

Using Eclipse for Java

Using Eclipse IDE for Java Development

- ▶ Download the latest version of Eclipse (Eclipse for Java Developers or the Standard version) from the website:
<http://www.eclipse.org>.
- ▶ Available for MS Windows, Linux and Mac OS X. We must first install Java Development Kit (JDK). See the following link on how to install and setup the Java JDK on your computer.
[Installing JDK on Windows/Mac](#)
- ▶ Eclipse is already installed in the onyx lab. There should be an eclipse icon on your desktop. Or just type `eclipse` on the console prompt.
- ▶ Eclipse comes with built-in tutorials and extensive help. Many good tutorials can also be found on the web.

Creating a new Java Project


- ▶ Click on **File** → **New** → **Java project**.
- ▶ Choose the name of the project. Check the box labeled *Use default location* if it isn't already checked. Eclipse creates a folder under your default workspace folder with the same name as the project.
- ▶ Next choose the option *Use project folder as root for source and class files*. We recommend that you configure that as default for this semester by clicking on the *Configure default* link. Select *Next* to go to the next window. Then select *Finish* at the bottom to finish creating the project.
- ▶ To create a new class file, right click on the *Package Explorer* pane (on the left) and choose **New** → **Class**. Make sure the field labeled *Package* is blank. Add a name for the class. Click on the box for adding a `main` method if you want one in that class.

Importing Existing Java Classes into Eclipse

Suppose you have an existing Java program in a folder. Then you can bring it into Eclipse two different ways.

- ▶ **Import existing files into Eclipse project.** Create a new Java project in Eclipse. Click inside the project on the left pane. Then select *Import...* and then *General* and then *File System*. Then browse to the folder that contains your Java files and select the ones you want to import into your project. This will make a copy of those files into your eclipse workspace folder for the new project.
- ▶ **Create Eclipse project in existing folder.** Create a new Java project in Eclipse. Uncheck the option *Use default location* and browse to the folder that contains your Java files. Then select *Finish* and now an Eclipse project has been created into your pre-existing folder!

Building, Running and Debugging a New Project

- ▶ Every time you save your Java class file, Eclipse automatically compiles it for you.
- ▶ To run your Java program inside Eclipse, click on the play button icon  or click on the *Run* menu and then choose *Run* (keyboard shortcut: **Ctrl-F11** on Windows and Linux, **Shift-Cmd-F11** on Mac OS X)
- ▶ To customize how a program runs, click on *Run* → *Run Configurations*. Then a new window pops up that allows you create and manage run/debug configurations. For example, command line arguments can be set in the *Arguments* tab.
- ▶ You can also run your Java program from the console directly by going to the folder in the workspace that contains your project and using the *java* command. You would also submit your assignment from the project folder.
- ▶ Note that if you change your Eclipse project files from outside Eclipse, you will have to refresh your files from inside the eclipse for it to see the changes. Use **F5** key to refresh your Eclipse project.

Handy Tips (1)

- ▶ *Content Assist*. Use the keys **Ctrl-Space** to ask for help with function names, arguments and other topical content assistance.
 - ▶ *Cool trick!* Type **syso** and then type **Ctrl-Space** and it will auto-complete to `System.out.println()`; 🤖
- ▶ *Word Completion*. Use the keys **Alt-/** to complete words after you type in the first few characters. Very useful to avoid having to type long variable or function names. Faster than **Ctrl-Space** but only matches in the current file.
- ▶ *Quick Fix menu*. Hover your mouse over an error and a quick fix menu drops down. Often the first suggestion will fix your error correctly! It will also help you find and insert the correct import statements when you use classes from the Java library.
- ▶ *Refactoring*. Refactoring allows you to change identifiers in your code in an intelligent fashion. The **Refactor** menu will check and change the identifiers in all locations and files for you! The Refactor menu has a lot more so explore away....
 - ▶ Use **Shift-Alt-R** on selected identifier allows you to quickly rename and refactor that identifier (variable, method or class).

Handy Tips (2)

- ▶ *Automatic Javadoc comment templates*. Check options under the *Source* menu for options to automatically generate javadoc comments for classes and methods. The keyboard shortcut is Shift-Alt-J. Type that after selecting the method or the class.
- ▶ *Commenting your code*. Check the *Toggle Comment* option under the *Source* menu to automatically comment selected parts of your program! The keyboard shortcut is Shift-Ctrl-C. Type that after selecting the code you want to comment.
- ▶ *Formatting your program*. Check the *Format* option under the *Source* menu to automatically nicely format selected parts of your program! The keyboard shortcut is Shift-Ctrl-F. Type that after selecting the code you want to format. Or select the whole program with Ctrl-a and then format it.

How to create a clickable Java GUI project

We have to convert our project into a Java JAR (Java Archive) file to make it clickable to run. First, place your sound files in a subfolder named `sounds`.

- ▶ From Eclipse, choose *Export...* in the project window on the left. Then choose *Java* → *JAR file*. Click *Next*.
- ▶ Choose *Export class files and resources* and then click *Next*. Choose where to save the jar file.
- ▶ Next page shows some jar packaging options. Accept defaults and click *Next*.
- ▶ On the next page, click on *Browse* by the field labeled *Main*. Choose the class with a main method that you want to start automatically when the user double clicks on your jar file. Then click on *Finish*.
- ▶ This will work fine except your program won't find the sound files. Use code similar to the following to read the sound clips.

```
url = getClass().getResource("sounds/explosion.wav");  
crashSound = JApplet.newAudioClip(url);
```

Now your program will be able to find the sound files when running as a jar file!