Java Fundamentals

Lesson 3: Creating a Java Main Class

Speaker: Nicolae Sîrbu
Alexandru Umaneț

Lesson Objectives

- Define the structure of a Java class
- Create executable Java applications with a `main` method
- Run a Java program from your IDE and from the command line
- Use `System.out.println`to write a `String` literal to system output

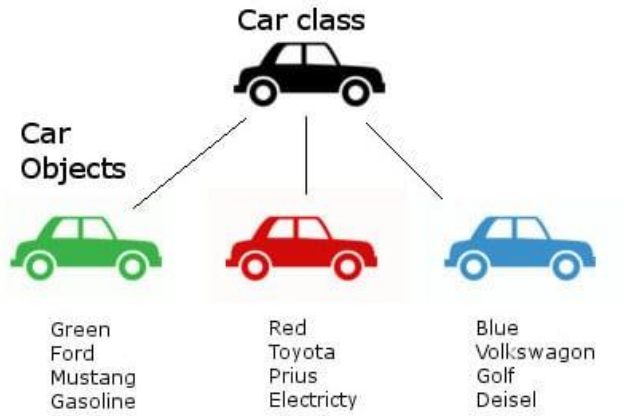
Java classes

Java Classes

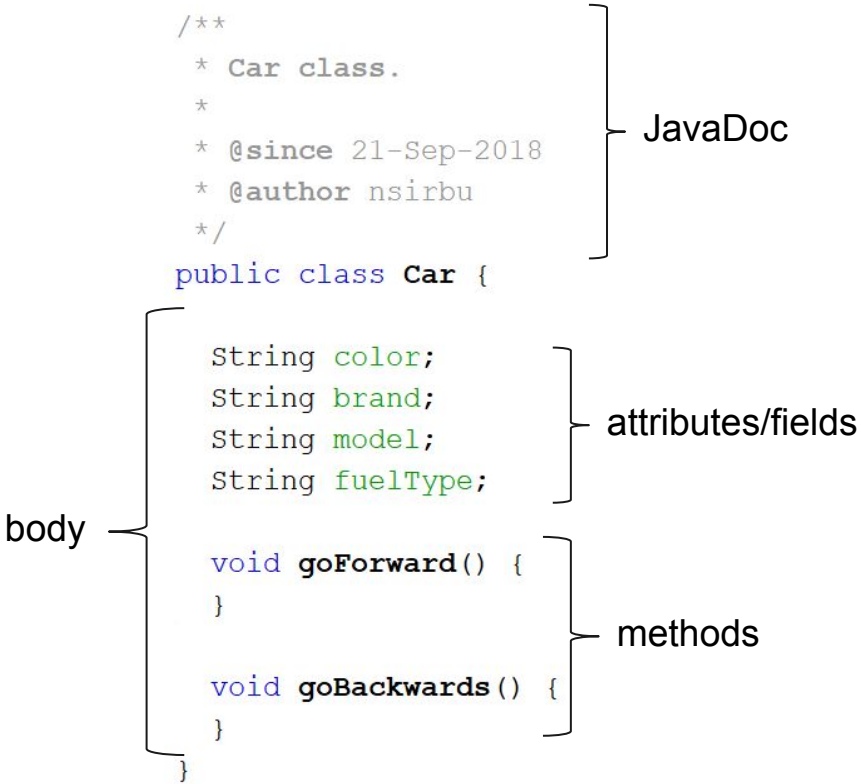
A **class** is a design used to specify the **attributes** and **behavior** of an object.

The **attributes/fields** of an object are implemented using ***variables***.

The **behavior** of an object is implemented using ***methods***.



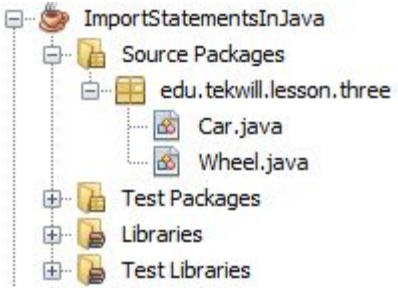
Class Structure



Definition of a class in a Java code file

When you define a *public* class in a Java source file, the name of the class and Java source file must match.

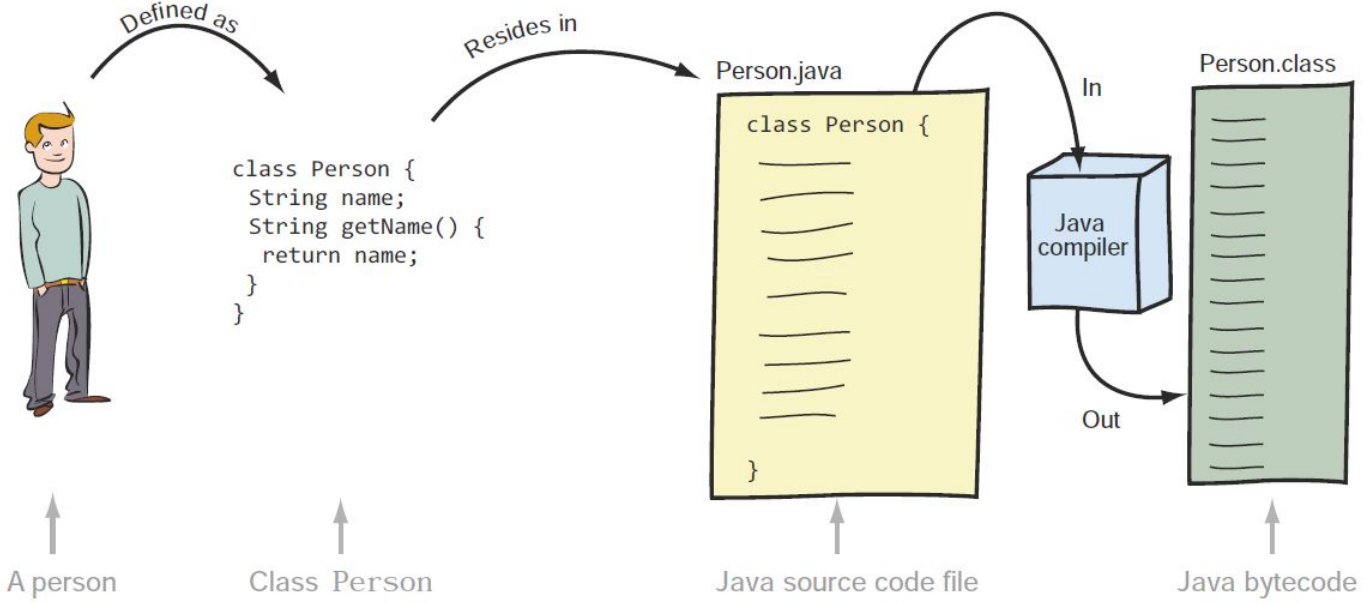
A source code file can't define more than one public class. If you try to do so, your code won't compile,




```
package edu.tekwill.lesson.three;

public class Car {
  Wheel[] wheels;
}
```

Java source code file vs Java bytecode file



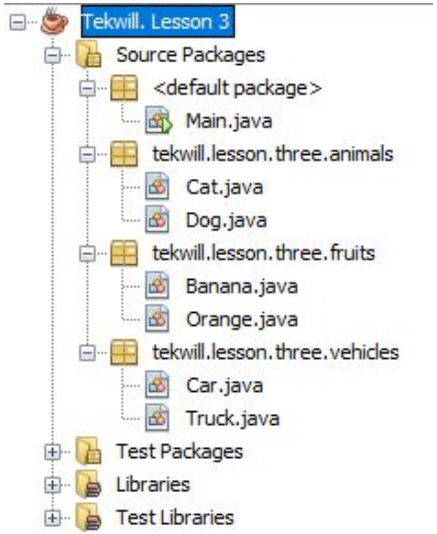
Relationship between the class file `Person` and the files Person.java and Person.class and how one transforms into another

Java packages

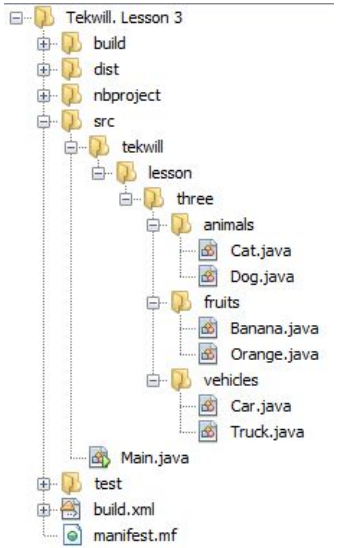
Java Packages

All Java classes are part of a package.

A Java class can be explicitly defined in a named package; otherwise, it becomes part of a default package, which doesn't have a name.



Classes structured
in packages in NetBeans



Classes structured
in packages in file system

Java Packages

```java
package tekwill.lesson.three.vehicles;

/**
 * Car class.
 *
 * @since 21-Sep-2018
 * @author nsirbu
 */
public class Car {

}
```

Java Packages

Example:

```
com.oracle.javacert.associate
```

Package or subpackage name	Its meaning
com	Commercial. A couple of the commonly used three-letter package abbreviations are ▪ gov—for government bodies ▪ edu—for educational institutions
oracle	Name of the organization
javacert	Further categorization of the project at Oracle
associate	Further subcategorization of Java certification

Java Packages

Here are a few of important rules about packages:

- Per Java naming conventions, package names should all be in lowercase.
- The package and subpackage names are separated using a dot.
- For classes and interfaces defined in a package, the package statement is the first statement in a Java source file. The exception is that comments can appear before or after a package statement.
- There can be a maximum of one package statement per Java source code file.

Comments in Java code

Commenting Java code

Comments come in two flavors:

multiline comments

```
class MyClass {
    /*
      comments that span multiple
      lines of code
    */
}
```

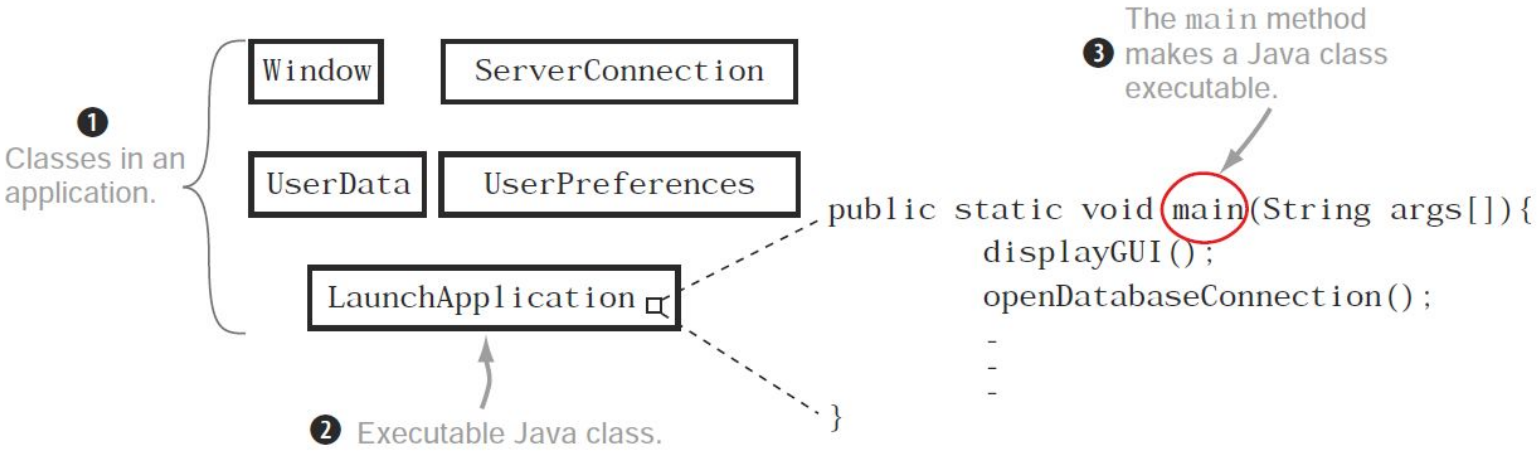
```
class MyClass {
    /*
     * comments that span multiple
     * lines of code
     */
}
```

end-of-line comments

```
package uni;              // package uni
class Monument {
    int startYear;
    String builtBy;     //  architect
}
// another comment
```

The `main` method

Executable Java classes versus non-executable Java classes



Class `LaunchApplication` is an executable Java class, but the rest of the classes—`Window`, `UserData`, `ServerConnection`, and `UserPreferences`—aren't.

Executable Java classes versus non-executable Java classes

A Java application can define more than one executable class.

We choose one (and exactly one) when the time comes to start its execution by the JVM.

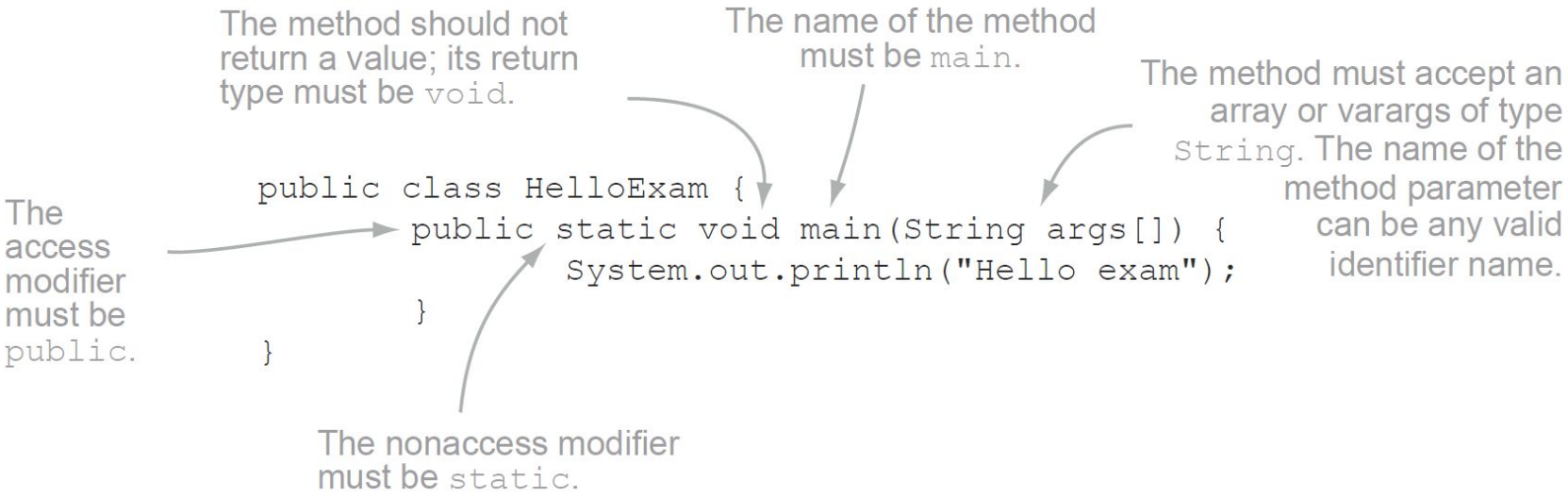
The `main` method

It is a special method that the JVM recognizes as the starting point of any Java program.

This `main` method should comply with the following rules:

- The method must be marked as a `public` method.
- The method must be marked as a `static` method.
- The name of the method must be `main`.
- The return type of this method must be `void`.
- The method must accept a method argument of a `String` array or a variable argument of type `String`.

A `main` class example



The `main` method

The method must accept a method argument of a `String` array or a variable argument of type `String`.

```
public static void main(String[] args)
public static void main(String args[])
public static void main(String... args)
```

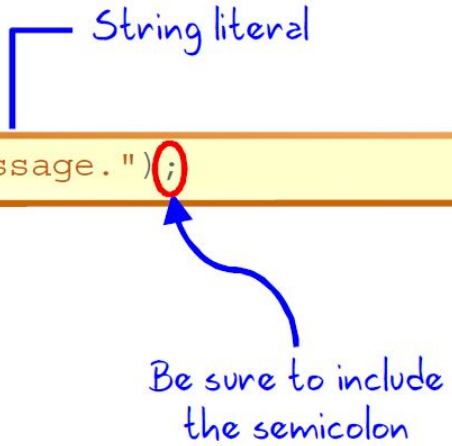
Output to the Console

Syntax:

```
System.out.println (<some string value>);
```

Example:

```
System.out.println ("This is my message.");
```

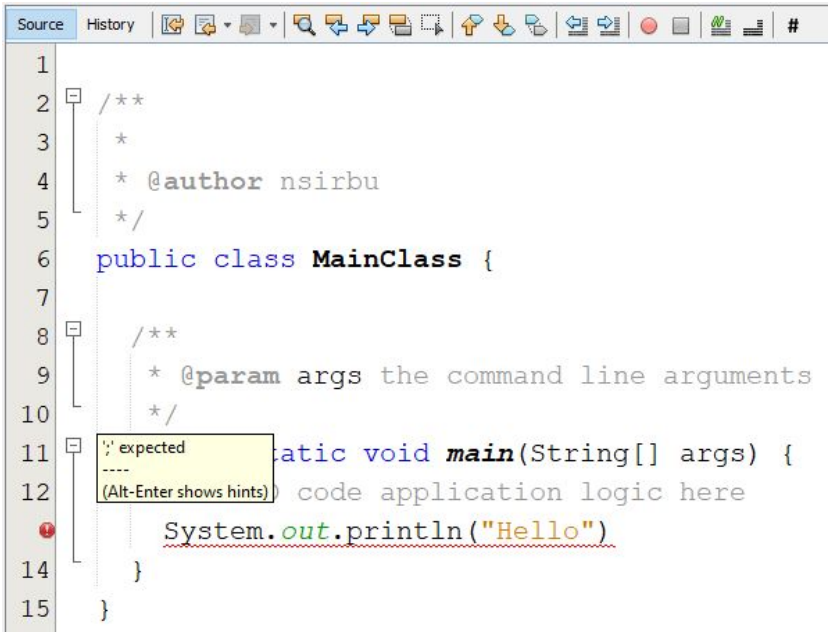


Avoiding syntax errors

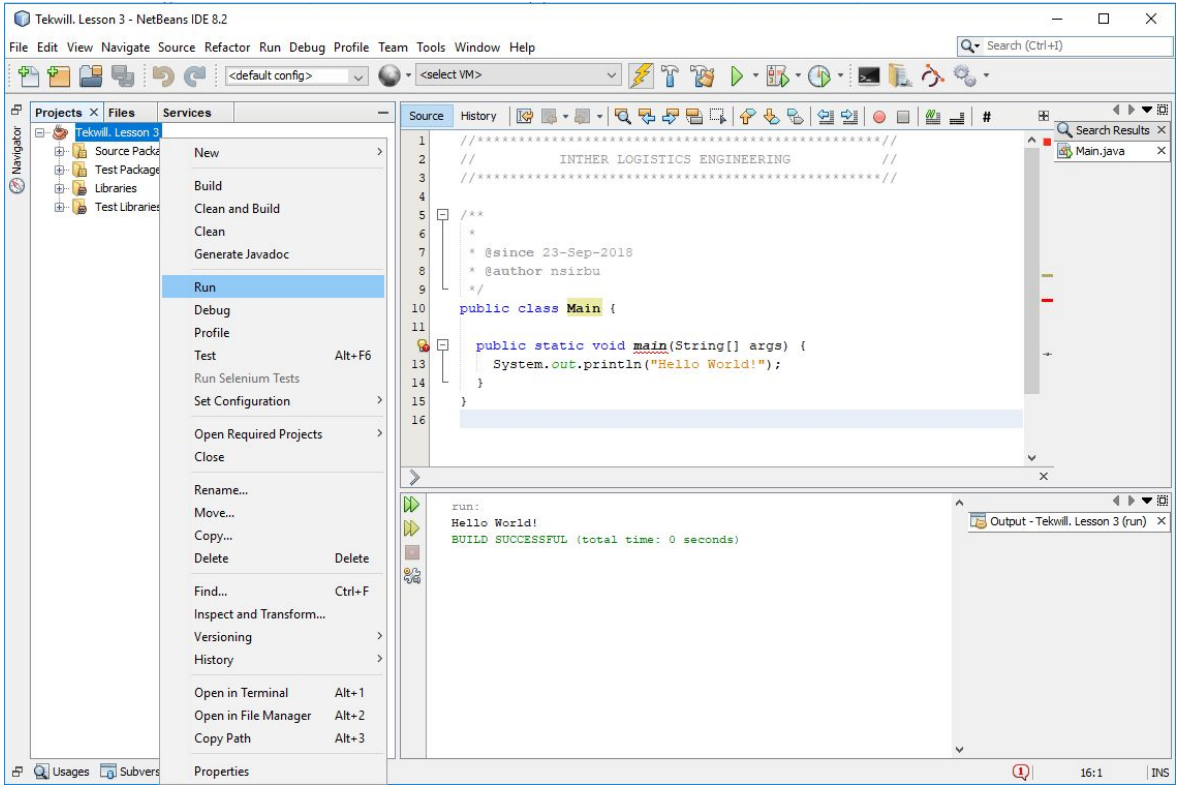
NetBeans will tell you if you have done something wrong.

Common errors include:

- Unrecognized words (case-sensitivity errors)
- Missing close quotation mark - “
- Unmatched brace - (}
- Missing semicolon - ;



Compiling and Running using NetBeans



Exercise #3.1 Creating the `main` method

1. Open NetBeans.
2. Create a new Java project. Deselect the box to create the `main` method.
3. Create a new class in the default package.
4. In the code editor add the method *main* to the class you created:

```
public static void main(String[] args) { … }
```

5. In the body of the `main` method, use a `System.out.println` to print the following message: `"Welcome to the main method of my application!"`.
6. Build your program.
7. Run the program.

Exercise #3.2 Running a Java program from CMD

Running a class that has the `main` method from command line, requires you to do the following operations:

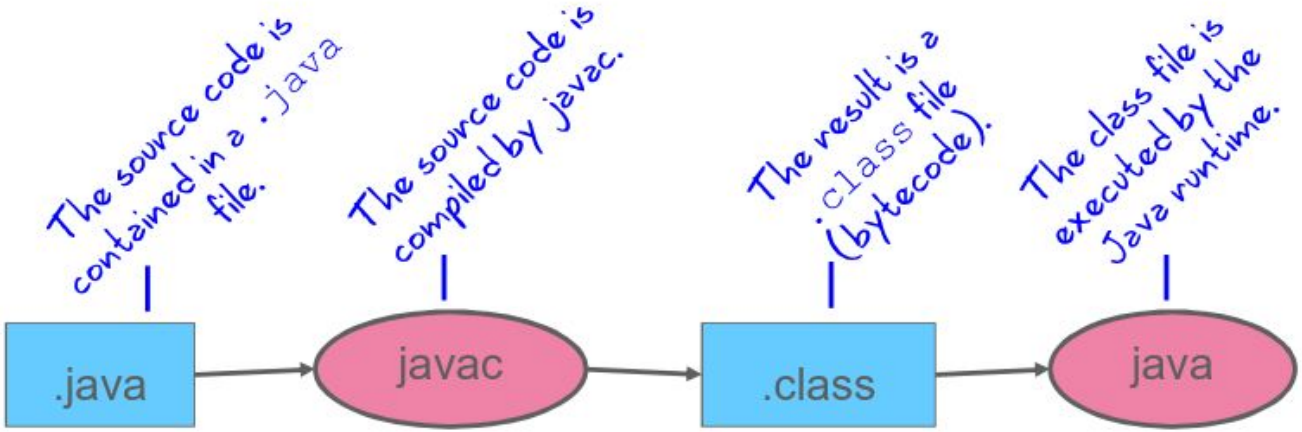
1. First of all, you need to use the `javac` command to compile the class:

```
javac HelloWorld.java
```

2. Then you can use the `java` command to run it:

```
java HelloWorld
```

Compiling and Running a Java Program



Exercise #3.3 Passing parameters to the `main` method

1. Continue editing Exercise #3.1.
2. In the code editor in the `main` method, print the first 2 parameters from the `args` variable, use `System.out.println(args[0])` and `System.out.println(args[1])`.
3. Use `javac` command to compile the class.
4. Use `java` command to run it. Pass 2 parameters to the main method from CMD:

```
java HelloWorld param1 param2
```

* **Note**: The method parameters that are passed to the main method are also called *command-line parameters* or *command-line values*.

Exercise #3.4 Adding the project to GitHub

1. Add the project you created in Exercise #3.1 into your Git repository.

* **Note**: Recall the purpose of the `.gitignore` file. Use it to exclude everything except `src` and `nbproject` folders from your commit.

Resources

Defining a Class
(http://journals.ecs.soton.ac.uk/java/tutorial/getStarted/application/classdef.html)

Packages In Java
(https://www.geeksforgeeks.org/packages-in-java/)

Java main() Method Explained
(https://www.baeldung.com/java-main-method)

How to Compile and Run your First Java Program
(https://beginnersbook.com/2013/05/first-java-program/)

Java Fundamentals

Lesson 3: Creating a Java Main Class

End.

Speaker: Nicolae Sîrbu
Alexandru Umaneț