Learning Java Output

Jackson Daumeyer

August 27, 2021

Contents

1 Introduction to Java

1.1 What is Java

Java is a fairly popular *object-oriented* programming language.

- Java was programmed in the 1990's by Sun Microsystems.
- It was designed to be an easier to understand object-oriented programming language.
- Java Applets used to be a thing, why are we stil talking about them?

1.2 Compilation

The Java Compiler runs thorugh the source code of a `.java` file and generates bytcode from it into a `.class` file.

- Eclipse using it's own compilier that continuously compiles code as it's being written.
- `javac` is the compilation command included in the **JDK**

1.3 Java Virtual Machine

The **Java Virtual Machine** is essentially a virtual computer.

- The **JVM** is like any other program that runs on a physical chip.
- When you run a java (`.class`) file, your computer creates an instance of this virtual computer and runs your code on it.
- The **JVM** only runs bytecode generated from a Java file.

The **JVM** is super significant the the **Java** progamming language because it follows the **Java** slogan of "Write once, run anywhere."

- The **JVM** is also universal, allowing other programming languages like *Clojure* to compile to the **JVM**.
- The **JVM** is also very performance friendly, the extra layer it adds to **Java** does not cost a lot of resources.

2 Writing a Java Program

2.1 Output, Strings and Input

```
public class HelloWorld {
    public static void main (String[] args) {
        System.out.print("Hello, World!");
    }
}
```

Figure 1: This is the simplest Java Program to print "Hello, World!"

```
Hello, World!
```

- `public class` are two keywords that declare this component as something bytecode should be generated for.
- Inside of the class, `public static void` declares a method, the name of the method `main` means that it will be executed when the program is run.
- The `String[] args` means that the **method** has **parameters**, in this case the **parameters** are *command-line arguments*.
- `System.out` is an **object** and `.println()` calls a **method** of the object.

2.1.1 Another "Hello, World!" Program

```
import compenents.simplewriter.SimpleWriter;
import compents.simplewriter.SimpleWriter1L;

public final class HelloWorld {
    private HelloWorld() {
    }

    public static void main(String[] args) {
        SimpleWriter out = new SimpleWriter1L();
        out.println("Hello, World!");
        out.close();
    }
}
```

This version of the "Hello, world!" program uses **OSU**'s self developed component library. This was developed based on real world engineering systems. We will go into this in more depth later

2.1.2 SimpleWriter

When creating an instance of a `SimpleWriter1L`, if you pass a file name then all output will be sent to that file. However, when left blank, the default behavior of console output is used.

2.2 Object Oriented Methodollogy

In **Java** there are two very imporant concepts; **Classes** and **Objects**

2.2.1 Classes

Classes can be thought of as a *mold*. This mold can be used to create **objects**. When an **Object** is created from a **Class** it is reffered to as a **Child** of the class.

- **Classes** can define certain properties of the **Objects** created from them. They can also leave properties for their **children** to define themselves later.
- New **classes** can also be created based on another **class**, this process is called **Inheritance**

2.2.2 Objects

Objects are essentially boxes that hold **methods** and **properties**. These can be called upon, changed or accessed like any data but they are associated with the object that contains them.