Lab 2

Sorting Laundry

Objective To gain experience using conditionals.

The Scenario Doing laundry: an important but often little-understood aspect of college. By the time some students figure out how to use laundry machines, their underwear is pink and their white T-shirts are streaked with lots of interesting colors. In the hopes of helping this 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. The second part of the assignment asks you to add additional functionality to your simple version.

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 continue to add functionality to the program.

Lab Preparation In preparation for lab next week, we expect you to do the following before coming to lab:

1. Read this handout carefully.
2. Prepare a written design for your program. In order to facilitate this step, we have provided a design template for you. You may use this template to help you organize your ideas for the program. When designing a program, you should think about its behavior in general, the instance variables you will need to store information, and the event-handling methods you will need to implement the program's behavior properly.

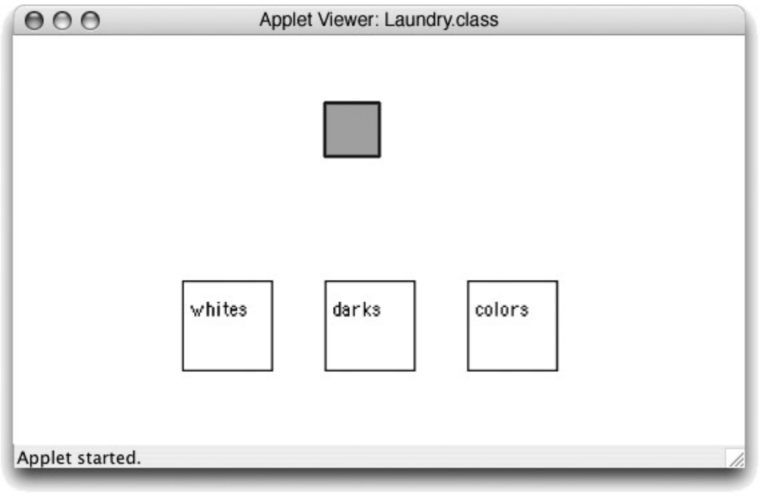
   As part of the design, you should also draw the layout for your graphical objects on the screen. You may use the provided graph paper for this if you wish.
3. Do not start coding before lab. It is often more productive to design a program away from the computer, and we would like you to get into the 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 and the item to be sorted, or the "swatch" (a sample piece of fabric). For our purposes, the baskets will simply be squares, and they 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 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 on our handouts web page.

A Warning! Because swatch 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 BlueJ:

1. From the "Project" menu select "New Project...".
2. First, make sure you are in your "cs134" directory. You can change the directory by clicking on the folder menu.
3. In the dialog box, name your project "Lab2YourLastNameLaundry", and hit "Create".
4. You should now see a project with no class files. Press the "New Class..." button. Create a "WindowController" class with the name "Laundry".
5. You should now see the "Laundry" class appear in the project. Double click on the "Laundry" class to open up the BlueJ text editor.

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. The default canvas dimensions – 400x400 – should work well. 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`.

```
private static final int SWATCH_X = 150;
```

You may declare `Location` constants as well by initializing the constant variable with the results of a constructor:

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

However, you should NOT have constants whose definition uses `canvas` (e.g., no `FramedRect` constants), or your program may not work correctly!

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.

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 the swatch's 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 because it uses less memory and makes the code more clear.

Once you have completed Part 1, make sure it works correctly and then move 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 presses 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 on our handouts web page to give you an idea of what we have in mind.

Optional Extensions 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 the user misses all the baskets, let the user try again without increasing either the correct or incorrect counters.

On this lab, and all others, you should also feel free to extend or embellish your programs in any way you like, but always be sure to have the basic assignment working properly beforehand. Also, be sure to describe the additional features in your comments, so that we understand them as features, rather than as bugs.

Submitting Your Work

Once you have saved your work in BlueJ, please perform the following steps to submit your assignment:

- First, return to the Finder. You can do this by clicking on the smiling Macintosh icon in your dock.
- From the "Go" menu at the top of the screen, select "Connect to Server...".
- For the server address, type in "Guest@fuji" and click "Connect".
- A selection box should appear. Select "Courses" and click "Ok".
- You should now see a Finder window with a "cs134" folder. Open this folder.
- You should now see the drop-off folders for the three labs sections. As with last week, drag your "Lab2Laundry" folder into the appropriate lab section folder. 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. on Wednesday if you're in the Monday afternoon lab; up to 6 p.m. on Thursday if you're in the Monday night lab; and up to 11 p.m. on Thursday if you're 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 "v2") and the resubmission will work fine.

Grading Guidelines

As always, we will evaluate your program for both style and correctness. Here are some specific items to keep in mind and focus on while writing your program:

Initial Design

- detailed enough to convey the basic design of Part 1
- includes a hand-drawn (or computer-drawn) picture of the layout, noting the essential coordinates

Style, Design, and Efficiency

- appropriate comments
- good variable names
- good use of constants
- appropriate formatting
- does not generate new objects unnecessarily
- design issues

Correctness

- generates a new color swatch only if previous one placed correctly
- swatch displayed in the correct initial position
- swatch returns to original location if incorrectly sorted
- updates # correct and # incorrect appropriately
- drags swatch properly
- appropriate behavior if user does unexpected things like starting to drag outside the laundry item

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.