Introduction to Borland's JBuilder
Mock

This document is a brief introduction to writing and compiling a program using Borland's JBuilder Integrated Development Environment (IDE). An IDE is a program that automates and makes easier many tasks that the programmer would otherwise have to perform herself. While many IDEs exist for Java, we will focus only on Borland's JBuilder because at the time of this writing it is free and contains many powerful features.

Installation

Previous editions of your textbook came with a CD that contained version 3.5 or version 4 of JBuilder. The current edition comes with Sun's Java One development environment which we will not be describing here since it is not available in the CS Lab. However, you are welcome to use the Sun environment if you like. The CS lab currently has installed JBuilder version 5. The latest for free personal use is version 8 although there is also version 9 professional. The screenshots shown in this tutorial are from JBuilder version 8. There may be some differences, but the general process is very similar even with version 3.5. We will not be using any of the advanced features of JBuilder here, as the emphasis of the course is on programming fundamentals, not on the intricacies of JBuilder.

Installing at Home

If you plan to install JBuilder on your own machine, your system should meet these minimum specifications:

233 Mhz or faster
256 Mb of RAM minimum, 512 Mb recommended
700 Mb of free disk space
Operating Systems: Windows 2000, NT, or XP; Mac OS 10.1; Linux

Installing at Home – JBuilder 8

You can download JBuilder 8 personal for free by visiting
http://www.borland.com/jbuilder/offers/

You will need to fill out a questionnaire so that Borland can send you email. You will also be emailed a license key that is needed to activate the software. For windows, you should download the file:

jb8_windows.zip (59.2 Mb)

It will be helpful to also download the built-in documentation. This is a separate download available from the URL:

http://info.borland.com/jbuilder/download/addons.html

For windows, you should download the file:

jb8docs.zip (66.2 Mb)

As you can see, these files are very large so you may wish to download them on a high-speed connection and save to zip disks or flash memory something if you only have a modem at home. If you require a copy on CD-ROM then let me know and I can burn one for you.

Using at the CS Lab – JBuilder 5.0

JBuilder 5 is installed on the Windows and Linux machines on campus in CAS 170. The personal edition is installed here. To use it, you must visit the web page at http://www.borland.com/jbuilder/offers/ and fill out the questionnaire for a personal key for JBuilder 5 (scroll down the page to the section for Keys only). You will be emailed the license key.

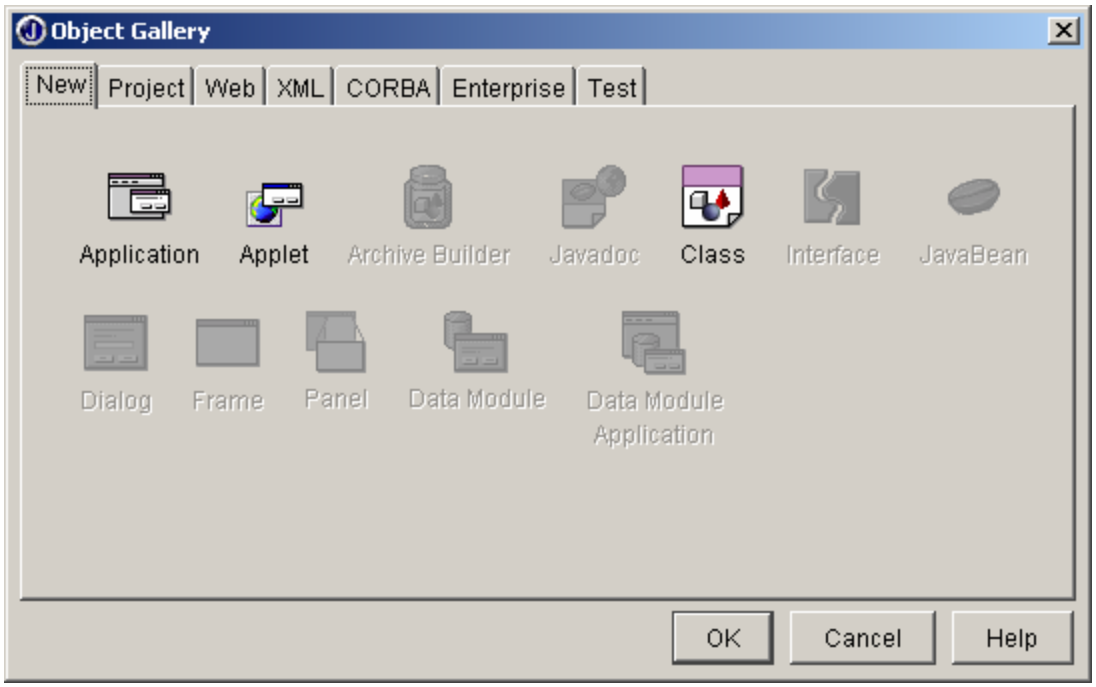
At this point you can start JBuilder on the lab machine. Once you start, JBuilder will prompt you to enter your personal key. You should only have to enter this once, at which point the information will be stored in your profile and should start up immediately the next time you run JBuilder.

Building a Sample Project – Hello World

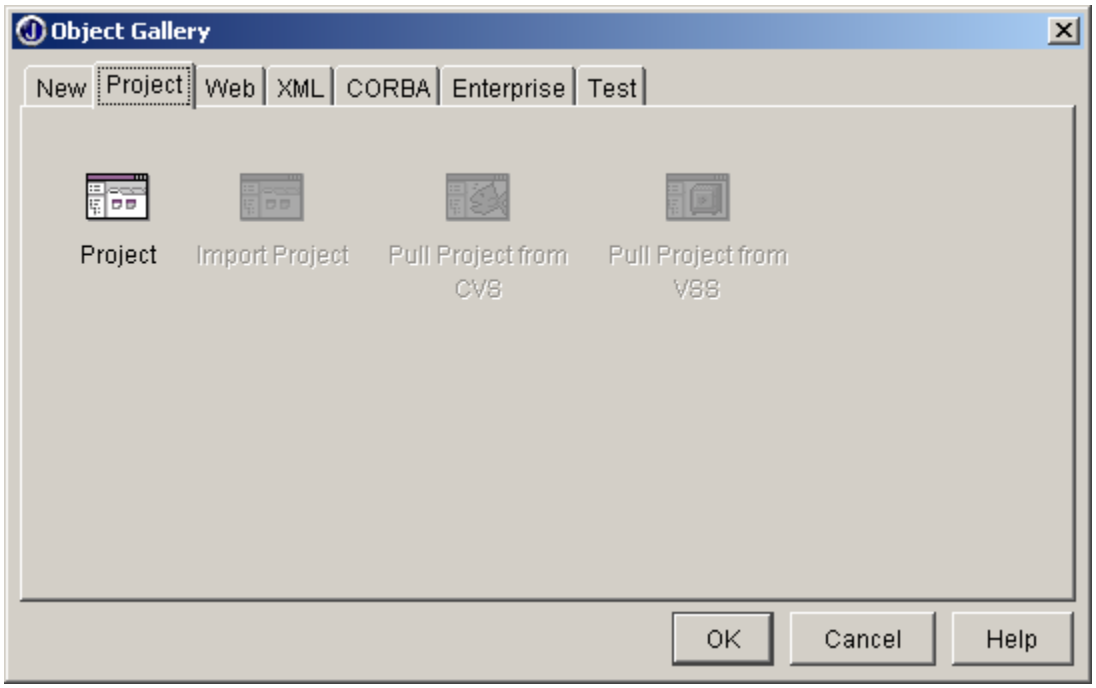
Let's start by creating in JBuilder the sample program we covered previously that printed out "Hello, world". The process will be similar when working on your own programs. Upon invoking JBuilder, by default you will be greeted with a "Welcome" project that gives you a tour of JBuilder. It is a good idea to take the tour sometime, but for now we would like to make our own project.

To make a new project, select F)ile, N)ew from the menu, or click the new document icon:

You will be presented with the new object dialog:



Select the “Project” tab, click on Project, and click “OK”:

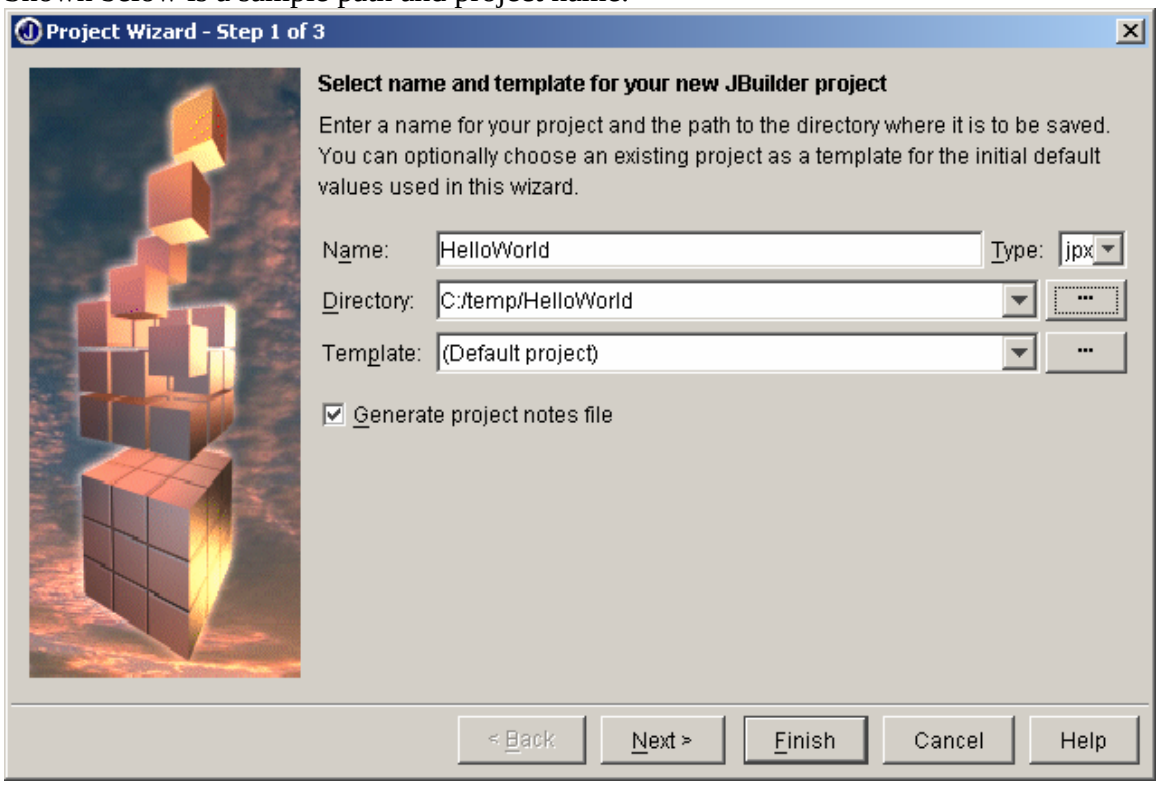


In the next dialog box, you will be prompted for a project name. Enter a name appropriate to this project, e.g. “HelloWorld”. You are also asked to select a path where the project should be stored. By default, you will likely see something on the C: drive in your “Documents and Settings” directory. I usually like to select a different directory but you are free to pick whatever you like. If you are using the machines in the CS lab, a

good directory to select is something on the H: drive since that is mapped to the network drive. This means you'll be able to access the same files from any machine you might log into (if you save the project on the C: drive, you won't be able to see them if you log into a different machine).

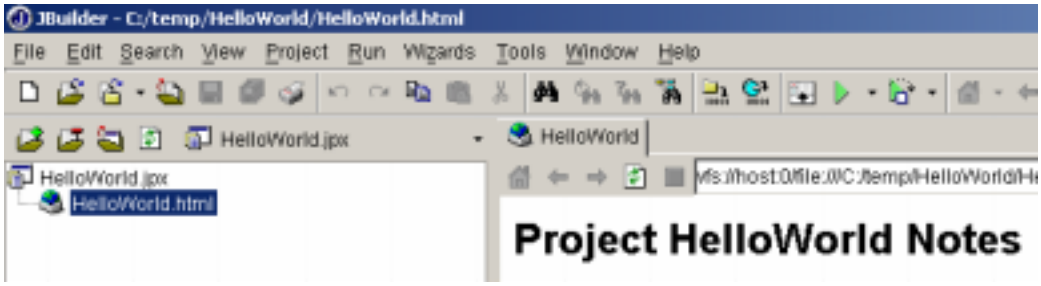
If you are using the CS lab and store files on the C: drive, please make sure to copy them to a floppy and delete them after you are done. The drives quickly become very messy if all the students leave various files hanging around.

Shown below is a sample path and project name:

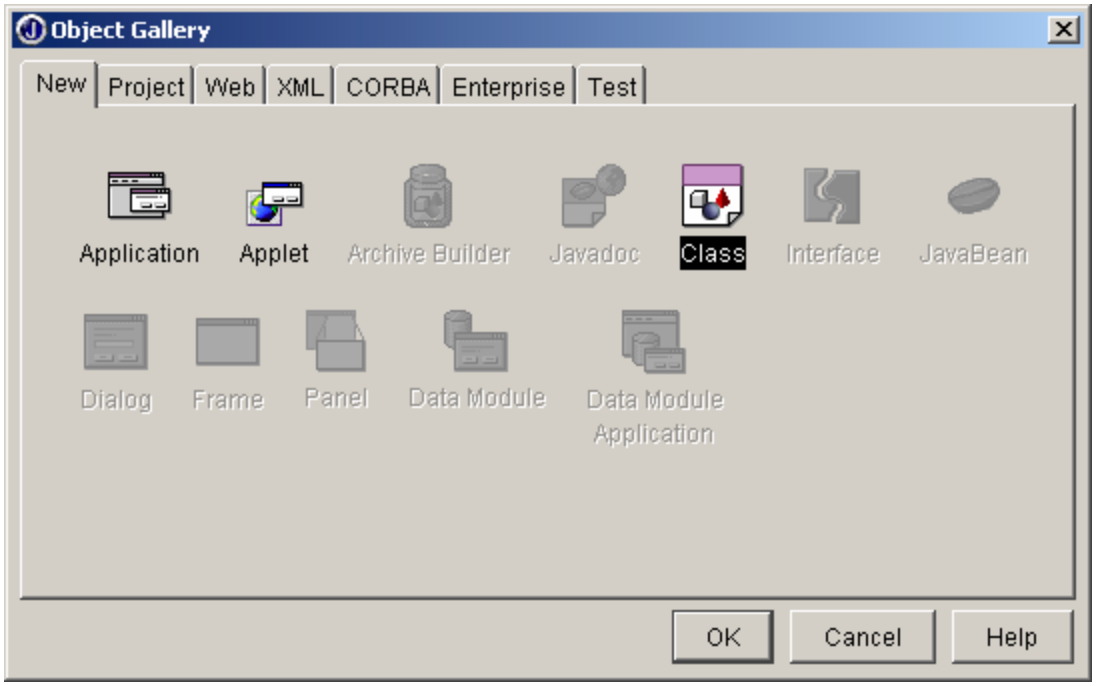


Click "Finish" to start writing your program. (Optionally, you could select "Next" to enter other file options).

JBuilder is set up so that the left hand pane shows what files and classes are available. The right hand pane shows source code. For now, the project is empty except for some project notes:



To start writing code, we must first create our classes. We can do this by going back to the Object Gallery by clicking on the "New" icon or selecting F)ile, N)ew from the menu. Make sure the "New" tab is selected, and click on "Class" and then click "OK":

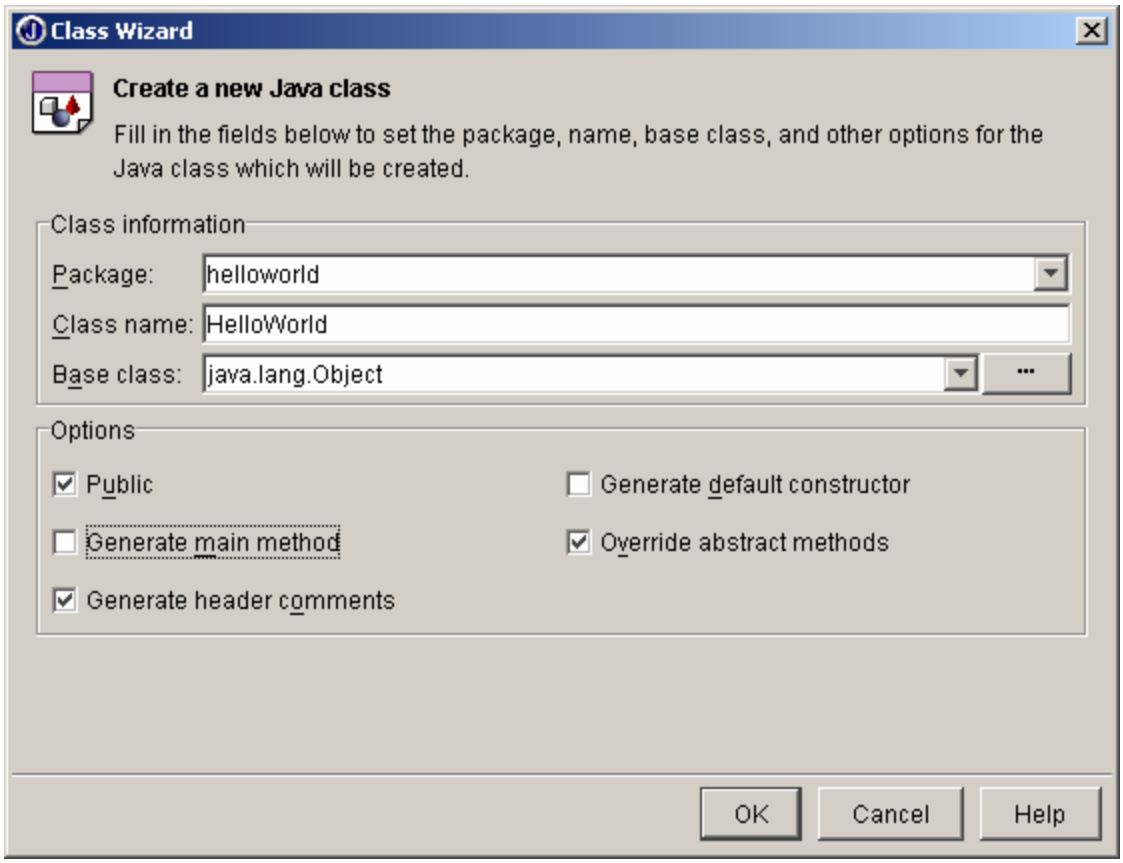


JBuilder will now ask us some information about the class. In particular, we will be asked for the package, class name, and base class. For now, just use the defaults except for class name.

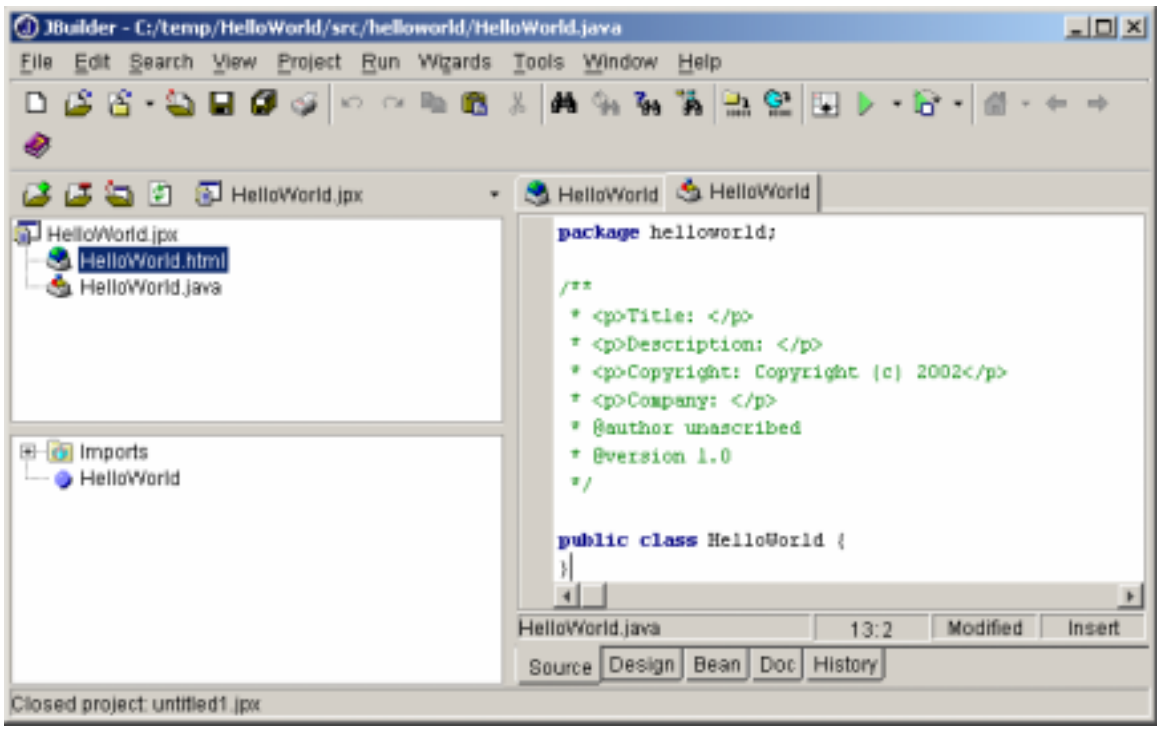
In our example, we created two classes, one named HelloWorld and the other named UseHello. Let's make the HelloWorld class first. The code for the HelloWorld class looked like the following:

```
// Put your name and date here
// followed by a description of the code
// This is a Hello World object that has a single
// method to print "hello world"
class HelloWorld {
      public void printHello() {
            System.out.println("hello, world");
      }
}
```

Note that there is no main method here, so I have de-selected the checkbox that automatically creates a main method. I have also de-selected the constructor box, we'll cover that later:



JBuilder will churn for a moment and then create an empty text file for you to begin entering code:



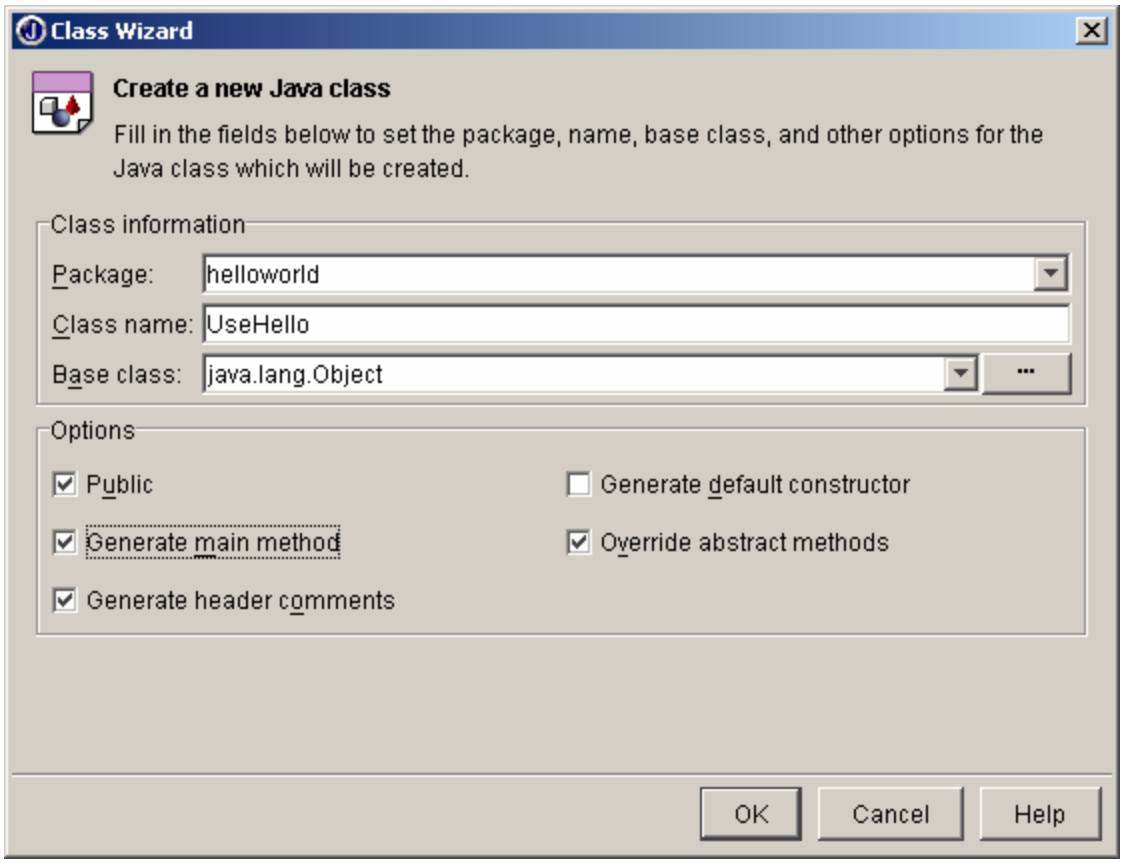
At this point, enter the code for HelloWorld into the text area on the right. Notice how JBuilder does nice things for you, like color-code the text depending on whether or not it is a comment, a special word, a string, etc.

```
package helloworld;

/**
 * <p>Hello World</p>
 * <p>Description: This program prints out 'Hello world' </p>
 * <p>Copyright: Copyright (c) 2002</p>
 * @author Kenrick Mock
 * @version 1.0
 * 5/11/2002
 */

public class HelloWorld {
        public void printHello() {
                System.out.println("hello, world");
        }
}
```

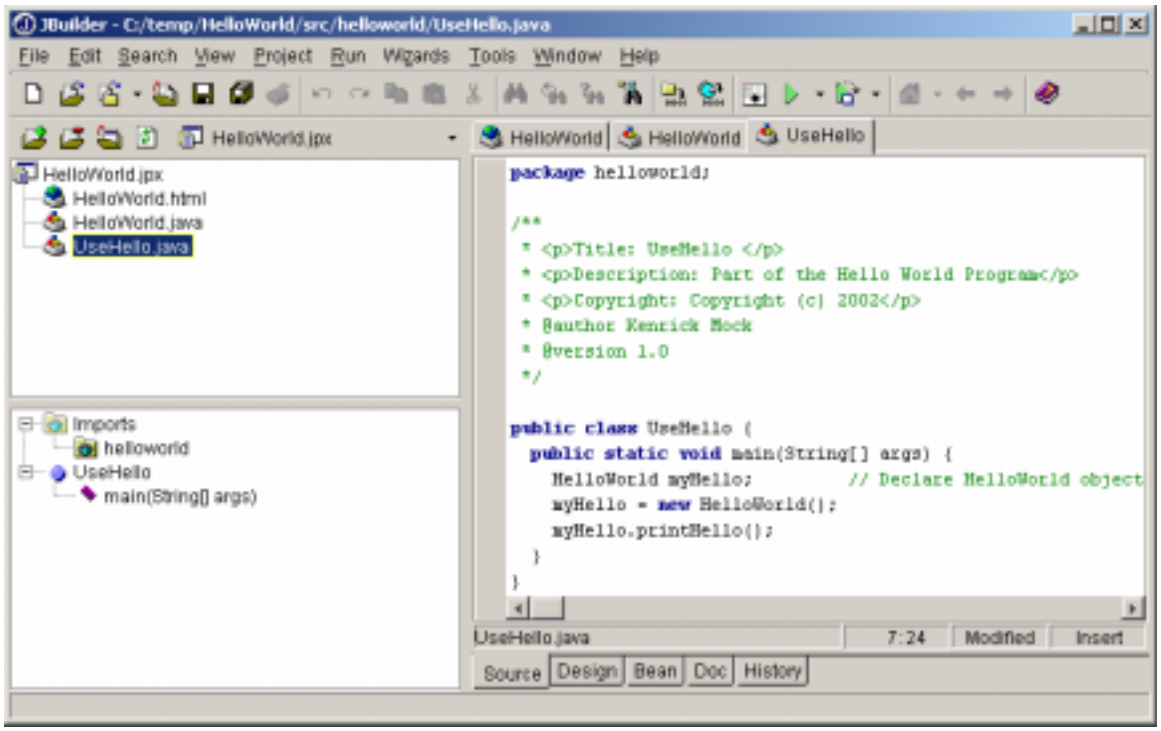
Next we repeat the process by adding a new class for UseHello. After clicking on the new document button, and selecting a new class, we enter the details for the UseHello class. Here is the original code for UseHello:

```
/* This file is the main object, when the program
   starts it beings executing in main
*/
class UseHello {
      public static void main(String[] args) {
            HelloWorld myHello;   // Declare HelloWorld object
            myHello = new HelloWorld();
            myHello.printHello();
      }
}
```

Since the main method exists in this class, I have selected the checkbox for “Generate Main Method” in the dialog below:



Now we enter the code for UseHello:



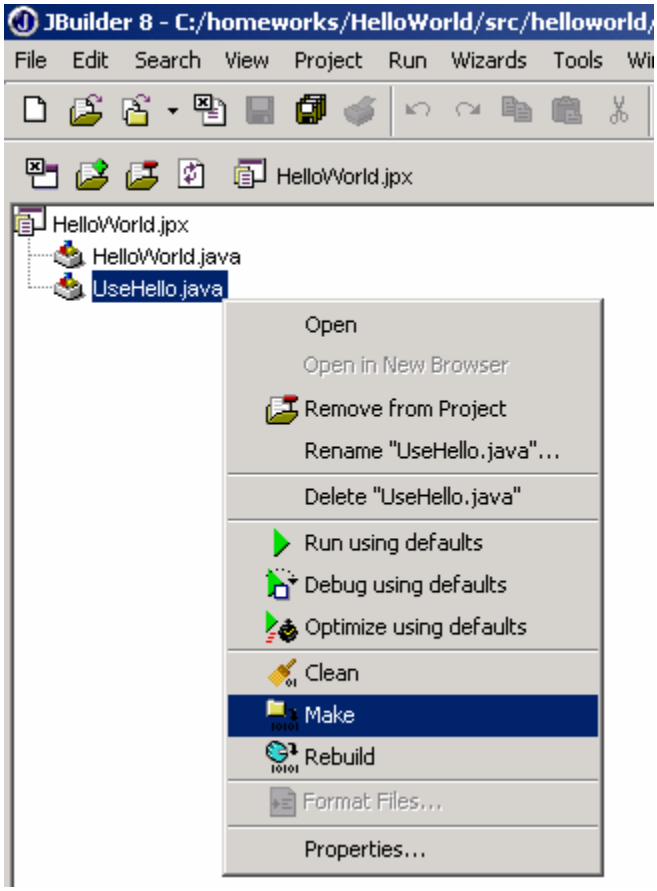
Note that we can click in the upper-left pane to select the source code we would like to see, either for HelloWorld.java or for UseHello.java.

At this point our program is complete. Before executing a project, I normally like to make sure everything is saved. To do this, click the stack of disks icon to save all files that have been modified since the last save.

Before running the program, we need to tell JBuilder where the main method lives. There are many different ways to do this. Below is described two different ways to actually compile and run the program:

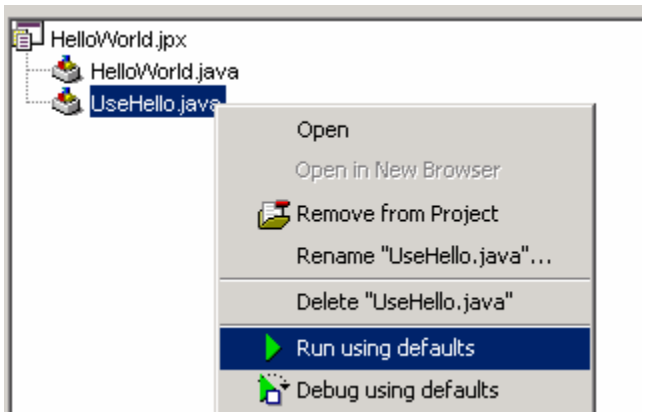
Compiling and Running Programs – Method 1

Perhaps the easiest way is to right-click on the name of the java file that contains "public static void main" in the Project Pane in the upper left corner. A pop-up window will appear. Select "Make":



This will compile the program. If the compiler found any errors, it would notify you at this time.

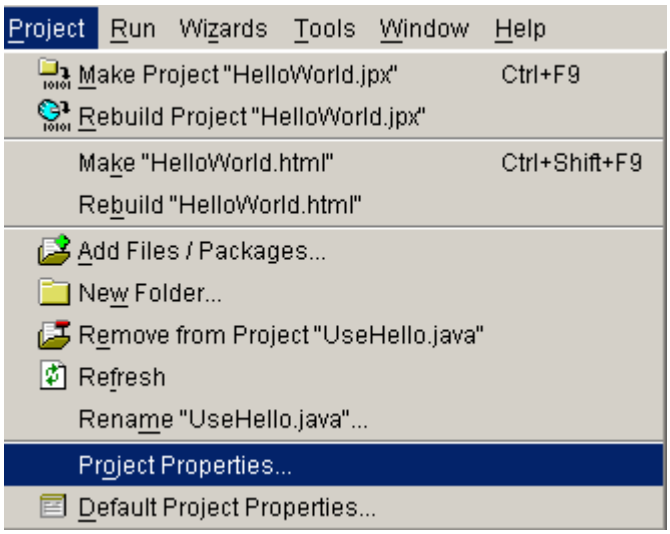
To run the program, right-click again on the file that contains “main”. This time select “Run using defaults” from the menu:



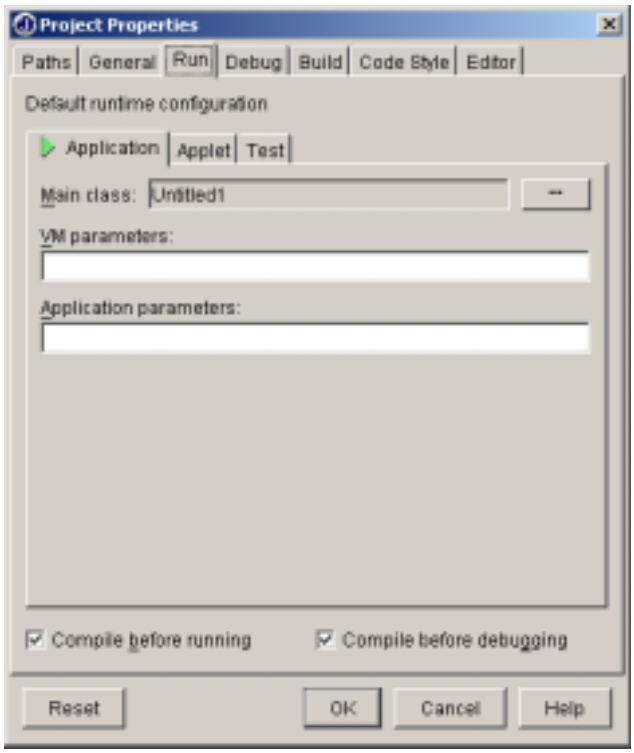
The program should now run!

Compiling and Running Programs – Method 2

In this method, you first tell JBuilder where your main method exists. To do this, select Project Properties from the P)roject menu:

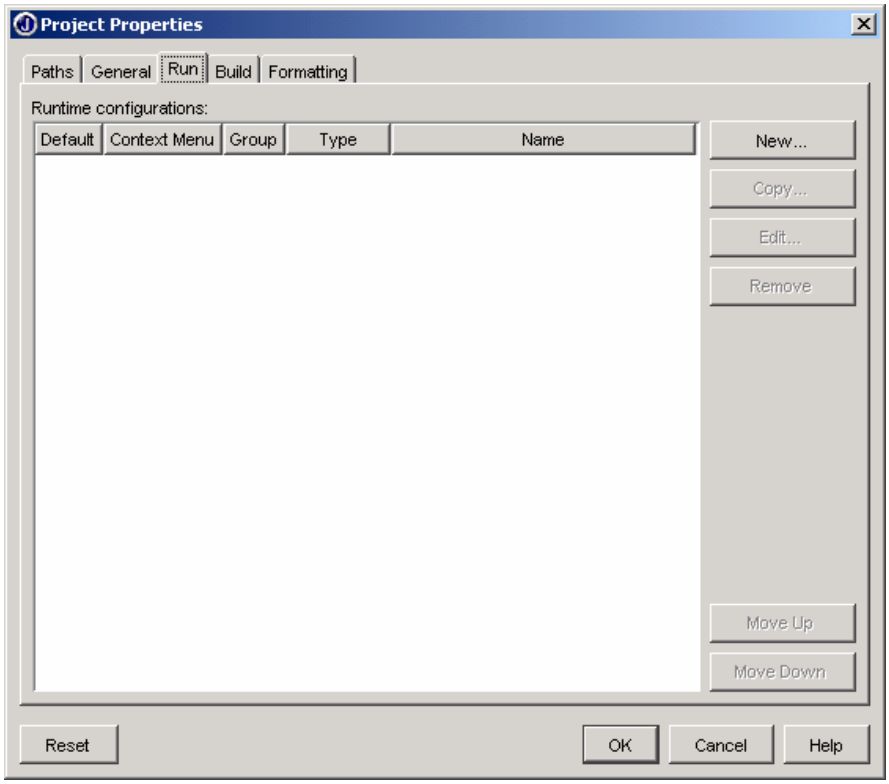


Next select the “Run” tab.
If you are using JBuilder version 5 in the lab then click the “…” next to the “Main Class” textbox:



JBuilder 5 Window

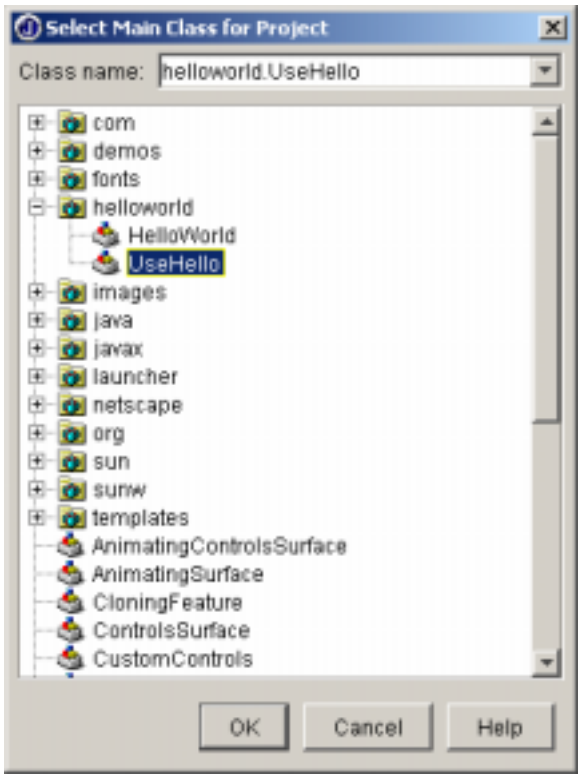
If you are using JBuilder 8, then instead click the “New” button:



JBuilder 8 Window

In JBuilder 8, you will then see a dialog like that for JBuilder 5. Click on the "…" button to continue.

Next find the method that contains main. In our case, it is in helloworld (the package), and then in UseHello:



Select "OK" until back to the main menu. We are now ready to run the project!

To run the project, click the run icon: 

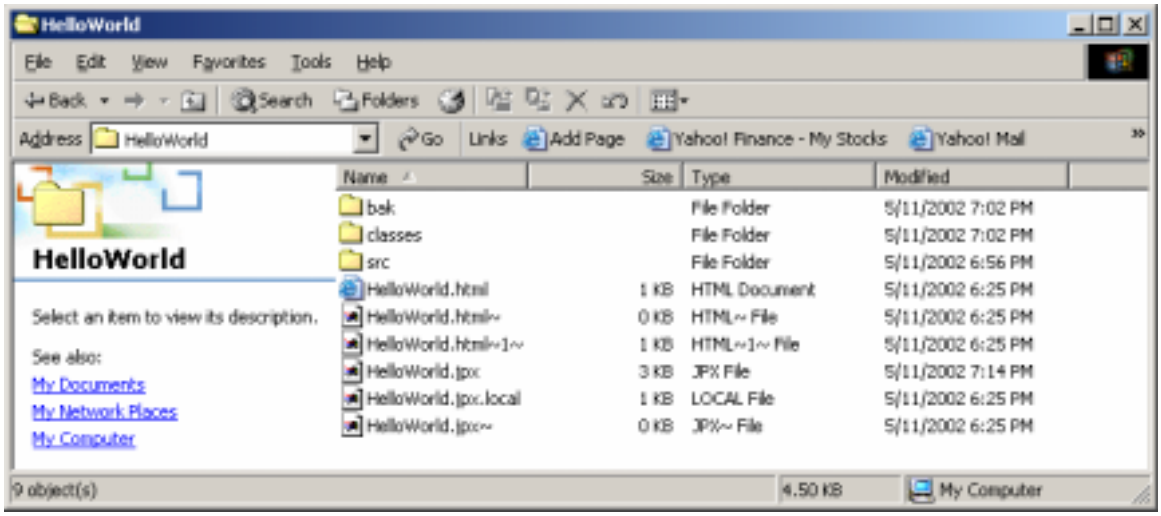
The program will compile. If there are errors, you will be shown the errors in the bottom window. Otherwise, the bottom window will contain the output of your program to the screen. In our case, it shows "hello, world" :

```
D:\Devstudio\JBuilder6\jdk1.3.1\bin\javaw -classpath
"C:\temp\HelloWorld\classes;D:\Devstudio\JBuilder6\jdk1.
aws.jar;D:\Devstudio\JBuilder6\jdk1.3.1\jre\lib\rt.jar;I
.3.1\lib\htmlconverter.jar;D:\Devstudio\JBuilder6\jdk1.3
hello, world
```

You will likely see other information about the compilation process in this lower window as well.

Congratulations, you have just created your first project in JBuilder! You should follow a similar process when working on programs assigned as homework. To close your project and exit, select “Close Projects” from the F)ile menu, and select the project you have worked on.

If you would like to copy the files you just created, they are located in the directory you initially specified when creating the project. In this example, I put them in c:\temp\HelloWorld:

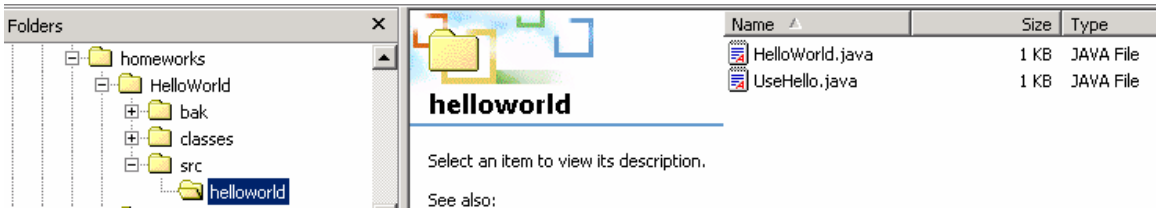


If you want to work on the project at a later date, open up “Helloworld.jpx” from JBuilder. We have only touched on the basics of using JBuilder here; feel free to explore on your own the many other options that are available. Later we will look at using JBuilder’s debugging tools which are quite helpful in tracking down mysterious runtime errors.

Turning Files In

When turning your files in, you only need to turn in the .java files. There is no need to turn in the .class files or the project files. The java files are located in the “src” directory of the project directory:

For example, the following image shows the files to turn in for the HelloWorld program:



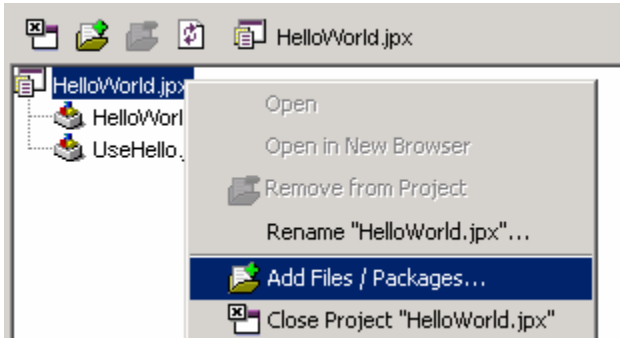
Feel free to compress the files with WinZip if you like prior to turning them in.

Using SavitchIn.java

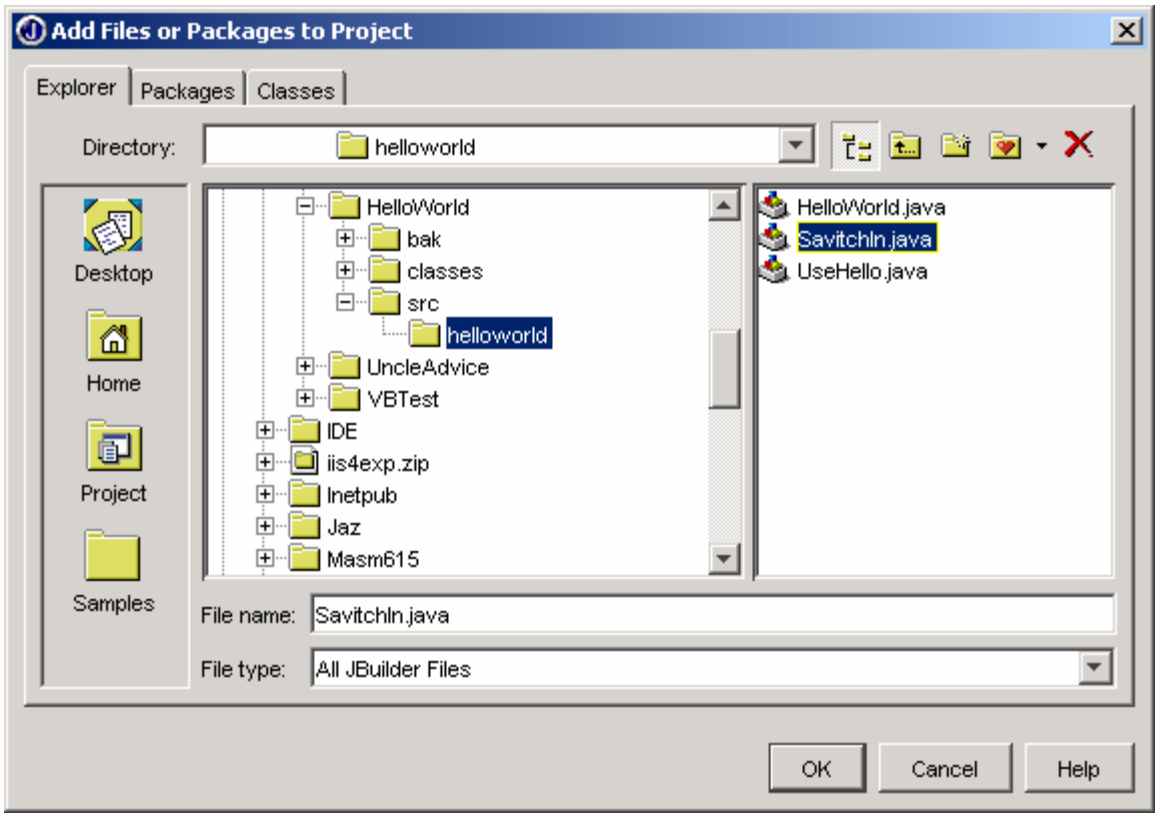
If you would like to use the SavitchIn.java library that comes with the textbook then follow these steps:

1. Copy the file SavitchIn.java into the src/projectname folder along with the other source code for your project. For example, with the HelloWorld project we would copy the SavitchIn.java file to src/helloworld to give us:

2. Add the SavitchIn.java file to your project by right-clicking in the project pane and select "Add Files"



3. Navigate to the SavitchIn.java file that you copied to the project directory and click OK.



4. Double-click the SavitchIn.java file and add the line "package <packagename>;" at the top of the SavitchIn.java file. Use the name of your package; this is the same line that JBuilder adds to the top of every line in the file. In this case, the package name is "helloworld"

```
1 package helloworld;
2
3 import java.io.*;
4 import java.util.*;
```

You can now use the SavitchIn class to input data.