Java Basics Course - Lesson 1

Java

`JRE` Java Runtime Environment
`JDK` Java Development Kit
`JVM` Java Virtual Machine
`JAVA_HOME` Location where a JRE or JDK is installed

Java programs are compiled to byte code which requires the Java Virtual machine to run. The byte code is portable and can be executed in JVMs running on different hardware (e.g. x86, ARM).

File Types

`*.java` Text files containing the Java source code.
`*.class` Contains the bytecode to be executed by JVM.
`*.jar` A Java archive file, basically a zip-archive.
`MANIFEST.mf` Describes dependencies and main class.

Java programs are packaged into JAR files which contain the compiled byte code and required resources and a description in META-INF/MANIFEST.mf. The manifest file describes the dependencies and the main entry point to launch the program.

A First Example

Steps to writing the program: `HelloDuke.java`

- create new directory and cd into it `lesson-01/src`.
- create a new file `HelloDuke.java`.
- edit `HelloDuke.java` and enter follwing code:

```
1 public class HelloDuke {
2         public static void main(String[] args) {
3                 System
4                     .out
5                     .println("Want some coffee?");
6         }
7 }
```

Compiling and running the program

Java programs need to be compiled before starting them. Unlike with R, Java code cannot be executed directly. Run following code on a command line interface (CLI) to compile and run the code in `HelloDuke.java`.

```
1 > javac HelloDuke.java
2 > java HelloDuke
3 Want some coffee?
```

`javac` The Java compiler, translates source code into byte code.
`java` Launches the JVM with the given class.

Packaging the program - creating a JAR

Java programs are not distributed by class files, instead archives are used where all required class files are collected within one JAR file. With Java 8 a plain JAR file is created with:

```
1 # Java 8
2 > jar cf HelloDuke.jar HelloDuke.class
3
4 # Java 9
5 > jar --create --file HelloDuke.jar HelloDuke.class
```

Creating a runnable JAR

As JAR files may contain many classes, the JVM requires an information about which class to launch. This JAR file does not yet contain this information, so any attempt to execute the JAR will result in an error.

```
1 > java -jar HelloDuke.jar
2 kein Hauptmanifestattribut, in Hello.jar
```

It is possible to define a list of JAR files (separated by ;) using the `-classpath` argument.

```
1 > java HelloDuke -classpath Hello.jar;
2 Want some coffee?
```

It is also possible to point to a directory using the `-cp` argument where `-cp .` refers the current directory.

```
1 > java HelloDuke -cp .
2 Want some coffee?
```

The JAR command allows to define an entry point, the so called main class.

```
1 # Java 8
2 > jar cfe HelloDuke.jar HelloDuke HelloDuke.class
3
4 # Java 9
5 > jar --create --main-class=HelloDuke --file HelloDuke.jar
      HelloDuke.class
```

Executing a runnable JAR

A runnable JAR can be launched on command line using the `java` command. It is no longer needed to specify the classpath or the main class.

```
1 > java -jar HelloDuke.jar
2 Want some coffee?
```

Lets extract the JAR file and have a look at the `META-INF/MANIFEST.mf` file which was created during the package process:

```
1 # Java 8
2 > jar xf HelloDuke.jar
3 # Java 9
4 > jar -xf HelloDuke.jar
```

After extraction, listing `MANIFEST.mf` will reveal main class definition, which causes the JVM to find the proper entry point to launch the `HelloDuke.class`. Compiled with Java 8 the manifest file will look like:

```
1 # on Windows command line one could use:
2 > type META-INF\MANIFEST.MF
3 # on a Bash shell:
4 > cat META-INF/MANIFEST.MF
5 Manifest-Version: 1.0
6 Created-By: 1.8.0_152 (Oracle Corporation)
7 Main-Class: HelloDuke
```

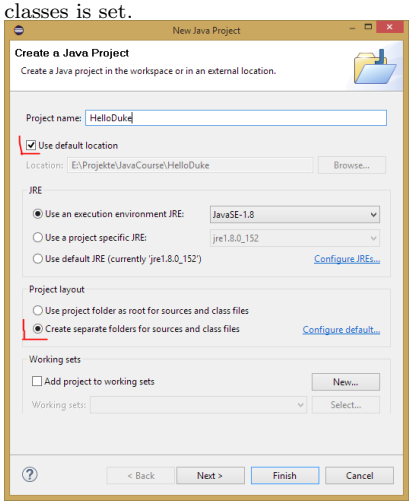
Beside the manifest, `HelloDuke.jar` contains a single file only, it is`HelloDuke.class`. The entry `Main-Class: HelloDuke` directly points to this single class file.

Creating an Eclipse project

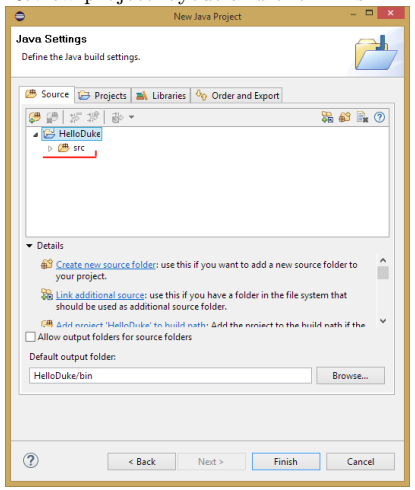
Setting up a new project

With the first start, Eclipse will ask you for a workspace location. A workspace is intended to contain one ore more projects. It can be any folder you like. When there are multiple workspaces, Eclipse will ask which workspace to open. Multiple instances of Eclipse can be started but a workspace can only opened by one Eclipse instance.

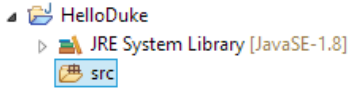
- Go to menu: `File / New Java Project`.
- Enter new project name `HelloDuke`.
- Ensure that in project layout, the option to create separate folders for sources and classes is set.



- Continue with next.
- Review project layout and click finish.

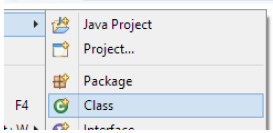


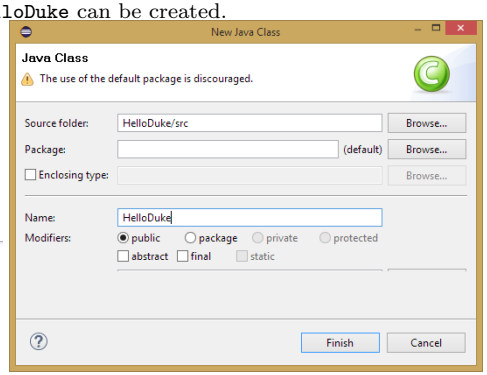
The new project will appear as follows in Eclipse Project Navigator.



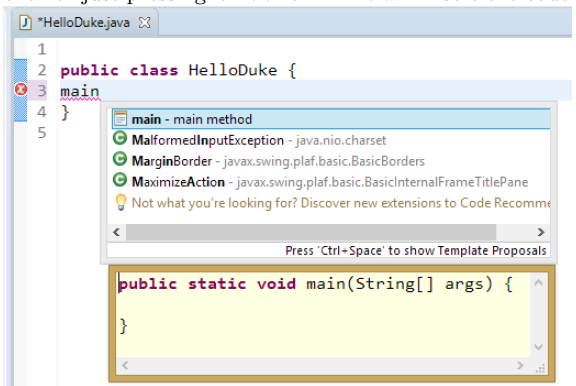
Creating and running the first Java program

Selecting thr project root and opening the context menu, new objects can be created. So here a new class with the name `HelloDuke` can be created.





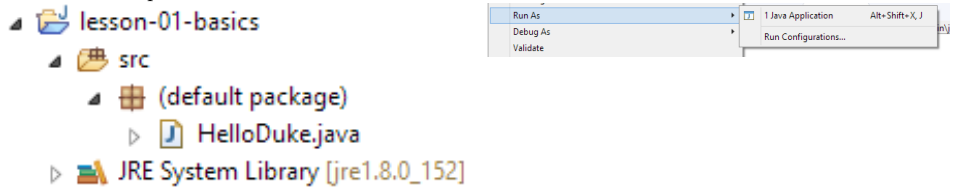
As soon as the class was created it can be edited in Eclipses Java code editor. The editor offers powerful code completion. As the main class skeleton was created automatically, it is sufficient to just type `main` and then press `CTRL+SPACE` to bring up code completion. Eclipse will offer automatic creation of a static main method, which is required to execute the program. Double click or just pressing `RETURN` or `ENTER` will insert the code.



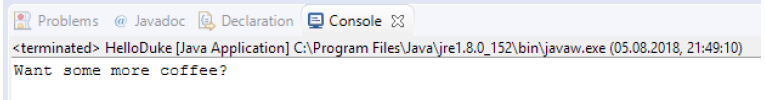
After completing the code as in the example, the program can be launched either using context menu on `HelloDuke.java` or via manual configuration. Here program execution via context menu is shown.

```
1
2 public class HelloDuke {
3    public static void main(String[] args) {
4       System.out.println("Want some more coffee?");
5    }
6 }
7
```

One should note, that `HelloDuke.java` is located inside the `Default Package`. Packages are an essential concept and are covered in Lesson 2.



Select `HelloDuke.java` then open the context menu, go to `Run As` then select `1 Java Application`. The program will be executed and the console view will show the desired result.



After first execution of `HelloDuke.java` a `Run Configuration` is created. This configuration can be reused for repeated execution and for JAR file creation.

Exporting a runnable JAR

- Select the project root and open its context menu OR open the file menu - in both cases select the `Export` command.
- Then select the `Runnable JAR file` from `Java` section, continue with `next.`
- There will be one launch configuration with the name `HelloDuke - HelloDuke`.
- An `export destination` must be provided, usually the project root folder with a suitable name such as `lesson-01-basics/HelloDuke.jar`.
- Click on `Finish` and verify that the JAR file starts (double click on Windows or just run java command on command line).