<< Lecture 2 >>

Basics of Java

Today's plan:

- Learn about the place of Java in programming languages and the Java Virtual Machine
- Learn about Java's syntax, basic types, control flow, and classes/modularity
- Learn what the IDE does and why it is useful

The Java Programming Language

- It was developed by Sun Microsystems, which was acquired by Oracle
- Still, the language is mostly open (GNU/GPL)
- Basic workflow:
  - Java source code compiled to Java bytecode
  - Javabyte code run by **Java Virtual Machine**
- The Java Virtual Machine or **JVM** allows the same binaries to run on different machines and operating systems
- Generalities:
  - Compiled language
  - Object Oriented
  - Strongly typed
- Compared to other languages:
  - **Python** is OO, interpreted, weakly typed
  - **C** is imperative, compiled, strongly typed
  - **C++** is OO*, compiled, strongly typed
  - **Haskell** is functional, compiled*, strongly typed
- We will be using **Java SE**
- **Java SE** stands for **Java Standard Edition**
- **Java EE** stands for **Java Enterprise Edition**
- **Java ME** stands for **Java Micro Edition**
- **jdk** stands for **Java Development Kit**

- **jre** stands for **Java Runtime Environment**
  - jre small subset of jdk
  - jre is enough to run Java programs
- **.java**: extension of Java source code
- **.class**: extension of Java bytecode
- **.jar**: extension of a packaged Java application or library (a project in general)

Hello World in Java

I will demonstrate how to compile and run the following code from the Windows command line. (I use the **java** and **javac** tools here.)

```
/********** Demo.java **********/

class Demo {

  public static void main(String[] x) {
    System.out.println("Hello World!");
  }

}
```

Why is an IDE useful?

- Edition, compilation, execution, all at once
  - No need to use so many programs
- **Netbeans is the official IDE of Java**
  - Can use Eclipse, IntelliJ as well
- **Packaging is easier!** We can create/deploy **jar** files in an easier manner

An "imperative" introduction to Java basics

- Basics of syntax:
  - Blocks surrounded by brackets **{...}**
    - *unless single instruction*
  - Instructions end in semicolons
  - Indentation, spaces, newlines are ignored (unlike Python)

- Parameters to functions in parentheses

- Every method, variable belongs inside a class

```
/********** Demo.java **********/
class Demo {
  public static void main(String[] x) {
    // we define variable n below
    int n = 30 ;
    // we update n
    n = n + 2 ;
    // we print it along with “Hello”
    System.out.println(“Hello ” + n);
  }
}
```

- **int n = 30 ;** means that we created variable **n** that is **of type integer** and has value **30**
- **The type of a variable cannot change**
- Primitive types:
  - **byte**, short, **int**, long: integers
  - float, **double**: decimals
  - **boolean**: Boolean (true/false) value
  - **char**: **single** character (16 bit, Unicode)

```
/********** Demo.java **********/
class Demo {
  public static void main(String[] x) {

    // defining several variables at once
    int base = 10, area, height = 15 ;
    area = base * height / 2 ;

    System.out.println(“Area ” + area);
  }
}
```

- **public static void main**
  - It is a method called **main**, that
  - is **public**, which is why we can call it from outside the class (e.g., command line)
  - is **static**, which is why we can just use the method/function
  - has return type **void**, i.e., it does not return anything

```
/********** Demo.java **********/
class Demo {

  // the following method computes the
  // area of a triangle
  static int getArea(int x,int y) {
     return x*y/2;
  }

  public static void main(String[] x) {
    int base = 10, area, height = 15 ;
    area = getArea(base, height);
    System.out.println("Area " + area);
  }
}
```

- We just created a method (function) called **getArea** that is static but not public, and **return**s an **int** value
- **Modularity is good**
  - Allows us to give **meaningful names** to portions of the code
  - Allows **reusability**
  - Allows for **divide and conquer**

OPERATIONS WITH BASIC TYPES

- *Operations with numbers:*
  addition (+), subtraction (–), multiplication (*), division (/), modulo or remainder (%).
- *Comparing primitive values (generate **boolean**):*
  ***value equality (==)**, strict inequalities (<, >), non strict inequalities (<=, >=)*
- *Operations with Booleans:*
  and (&&), or (||), not (!)

<< The lecture ended here—I will resume next class on this point >>