Programming Exercise:
Dirty Laundry

Objective: To gain experience using conditionals.

The Scenario. One thing some students have to figure out for the first time when they come to college is how to wash their own clothes. By the time many students have figured it out, all their underwear is pink and 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*.

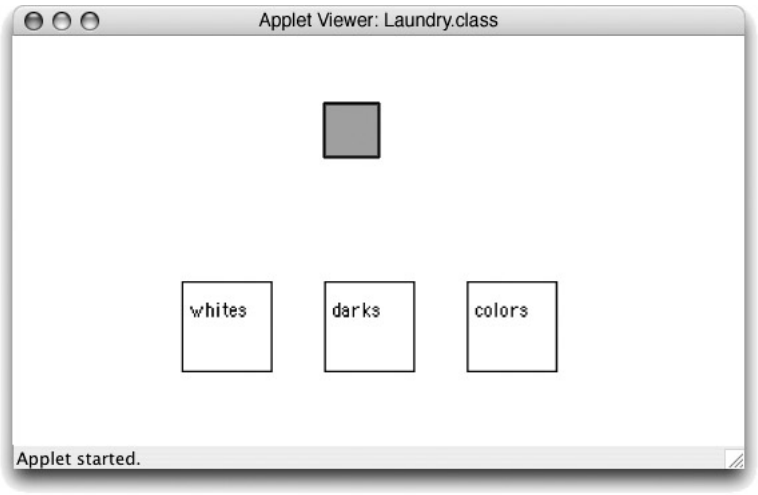
The Approach. It is usually a good practice to develop programs incrementally. Start with a simplified version of the full problem. Plan, implement, and test a program for this simplified version. Then, add more functionality until you have solved the full problem.

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 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.

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 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.

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 make it clearer whether the user succeeded.

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 some graph paper. (We recommend using a window that has height and width of 400 pixels.) The picture should look more or less like the one above.

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 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). 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`s. 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).*

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 initial display, it would be good to run it to see if it does what you expect.

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 the new 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).

Recycling Because your program only uses one laundry item at a time, you might as well just recycle it – reusing the same rectangle for each laundry item. Simply change its color rather than creating a new rectangle. In general it is a good strategy to reuse objects rather than creating new ones when possible, as this generally uses less time and does not clutter memory.

Picking a random color To pick a random color, you should use the `RandomIntGenerator` class as described in the textbook (Section 3.9). Since there are 3 colors to select from, you should create your random number generator to return random numbers in the range of 1 to 3 and associate each of those values with one of the colors the laundry may have (`Color.white`, `Color.black`, and `Color.red`).

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 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. 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. Note that for the second enhancement 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.

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 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.

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.

Good luck and have fun!