Lab 3 – Sorting Laundry

CSCI 1101B – Spring 2015

Due: February 17, 10 pm

Objective: To gain experience working with conditionals.

For this lab, you will a *laundry sorting simulator*.

It is usually a 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, I have divided this lab into two parts. For the first, I 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 I will ask you to add additional functionality.

You should approach each of our two parts in this step-wise fashion. For example, in the first part you might begin by just writing the instructions to construct the necessary graphical objects without worrying about any of the mouse event handling. Once your program can draw the picture, you can move on to figure out how to handle events.

NOTE: Part 2 asks you to change/improve some of the things you did in Part 1. You only need to turn in the version of the laundry sorter that you create in Part 2.

Part 1

The simulator should begin with three wash baskets on the screen (for our purposes they can just be rectangles or squares). One is labeled "whites", one "darks", and the last "colors". An image showing the kind of display I have in mind appears in Figure 1.

When the simulation begins, a square of random color (i.e., a piece of clothing) will appear on the screen. The user should then click in the corresponding basket. If the user is correct, the program should randomly select a new color for the next item and display it on the screen. If the user clicks on an incorrect basket, the original item remains in position for another try.

What does "correct" mean? You should base this on the sum of the three color components in the random color. If the sum of the component numbers is less than 230, it is dark, if it is greater than 600, it is white. Otherwise, it is colored. (**Remember: named constants!**)

A Warning! One odd feature of the simple interface that may bother you a bit is a result of the fact that the program selects laundry items randomly. Because the selection is truly 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 ignored you. Even though it has actually displayed a new item, the new item looks just like the old one, so you may think nothing changed. Don't let this trick you into thinking that your version of the program isn't working correctly. The more advanced interface in Part 2 includes counters in the display that make it clearer whether the user succeeded.

Design of Part 1. You will need to design a program that will display the wash baskets and the item to be sorted. The picture should look more or less like the one above. Don't worry if your labels aren't perfectly centered in the baskets.

When the program begins, place all the wash baskets (with labels) on the screen. Then, add the item of clothing that is to be sorted. You can, but don't need to, let the first item always have color white. The item should actually consist of two rectangles, a `FilledRect` which is the appropriate color and a black `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! When you lay out the wash baskets and item, make up constants (`private static final ...`) for all the relevant information. This will make it easier to change things around and also make your program much, much easier to read (presuming you give constants good names). Remember, constant names are by convention written with all capital letters and underscores. Your constants may be (and often should be) more complex objects like `Location`s. You can initialize constants with the results of a constructor:

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

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 named constants.

After writing the code to draw the initial display, it would be good to run it to see if it does what you expect.

Next, you should write the code for the method `onMousePress` (or `onMouseClick` if you prefer) that checks to see if the user selected the right basket and, if they did, generates a new color for the piece of clothing.

Recycling: Because your program only uses one laundry item at a time, you should just recycle it, i.e. use the same rectangle for each laundry item instead of creating a new rectangle each time you want to make a new piece of laundry. Simply change its color. In general, you should reuse objects rather than creating new ones when possible, as this generally uses less time and does not clutter memory.

Part 2:

Once you get the basic version working, I would like you to add some additional features:

1. Add labels (`Text` items) at the bottom of the picture showing the number of correct and incorrect placements. This makes it clearer when the user 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 is similar. When your program starts, the window should look like Figure 1. After the user has sorted some laundry, your window should look something like Figure 2.

2. Users should drag the items to the correct laundry basket rather than just clicking on the basket. Since, when the user drags the item, the arrow should remain pointing to the spot it was pointing to when the mouse was clicked (instead of jumping to the upper-left corner), 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 using the `move()` method. If the user presses the mouse button down outside the laundry item, it should not increase the correct or the incorrect counter.

3. Notice that, as described so far, if the user drops their laundry outside of all baskets, it will count as an incorrect sorting. Be nicer. **If the user drops the laundry outside any basket, it should automatically move back to the position it started from and you should not increase either counter.**

4. Finally, if the user tries to place the item in an incorrect basket, you should display the color "coordinates" and the color thresholds (what color element sums correspond to darks, colors, and whites) in the lower part of the window. See Figure 3. This information should *not* appear if they drop the item outside of *any* basket. Of course, this information should disappear as soon as they drag the item to the correct basket. Dropping the laundry in an incorrect basket should also return the laundry to the starting position.

Note that for the second enhancement you need to replace the `onMousePress` method with 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, and
- `onMouseRelease`, to check to see where they've dropped the item and do the appropriate thing.

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

When you are finished, compress and submit your lab on Blackboard in the usual way (refer to the previous lab handouts if you need to refresh your memory). Also as in previous weeks, you should keep a copy of your lab in a safe place (e.g., in your folder on the `microwave` server) after you are done working on it so that you can easily refer to it later.

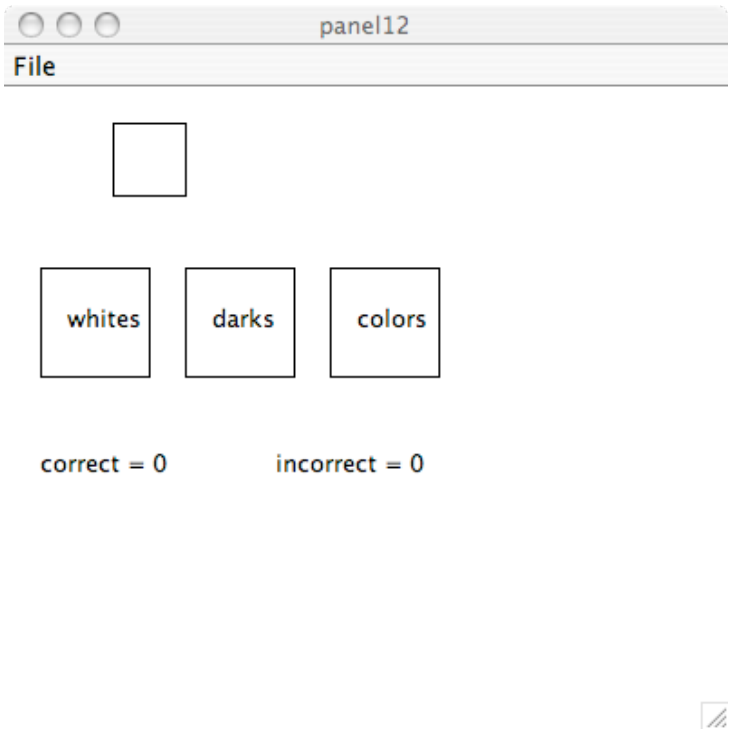

Figure 1: Starting laundry window.

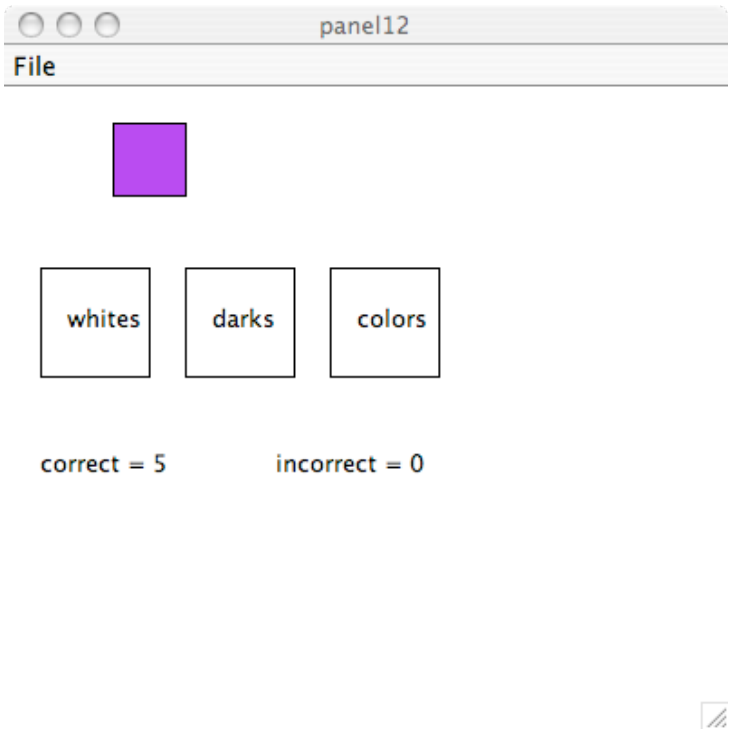

Figure 2: Laundry window in progress.

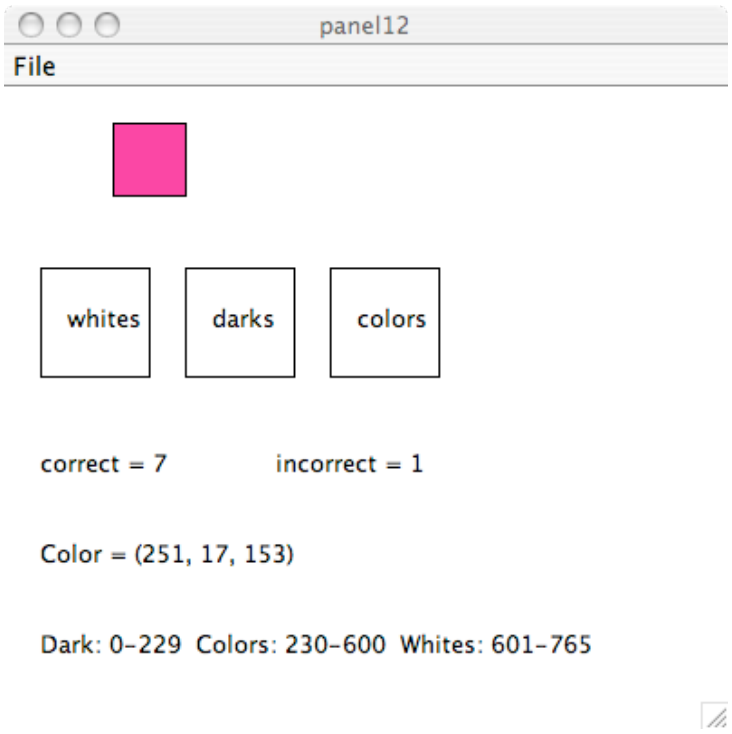

Figure 3: Laundry window after user selects the wrong basket.