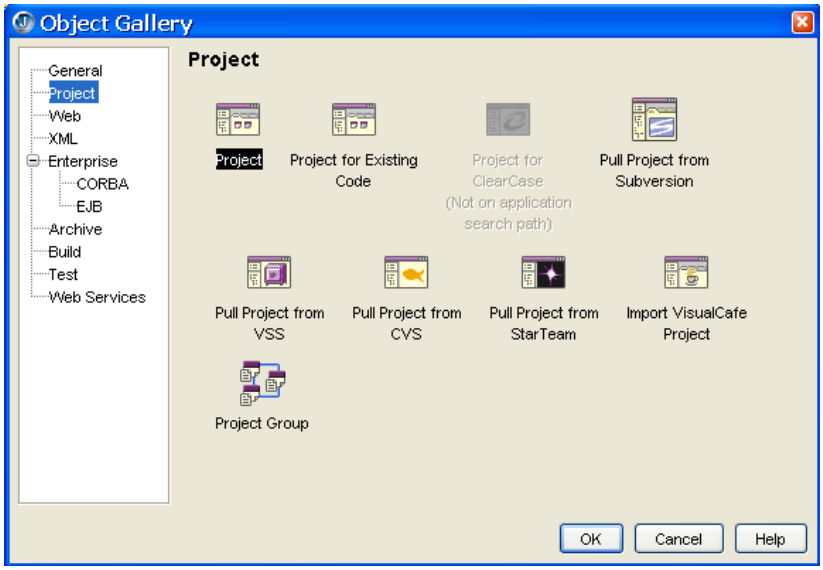
Lab 1A. Create a simple Java application using JBuilder

In this lab exercise, we'll learn how to use a Java integrated development environment (IDE), Borland JBuilder 2005, to develop a simple Java application – the hello world example from Jo Wood's book "Java Programming for Spatial Sciences".

The exercise has two parts: the first one is to make a program that prints out on the screen some messages. The second is to make a program that opens up a window and display some messages in that window. See appendix for Java source code provided by Jo Wood. (http://oldspice.soi.city.ac.uk/jpss/source/)

Part 1: make the simplest Java application – Hello World

1. Start Jbuilder.
2. Select File → New

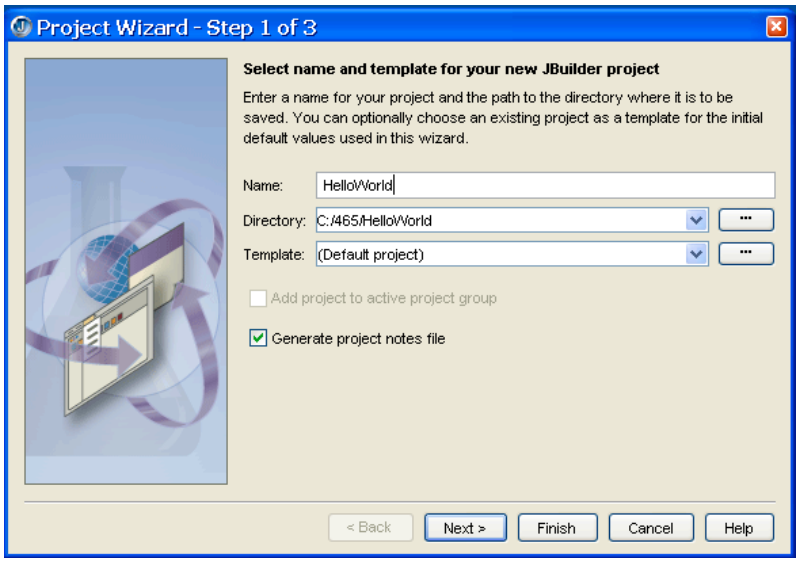


You might want to look at all the options you have on this dialog. The "Help" button will give you more information about the selected option.

3. In the dialog box, select Project Project, click OK

4. In the following dialog box, type in the name of the project and path that you want to save your project. A JBuilder project contains all the resources you need to build, deploy, and document your Java application. You can, and in many cases should, explore and edit these resources.

A folder with the application name will be created in the directory you designate.

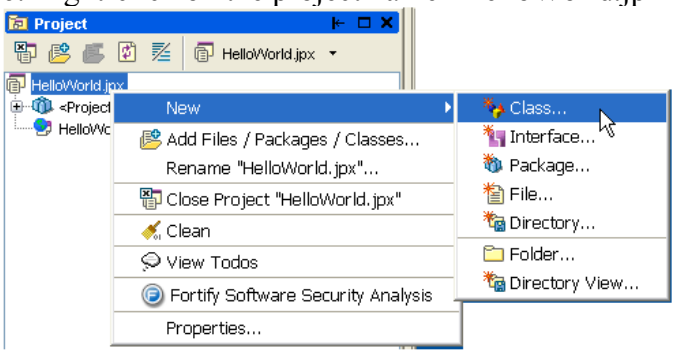


5. In the next steps, review the fields and accept the defaults.

Now you have a blank project.

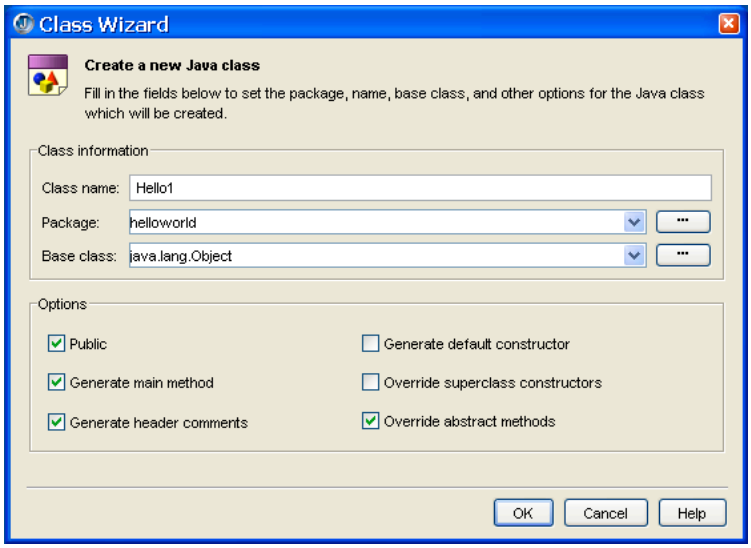
Now we will create a new class - class Hello1

6. Right-click on the project name “HelloWorld.jpx”, select New □ Class



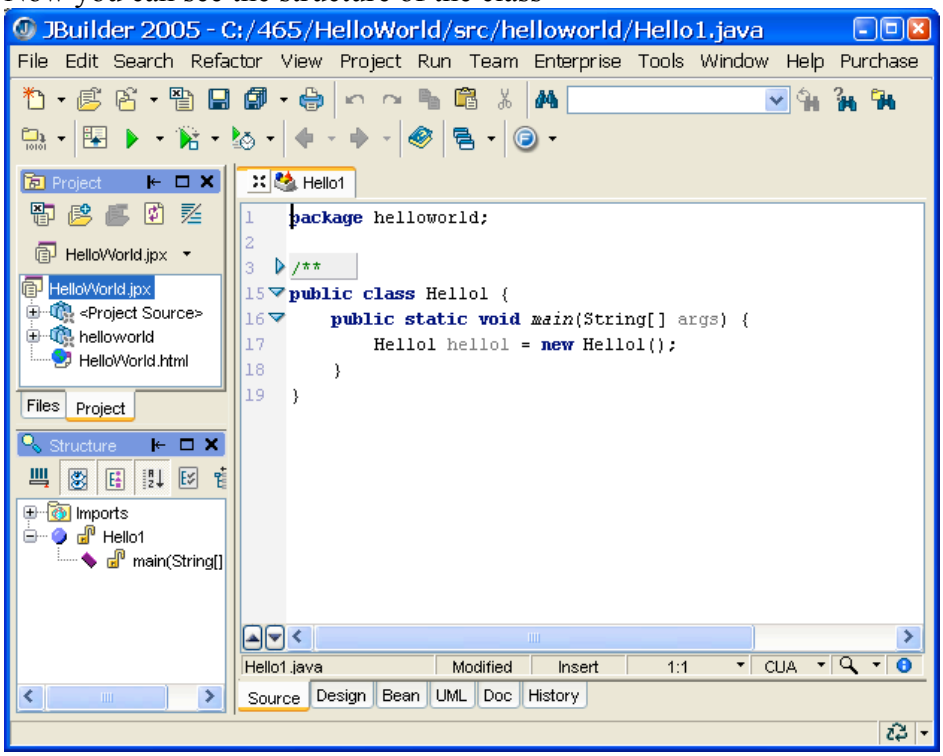
7. Fill in class name “Hello1”, type in “helloworld” as package name. Select “Generate main method”, deselect “Generate default constructor”.

Package is a container of classes. You can think of it as a folder, which contains Java class files.



Note: Java is case sensitive, so Hello1 is different from hello1

Now you can see the structure of the class



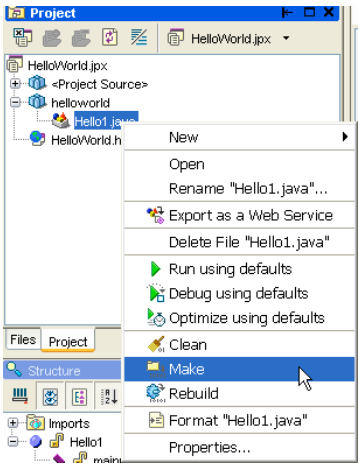
At this point let's take a look at JBuilder's IDE: from top down, after menus and tool buttons, you see three panes: two on the left and one on the right. Upper left is the project explorer, which displays all project resources and you can access the functionalities of manipulating these resources by right-click on the item. Lower left is the structure view of the content displayed on the right pane, which is the working area you do the detailed editing.

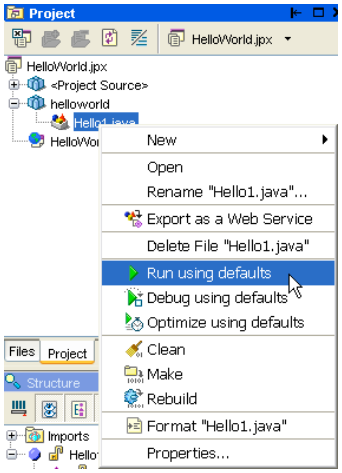
Now we can finish up Hello1.class:

8. Copy and paste the code found in Jo Wood's source code Hello1.java into the main() method of your class Hello1. Replace whatever exists in the body of main(){}. LEAVE the brackets { and }.

```
System.out.println("Java Programming for Spatial Scientists");
System.out.println("==== =========== === ======= ==========");
System.out.println("");
System.out.println("Program One: Hello World.");
```

9. Save your work.

10. Compile the class 

and run it: 

At the bottom you can see the output

How Java program runs:
When the interpreter executes a class, it looks for a particular method by the name of main, which will sound familiar to C programmers. The main method is passed as a parameter to an array of strings (similar to the argv[] of C), and is declared as a static method (more on this in a later tutorial).

To output text from the program, we execute the ' println ' method of System.out, which is Java's output stream. Unix users will appreciate the theory behind such a stream, as it is actually standard output. For those who are instead used to the Wintel platform, it will write the string passed to it to the user's screen.

Part 2. make an application with a window

Now repeat from step 6 and make another class Hello2, this time copy all the code, except the package line, of Hello2.java from Jo Wood's sample code.

Try to understand the code:

- to open up a window, class Hello2 needs to "extends JFrame";
- JFrame is a class that exists in package javax.swing, this why "import javax.swing.*";
- "super(windowTitle)" creates a new instance of JFrame with the given title
- Message "Program two" and "Hello again" is displayed as labels in the window
- Class "Container" and "BorderLayout" come from package java.awt;

Complete Java API documentation can be found at Sun website:
http://java.sun.com/j2se/1.4.2/docs/api/index.html

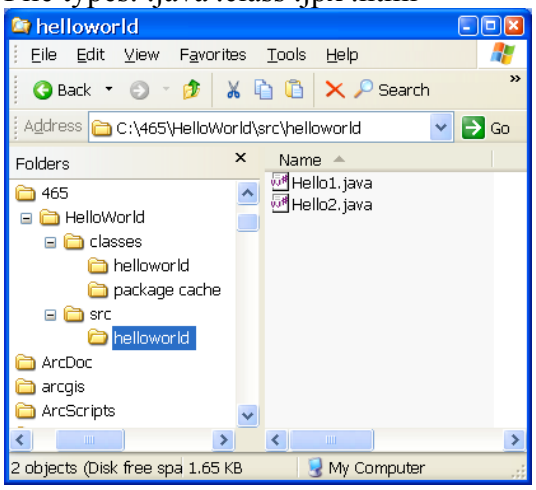
Run this class, what do you get?

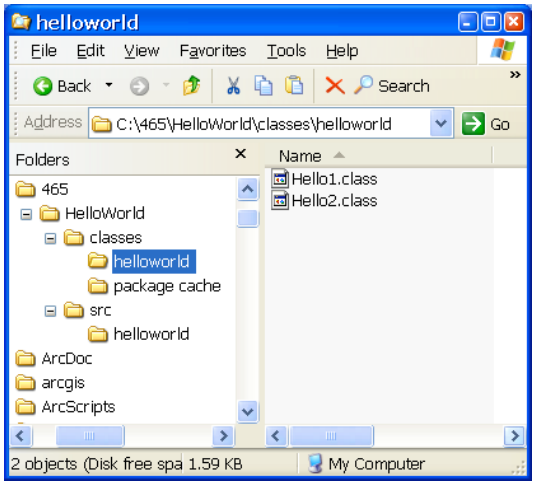
Things to think about:

1. Files and folders

Folders: what's in HelloWorld? What's in classes\helloworld? What's in src\helloworld?

File types: .java .class .jpx .html





Packages and classes

Dependencies: import

Common Java Libraries: java.awt java.swing

Appendix: Source code

Hello1.java

```
/**************************************************************
 * Hello1 - Program to display a simple text message on the screen.
 * Author   Jo Wood.
 * Version  1.1, 14th September, 1999
 *
 **************************************************************/

public class Hello1
{

    // All java applications have a main() method.
    // This one displays a simple message.

    public static void main(String args[])
    {
        System.out.println("Java Programming for Spatial Scientists");
        System.out.println("==== =========== === ======= ==========");
        System.out.println("");
        System.out.println("Program One: Hello World.");
    }
}
```

Hello2.java

```
import java.awt.*;              // Needed to use graphics classes.
import javax.swing.*;

/**************************************************************
```

```
 * Hello2 - Program to create a graphics window and display a
 *          simple text message on the screen.
 * Author   Jo Wood.
 * Version  1.2, 25th August, 2001
 *
 ****************************************************************/

public class Hello2 extends JFrame
{
    // All java applications have a main() method.
    // This one creates a graphics window.

    public static void main(String args[])
    {
        new Hello2("Java for Spatial Sciences");
    }

    // This method has the same name as the class (Hello2) and is
    // called a constructor. In this case it sets the window title.

    public Hello2(String windowTitle)
    {
        // Initialise window.
        super(windowTitle);
        setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        Container window = getContentPane();

        // Add two lines to the window.
        window.add(new JLabel(" Program two:"),BorderLayout.NORTH);
        window.add(new JLabel(" Hello again."),BorderLayout.CENTER);

        // Size window and make it visible.
        pack();
        setVisible(true);
    }
}
```

Lab 1B. Explore the Spatial Modeling Package using JBuilder

In this lab exercise, you are asked to explore the spatial modeling package, which will be used throughout this course. The spatial modeling package contains five classes and one interface:

```
public class Footprint{}
public class Gazeteer{}
public class GISVector{}
public class SpatialObject{}
public class VectorMap{}
public interface SpatialModel{}
```

In this exploration, you are asked to create a new project in JBuilder, and import the package, jwo.jpss.spatial, following the directions below:

1. Similar to Lab 1A, create a new project, using File → New. Name your project, spatialTest*_yourinitials*. Designate the location of the new project folder within your student storage on the P:/ drive and accept all default settings. JBuilder creates a folder in your student storage, with the name you specified.
2. Open Windows Explorer and point to the new project folder you just created. Open the project folder and create a new folder called "src".
3. Since we already have the source code for these six .java files, we do not need to create a new class. Using Windows Explorer copy the package folder called "jwo" from P:/geog465win05/lab1b into the spatialTest*_yourinitials*/src folder you designated in the P:/ drive.
4. Return to JBuilder, and on the project explorer (the left pane of the screen), click the refresh image at the top of the pane. The folder you imported in Windows Explorer should now be visible as a package, under the name: "jwo.jpss.spatial". Expand the package using the "+" symbol to display the six .java files.
5. Begin exploring this package by double clicking on any of the .java files to view the source code. Try to identify the following items:
    a. Constructor methods
    b. Accessor methods
    c. Mutator methods
    d. Classes which *extends* other classes
    e. Classes which *implements* interfaces
    f. Classes versus instantiated objects
    g. Constants
6. After you feel comfortable with the source code, right click on the package in the project explorer pane, and select "Make" to compile the package.