

Netbeans IDE Tutorial for using the Weka API

Kevin Amaral

University of Massachusetts Boston

First, download Netbeans packaged with the JDK from Oracle.

<http://www.oracle.com/technetwork/java/javase/downloads/jdk-7-netbeans-download-432126.html>

Overview Downloads Documentation Community Technologies Training

JDK 7u13 with NetBeans 7.2.1

This distribution of the JDK includes the [NetBeans IDE](#), which is a powerful integrated development environment for developing applications on the Java platform. [Learn more](#)

You must accept the [JDK 7u13 and NetBeans 7.2.1 Cobundle License Agreement](#) to download this software.

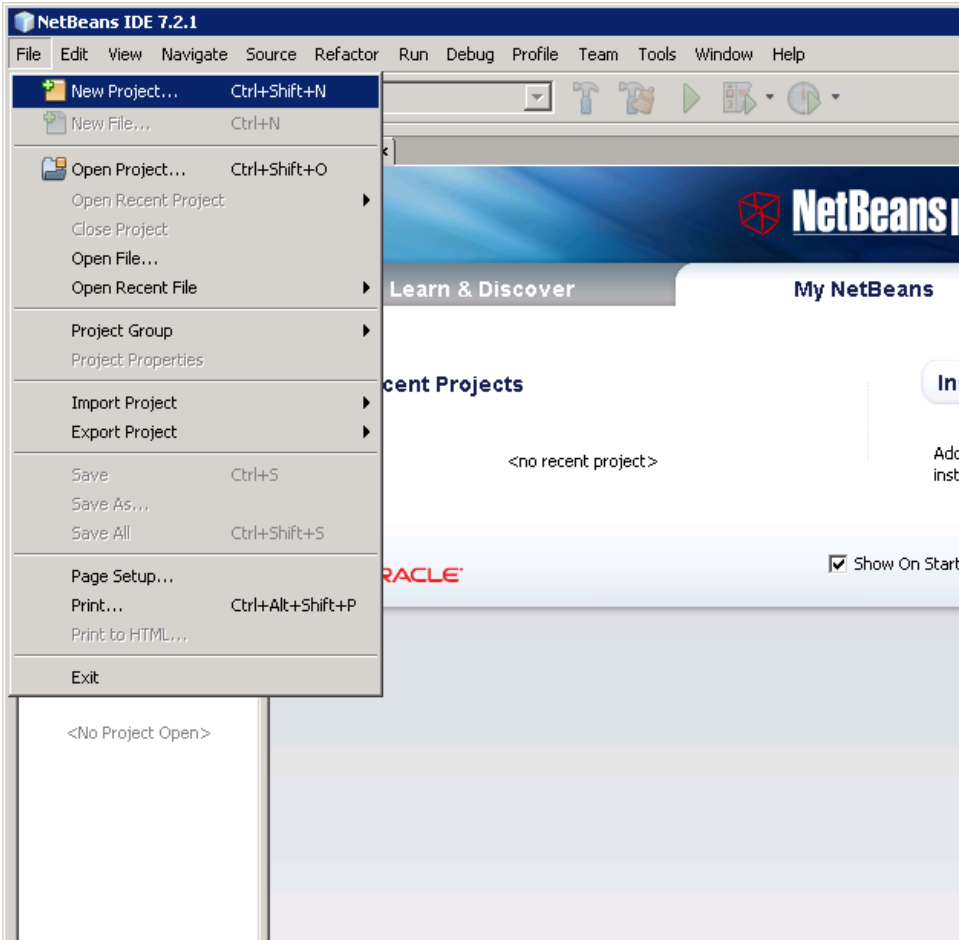
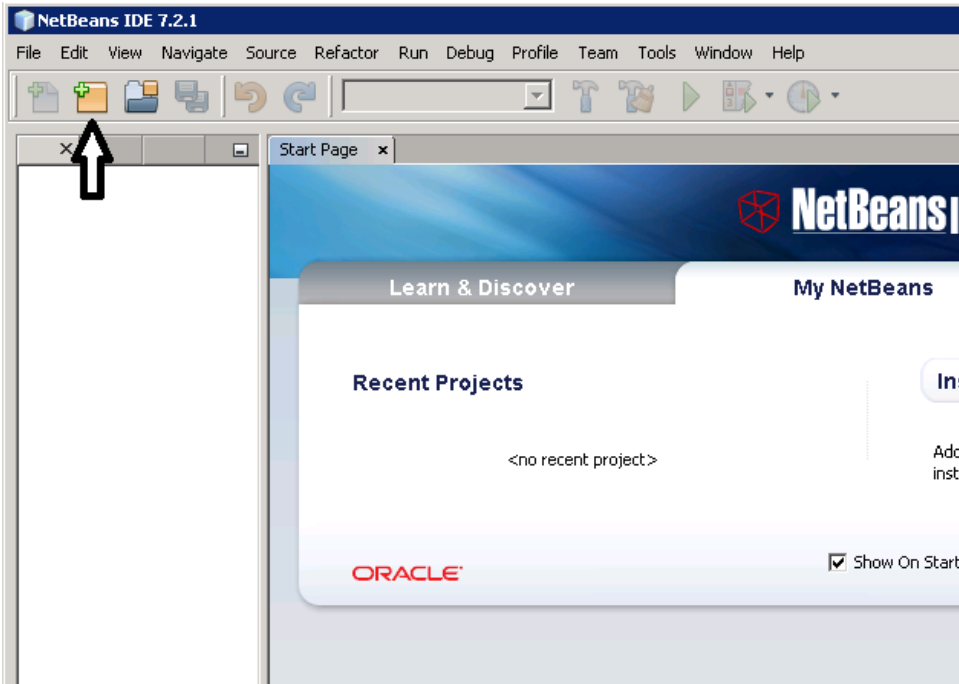
Accept License Agreement Decline License Agreement

Java SE and NetBeans Cobundle (JDK 7u13 and NB 7.2.1)		
Product / File Description	File Size	Download
Linux x86	188.77 MB	jdk-7u13-nb-7_2_1-linux-i586-ml.sh
Linux x64	185.53 MB	jdk-7u13-nb-7_2_1-linux-x64-ml.sh
Mac OS X x64	215.7 MB	jdk-7u13-nb-7_2_1-macosx-x64-ml.dmg
Windows x86	181.31 MB	jdk-7u13-nb-7_2_1-windows-i586-ml.exe
Windows x64	184.07 MB	jdk-7u13-nb-7_2_1-windows-x64-ml.exe

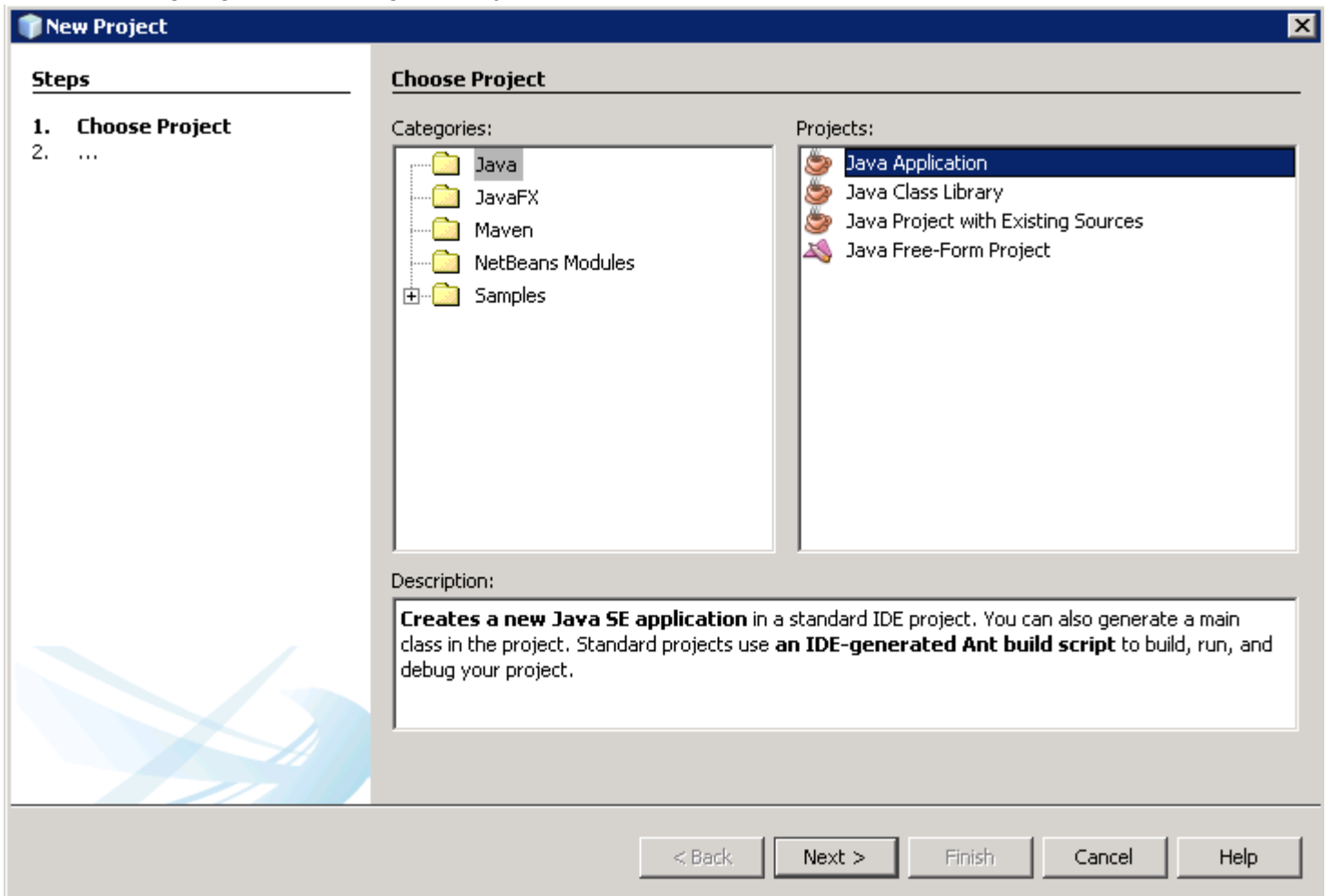
You need to accept the license agreement (unless you disagree with the licensing terms). After which, you should choose the package that corresponds with the machine you're going to be using the IDE on. Machines in the Unix Lab and Web Lab should have NetBeans installed already.

Follow the installer and avoid installing the Ask! Toolbar or McAfee in the event that they are offered by the installer. You won't need either of them to use Netbeans.

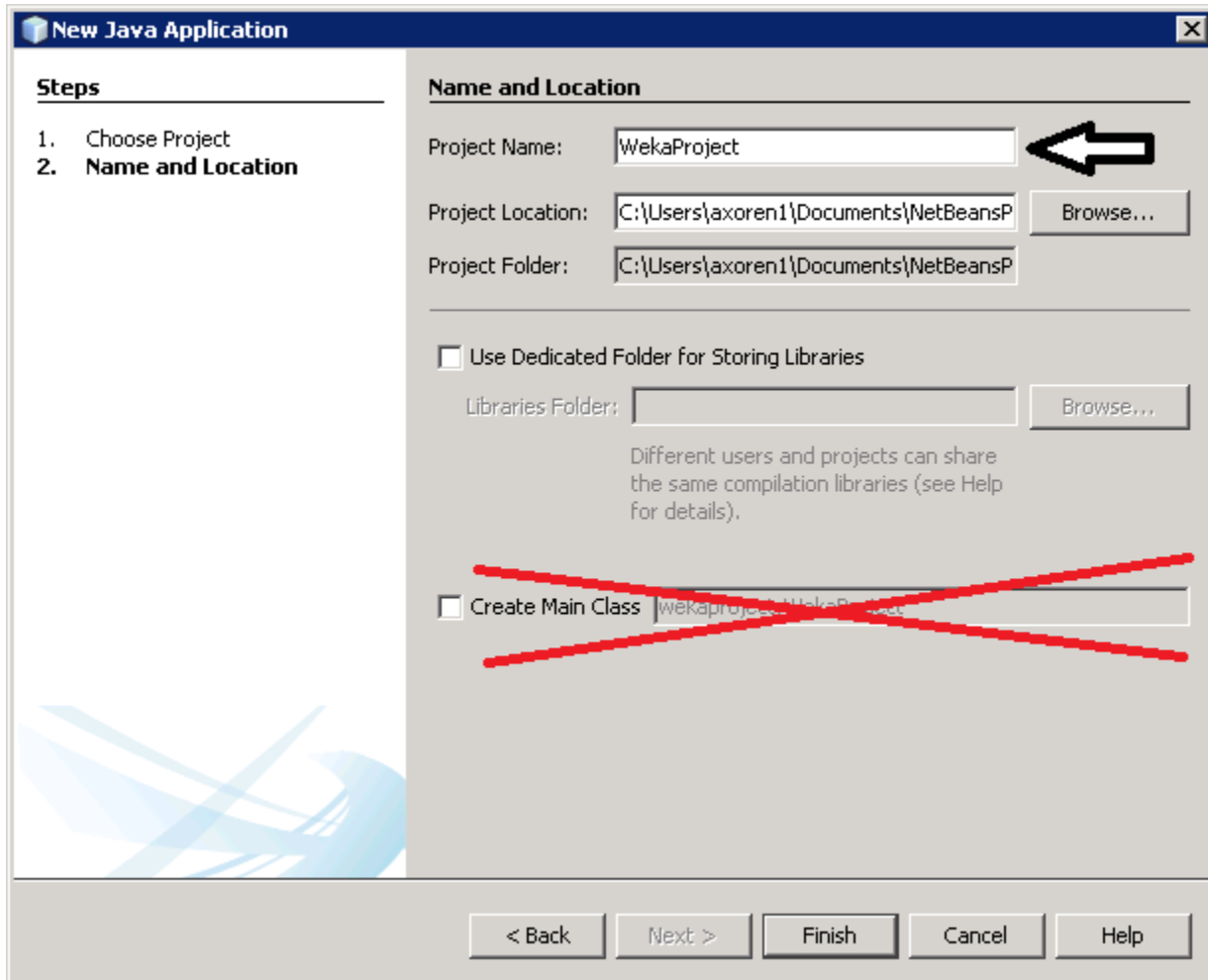
Once Netbeans is installed, run the IDE and create a new project.



We're going to be creating this project as a new application.

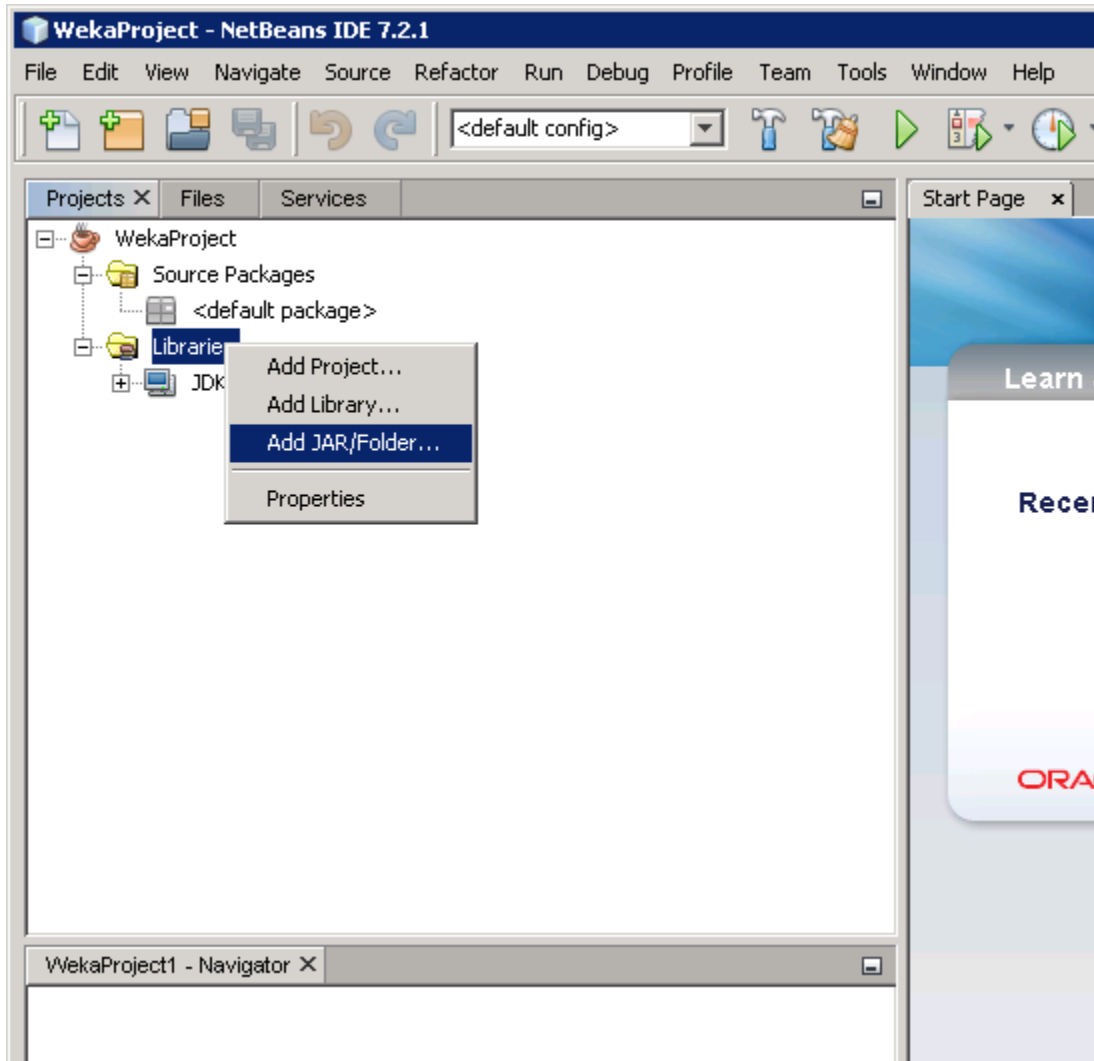


On the next screen, give the project a name and uncheck “Create Main Class”. That option will just give you more headaches later on if you’re not familiar with java packages.

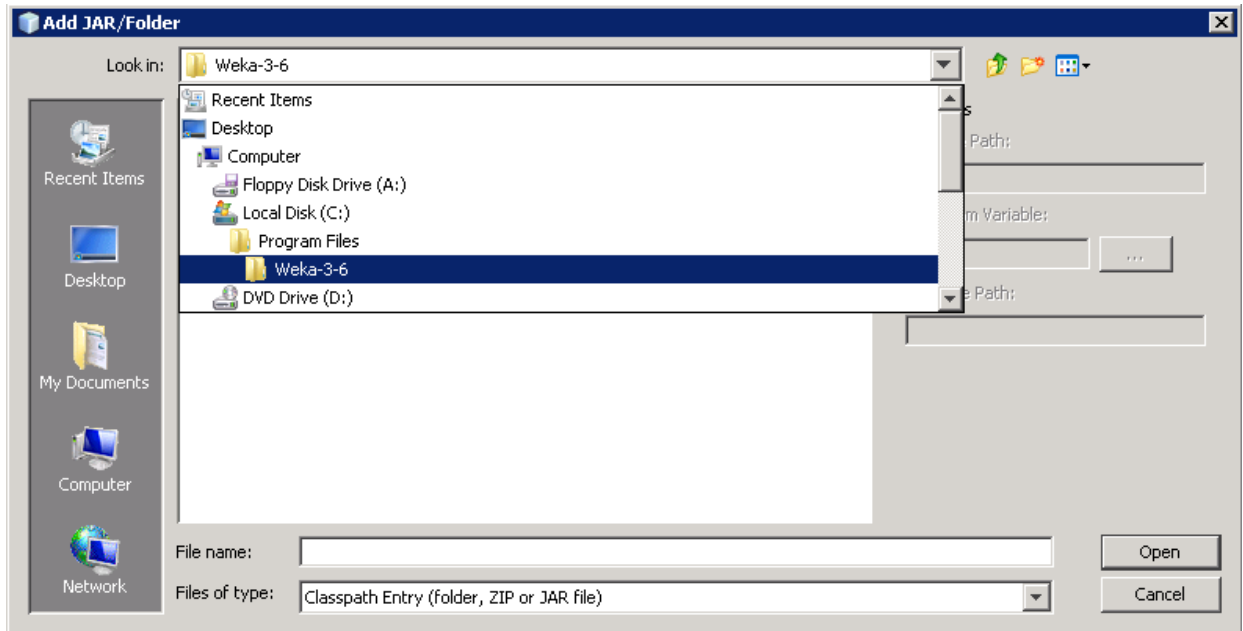


At this point, click Finish. You should be back at the main IDE window.

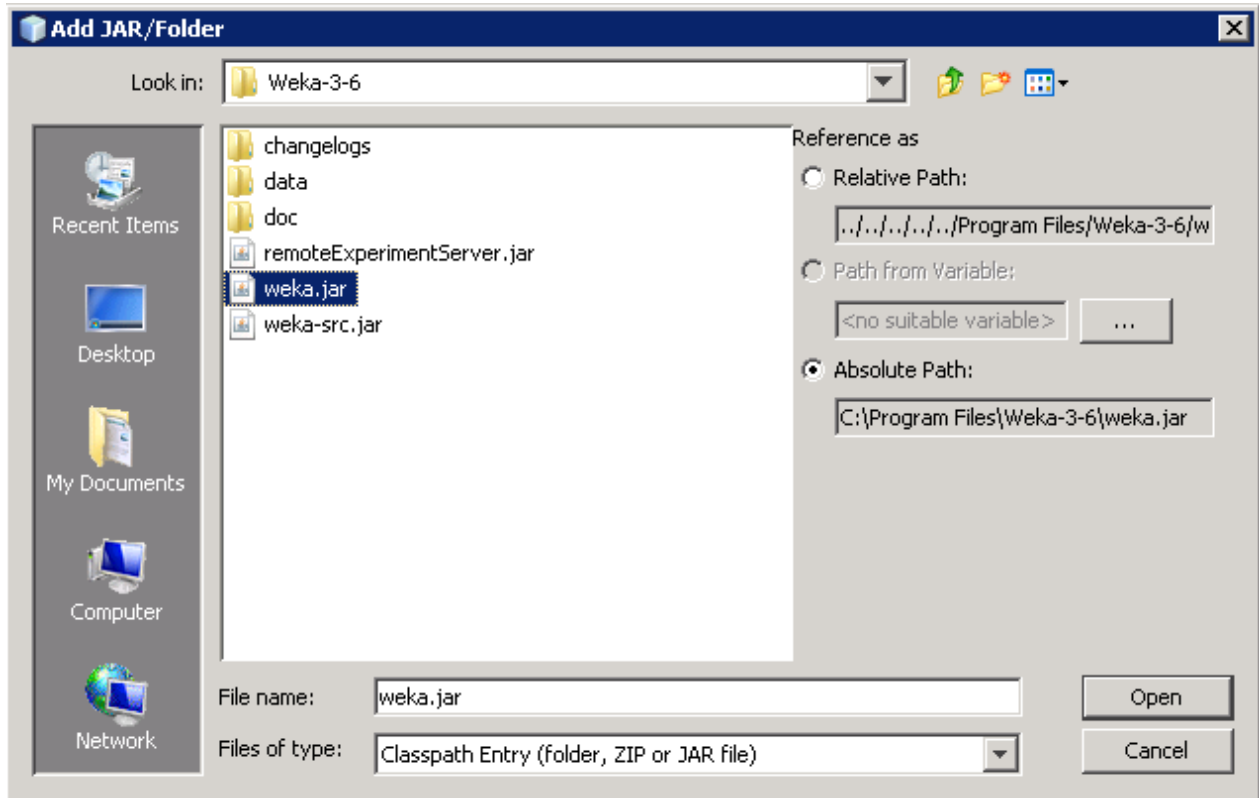
If you've navigated the Weka program folder, you'll notice something called JAR files. They're actually what we're going to be using to expose the Weka API to Netbeans so that you can program in Java with the same flexibility as the Weka GUI, and possibly more.



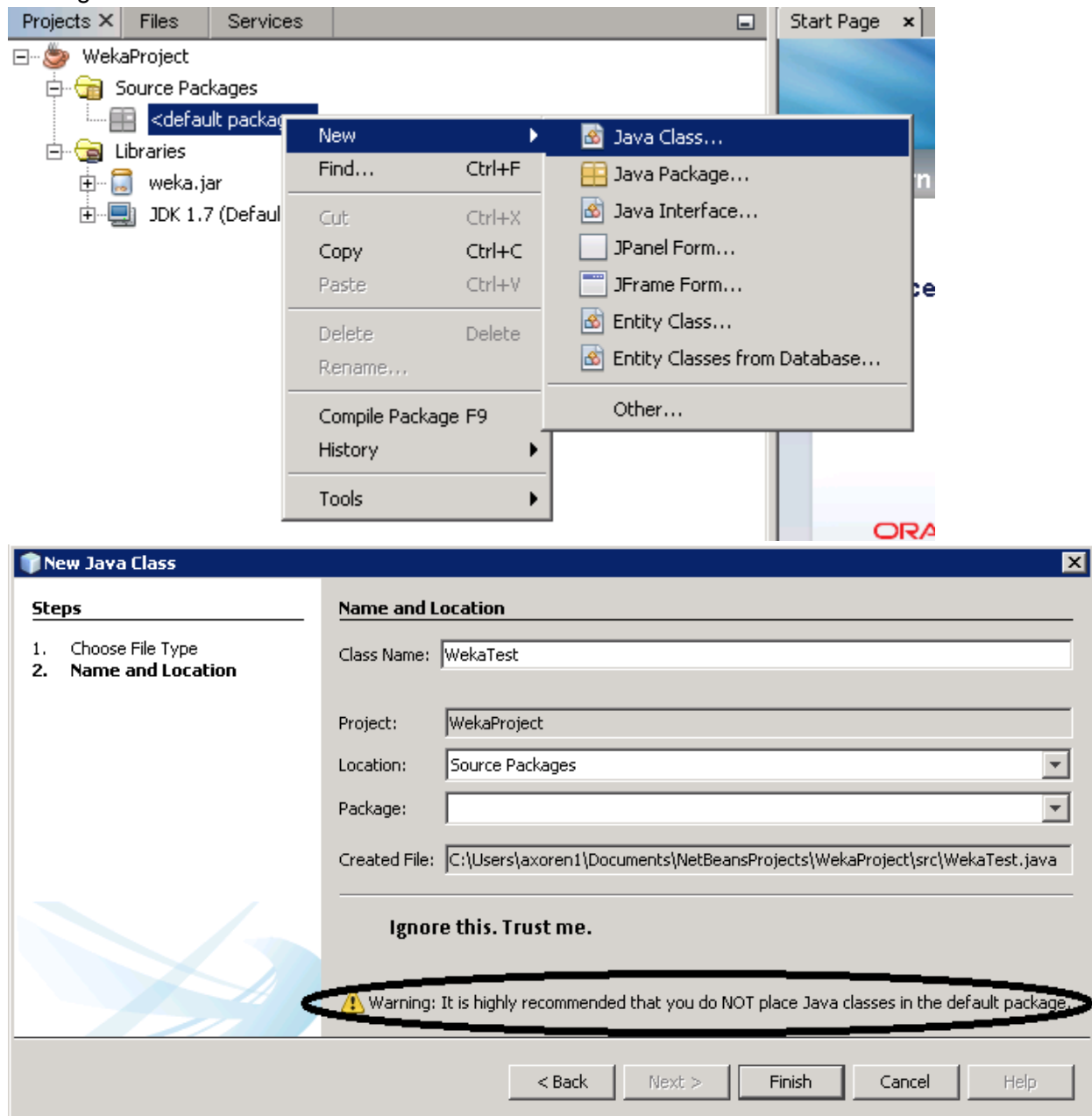
Navigate to the Weka folder in the file dialog and select the weka.jar file.



You may notice that there's a weka-src.jar file, as well. If you're feeling adventurous, at another time, you can extract files from that JAR with WinRAR or similar archiving tools. That is all the human readable source that, when compiled, becomes the Weka program you used in Homework 1. For our purposes, only worry about the weka.jar .



Now, we can start programming. Create a new java class in the Default Package. This is commonly bad practice when making an actual application, but we're doing experiments, not making the next Word Processor or Internet Browser.

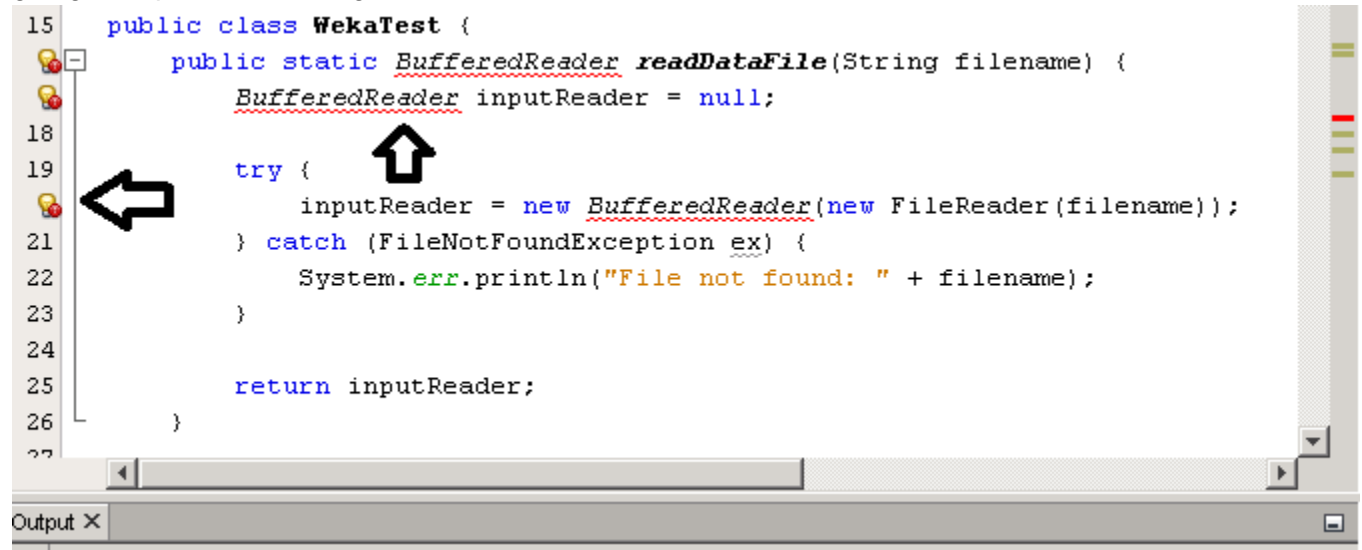


Click Finish and you should be brought to a text editor window where you'll be programming in Java. For those that haven't programmed in Java before, it's a very structured language that is driven by object orientation. Oracle has some good resources for understanding the basic principles of Java. They can be found here:

<http://docs.oracle.com/javase/tutorial/java/index.html>

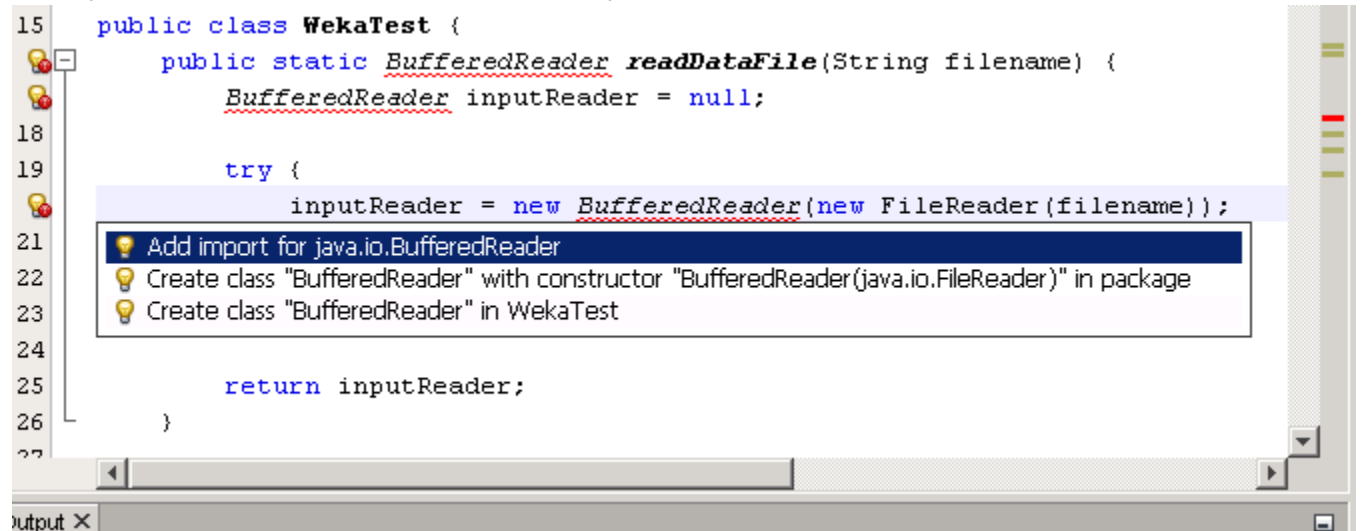
I'm going to be providing source code for you all to look at and pick apart as a demo, but I'm going to explain a few things first. First of all, this:

```
15 public class WekaTest {
16     public static BufferedReader readDataFile(String filename) {
17         BufferedReader inputReader = null;
18
19         try {
20             inputReader = new BufferedReader(new FileReader(filename));
21         } catch (FileNotFoundException ex) {
22             System.err.println("File not found: " + filename);
23         }
24
25         return inputReader;
26     }
27 }
```



When you see errors like this, it usually means the type of object you want to use isn't imported. By default, Java doesn't import every object you could possibly use. This is a good thing even if it makes your life a little tougher. Sometimes object types have the same names across different packages, so Java makes you choose explicitly. When you see those underlined errors, you can usually use the Hint Icons on the left to remedy them, like so:

```
15 public class WekaTest {
16     public static BufferedReader readDataFile(String filename) {
17         BufferedReader inputReader = null;
18
19         try {
20             inputReader = new BufferedReader(new FileReader(filename));
21         }
22         } catch (FileNotFoundException ex) {
23             System.err.println("File not found: " + filename);
24         }
25
26         return inputReader;
27     }
28 }
```



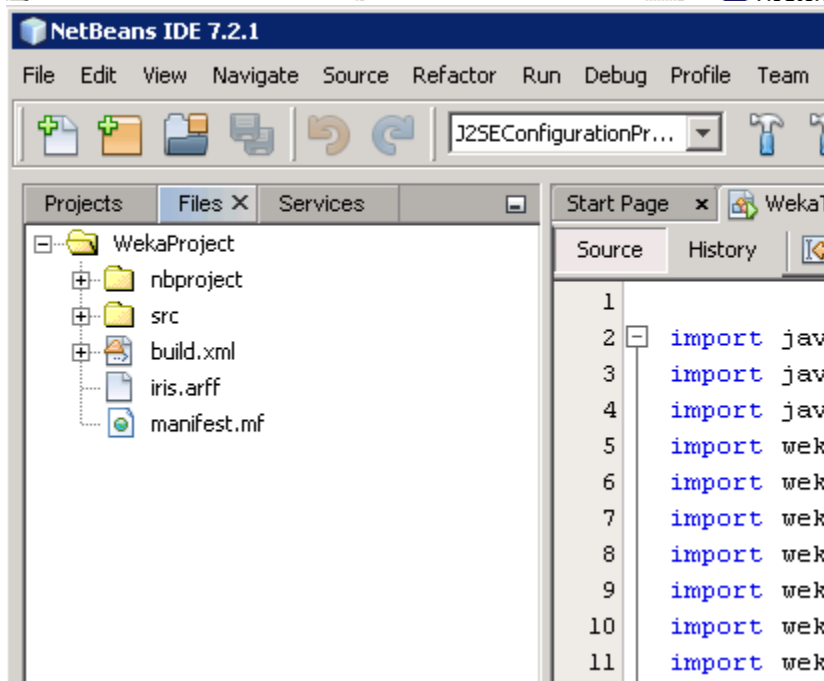
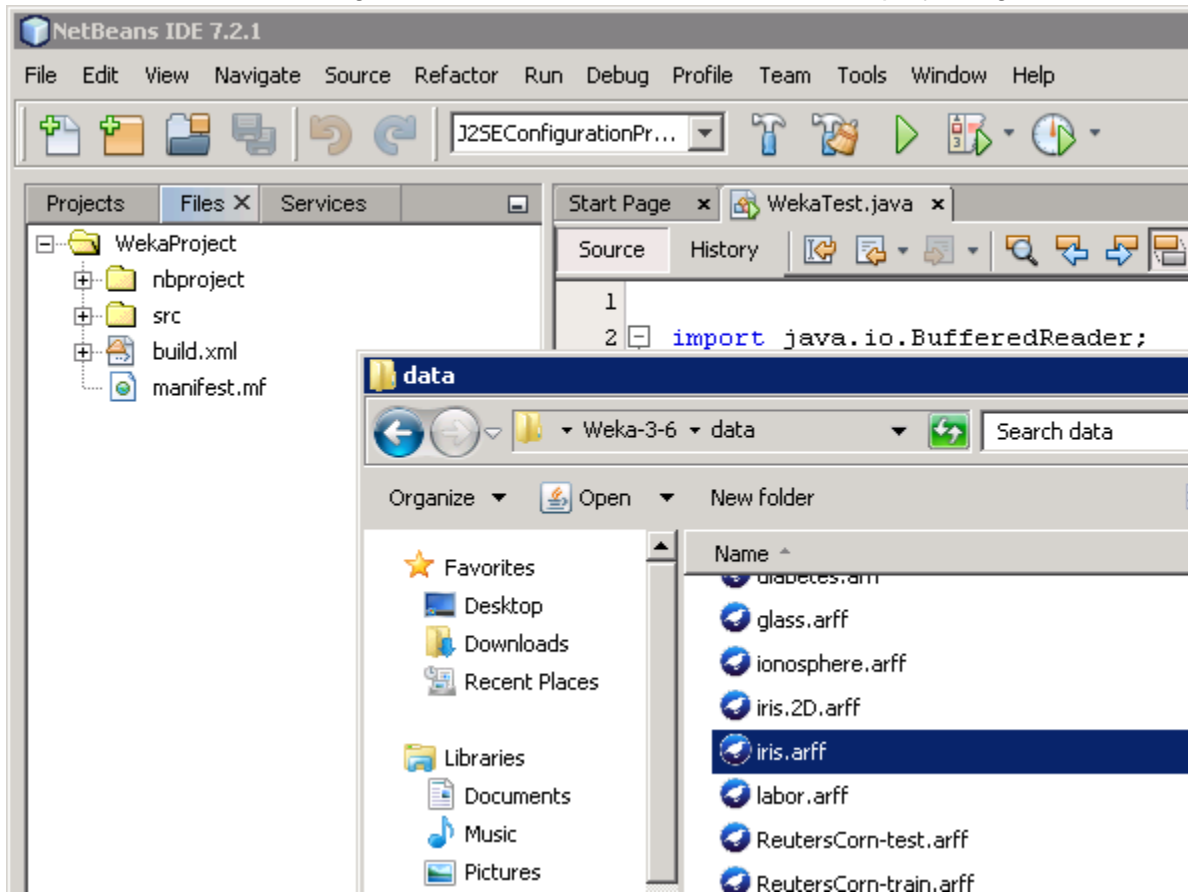
Believe it or not, this is actually a reasonable amount of imports when using an imported API.

```
1
2 import java.io.BufferedReader;
3 import java.io.FileNotFoundException;
4 import java.io.FileReader;
5 import weka.classifiers.Classifier;
6 import weka.classifiers.Evaluation;
7 import weka.classifiers.evaluation.NominalPrediction;
8 import weka.classifiers.rules.DecisionTable;
9 import weka.classifiers.rules.OneR;
10 import weka.classifiers.rules.PART;
11 import weka.classifiers.trees.DecisionStump;
12 import weka.classifiers.trees.J48;
13 import weka.core.FastVector;
14 import weka.core.Instances;
15
16 public class WekaTest {
17     public static BufferedReader readDataFile(String filename) {
18         BufferedReader inputReader = null;
19
20         try {
21             inputReader = new BufferedReader(new FileReader(filename));
22         } catch (FileNotFoundException e) {
23             e.printStackTrace();
24         }
25     }
26 }
```

Output X

That method I just showed you is going to be necessary for reading data files. However, there's something you need to know about the value of "filename" when you call it. It is either a direct path or a relative path. A direct path will be something like "C:\directory\data.arff" and a relative path will be something as simple as "data.arff". Direct paths are usually OS dependent, so you should use relative paths whenever possible. But where does the program look for files in a relative path?

If we switch to the files tab, on the left, we see the directory structure of the Java project folder. Here, we can click and drag datafiles that we intend to use with the project right into the folder.



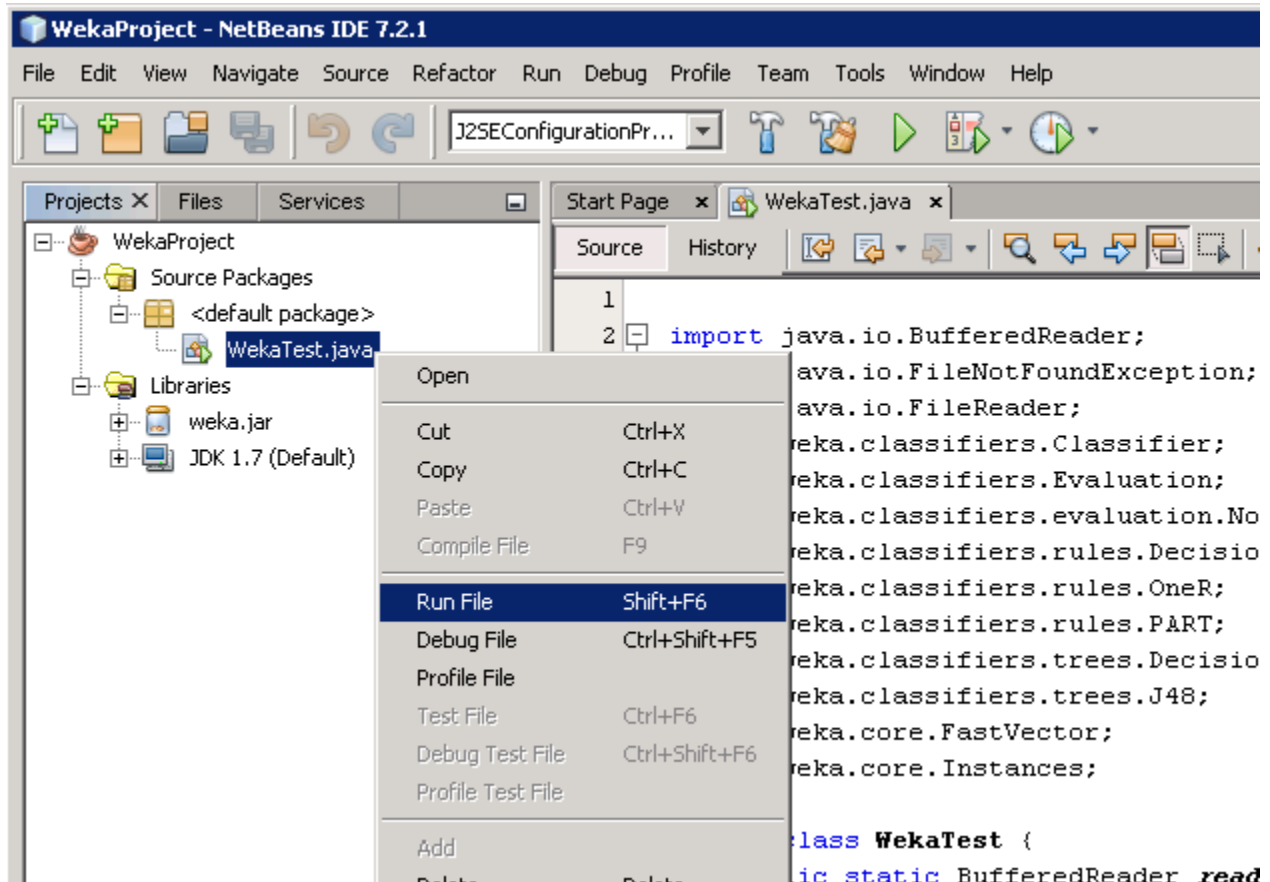
Now that that's done, we can use the following line in our code and it will correctly read from the data file we want to use. This method is the main method. It is the program's entry point, where the first lines of code are executed. Java is an imperative language of commands and declarations, with each line of code happening in the order that it's written, for the most part. Method definitions allow you to name frequently used subroutines so that you can use them more than once without having to write them out each time. This is what I've done with "readDataFile". If you looked at the method earlier, you can see why I wouldn't want to write that more than once.

```
61
62 public static void main(String[] args) throws Exception {
63     BufferedReader datafile = readDataFile("iris.arff");
64
```

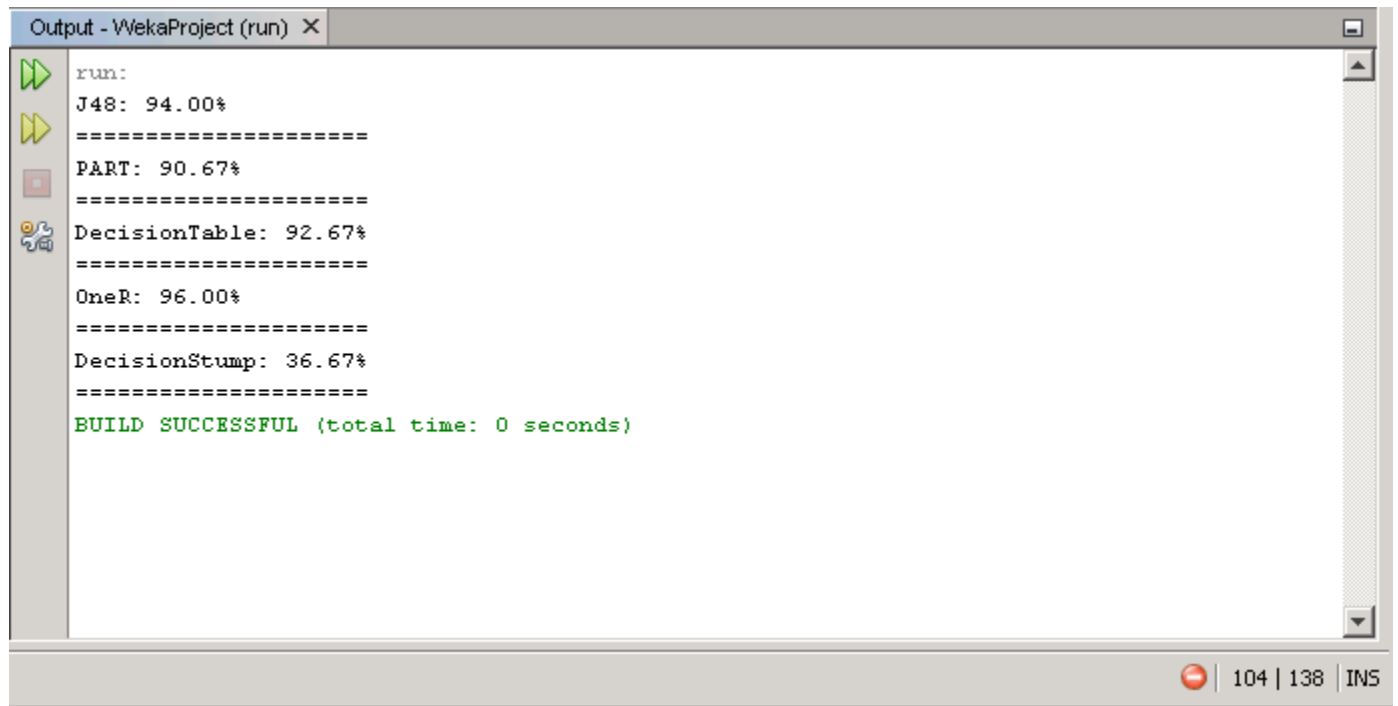
You'll notice that I've added "throws Exception" to the end of the main method. This is bad style, but it relieves a few coding-headaches (fewer try-catch statements littered across your code). If your code works, you won't need to worry about it. My finished WekaTest.java source file is here:

http://www.cs.umb.edu/~axoren1/weka_netbeans_tutorial/WekaTest.java

One last thing: Now that you have a java program, how do you run it in Netbeans?



After you've done that, your program's output should be in the bottom-left section of the Netbeans window.



```
Output - WekaProject (run) X
run:
J48: 94.00%
=====
PART: 90.67%
=====
DecisionTable: 92.67%
=====
OneR: 96.00%
=====
DecisionStump: 36.67%
=====
BUILD SUCCESSFUL (total time: 0 seconds)
```

104 | 138 | INS

You should be all set, now. Happy coding!