CS 051 — Laboratory #2
Dirty Laundry

Objective: To gain experience using conditionals.

—

The Scenario. One thing many students have to figure out for the first time when they come to college is how to wash their clothes. Often by the time most students have figured it out, all their underwear is pink and T-shirts and other light-colored items are streaked with all sorts 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 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, 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 are also encouraged to 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.

The Design. You should bring a design for the first part of this assignment to lab. The design should include the name of the main class, the names of constants and instance variables, and the names of the methods. In addition, everything should be commented: what will each instance variable keep track of and what will each method do? Finally, the design should give a sense of how each method will do what it's supposed to do (put in comments!).

The design is an important step in the coding process! You should approach it as attempting to answer the above questions, not as trying to see how far along you can get in coding the lab. I promise you thinking about the design at a higher-level will make your life easier in the long run.

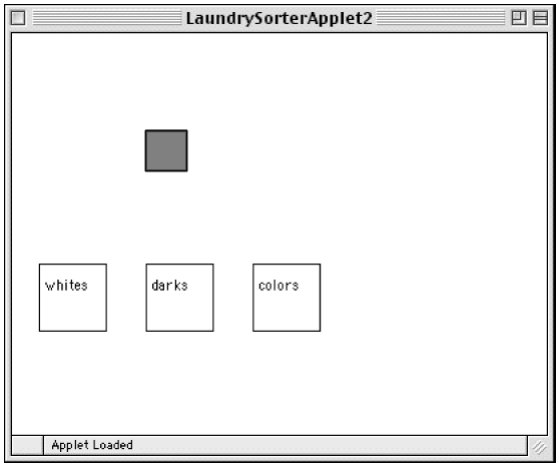
Part 1:

Your program should begin with three washbaskets 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 we have in mind appears below. When the simulation begins, a color swatch will appear on the screen. The user ("laundry trainee") 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. For this part of the assignment you may just select among the three colors `Color.WHITE`, `Color.RED`, and `Color.BLACK` when creating items of clothing. If the user clicks on an incorrect basket, the original item remains in position for another try.

Design of Part 1.

You will need to design a new class that is an extension of the `WindowController` class which will display the wash baskets and the item to be sorted. You might want to try laying out where all the items go on some graph paper. The picture should look more or less like the one we provided here..

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. For simplicity you might as well always make the first item have



color white. 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! 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). Constant names are by convention written with all capital letters and underscores, e.g. THIS_IS_A_CONSTANT. Your constants may be (and often should be) more complex objects like `Location`. You can initialize constants with the results of a constructor:

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

Remember that you may NOT have constants whose definition uses `canvas` (e.g., no `FramedRect` constants). Other good candidates for constants are: he widths and heights of wash baskets and the item to be sorted, coordinates of the upper left corner of each of these, etc..

Identifying the Correct Basket Once you have done the layout and figured out how to generate new items, all you have to do is to write the code for the method `onMouseClick`. Because you may 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`.

When you generate a new item (in either `begin` or `onMouseClick`), you will associate this variable with the rectangle/basket in which an item of its color should be placed. That way when the user clicks on a 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).

Think carefully about those last two paragraphs and what they're saying!

Hints and a warning

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 (or ours) isn't working correctly. The more advanced interface in part 2 includes counters in the display that eliminate this problem.

Generating random numbers We have provided a class in the `objectdraw` package which helps to generate random numbers. This will be used to determine the next color for the item.

Suppose you wish to generate random integers in the range from m to n (where $m \leq n$). Let `generator` be an instance variable declared to be of type `RandomIntGenerator`. Create a new random number generator from class `RandomIntGenerator`, and assign it to `generator`:

```
generator = new RandomIntGenerator(m,n);
```

Now every time you want a new random integer in that range, simply invoke the method

```
generator.nextValue()
```

which will return a randomly chosen integer between m and n (inclusive). Thus to generate integers between 1 and 3 (say, standing for white, dark, and colored), use

```
generator = new RandomIntGenerator(1,3)
```

If `n` is the value of `generator.nextValue()`, make the color of the item be `Color.WHITE` if `n` is 1, `Color.BLACK` if `n` is 2, and `Color.RED` if `n` is 3.

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 labels (`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 is similar.

2. Users should drag the items to the correct laundry basket rather than just clicking on the basket. Recall from the examples 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 presses the mouse button down outside the laundry item, it should not increase the correct or the incorrect counter.

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. You can create a color 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, decide it is dark, if it is greater than 600, decide it is white. Otherwise it is colored. This rule isn't perfect, so after you get the program working you might want to experiment with other criteria.

We will let you figure out most of the details of how to add the features for the more advanced versions. One piece of advice is that for the second enhancement you will be dropping 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.

Grading Guidelines

Table 1: Grading Guidelines

Value	Feature
	Style, Design, and Efficiency (10 pts total)
2 pts.	Descriptive comments
2 pts.	Good variable names
2 pts.	Good use of constants
2 pts.	Appropriate formatting
1 pt.	Does not generate new objects unnecessarily
1 pt.	Design issues (including bringing a design to lab)
	Correctness (10 pts total)
2 pts.	Generates a new color swatch only if previous one placed correctly
2 pts.	Swatch displayed in the correct initial position; returns to original location if incorrectly sorted
2 pts.	Updates number correct and incorrect properly
2 pts.	Drags swatch properly (-1 pt if can use clicking instead of dragging)
2 pts.	Appropriate behavior if user does unexpected things like starting to drag outside the swatch
	Extra credit (1 pt)
1 pt.	Does not update either # correct or # incorrect if user misses all baskets

Final remarks

Be sure to do the basic version of the lab before attempting the more advanced features; this kind of incremental design/coding is critical for more complex programs. Just work on adding one feature at a time, and make sure to test each thoroughly before working on the next.

Turning in your program Your program is due at 11p.m. on Monday evening. Turn in your program as you did last week. Here are the instructions again:

- First, return to Eclipse and make sure you included your name and the course number in a comment at the start of the program.
- Next, click on the "Laundry" project 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 (you may need to click on the triangle next to "General" in the dialaog box first).
- Make sure all the files in the dialog on the right hand side are checked, and then click the "Browse" button next to the "To directory:" entry.
- Select "Desktop" from the pulldown menu and click on "Choose".
- This should create a new folder called "Laundry" on the desktop. You should now rename that folder by clicking on it and pressing return, and typing in a more descriptive name (one that identifies you and the lab you are working on, such as "Kauchak-Lab2" – except replace "Kauchak" by your name!). Note that dashes in names are OK, but don't include spaces or periods.
- Quit Eclipse.
- Now open the "cs051" folder icon on the desktop by double-clicking on it. Within the "cs051" folder you should see a "dropbox" folder.
- Drag the folder you just created into the dropbox folder. When you do this, the computer may warn you that you will not be able to look at this folder. That is fine. Just click "OK".
- The new folder will still show up in your home folder. Drag it into the trash both to save space and to keep anyone from copying it. (Files in your CS51Workspace folder are protected so that others cannot read them.) Press the right mouse button on the trash icon to bring up a menu and select "empty trash".

If you should accidentally turn in a bad version of your program, you may drag another copy in as long as you change the name to be slightly different from the one you used before (e.g. "Doe-Lab2-version2"). The new name should make it clear which is the newer version.

Good luck and have fun!