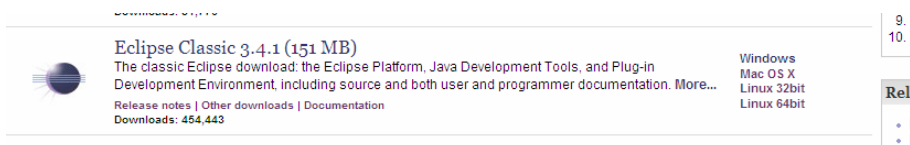


Lab 9 – posted Nov. 2, due Nov. 14

Part 1: OPTIONAL – use Eclipse to edit and run your program, and submit a JAR file. Worth an extra 10 points. If you're not going for the extra 10 points, skip to Part 2...

If you're in the lab, start Eclipse by typing "eclipse &" at the prompt. You might have to specify `/util/eclipse/eclipse &`

If you're at home, download Eclipse here: <http://www.eclipse.org/downloads/>
choose Eclipse Classic and pick the U.B. mirror site.



Beginning Your Setup of Eclipse (thanks to Adrienne Decker for these instructions)

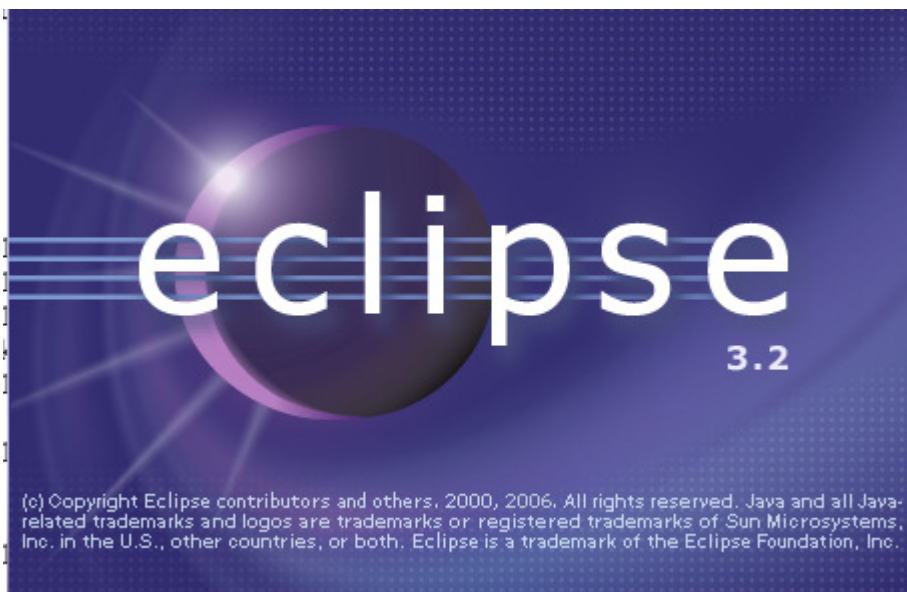
Create a directory in your home directory named `cse115`.

Change directories into that directory (using `cd`).

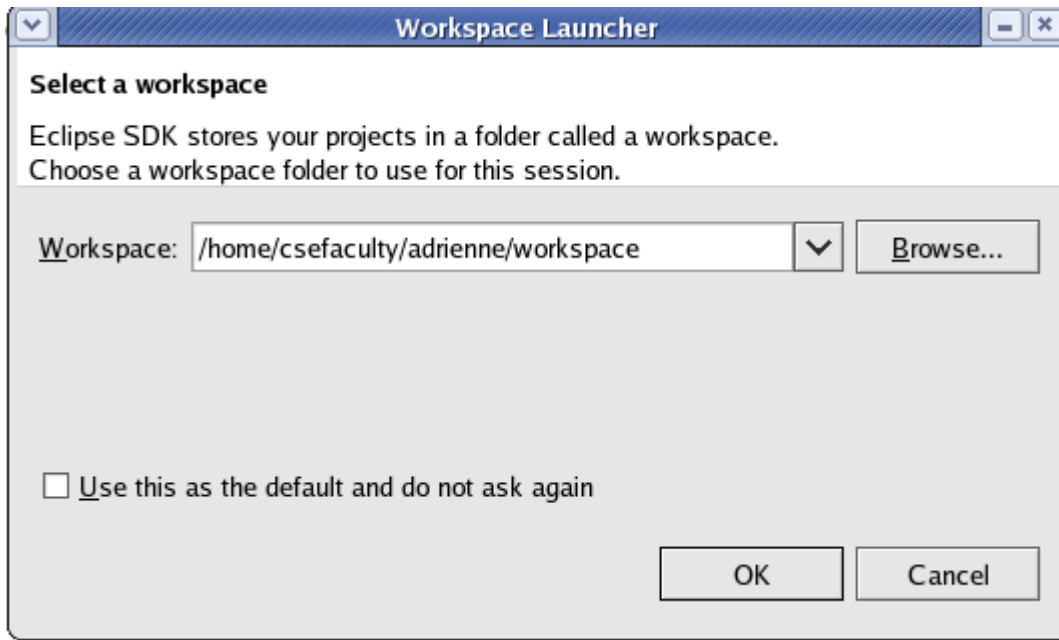
Make a directory named `workspace`.

Now you need to open Eclipse. There are a few ways in which you can do this. You can type the command to launch the program into the prompt. That command is `/util/eclipse/eclipse &` However, you can also simply go to Applications → Programming → Eclipse and it will start up Eclipse for you.

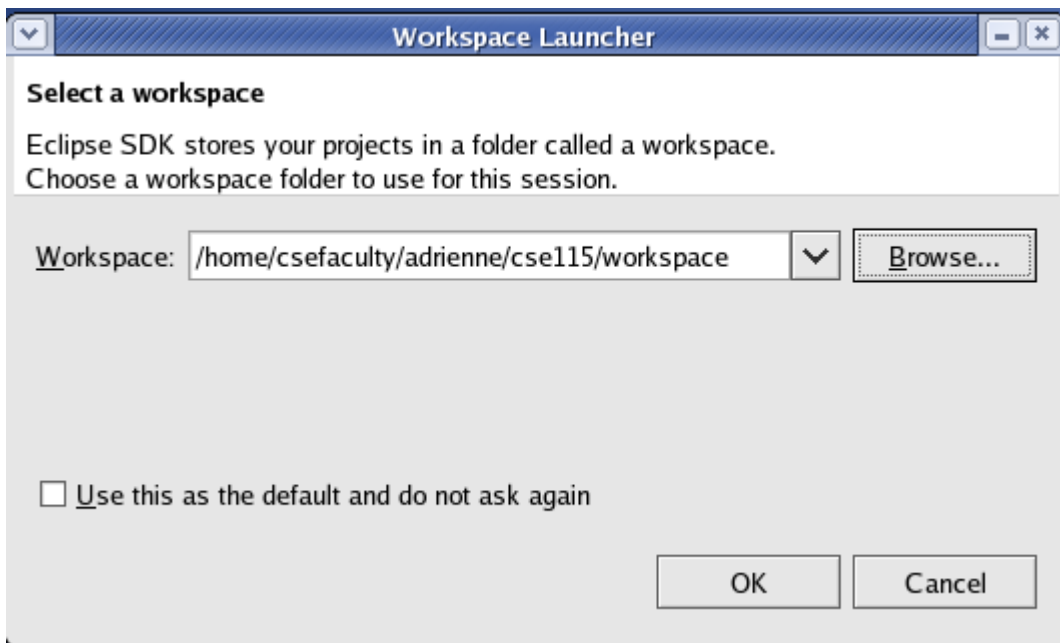
When you start up Eclipse, you will see the following screen:



Then you will see a Window that will ask you which workspace you want to open. By default, it will probably say something like this:

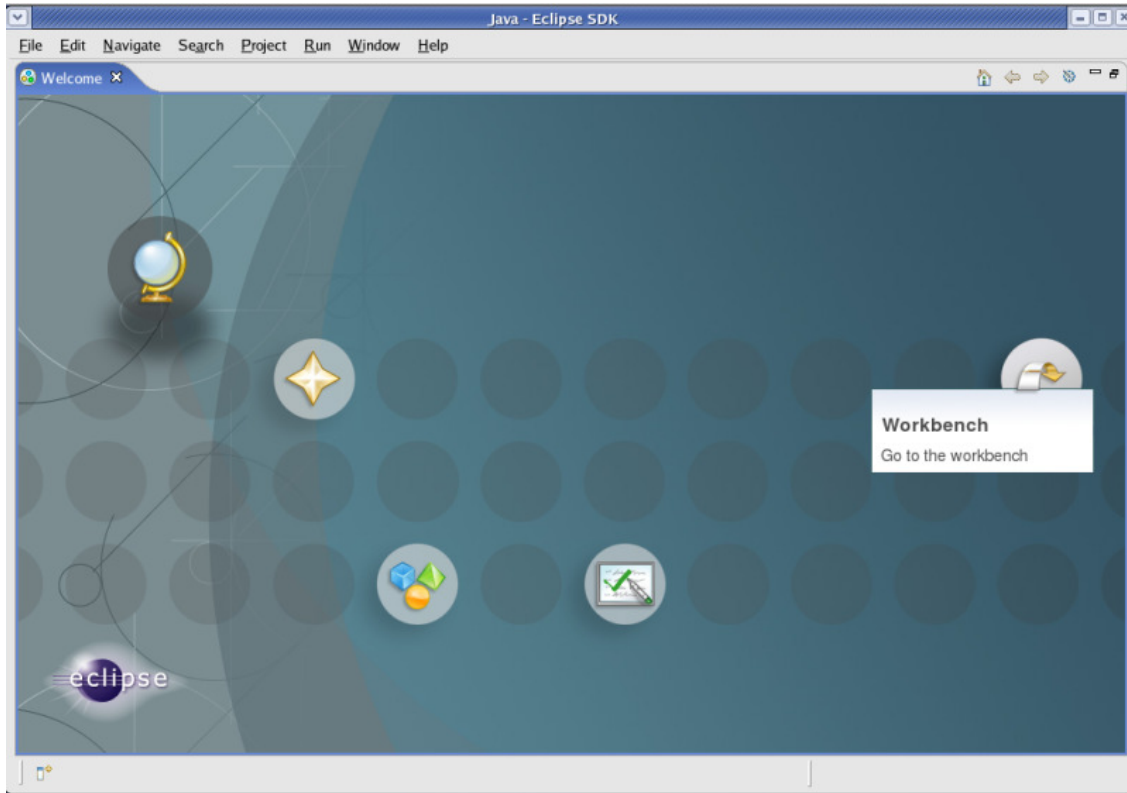


You do not want to use this default workspace. Use the Browse button to make the workspace point to the directory you just made named workspace, like this:

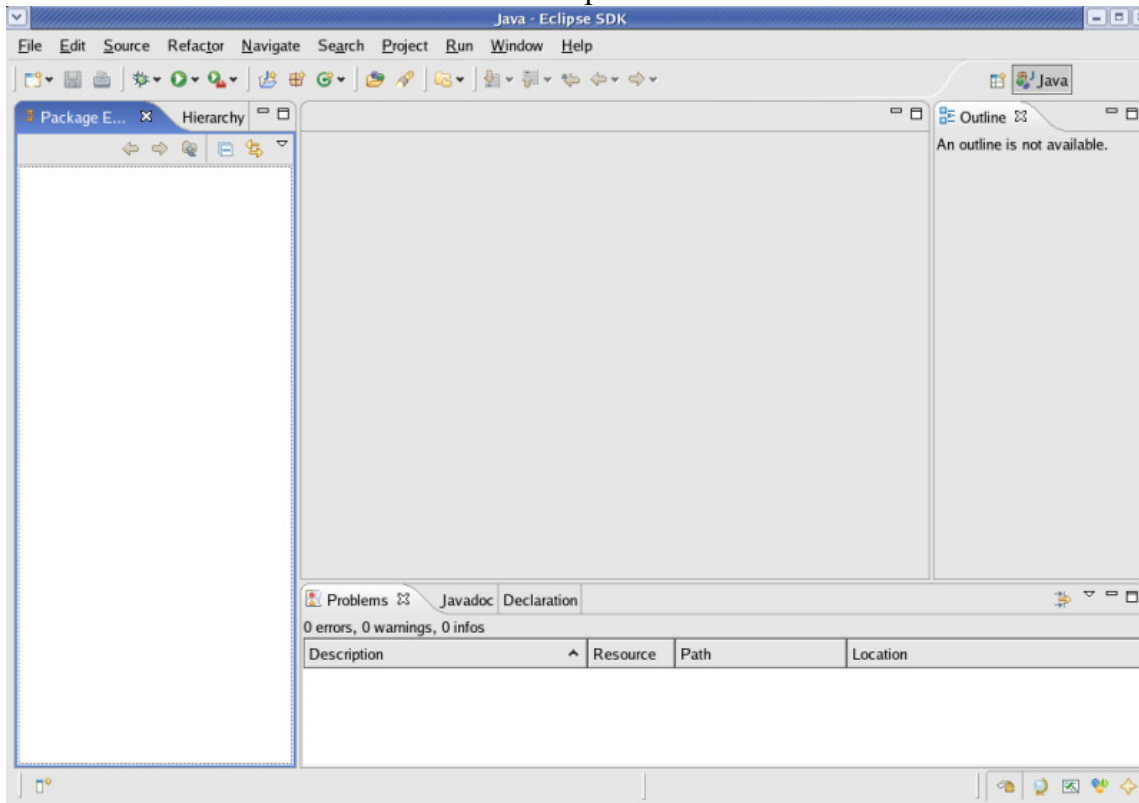


Be sure to check the box "Use as default and don't ask me again", and then click "OK".

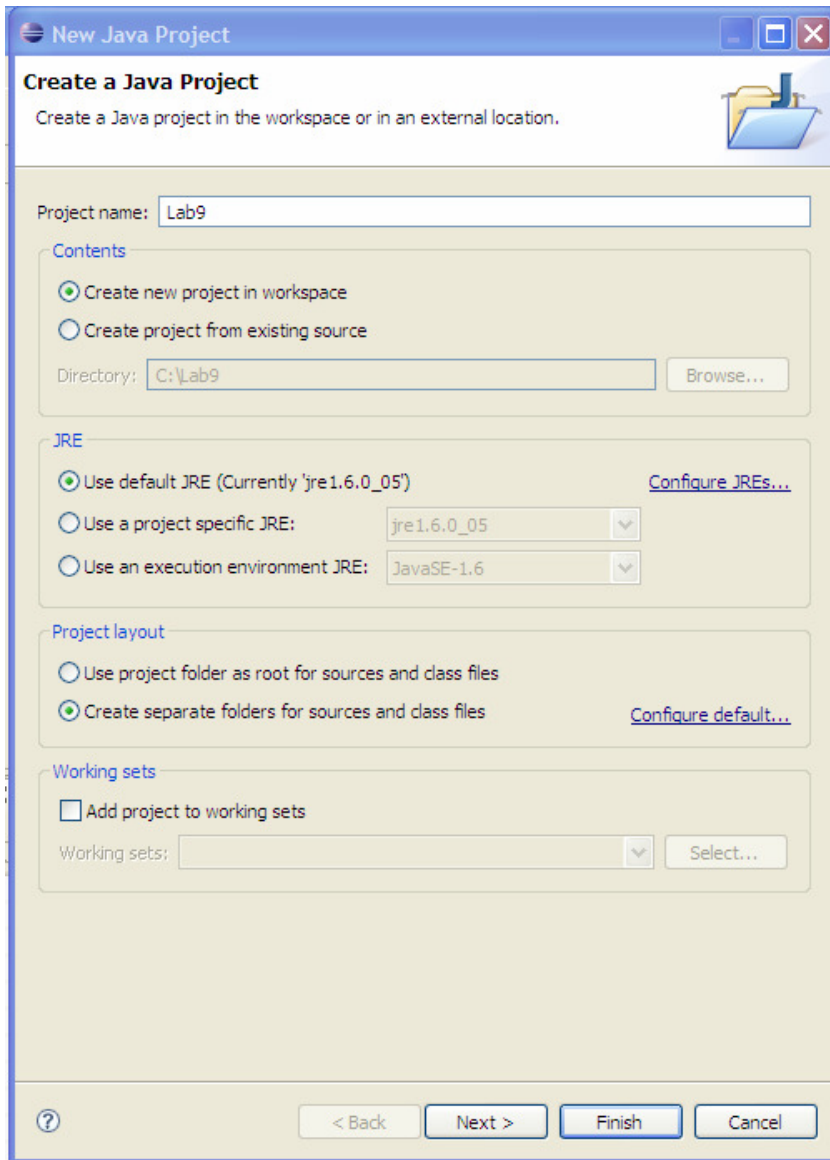
You will then see a screen that looks like this. You will choose the arrow to take you to the Eclipse workbench.



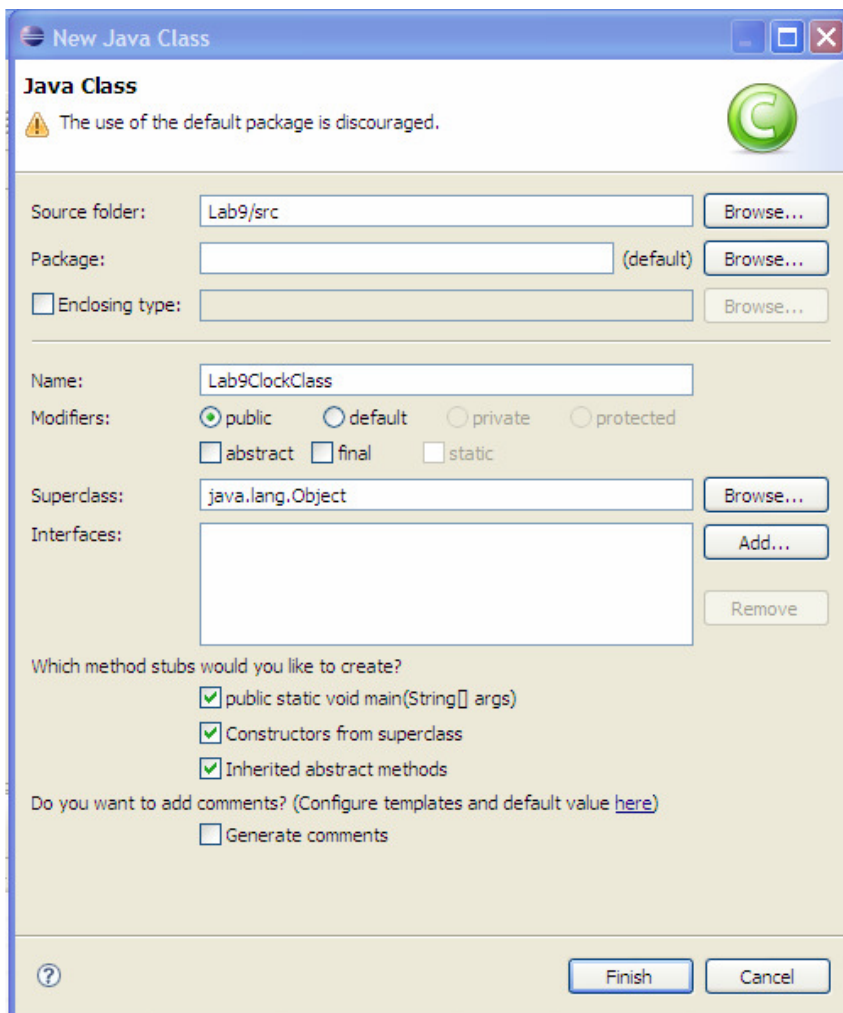
You will then see the default workbench setup.



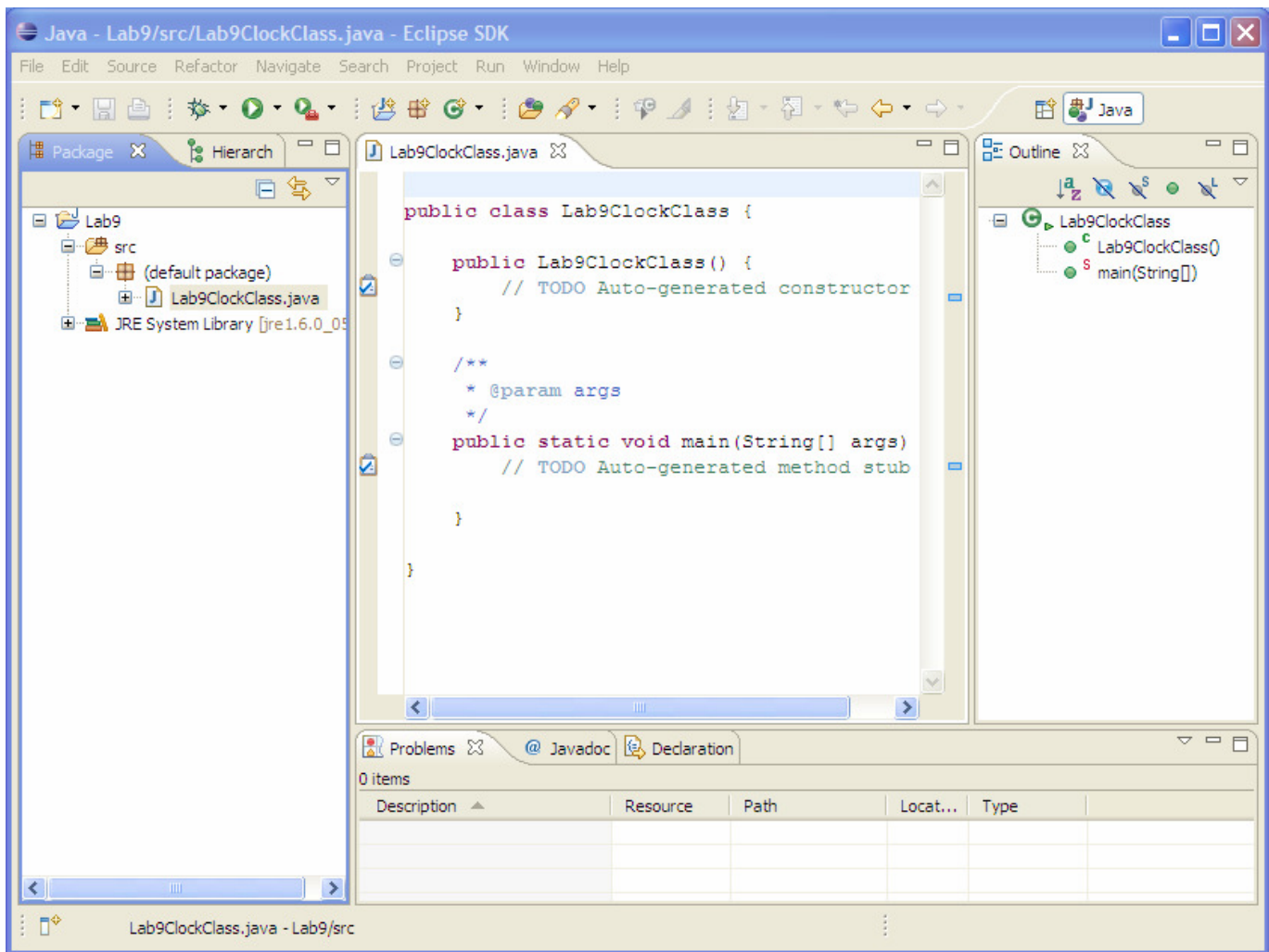
Unlike JGrasp, Eclipse never works outside of a "Project", so select New→Project→Java Project. Name the project Lab9. Hit "Finish" (not Next).



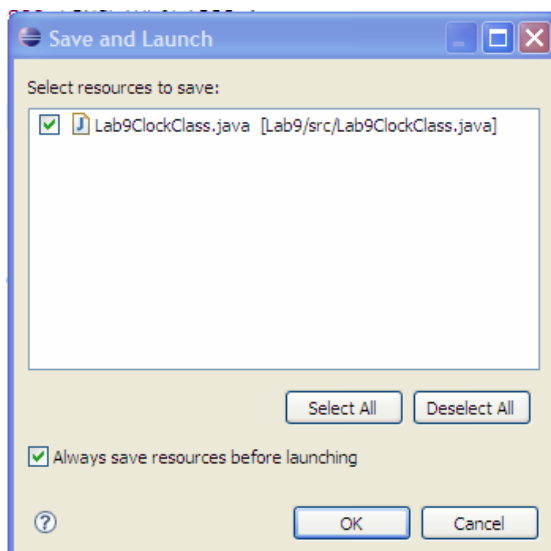
Now, you need to get Eclipse to give you a file to edit. Select File→New→Class, and enter the Class name as Lab9ClockClass. If you select "public static void main..." and "Constructors from superclass", it will create those brackets in the file for you.



and here's your edit window:

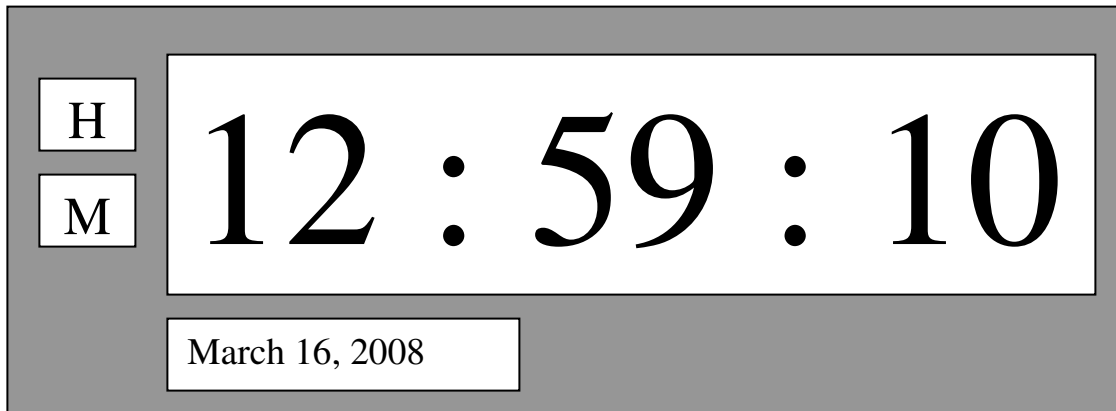


You can edit and run your file just like in JGrasp, although every once in awhile you'll see this, to save your files:



Part 2: MANDATORY – Make a working time & date clock

Make something that looks like this (*variations are acceptable and encouraged*):



Use the Timer class to count exact seconds. You'll need separate Hours, Minutes, and Seconds integer variables. It must operate like a real clock. Minutes and seconds roll-over at 59. Increment Minutes when Seconds pass 59, increment Hours when Minutes pass 59.

To set your clock, you'll need an "Hours" button to increment the hours and a "Minutes" button to increment the minutes. Make sure the clock is big, bright, and visible.

Given the Timer, Date/DateFormat, JButton, and JLabel classes, you have all of the working tools.

Use the Date class with the DateFormat helper class to get a date String. *It's a simple three-line procedure that is on the class web site in shortDate.java (in Hint 3 below).*

Hint 1:

You will need a class (call it Lab9ClockClass), that extends JFrame and implements ActionListener that has a constructor, a main, and an actionPerformed method. You can use the file ActionFramework.java on the web site as a start.

Hint 2:

```
import javax.swing.*; // for JFrame
import java.awt.*; // for Colors
import java.awt.event.*; // for ActionListener, ActionEvent, Timer
import java.util.*; // for Date
import java.text.*; // for DateFormat
```

Hint 3:

```
// establish a date
Date myDate = new Date();
// give the date to a special formatter, that returns a string
DateFormat dfL = DateFormat.getDateInstance();
String todaysDate = dfL.format( myDate );
```

Now you can use the String anywhere a String can be displayed, such as `yourJLabelObject.setText(todaysDate);`

Hint 4: FONTS!

Once you have a JLabel object, you can set its style like this:

```
yourJLabelObject.setFont(new Font("Serif", Font.BOLD, 48));  
yourJLabelObject.setForeground(Color.red);
```

Do this after you've instantiated the JLabel (obviously) and before you add it to the frame.

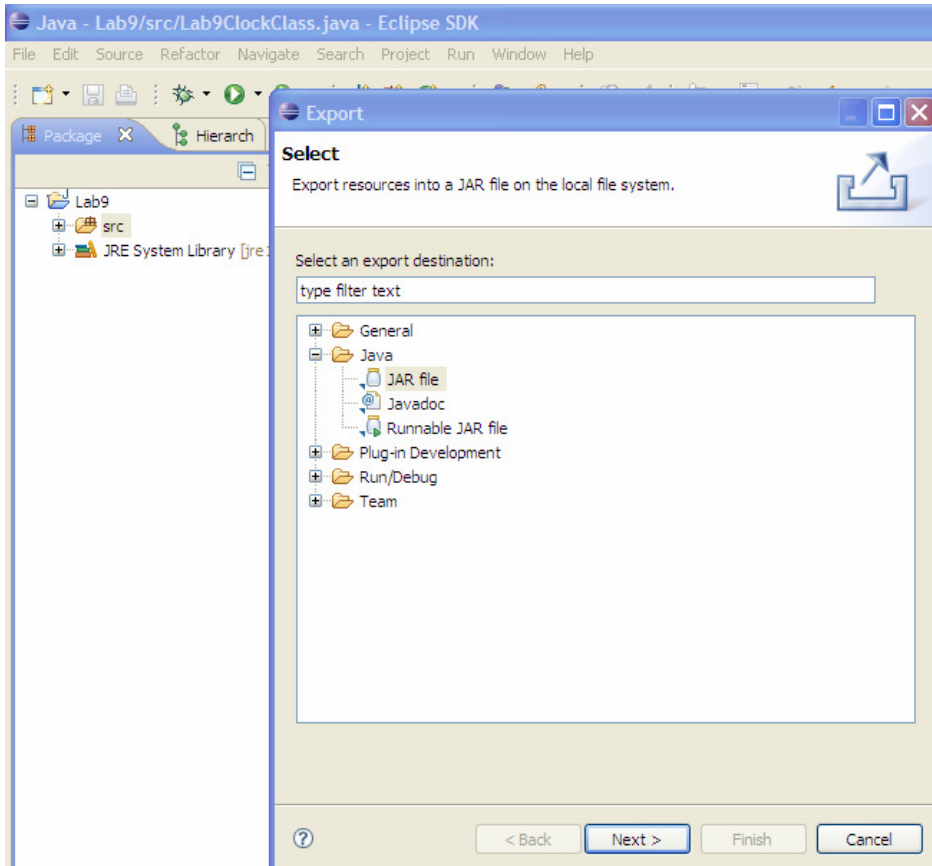
Hint 5: JPanels

You can group buttons together and give them their own small window on your frame. Essentially, you add JButton objects to a JPanel object (instead of adding the buttons to your frame) and then add the panel to your frame. Use the file *JPanelDemo.java* on the web site as an example.

If you're not going for the additional extra-credit 10 points, you can submit your java fikoe in the usual way.

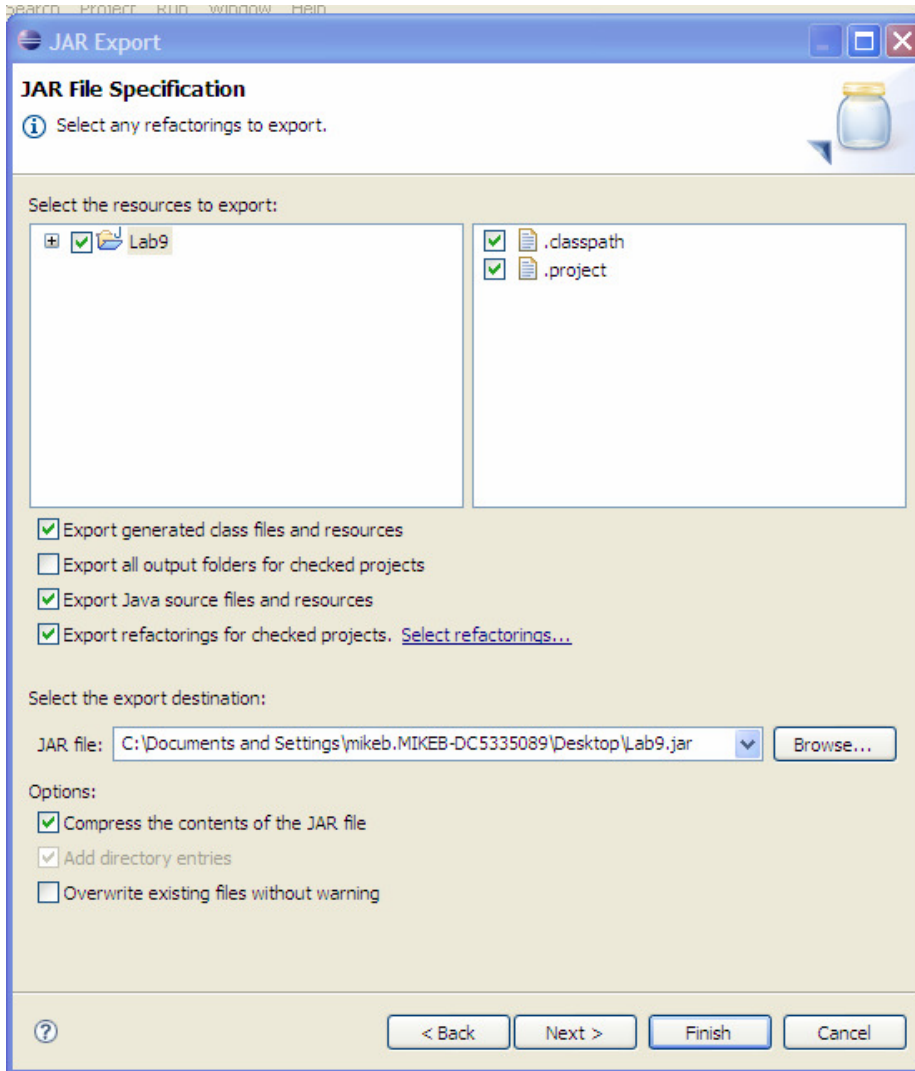
Part 3: (still optional) To get your 10 extra credit points, you need to submit the Eclipse JAR file. A JAR file is a compressed archive of your entire project.

Select File→Export, and get this menu:



Select JAR file and then Next.

Now this is VERY IMPORTANT. Check the boxes as indicated below, and specify a location for your JAR file. Then select Finish.



Submit the JAR file using the usual submit command.