# Lab 1: Java Basics, Heap Sort and Eclipse

## Objectives

- Getting familiar with the Eclipse environment
- Creating a Java project; creating and editing a Java program
- Adding a library to the build path
- Running project inside and outside Eclipse

## Source files

- components.jar
- HeapSort.java

## 1 Introduction

Welcome to the first lab. These labs are generally intended to let you explore concepts introduced in the course lectures. During each lab, you need to follow the instructions from problem setup to deliverables. This lab contains 3 deliverables, which you should demo to the lab instructor.

### 1.1 Eclipse

The Eclipse platform is a generic *integrated development environment* (IDE) foundation without any focus on a specific programming language. The platform contains IDE functionality and is built with components creating applications by using component subsets. Developers create, share and edit generic projects and files in the platform, while participating within a multiple team development environment repository. However, it is written mostly in Java and most of the time is used to develop applications in this programming language, but it covers other languages such as: C, C++, Ruby, R, JavaScript, PHP, Python ...

In this lab we will focus on Eclipse usage in writing, editing and running **Java** programs.

### 1.2 Java Requirements of Eclipse

Eclipse requires an installed Java runtime (at least Java 5). Java can be downloaded in two flavors, a JRE (Java runtime environment) and a JDK (Java development tools) version. The Eclipse IDE contains its own Java compiler hence a JRE is sufficient for most tasks with Eclipse. The JDK version of Java is required if you compile Java source code on the command line and for advanced development scenarios, for example if you use automatic builds or if you develop Java web applications.

### 1.3 Eclipse Resources

| Resource | URL |
|---|---|
| Eclipse Homepage | http://www.eclipse.org/ |
| Download Eclipse | http://www.eclipse.org/downloads/ |
| Eclipse Online Tutorials | http://www.eclipse.org/resources/ |

## 2 Deliverable 1 -- Create and Run a Simple Java Program (Heap Sort)

Sort a list of integers using the heap sort algorithm:

- http://www.youtube.com/watch?v=WYII2Oau_VY (Super short, no words, pictures only)
- http://www.youtube.com/watch?v=B7hVxCmfPtM (The traditional version from M.I.T., very, very long)
- Or find your own

**Note**: Write carefully, you are going to reuse this code later!

*Step 1: Launch Eclipse*

Open the Eclipse IDE. It can be done at terminal by typing `eclipse`.

*Step 2: Create a new project*

Select *File* => *New* => *Java Project* => Name your project => *Finish*.

*Step 3: Add a new class*

Select your project => *File* => *New* ⇒ *Class* => Name the class => check `public static void main (String[] args)` => *Finish*.

For this lab, just copy the source file HeapSort.java and paste into the *src* folder of the project.

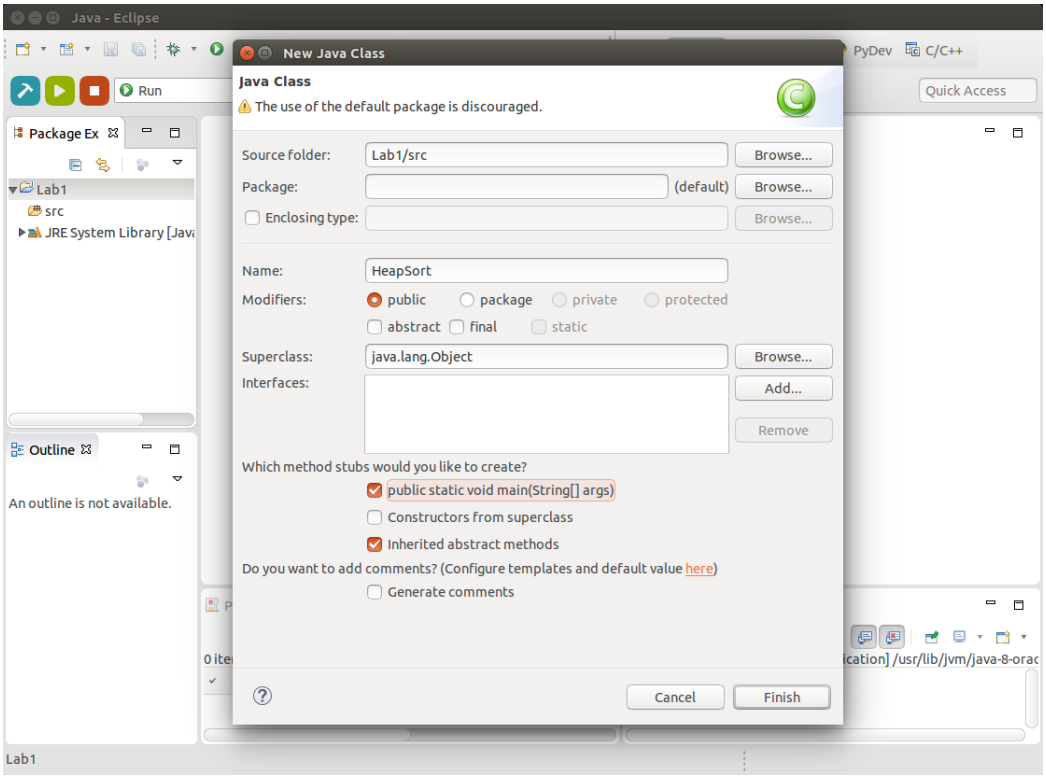
*Step 4: Implement your class*

Finish the `sort` method.

*Step 5: Run you code*

Run the code (*Run as* => *Java application*).

**DEMO this deliverable to the lab instructor.**




# 3 Deliverable 2 -- Add a Library to the Build Path

In this part, we want you to edit your source code and add a library to the build path.

## 3.1 Edit Your Code

An *import* statement is a way of making more of the functionality of Java available to your program. Java can do a lot of things, but you do not need all of them so often. It has its classes divided into *packages*. Your own classes are part of packages, too. So, anything that isn't in the `java.lang` package or the local package needs to be imported.

*Step 1: Import libraries*

Add the two following imports to the beginning of your source code:

```
import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;
```

*Step 2: Rewrite the output code*

Edit your `print`/`println` commands by instantiating an object of `SimpleWriter`. For example, substitute:

```
System.out.println(array[c]);
```

with:

```
SimpleWriter out = new SimpleWriter1L();
out.println(array[c]);
out.close();
```

*Step 3: Run your code*

Run your code and see there are some errors.

The reason for these errors is that the components in the import lines, `SimpleWriter` and `SimpleWriter1L`, are from the components.jar, but we have not told Eclipse to use it yet.

### 3.2 Add the Library to Your Project Build Path

*Step 1: Download the library*

Save the components.jar in your home directory (or home directory of your project).

*Step 2: Configure the project*

Right-click on your project => *Build Path* => *Configure Build Path* => *Add External Jars*=> Select the library => *Finish*.

Errors should disappear and you would be able to run your code.

**DEMO this deliverable to the lab instructor.**

## 4 Deliverable 3 -- Run Your Java Code outside of Eclipse

Eclipse has implemented its own java compiler called *Eclipse Compiler for Java* (ECJ), which is based on IBM's Jikes java compiler. Thus, Eclipse is automatically compiling your code on-the-fly to create an instant view of the result. In this part, you need do thing manually.

### 4.1 Runable JAR

A runnable JAR is just like a EXE file in Windows, which means you can double click it to run. However, what your heap sort is doning is just outputting results into the console, so you still need the terminal/cmd/PowerShell to run it.

Select *File* => *Export...* => *Java* => *Runnable JAR file* ⇒ *Next*:

- *Launch configuration*: select your HeapSort;
- *Export destination*: type your target JAR file path and name;
- *Library handling*: choose the last option *Copy required libraries into a sub-folder next to the generated JAR*.

Click *Finish*, and you are ready to go:

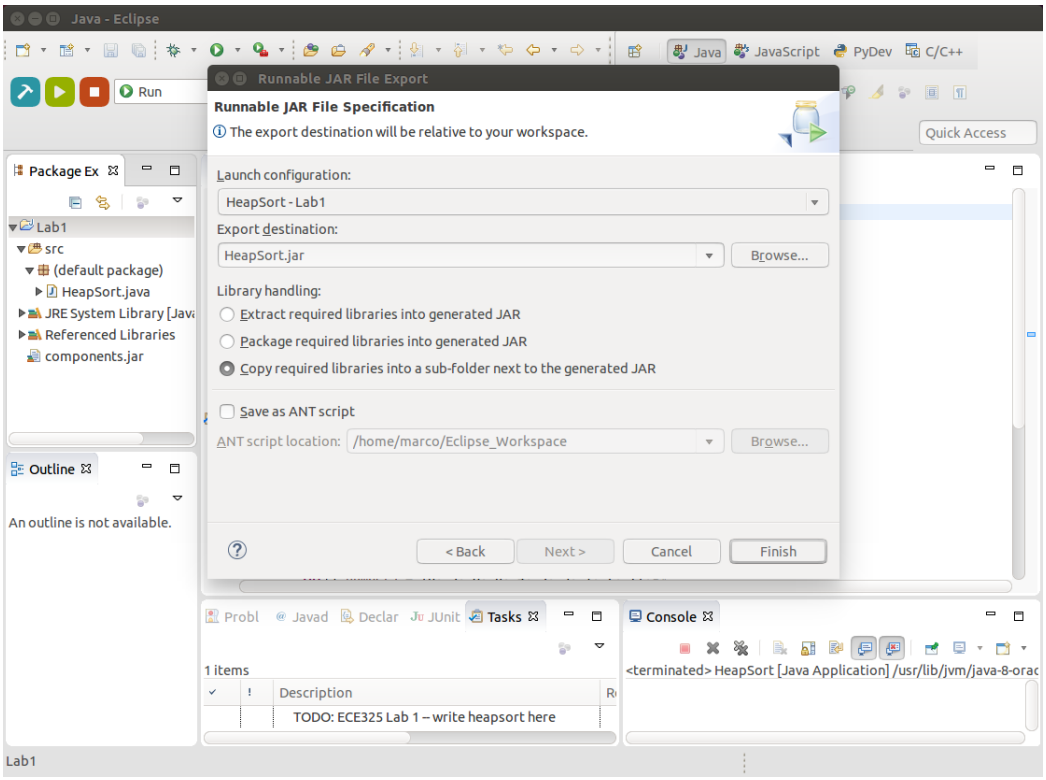
```
java –jar HeapSort.jar
```

### 4.2 Compile and Run by JDK

*Step 1: Copy your source codes to home directory*

```
cp src/HeapSort.java ~      # "~" refers to the home directory of your account
cp components.jar ~
```

Note: your HeapSort.java must use the `SimpleWriter` and `SimpleWriter1L`.




*Step 2: Configure the `CLASSPATH`*

JDK requires an important environmental variable `CLASSPATH` to compile and run codes. In Linux, you can do it as follows:

```
export CLASSPATH="your_class_paths"     # You need fill in the actual paths
echo $CLASSPATH                         # Print your class paths
```

In this lab, your have two groups of classes:

- Your own classes, which in this case is your `HeapSort`;
- The third party library classes, which is the components.jar.

The `CLASSPATH` must contain the paths of both groups. Try figure it out.

*Step 3: Compile and run*

```
javac HeapSort.java     # Now you can find the file HeapSort.class at your home directory
java HeapSort           # Do not add the ".class" extension
```

**DEMO this deliverable to the lab instructor.**