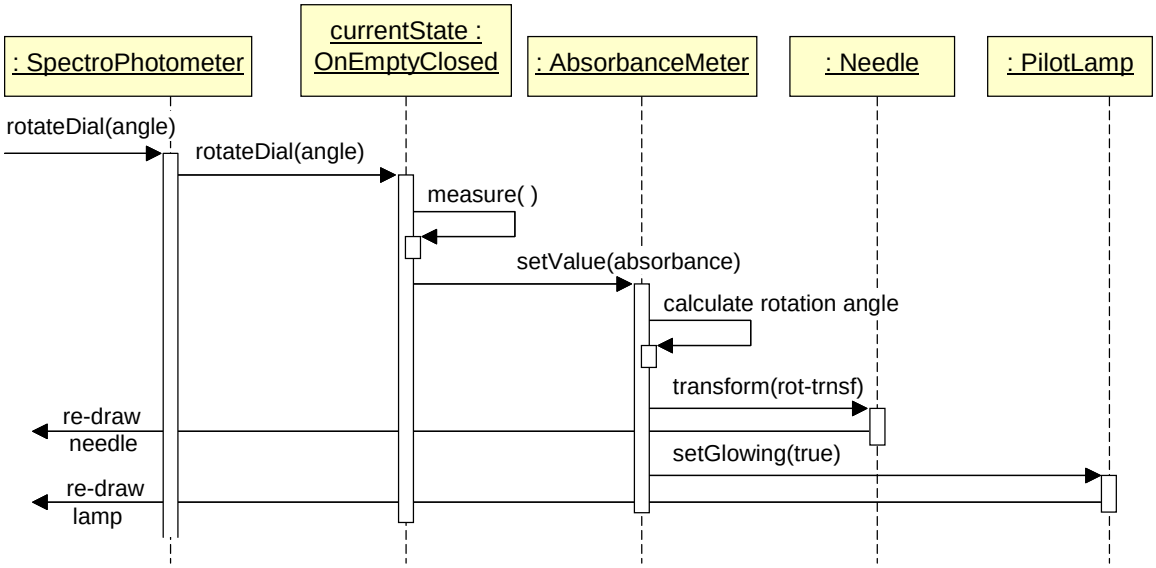


Figure 5: Interaction sequence fragment for the event `rotateDial()`. (This diagram is referred to from Figure 4.)

> `measure()` is called with the parameter "angle" which indicates by how much the dial is rotated.

Note that the diagram in Figure 4 refers to a fragment shown in Figure 5. Based on the calculated value of absorbance, the SpectroPhotometer object calculates by how much AbsorbanceMeter should rotate its needle to display the measured wavelength and calls the `transform()` method with rotation transformation parameter.

Note that there is a design issue with this distribution of responsibilities. On one hand, we want to keep object communications simple, so they just call the `transform()` method on each other. On the other hand, the SpectroPhotometer object is assigned the responsibility to calculate by how much AbsorbanceMeter's needle should rotate. A better assignment of responsibilities would be that the absorbance meter calculates the needle's rotation angle. However, but in this case the calculated measured wavelength needs to be communicated to the AbsorbanceMeter using a different method—a new method needs to be introduced on the AbsorbanceMeter.

3 Spectrophotometer Software Classes

The class diagram is shown in Figure 6. This section describes the following Java classes:

AbsorbanceMeter.java	Dial.java
LabBackground.java	PilotLamp.java
Needle.java	PowerSwitch.java
SampleHolder.java	SpectroPhotometer.java
TestTube.java	WaveDial.java

3.1.1 LabObject.java

An abstract base class for all parts of the spectrophotometer (Figure 6). This is a graphical figure base class that can be manipulated by mouse pointer. The manipulations include click and drag. A dragging manipulation is converted into an affine transformation that will be applied to the corresponding figure.

Class Name:	LabObject	
Attributes:	# boundingBox : Rectangle2D	Represents the bounding box (rectangle) of the lab object's graphical representation.
	# transform : AffineTransform	Represents the current affine transformation of the object, relative to its non-transformed prototype.
	# stateListeners : HashSet	Listener objects interested in state changes of this lab object.
Operations:	+ transform(at : AffineTransform)	Applies an affine transformation to this object.
	+ addStateListener(sl : StateListener)	Adds the listener `sl` to the list of listeners.
	+ removeStateListener(sl : StateListener)	Removes the listener `sl`.
	# fireMeasureEvent( )	Calls the method `measure()` on all listener objects.

The bounding box is a `java.awt.geom.Rectangle2D` object type. The bounding box of all objects is always maintained of constant size, i.e., they are in a non-transformed prototype, which means zero translation, zero rotation, and scale is 1 for both *x* and *y* dimension (i.e., resized to 100%). Any size changes should be recorded in the `transform` attribute as a size scaling transformation (resizing to a smaller or larger size). To change the object's translation, rotation, or scale (resizing), just call the method `transform()` with the appropriate transformation parameter.

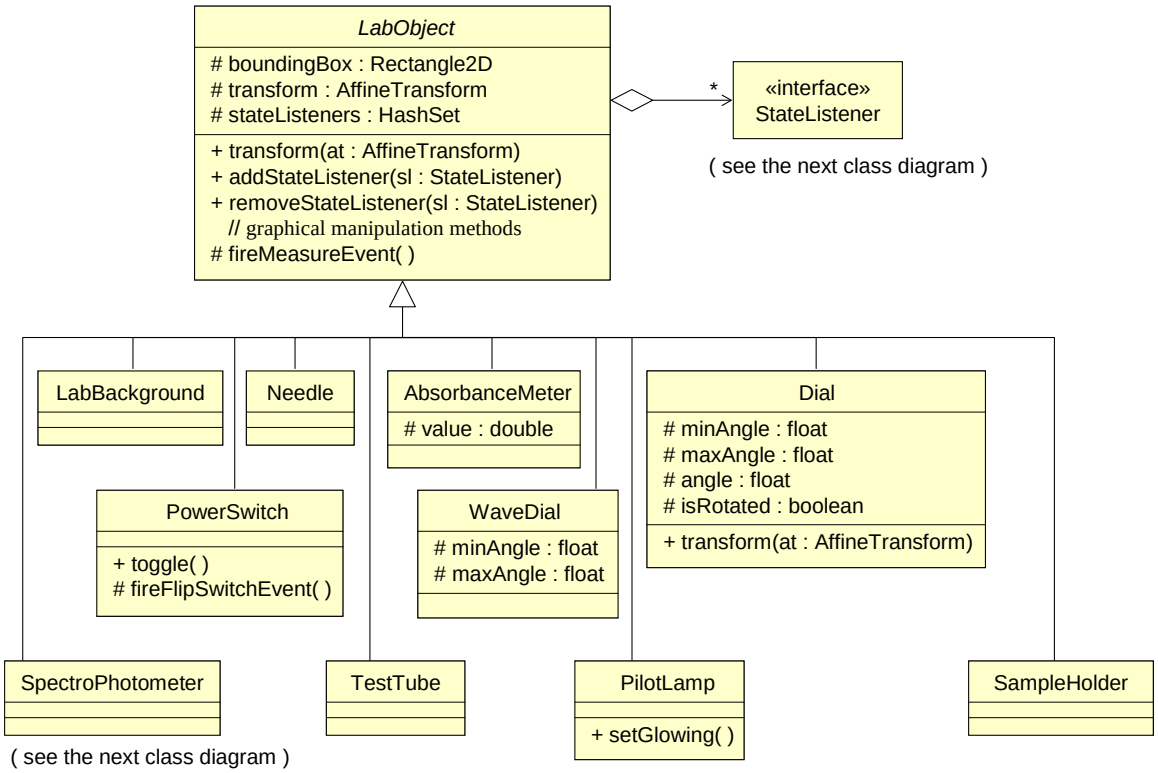


Figure 6: Class diagram for spectrophotometer parts. Note that LabObject is an *abstract* class, symbolized with an italicized class name. (This diagram is continued in Figure 8.)

protected transient HashSet stateListeners;

```
protected void fireMeasureEvent()
{
        HashSet listeners = (HashSet) stateListeners_.clone();
        for (Iterator i = listeners.iterator(); i.hasNext(); ) {
                ((StateListener)i.next()).measure();
        }
}
```

Note: The developer may wish to employ the Model–View–Controller (MVC) design pattern (http://en.wikipedia.org/wiki/Model-view-controller) for Lab Objects.

3.1.2 SpectroPhotometer.java

This is the central class of the spectrophotometer system. This class needs to maintain a complex state, so therefore we implement it using the *State design pattern*. (See the textbook, Section 5.2.3 for description of the State pattern. SpectroPhotometer is the "context" class in Figure 5-14 (page 264).

In the following table, we show that the event methods pass an Event type argument. This argument may carry some additional parameters of the event (e.g., timestamp, user ID, etc.), but it may not be necessary, depending on the implementation.

Class Name:	SpectroPhotometer	
Attributes:	currentState : SpectroState	Reference to the current state object
Operations:	+ setCurrentState(SpectroState)	Allows setting the current state from outside of the SpectroPhotometer
	+ flipSwitch(Event)	Event “flip switch” occurred—generated by the PowerSwitch when it is toggled/flipped
	+ flipLid(Event)	Event “flip lid” occurred—generated by the SampleHolder object, when its lid is opened or closed
	+ insertTestTube(Event)	Event “tube inserted” occurred—generated by the SampleHolder object, when a test tube is inserted
	+ removeTestTube(Event)	Event “tube removed” occurred—generated by the SampleHolder object, when the test tube is removed
	+ measure(Event)	Event generated by a Dial object when rotated; or by the SampleHolder when its lid is opened or closed.

The SpectroPhotometer constructor initializes the `currentState` attribute as OffEmptyClosed (see Section 4.1.3). Different SpectroState classes are described in Section 4. However, in Section 1.4.1 we mentioned that, in an earlier session, the user might have turned off the power switch while a test tube was still in the chamber. We decided that in this case the system would memorize this information and load the same test tube in a future session. Therefore, the constructor may look like so:

```
public class SpectroPhotometer
        extends LabObject implements StateListener {

        private SpectroState currentState;
        ...
        public SpectroPhotometer( ... ) {
                ...
                read the file with the record of previous sessions
                if ( test tube left in chamber ) {
                        tube = new TestTube( use stored params );
                        currentState = new OffOccupiedClosed(this, tube);
                } else {
                        currentState = new OffEmptyClosed(this);
                }
        }
        ...
}
```

3.1.3 PowerSwitch.java

Extends LabObject

This object is on/off switch, labeled as ① in Figure 1. It also turns on or off the pilot lamp when the spectrophotometer is turned on or off. The pilot lamp is described in Section 3.1.9.

Class Name:	PowerSwitch	
Attributes:	powered : boolean	Indicates if the switch is turned "on"
Operations:	+ toggle( )	When mouse pointer is clicked on this object, this method calls `fireFlipSwitchEvent()`
	# fireFlipSwitchEvent( )	Calls the method `flipSwitch()` on all listener objects

The list of listeners used in the method `fireFlipSwitchEvent()` is inherited from the base LabObject. Note that in Figure 2 we showed that PowerSwitch calls directly SpectroPhotometer method `flipSwitch()`. However, because we anticipate that other objects may need to be notified about the flipped switch event, we created the method `fireFlipSwitchEvent()` that will call `flipSwitch()` on all listener objects, including the SpectroPhotometer.

3.1.4 TestTube.java

Extends LabObject

Does not have any operations or attributes on its own. In other words, it inherits everything from its base class LabObject.

3.1.5 SampleHolder.java

Extends LabObject

This object is the sample holder (or, "chamber") for the test tube and it is labeled as ③ in Figure 1. It is a hole where the test tube is inserted for measuring the wavelength of light absorbed by the substance contained in the test tube. There is also a lid on top of the sample holder. Several facts should be observed:

- The sample holder can be either empty or hold one test tube
- The holder's lid can be open (lifted) or closed
- The test tube cannot be inserted or removed when the lid is closed
- The measurement should not be taken while the lid is open, because the external light will interfere with the measurement light that illuminates the test tube

3.1.6 Dial.java

Extends LabObject (see Figure 6)

There are two dial knobs on the front side of spectrophotometer:

(a) the zero-control dial (also known as "dark control"), labeled as ④ in Figure 1
(b) the light control dial (also known as "light control"), labeled as ⑤ in Figure 1

Both implement the same behavior: when the knob is rotated, it fires an event on its listener objects, contained in the `stateListeners` attribute of the base class LabObject (Section 3.1.1).

Class Name:	Dial	
Attributes:	# minAngle : float # maxAngle : float # angle : float # isRotated : boolean	Dial's minimum rotation angle. Dial's maximum rotation angle. Dial's current rotation angle. Indicates if the dial is currently being rotated.
Operations:	+ transform(at : AffineTransform)	Applies a rotation transformation.

A knob is shown graphically as a figure consisting of a circle and a line, which represent the knob and the reference mark.

The operation `transform()` allows the user to rotate the knob and set it in the desired position. This operation first calls the operation `transform()` on LabObject (its superclass), and then it informs all the listeners that a measurement should be performed for the new position of the dial. The operation `fireMeasureEvent()` is described in Section 3.1.1.

The code of transform(at : AffineTransform) is:

```
{
        super.transform(at);
        fireMeasureEvent();
}
```

Behavior "measuring" performs the calculations based on the solution's density and the light wavelength and calls the `transform()` method on the instrument (AbsorbanceMeter, described in Section 3.1.7) needle (described in Section 3.1.8) to display the wavelength. It also turns on or off the pilot lamp when the spectrophotometer is turned on or off. The knob "turning" behavior causes the "measuring" behavior to redo the measurement when a dial is rotated. Similarly, the lid "opening" behavior causes the "measuring" behavior to redo the measurement when the sample holder's lid is opened or closed (described in Section 3.1.5).

3.1.7 AbsorbanceMeter.java

Extends LabObject

The absorbance meter is labeled as ⑥ in Figure 1. It is linked with the needle and pilot light (green oval in the upper right corner of the absorbance meter scale). Figure 1 also shows a magnified view of the absorbance meter. Note that there are two scales with different series of numbers: the *transmittance* and *absorbance* scales. Note that one scale is a logarithmic scale (it is the only one with an "infinity" reading).

Class Name:	AbsorbanceMeter	
Attributes:	# value : float	Currently measured absorbance.
Operations:		Setter & getter for "value".

This object does not have any operations other than a setter and getter for the "value" attribute.

3.1.8 Needle.java

Extends LabObject

The needle is a part of the absorbance meter, labeled as ⑥ in Figure 1.

It does not have any operations or attributes on its own. In other words, it inherits everything from its base class LabObject.

3.1.9 PilotLamp.java

Extends LabObject

The pilot light, also known as ON/OFF indicator jewel lamp) indicates when the spectrophotometer instrument is turned on (using the on/off switch). It is a part of the absorbance meter, labeled as ⑥ in Figure 1. The pilot light is shown as an oval in the upper right corner of the absorbance meter scale in Figure 1. In the "on" state, the oval is glowing green; otherwise, it is transparent (or colored white).

Class Name:	PilotLamp	
Attributes:	# glowing : boolean	The state value is "true" when the lamp is "on"; otherwise it is "false."
Operations:	+ setGlowing(ok : boolean) + isGlowing( ) : boolean	Sets the lamp color as green ("on"). Returns "true" if the lamp is "on."

3.1.10 WaveDial.java

Extends LabObject

This object is the wavelength control dial for setting the color of the illumination light, labeled as ⑦ in Figure 1. Because this object is normally seen in a side projection but the main view is from the top, there will also be a magnified view of the wavelength dial (shown as top view), labeled as ⑧ in Figure 1.

Class Name:	WaveDial	
Attributes:	# minAngle : float # maxAngle : float	Dial's minimum rotation angle. Dial's maximum rotation angle.
Operations:		

3.1.11 LabBackground.java

Extends LabObject

Does not have any operations or attributes on its own. In other words, it inherits everything from its base class LabObject.

4 Spectrophotometer State Diagram

Based on the discussion of possible user interactions with spectrophotometer parts (Section 1.4), we identify the following events that can occur because of user interaction with the spectrophotometer:

1. Flip the power switch ON or OFF (Section 3.1.3)
   Event: “flipSwitch” — see interaction diagrams in Section 2.1
2. Flip the lid of the sample holder OPEN or COSED (Section 3.1.5)
   Event: “flipLid”
3. INSERT or REMOVE the test tube to/from the sample holder
   Events: “insertTestTube” and “removeTestTube”
4. Rotate a dial knob (Section 3.1.6)
   Event: “rotateDial” — see interaction diagrams in Section 2.2

We also recall that in real world, the instrument needle will move as a result of user interaction, *only while the power switch is ON*. Therefore, we define an *action* “measure” that should take place for any event if the power switch is currently ON.

There are three state variables that define the spectrophotometer state:

1. Power switch value: ON or OFF
2. Sample holder’s lid value: OPEN or CLOSED
3. Sample holder’s occupancy value: OCCUPIED or EMPTY

Based on these state variables, we define the following states of the spectrophotometer:

State Name	State Variable Values	Java Class
OnEmptyClosed	{Switch=ON, Sample-holder=EMPTY, Lid=CLOSED}	OnEmptyClosed.java
OnEmptyOpen	{Switch=ON, Sample-holder=EMPTY, Lid=OPEN}	OnEmptyOpen.java
OffOccupiedClosed	{Switch=OFF, Sample-holder=OCCUPIED, Lid=CLOSED}	OffOccupiedClosed.java
OffOccupiedOpen	{Switch=OFF, Sample-holder=OCCUPIED, Lid=OPEN}	OffOccupiedOpen.java
OffEmptyClosed	{Switch=OFF, Sample-holder=EMPTY, Lid=CLOSED}	OffEmptyClosed.java
OffEmptyOpen	{Switch=OFF, Sample-holder=EMPTY, Lid=OPEN}	OffEmptyOpen.java
OnOccupiedClosed	{Switch=ON, Sample-holder=OCCUPIED, Lid=CLOSED}	OnOccupiedClosed.java
OnOccupiedOpen	{Switch=ON, Sample-holder=OCCUPIED, Lid=OPEN}	OnOccupiedOpen.java

These events and states form the state diagram shown in Figure 7. Note that the diagram in Figure 7 identifies the unique initial state of the system as OffEmptyClosed, which means that the power switch is OFF, the sample holder is EMPTY, and its lid is CLOSED. Only the state transitions shown in Figure 7 are allowed—all other transitions are forbidden and the events that are not shown in this diagram are ignored.

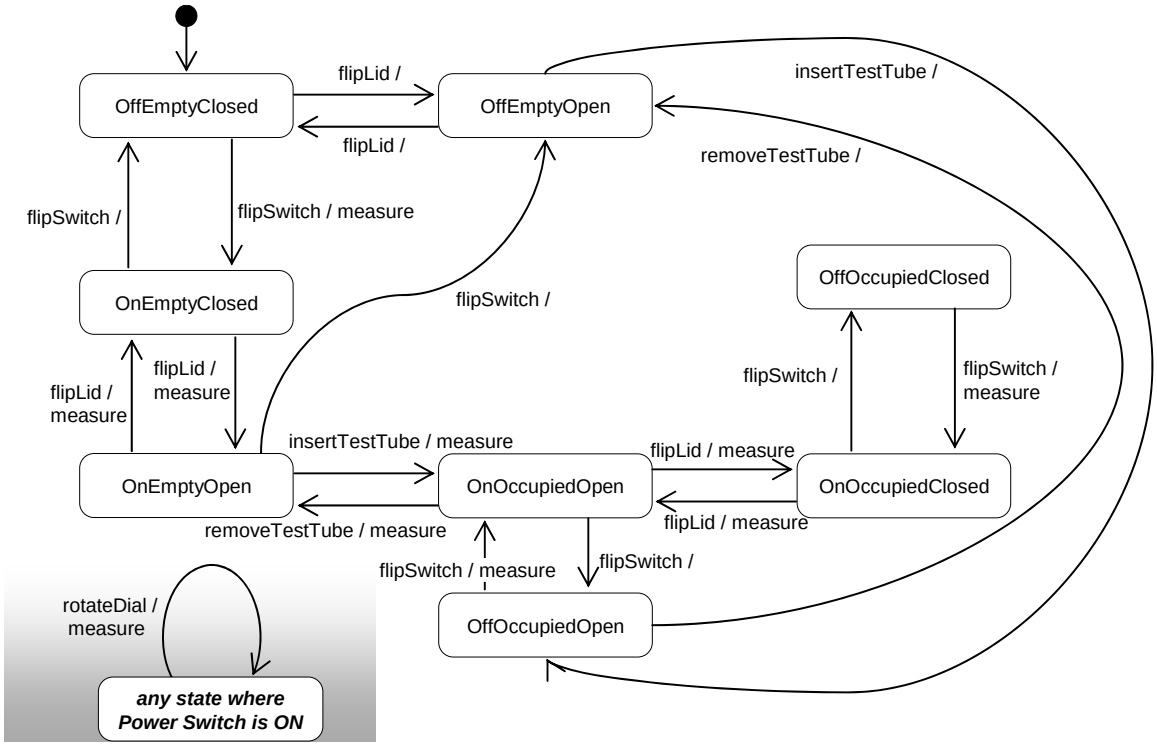


Figure 7: State diagram showing the spectrophotometer operating states and events that cause transitions between the states.

The corresponding class diagram is shown in Figure 8 and the Java classes are described in the following sections.

4.1.1 SpectroState.java

This class is a base class for different spectrophotometer states (Figure 8). It is part of the *State design pattern*. (See the textbook, Section 5.2.3 for description of the State pattern.)

Note: this is an *abstract* class.

Class Name:	SpectroState	
Attributes:	spectroContext : SpectroPhotometer	Context object for this State
	absorbance : double	The default absorbance value is 5.67
Operations:	+ flipSwitch( ) // abstract	Defines this state reaction on a flip-switch event.
	+ flipLid( ) // abstract	Defines this state reaction on a flip-lid event.
	+ insertTestTube( ) // abstract	Defines this state reaction when a cuvette is inserted in the chamber.
	+ removeTestTube( ) // abstract	Defines this state reaction when the cuvette is removed from the chamber.
	+ measure( )	Action taken on state transitions. Calculates the current value of "absorbance" based on

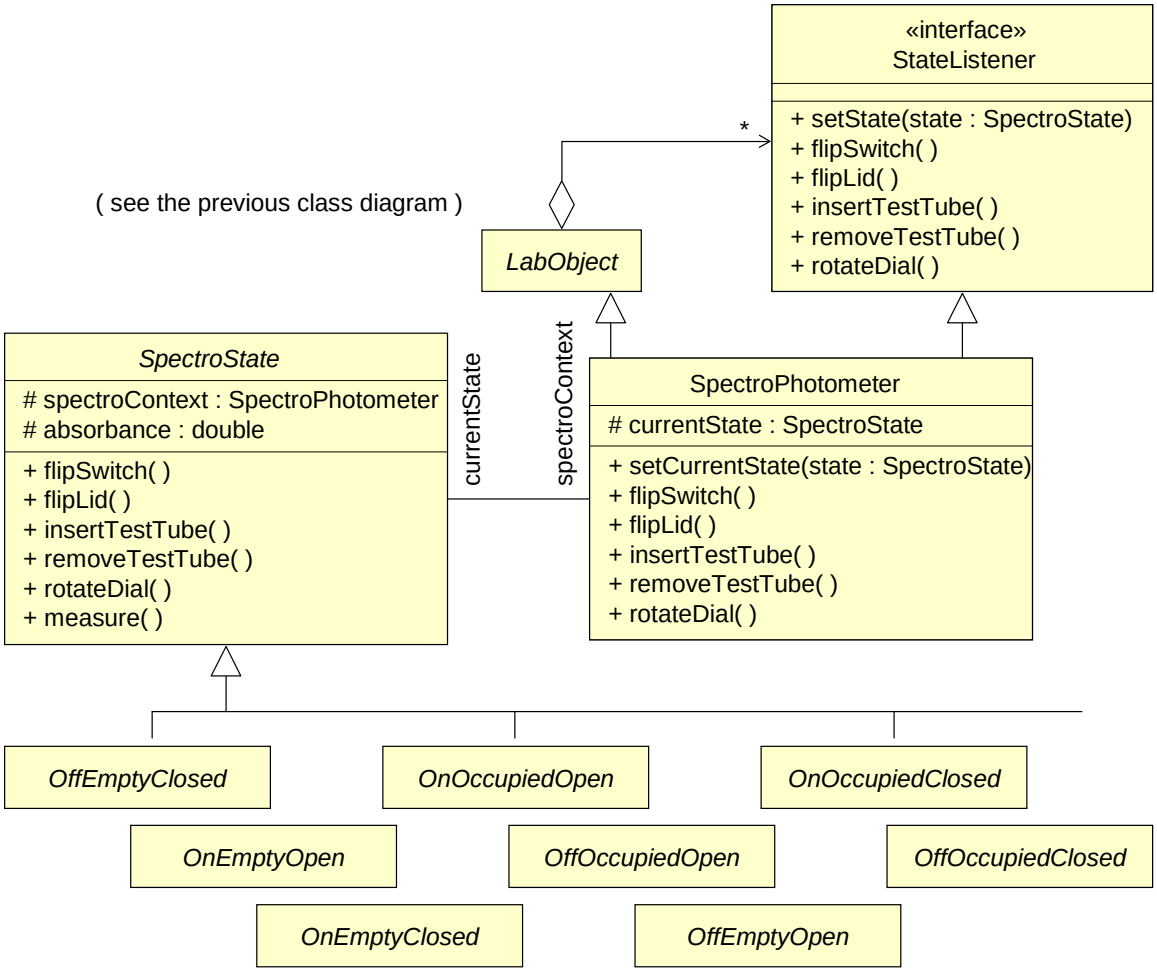


Figure 8: Class diagram for spectrophotometer operating states. Note that LabObject and SpectroState are *abstract* classes. (This diagram is continued from Figure 6.)

		the light wavelength and the test tube substance.

In this implementation, we show that the state class calls the context object and sets its current state:

```
state = new ___ // construct appropriate next state
spectroContext.setCurrentState(state);
state.measure();      // action on state transition
```

This state transitioning is slightly different from that in Section 5.2.3 of the textbook, where the event method returns the next state, and the context class itself sets its current state. The developer can choose either implementation, as seen more fit.

The method `measure()` is not implemented in the base class SpectroState. Each concrete state implements its own measurement method, because the concrete state best knows how to perform the appropriate absorbance measurement.

4.1.2 StateListener.java

An interface defining the public methods for listener objects that listen on spectrophotometer part objects for *events* generated by user interaction.

4.1.3 OffEmptyClosed.java

Extends SpectroState (see Figure 8)

This state is designated as the initial state of the virtual spectrophotometer machine (Figure 7). We need to decide how the instrument needle will be positioned in this initial state. A simple option is to have it rotated maximally to the left corner of the scale (indicated with ⑥ in Figure 1).

4.1.4 OnEmptyClosed.java

Extends SpectroState (see Figure 8)

The spectrophotometer transitions into this state when the power switch is turned ON (Figure 7), assuming that the test-tube chamber is empty and its lid is closed. The jewel lamp indicator should start glowing green (see the green circle within the absorbance meter area, labeled as ⑥ in Figure 1). The instrument needle should jump into a random position, because initially the instrument is not calibrated (see Sections 1.4.2 and 2.2).

4.1.5 OffOccupiedClosed.java

Extends SpectroState (see Figure 8)

In this state, the spectrophotometer device is OFF, the test-tube slot is OCCUPIED and CLOSED.

Class Name:	OffOccupiedClosed	
Attributes:		
Operations:	+ measure( )	Issues warning about improper measurement.

The constructor:

```
public OffOccupiedClosed(SpectroPhotometer spectro) {
       super(spectro);
}
```

The `measure()` method issues a warning that measurement is attempted while the device is OFF.

```
public void measure() {
       System.out.println("measuring while off, occupied and closed");
       return number;
}
```

When the switch is flipped, set the state of the test-tube slot as OCCUPIED and CLOSED.

```
public void flipSwitch() {
       setState(new OnOccupiedClosed(spectroContext));
}
```

When the lid is flipped, set the state of the test-tube slot as OCCUPIED and OPEN.

```
public void flipLid() {
        setState(new OffOccupiedOpen(spectroContext));
}
```

Note that this class (OffOccupiedClosed) may be set as the initial state, as described in Section 3.1.2.

4.1.6 OffOccupiedOpen.java

Extends SpectroState (see Figure 8)

4.1.7 OffEmptyOpen.java

4.1.8 OnOccupiedClosed.java

4.1.9 OnOccupiedOpen.java

4.1.10 OnEmptyOpen.java

5 Additional Information

There is a great deal of online information. Some examples are given at this project's website, http://www.ece.rutgers.edu/~marsic/books/SE/projects/. Also, project reports along with running software developed by students at Rutgers University are available for download at this website.

David A. Reckhow, "Light-Scattering and Molecular Spectrophotometry," Chapter XVII of *Analytical Chemistry for Environmental Engineers and Scientists*, Department of Civil and Environmental Engineering, University of Massachusetts Amherst. Online at: http://www.ecs.umass.edu/cee/reckhow/courses/572/572bk17/572BK17.html

David B. Fankhauser, "Spectrophotometer Use," University of Cincinnati Clermont College. Online at: http://biology.clc.uc.edu/fankhauser/Labs/Microbiology/Growth_Curve/Spectrophotometer.htm

Steve Baskauf, "Principles of Spectrophotometry," Department of Biological Sciences, Vanderbilt University. Online at: http://www.cas.vanderbilt.edu/bsci111a/equipment/spec20.htm

The reader may also find useful information in this article:

R. Subramanian and I. Marsic, "ViBE: Virtual biology experiments," *Proceedings of the Tenth International World Wide Web Conference (WWW10)*, Hong Kong, pp. 316-325, May 2001. Online at: http://www.ece.rutgers.edu/~marsic/publications/www10/