Lab 2

Sorting Laundry

Objective
The purpose of this lab is to gain experience using conditionals.

The Scenario
Many students must wash their own clothes for the first time when they come to college. By the time some figure out how to use laundry machines, their underwear is pink and their T-shirts and other light-colored items are streaked with lots of interesting colors. In the hopes of helping next year's first-year students adjust more easily to college, we would like you to write a laundry sorting simulator.

The Approach
It is usually good practice to develop programs incrementally. Start with a simplified version of the full problem. Plan, implement, and test a program for this simplified version. Then, add more functionality until you have solved the full problem.

To encourage this approach, we have divided this lab into two parts. For the first, we will describe a laundry sorter with a very simple interface. You should plan, implement, and test a program for this problem first. Then, in the second part of the assignment we will ask you to add additional functionality.

You should approach each of the two parts in this step-wise fashion. For example, in the first part you should begin by drawing the objects on the screen. Then, once your program can draw the objects, you can move on to figure out how to add functionality to the program.

Lab Preparation
To prepare for this lab assignment, we expect you to do the following before coming to lab next week:

1. Read this handout carefully.
2. Sketch out, as much as possible, what the code will do. You should at least outline the begin method, which will draw the baskets and clothing on the screen.
3. Do not start coding before lab. It is often more productive to design a program away from the computer, and we would like you to get into the habit of working in this way. If you come to lab with a written idea of how to approach the problem, you will be able to make progress much more quickly than you would otherwise.

Part 1

The simulator should begin with three wash baskets on the screen (for our purposes they will be squares) and the item to be sorted or the “swatch” (a swatch is a sample piece of fabric). The baskets will be labeled “whites”, “darks”, and “colors”. An image showing the display we have in mind appears below.

When the program starts, a color swatch will appear on the screen. The user (“laundry trainee”) should then click on the basket corresponding to the color of the swatch. If the user clicks on the correct basket, the program should then randomly select a new color for the next item and display it on the screen. For this part of the assignment, you should select among the three colors Color.white, Color.red, and Color.black when creating swatches. If the user clicks on an incorrect basket, the original item remains in position for another try.
A working demo of the laundry sorter is at:


**A Warning!** One odd feature of the simple interface that will occur is a result of the fact that the program selects laundry items randomly. Because the selection is random, it sometimes picks the same color twice in a row. When this happens and you click on the correct basket for the first item, you will get the feeling that the program did not do anything. Even though it has actually displayed a new item, the new item looks identical to the old one, so you may think nothing happened. Don't let this trick you into thinking that your version of the program (or ours) isn't working correctly. The more advanced interface in Part 2 includes counters in the display that make what the program is actually doing clearer.

**Creating a Project**  For this lab, you will create a project from scratch inside Eclipse:

1. Open the File menu. Select “New” and then “Project.”
2. Select “Java,” then select “Java Project,” and then “Next.”
3. Enter “Laundry” as the project name, then click “Next.”
4. Select the “Libraries” tab.
5. Click “Add Variable.”
6. Select OBJECTDRAW from the list and click “OK.”
7. Click “Finish.”

You now have an empty project to which you should add a Java file for your class:

- Click on Laundry in the Package Explorer panel on the left side of the Eclipse window.
- Open the File menu. Select “New” and then “Class.”
- Enter “Laundry” as the name of the class.
- Enter “WindowController” as the Superclass.
- Click Finish.

You should now have a Laundry.java file inside the Laundry project. Modify the file so that it starts with import objectdraw.* and java.awt.*.
Design of Part 1  You will need to design an extension of the WindowController class which will display the wash baskets and the item to be sorted. Begin by laying out where all the items go on paper. (We recommend using a window that has height and width of 400 pixels.) The picture should look like the one above but with the coordinates labeling where the baskets, text, and swatch will go.

When the program begins, draw all the wash baskets (with labels) on the screen. Then draw the item of clothing that is to be sorted. For simplicity, always make the first item have color red. The item should actually consist of two rectangles, a FilledRect which is the appropriate color and a FramedRect which is the same size, but lays on top of the filled rectangle to form a border. (Otherwise it will be awfully difficult to see a white item!)

Think Constants Use constants (private static final CONSTANT_NAME) for all the relevant location and size information. This makes it easier to change the layout and also makes your program much, much easier to read (presuming you give constants good names). Constant names are by convention written with all capital letters and underscores, e.g. THIS_IS_A_CONSTANT. Your constants may be more complex objects like Locations. You can initialize constants with the results of a constructor:

    private static final Location SOME_LOCN = new Location(100,200);

However, you may NOT have constants whose definition uses canvas (e.g., no FramedRect constants).

The widths and heights of wash baskets and the item to be sorted, coordinates of the upper left corner of each of these, etc., are all good candidates for constants.

After writing the code to draw the baskets and item to be sorted, run the program to see if it does what you expect. As with last week’s lab assignment, you must create a configuration to run your program. These instructions are the same as last week, except that you will use a different name and different dimensions.

- Open the Run menu.
- Select “Run...”
- Select “Java Applet” from the Configurations list.
- Click the “New” button at the bottom left.
- Make sure that the Name field at the top, the Project, and the Applet class all say Laundry.
- Click on the “Parameters” tab.
- Enter 400 for the width and 400 for the height.
- Click the “Apply” button and then the “Run” button.

Identifying the Correct Basket Once you have done the layout, you should next figure out how to generate new items. (Generating the random color is discussed below.) Because you will initially be generating the item in one method (begin) and checking to see if the user clicked in the appropriate basket in a different method (the onMouseClick method), you will need to associate some information with an instance variable that will enable onMouseClick to determine which is the correct basket.

An appropriate way to do this is to use an instance variable of type FramedRect that stores the basket in which the user should click. When you generate a new item (either initially in begin or thereafter in onMouseClick), you will associate the instance variable with the rectangle/basket in which an item of its color should be placed. That way, to determine if the user clicks on the correct basket, onMouseClick can simply check to see if the rectangle currently associated with the instance variable contains the point where the mouse was clicked. Then, onMouseClick will either select a new color for the item (if the user was correct) or wait until the user clicks again (if incorrect).
**Picking a random color**  To pick a random color, use the `RandomIntGenerator` class as described in the textbook. Since there are three colors to select from, you should create your random number generator to return random numbers in the range of 1 to 3. Then, pick the color based on the number generated (i.e., make the laundry be `Color.white` if the number is 1, `Color.black` if the number is 2, and `Color.red` if the number is 3).

**Recycling**  Because your program only uses one laundry item at a time, reuse the same rectangle for each new laundry item by simply changing its color rather than creating a new rectangle. In general, it is a good strategy to reuse objects when possible rather than creating new ones when possible, because it uses memory and makes the code clearer.

Once you have completed Part 1, have a TA or instructor check to make sure it is working correctly before moving on to Part 2.

---

**Part 2**

Once you get the basic version working, we would like you to jazz it up a bit. Here are the extensions we would like you to make:

1. Add counters (`Text` items) at the bottom of the picture showing the number of correct and incorrect placements. This makes it clearer when the student succeeds in placing the item correctly. They should read something like “Correct = nn”, “Incorrect = mm”. The value in the first `Text` item will be formed by concatenating the string “Correct = ” with an integer instance variable which keeps track of the number of correct answers. The other `Text` item is similar. (You can also have both counters interspersed in the text of one text string.) If the user presses the mouse button down outside the laundry item, it should not increase the correct or the incorrect counter.

2. Users should drag the items to the correct laundry basket rather than just clicking on the basket. Recall from the example in class that you will need an instance variable to label the last mouse position before the drag so you can determine how far to drag the item. If the user drops their laundry outside of all baskets, it will count as an incorrect sorting.

3. Assign the item a randomly generated color by randomly choosing integers `redNum`, `greenNum`, and `blueNum` between 0 and 255 for each of the red, blue, and green components of the color. Colors are created from those components by writing `new Color(redNum,greenNum,blueNum)`.

Now define criteria for determining where each color should go. The simplest criterion is based on the sum of the three color components. If the sum of the component numbers is less than 230, then it goes in the darks basket, if it is greater than 675, then it goes in the whites basket. Otherwise it goes in the colors basket.

We will let you figure out most of the details of how to add the features for the more advanced versions. Note that to drag the item, you should drop the `onMouseClick` method in favor of using the three methods:

- `onMousePress` – for when the user first clicks on the item - though remember that they might miss.
- `onMouseDrag` – to do the actual dragging.
- `onMouseRelease` – check to see if they’ve dropped it in the right place when the mouse is released.

There is a working version of the final laundry sorter at:

**Extra credit**  We would like you to focus on getting the lab working, writing good comments, and using good style in your programming. If you have done all that and would like to do more, here is an opportunity to earn extra credit.

As described in the second feature of Part 2 above, if the user drops their laundry outside of all baskets, it will count as an incorrect sorting. For extra credit, you can be nicer to the user. If he/she misses all the baskets, let him/her try again without increasing either the correct or incorrect counters.

As always, you should also feel free to extend or embellish your programs in any way you like, once you have the assignment completely finished.

---

**Grading guidelines**

As with last week’s lab, your grade will have components based on style and on correctness. Please note that each week we will likely be adding new style guidelines that we will be looking for in your programs.

<table>
<thead>
<tr>
<th>Value</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 pts.</td>
<td>appropriate comments</td>
</tr>
<tr>
<td>2 pts.</td>
<td>good variable names</td>
</tr>
<tr>
<td>2 pts.</td>
<td>good use of constants</td>
</tr>
<tr>
<td>2 pts.</td>
<td>appropriate formatting</td>
</tr>
<tr>
<td>1 pt.</td>
<td>does not generate new objects unnecessarily</td>
</tr>
<tr>
<td>1 pts.</td>
<td>design issues</td>
</tr>
</tbody>
</table>

**Correctness (10 pts)**

<table>
<thead>
<tr>
<th>Value</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 pts.</td>
<td>generates a new color swatch only if previous one placed correctly</td>
</tr>
<tr>
<td>2 pts.</td>
<td>swatch displayed in the correct initial position; returns to original location if incorrectly sorted</td>
</tr>
<tr>
<td>2 pts.</td>
<td>updates # correct and # incorrect appropriately</td>
</tr>
<tr>
<td>2 pts.</td>
<td>drags swatch properly (-1 pt if use clicking instead of dragging)</td>
</tr>
<tr>
<td>2 pts.</td>
<td>appropriate behavior if user does unexpected things like starting to drag outside the laundry item</td>
</tr>
</tbody>
</table>

**Extra credit (1 pt)**

<table>
<thead>
<tr>
<th>Value</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 pt.</td>
<td>does not update # correct or # incorrect sorted if user misses all baskets.</td>
</tr>
</tbody>
</table>

---

**Final remarks**  Try to complete the basic version of the lab before attempting the more advanced features. Just work on adding one feature at a time, testing each thoroughly before working on the next.

**Submitting Your Work**

The submission procedure is somewhat different from last week. How you save your work is the same, but how you reach the drop-off folder will be different.

- First, return to Eclipse and make sure you included your name and lab in a comment at the start of the program.
- Next, click on “Laundry” in the Package Explorer panel on the left side of the Eclipse window.
- Now, select “Export” from the “File” menu.
- Select “File system” in the dialog box and click “Next.”
• Click the “Browse” button.

• Click on the house icon.

• Click the “New Folder” button. Name your folder with your last name followed by Lab2. For example, if your last name is Smith, name your folder “SmithLab2.” Do not put spaces in the name. Click the “Create” button. Then click the “Choose” button. Then click ‘Finish’.

• Quit Eclipse.

• Open a Finder window and go to your home directory (by clicking on the house icon). You should see the folder you just created.

• Click on the Desktop, then go to the “Go” menu and select “Connect to Server.”

• Type “cortland” in for the Server Address and click “Connect.”

• A window will come up which says “Connect to the file server “cortland” as: “, select Guest, then click “Connect.”

• A window will next come up where you should select the volume “Courses” to mount and then click “OK.”

• A window will come up which says “You are connected . . .” Click “OK.”

• Then a Finder window will come up where you should double-click on “cs134”, then drag the folder you created into either “Dropoff-Monday” if you are in Monday’s lab or “Dropoff-Tuesday” if you are in Tuesday’s lab. When you do this, the Mac will warn you that you will not be able to look at this folder. That is fine. Just click “OK”.

• Log off of the computer before you leave.

You can submit your work up to 11 p.m. two days after your lab (11 p.m. Wednesday for those in the Monday Lab, and 11 p.m. Thursday for those in the Tuesday Lab). If you submit and later discover that your submission was flawed, you can submit again. We will grade the latest submission made before the 11 p.m. deadline. The Mac will not let you submit again unless you change the name of your folder slightly. It does this to prevent another student from accidentally overwriting one of your submissions. Just add something to the folder name (like the word “revised”) and the re-submission will work fine.