



CS 315

Software Design

Homework 1

First Sip of Java

Due: Sept. 10, 11:30 PM

Objectives

The objectives of this assignment are:

- to get your first experience with Java
- to become familiar with Eclipse

Java Background

One of the benefits of working with Java is that every installation of Java comes with a large set of standard libraries. A Java application written for one operating system should work unchanged on any other operating system. In this lab, you will gain familiarity with some classes that Java provides for drawing.

In general, when you want to find out information about how to use a specific Java class or method, you should consult the online Java API available at <http://java.sun.com/javase/6/docs/api/>. For this lab, I have tried to give you all the information that you will need about the classes, but you will definitely want to have this URL bookmarked for the future!

Drawing

A `JComponent` is a Java class that allows you to layout images, text and shapes at precise pixel locations. The Hello from Venus example from class is an example of drawing functionality. Notice how the `HelloFromVenus` class "extends `JComponent`". The `paintComponent` method contains the code that defines the appearance of the component. The `main` method creates the `JFrame` and the `JComponent` and makes them visible.

The methods that you will use to draw are these:

```
public void drawImage(Image img, int x, int y, ImageObserver observer);
public void drawLine (int x1, int y1, int x2, int y2);
public void drawOval (int x, int y, int width, int height);
public void drawRect (int x, int y, int width, int height);
public void drawString (String s, int x, int y);
```

```
public void fillOval (int x, int y, int width, int height);  
public void fillRect (int x, int y, int width, int height);
```

`drawImage` takes an `Image` as its first parameter. You get an image by calling:

```
Toolkit.getDefaultToolkit().getImage("blerner.jpg")
```

passing the name of the file containing the image as the argument to `getImage`. The final argument to `drawImage` should be `"this"` (without the quotes).

The arguments to `drawLine` are the `x` and `y` values of the 2 endpoints of the line.

The `x` & `y` arguments to the remaining methods identify the upper left corner. For an oval, the upper left corner is the upper left corner of the oval's bounding box, that is the smallest rectangle that encloses the oval. The `"draw"` methods give a border with the background showing, while the `"fill"` methods gives the same shape but with a solid color, blocking the background.

When drawing shapes or text, you must set the color and font prior to doing the drawing. You use the methods:

```
public void setFont (Font f);  
public void setColor (Color c);
```

You can define a font by calling the font constructor:

```
Font (font-name, font-face, size)
```

such as

```
new Font ("Helvetica", Font.BOLD, 18)
```

Colors are defined using RGB (Red-Green-Blue). Each value can range from 0 to 255, where 0 means none and 255 means maximal. RGB 0, 0, 0 is black, while 255, 255, 255 is white. Tools that allow color selection, like word processors, Photoshop, etc. generally provide a way for you to play with colors and then determine what RGB value a color has. You define a color by calling the `Color` constructor:

```
Color (int red, int green, int blue)
```

Please refer to the example from class and the Java online API for more details on how to draw.

Displaying a JComponent

Using the information above, you will be able to draw on a `JComponent`, but it still won't appear on the display! To accomplish that, you need to add the `JComponent` to a window and display the window. This section will tell you how to do that.

The type for window in Java is `JFrame`. So, the first thing you will need to do is construct a `JFrame`. Its constructor takes no arguments. There are several important methods that you need to know for `JFrame`:

```
public Container getContentPane()
```

Returns the top-level container. Ultimately, to add something to a window, it must be added to the content pane of the window, like this:

```
frame.getContentPane().add(component);
```

```
public void setSize(Dimension d)
```

Size the window to a particular size giving the desired width and height in pixels. For example, you could use this as:

```
frame.setSize (new Dimension (300, 400));
```

to create a window that is 300 pixels wide and 400 pixels tall.

```
public void setVisible (boolean visible)
```

If `true` is passed in, the window becomes visible. If `false` is passed in, the window is removed from the screen.

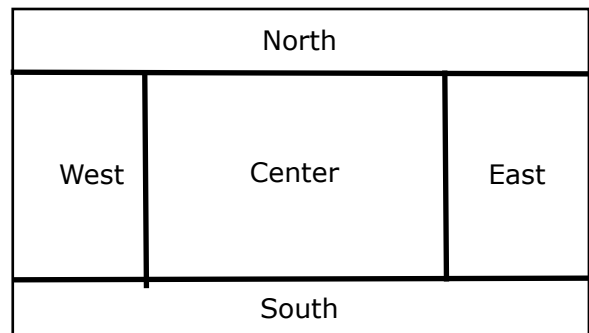
Associated with containers, like the `JFrame`'s content pane, is a layout manager. In Java, you do not specifically provide pixel coordinates when adding a component to the content pane. Instead, you select a layout manager and it determines how to lay out the components. In this lab, you will use the `BorderLayout` layout manager.

`BorderLayout` divides the screen into 5 areas:

To add a component named `c` to a container named `panel` that uses a border layout, you specify which section of the layout to use:

```
panel.add (c, BorderLayout.CENTER);
```

If an area of the screen remains empty, its size shrinks to nothing. The Center area is generally the largest. Each area can contain only one component or container.

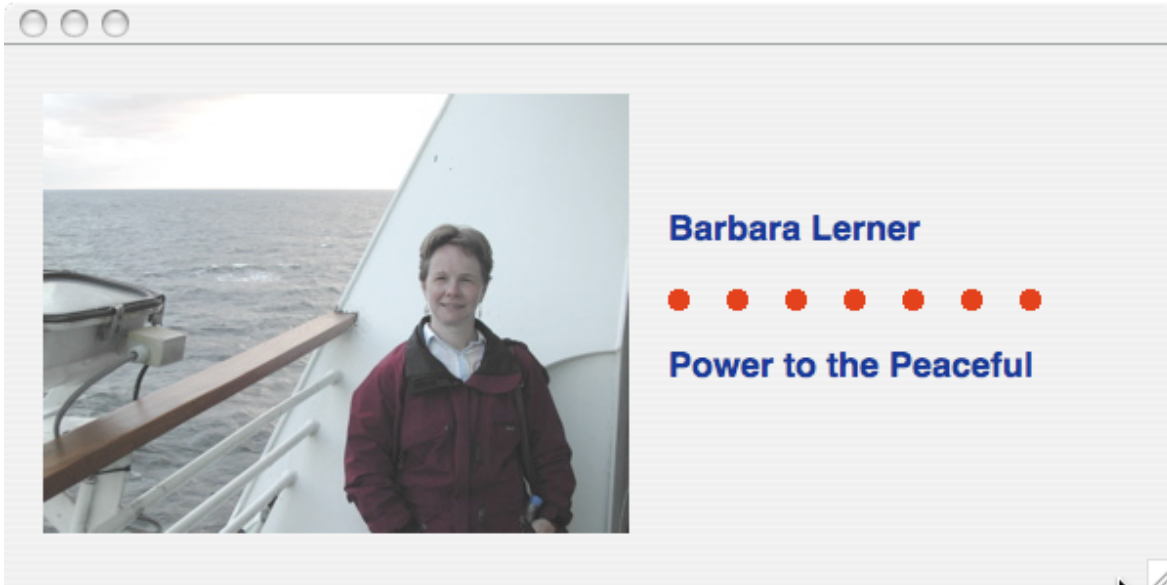


By default, a `JFrame`'s content pane uses `BorderLayout`. So, you will need to add the `JComponent` you create to the `CENTER` of the `JFrame`'s content pane. (Yes, it sounds confusing, but it's really just a few lines of code that you will find yourself using a lot over the course of the semester.)

Assignment

Wow! Now that we've gotten through all the background information, the description of the actual assignment can be quite brief but relies heavily on the background material.

Your program should provide an ID card display that looks as follows:



This program contains a panel similar to the Venus panel of the example from class.

You do not need to follow this exact layout but your program should exhibit the following characteristics:

- It should include an image
- It should include some text
- It should include some shapes (ovals, rectangles and lines are the basic drawing capabilities that Java provides).
- It should use a few colors.
- The window should be the right size to comfortably display the information without seeming too large.

Eclipse

We will use Eclipse in this class. Eclipse is available on all major platforms: Linux, Windows and Mac. You can download Eclipse for your own personal computer for free from <http://www.eclipse.org>. Eclipse has become one of the most widely used software development environments in recent years for Java programming.

Eclipse maintains all your code in a "workspace". The workspace contains a collection of projects, where each project typically corresponds to a single program. When you first start Eclipse, you need to select the arrow icon on the right middle side of the window that says "Go to Workbench". It will then ask you where it should create the workspace. Be sure to select a location in your account on the file server. Also, be sure to create a folder for the workspace, called "workspace" or "cs315workspace" or something similar.

The Eclipse window has a package explorer panel on the left, a panel where error messages (and other informational messages) will be displayed at the bottom. The main area on the top right is where your code will appear. There is a lot of functionality to Eclipse, so it may take some time to master it, but the basics are mostly intuitive.

Next, create a project called IDCard. To do this, select the icon in the row of icons that looks like an open file folder with a J in it. Enter IDCard for the project name and click Finish.

For this program, you will need a single class. You might as well call that IDCard. To create a class, click on the icon in the top toolbar that is a C in a green circle. Make sure the source folder is IDCard. Enter IDCard for the name and click Finish. This will create the stub for a new class named IDCard. As with the example in class, you will just need to define the `main` method and the `paintComponent` method. Please refer to that example for ideas on how to do that.

One feature of Eclipse is that it checks your code as you enter it. When it finds an error, an icon will appear in the left margin of the code area. The icon will either be a red X or a red X with a light bulb. In either case, an error message will appear in the bottom panel. If a light bulb is present and you single-click on the light bulb, Eclipse will offer some suggestions on how to correct the error. Often it has the solution in its list; you can then double-click on the solution to have Eclipse perform the necessary edits for you.

When you have no red X's, you can run your program. To do this, right-click on the main class in the project explorer and select Run As...Java Application. After doing that the first time, you can then click on the green play arrow in the palette running across the top to run the program again.

You will also want to import an image into your Eclipse project that your program will use. To do this, right-click on the project name in the Project Explorer and select Import. In the window that comes up, select FileSystem from the General list and click Next. Click the Browse button and find and select the folder containing the image file you want to use. This folder name should now appear in the left column in the Import window. Click on the folder name (**not** on the checkbox to the left of the name). Now the right column should list the files in that folder. Find the file you want and select its checkbox. Then click Finish.

As the semester progresses, we will explore some of the more interesting features of Eclipse, but this should be enough to get you started.

Grading

Style is very important in software development. Good style and good documentation greatly increases the maintainability of software. As a result, all the good things you have learned about documentation, choosing variable, method, and class names, use of private variables, use of named constants rather than literals, consistent indentation, etc. are important and will be graded.

Comments	25 points
Style	25 points
Including an image	10 points
Including text	10 points
Including some shapes	10 points

Using colors	10 points
Making the window a good size	10 points

Turning in your work

To turn in your work, create a jar file that contains your source code. From Eclipse, you can create a jar file as follows. Select your project. Open the File menu and select Export. Select Jar file and click Next. Check the box that says export Java source files. Select a name and destination for the jar file. Then click Finish. Submit your assignment using Ella.