Java Notes

Residence Tutors

by: Mbongeni

Basics

Java is one of the most popular programming languages in the world. With Java you can build various types of applications such as desktop, web, mobile apps and distributed systems.

Java Development Kit

We use Java Development Kit (JDK) to build Java applications. JDK contains a compiler, the Java Runtime Environment (JRE) and a library of classes that we use to build applications.

How Java Code Gets Executed

The Java compiler takes Java code and compiles it down to Java Bytecode which is a cross-platform format. When we run Java applications, Java Virtual Machine (JVM) gets loaded in the memory. It takes our bytecode as the input and translates it to the native code for the underlying operating system. There are various implementations of Java Virtual Machine for almost all operating systems.

Architecture of Java Applications

The smallest building blocks in Java programs are **methods** (also called functions in other programming languages). We combine related methods in **classes**, and related classes in **packages**. This modularity in Java allows us to break down large programs into smaller building blocks that are easier to understand and re-use. **Which you will learn in Principles of Programming B**

HOW TO NAME YOUR CLASSES AND METHODS IN JAVA

Camel casing: In camel casing, the first letter of each word is capitalized except for the first word, and there are no spaces between words. For example: camelCaseExample. This convention is commonly used for naming variables, methods, and parameters in Java.

Pascal casing: In Pascal casing the first letter of each word, including the first one, is capitalized, and there are no spaces between words. For example: PascalCaseExample. This convention is often used for naming classes in Java.

These naming conventions are important for several reasons:

Readability: Consistent naming conventions make code easier to read and understand for other programmers who might work on the code. When variables, methods, or classes are named in a consistent manner, it becomes easier to grasp their purpose and function.

Consistency: Using the same naming convention throughout a code helps maintain consistency. It ensures that different parts of the code look and feel similar, making it easier for programmers to navigate and understand the code.

Data Types

Variables

We use variables to temporarily store data in a computer's memory. In Java, the type of a variable should be specified at the time of declaration.

In Java, we have two categories of types:

- **Primitives:** for storing simple values like numbers, strings and booleans.
- **Reference Types:** for storing complex objects like email messages.

Primitive Types

Type	Bytes	Range
byte	1	[-128, 127]
short	2	[-32K, 32K]
int	4	[-2B, 2B]
long	8	
float	4	
double	8	
char	2	A, B, C, …
boolean	1	true / false

Declaring Variables

When we declare variables in java, please make use of the identifiers to follow the correct naming conversions which is the camelCasing naming conversion, this will help you be able to trace your variables in your program and be able to also identify as to which data type does the variable belong to:

```
int iAge = 30;
double rPrice = 10.99;
char cLetter = 'A';
boolean bIsEligible = true;
```

- In Java, we terminate statements with a semicolon
- We enclose characters with single quotes ‘ ‘, and strings (sequence of characters) with double quotes “ “.
- The default integer type in Java is int. To represent a long value, we should add L to it as a postfix.
- The default floating-point type in Java is double. To represent a float, we should append F to it as a postfix.

Comments

We use comments to add notes to our code.

```
// This is a comment and it won’t get executed.
```

Reference Types

In Java we have 8 primitive types. All the other types are reference types. These types don’t store the actual objects in memory. They store the reference (or the address of) an object in memory.

Strings

Strings are reference types . We can declare string variables like the primitives since we use them a lot.

```
String sName = “Mbongeni”;
```

Constants

Constants (also called final variables) have a fixed value. Once we set them, we cannot change them.

```
final double INTEREST_RATE = 0.04;
```

By convention, we use CAPITAL LETTERS to name constants. Multiple words can be separated using an underscore.

Arithmetic Expressions

```
int x = 10 + 3;
int x = 10 - 3;
int x = 10 + 3;
int x = 10 / 3                    // returns an int
float x = (float)10 / (float)3;   // returns a float
int x = 10 % 3;    // modulus (remainder of division)
```

Increment and Decrement Operators

```
int x = 1;
x++;    // Equivalent to x = x + 1
x--;    // Equivalent to x = x - 1
```

Augmented Assignment Operator

```
int x = 1;
x += 5;  // Equivalent to x = x + 5
```

Order of Operations

Multiplication and division operators have a higher order than addition and subtraction. They get applied first. We can always change the order using parentheses.

```
int x = 10 + 3 * 2;        // 16
int x = (10 + 3) * 2;      // 26
```

Reading Input From The Scanner
To obtain user input in java, you can use a class called Scanner. This class allows you to read input from a variety of sources, including the console or user input.

So the scanner is a class we use to create an instance (Object) that will take user input.

To create a new Scanner object, specifying where you want to read input from, example:

```
Scanner scanner = new Scanner(System.in);
```

We use the below scanner methods to specify what kind of user input is to be entered through the object

```
double number     = scanner.nextDouble();
byte  number      = scanner.nextByte();
String name       = scanner.next();
String line       = scanner.nextLine();
```