**JAVA CONCEPTS**

**what is main method will do?**

In Java, **main** is a static **method**. This means the **method** is part of its class and not part of objects

public static void main(String args[])

The method signature for the main() method contains three modifiers:

* public indicates that the main() method can be called by any object.
* main method in Java is public so that its visible to every other class, even which are not part of its package. if its not public JVM classes might not able to access it.
* static indicates that the main() method is a class method.
* main method is static in Java, so that it can be called without creating any instance. While JVM tries to execute Java program it doesn't know how to create instance of  main class as there is no standard constructor is defined for main class.
* void indicates that the main() method has no return value. main method is void in Java because it doesn't return any thing to caller which is JVM .

when you execute a class with the Java interpreter, the runtime system starts by calling the class's main() method. The main() method then calls all the other methods required to run your application. If you try to run a class with the Java interpreter that does not have a main() method, the interpreter prints an error message.

public static void main(**String args[]**)

* the main() method accepts a single argument: an array of Strings.
* This array of Strings is the mechanism through which the runtime system passes information to your application. Each String in the array is called a *command line argument*.

**what is data type and different data types**

a **data type** or simply **type** is a classification of **data** which tells the compiler or interpreter how the programmer intends to use the **data**. Most programming languages support various **types** of **data**, for example: real, integer or Boolean. Is a classification that specifies which type of value a variable has and what type of mathematical, relational or logical operations can be applied to it without causing an error.

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in the memory.

Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals, or characters in these variables.

There are two data types available in Java −

* Primitive Data Types
* Reference/Object Data Types

## **Primitive Data Types**

There are eight primitive datatypes supported by Java. Primitive datatypes are predefined by the language and named by a keyword. Let us now look into the eight primitive data types in detail.

### **byte**

* Byte data type is an 8-bit signed two's complement integer
* Minimum value is -128 (-2^7)
* Maximum value is 127 (inclusive)(2^7 -1)
* Default value is 0
* Byte data type is used to save space in large arrays, mainly in place of integers, since a byte is four times smaller than an integer.
* Example: byte a = 100, byte b = -50

### **short**

* Short data type is a 16-bit signed two's complement integer
* Minimum value is -32,768 (-2^15)
* Maximum value is 32,767 (inclusive) (2^15 -1)
* Short data type can also be used to save memory as byte data type. A short is 2 times smaller than an integer
* Default value is 0.
* Example: short s = 10000, short r = -20000

### **int**

* Int data type is a 32-bit signed two's complement integer.
* Minimum value is - 2,147,483,648 (-2^31)
* Maximum value is 2,147,483,647(inclusive) (2^31 -1)
* Integer is generally used as the default data type for integral values unless there is a concern about memory.
* The default value is 0
* Example: int a = 100000, int b = -200000

### **long**

* Long data type is a 64-bit signed two's complement integer
* Minimum value is -9,223,372,036,854,775,808(-2^63)
* Maximum value is 9,223,372,036,854,775,807 (inclusive)(2^63 -1)
* This type is used when a wider range than int is needed
* Default value is 0L
* Example: long a = 100000L, long b = -200000L

### **float**

* Float data type is a single-precision 32-bit IEEE 754 floating point
* Float is mainly used to save memory in large arrays of floating point numbers
* Default value is 0.0f
* Float data type is never used for precise values such as currency
* Example: float f1 = 234.5f

### **double**

* double data type is a double-precision 64-bit IEEE 754 floating point
* This data type is generally used as the default data type for decimal values, generally the default choice
* Double data type should never be used for precise values such as currency
* Default value is 0.0d
* Example: double d1 = 123.4

### **boolean**

* boolean data type represents one bit of information
* There are only two possible values: true and false
* This data type is used for simple flags that track true/false conditions
* Default value is false
* Example: boolean one = true

### **char**

* char data type is a single 16-bit Unicode character
* Minimum value is '\u0000' (or 0)
* Maximum value is '\uffff' (or 65,535 inclusive)
* Char data type is used to store any character
* Example: char letterA = 'A'

## **Reference Datatypes**

* Reference variables are created using defined constructors of the classes. They are used to access objects. These variables are declared to be of a specific type that cannot be changed. For example, Employee, Puppy, etc.
* Class objects and various type of array variables come under reference datatype.
* Default value of any reference variable is null.
* A reference variable can be used to refer any object of the declared type or any compatible type.
* Example: Animal animal = new Animal("giraffe");

## **Java Literals**

A literal is a source code representation of a fixed value. They are represented directly in the code without any computation.

Literals can be assigned to any primitive type variable. For example −

byte a =68;

char a ='A'

byte, int, long, and short can be expressed in decimal(base 10), hexadecimal(base 16) or octal(base 8) number systems as well.

Prefix 0 is used to indicate octal, and prefix 0x indicates hexadecimal when using these number systems for literals. For example −

intdecimal=100;

int octal =0144;

int hexa =0x64;

String literals in Java are specified like they are in most other languages by enclosing a sequence of characters between a pair of double quotes. Examples of string literals are −

### **Example**

"Hello World"

"two\nlines"

"\"This is in quotes\""

String and char types of literals can contain any Unicode characters. For example −

char a ='\u0001';

String a ="\u0001";

Java language supports few special escape sequences for String and char literals as well. They are −

|  |  |
| --- | --- |
| **Notation** | **Character represented** |
| \n | Newline (0x0a) |
| \r | Carriage return (0x0d) |
| \f | Formfeed (0x0c) |
| \b | Backspace (0x08) |
| \s | Space (0x20) |
| \t | Tab |
| \" | Double quote |
| \' | Single quote |
| \\ | backslash |
| \ddd | Octal character (ddd) |
| \uxxxx | Hexadecimal UNICODE character (xxxx) |

**What is variable?**

In programming, a variable is a value that can change, depending on conditions or on information passed to the program. They also provide a way of labeling data with a descriptive name, so our programs can be understood more clearly by the reader and ourselves. It is helpful to think of variables as containers that hold information. Their sole purpose is to label and store data in memory. This data can then be used throughout your program.

# **Java - METHODS**

A Java method is a collection of statements that are grouped together to perform an operation. When you call the System.out.**println()** method, for example, the system actually executes several statements in order to display a message on the console.

Now you will learn how to create your own methods with or without return values, invoke a method with or without parameters, and apply method abstraction in the program design.

## **Creating Method**

Considering the following example to explain the syntax of a method −

**Syntax**

public static int methodName(int a, int b) {

// body

}

Here,

* **public static** − modifier
* **int** − return type
* **methodName** − name of the method
* **a, b** − formal parameters
* **int a, int b** − list of parameters

Method definition consists of a method header and a method body. The same is shown in the following syntax −

**Syntax**

modifier returnType nameOfMethod (Parameter List) {

// method body

}

The syntax shown above includes −

* **modifier** − It defines the access type of the method and it is optional to use.
* **returnType** − Method may return a value.
* **nameOfMethod** − This is the method name. The method signature consists of the method name and the parameter list.
* **Parameter List** − The list of parameters, it is the type, order, and number of parameters of a method. These are optional, method may contain zero parameters.
* **method body** − The method body defines what the method does with the statements.

<http://www.gcreddy.com/2016/02/user-defined-methods-in-java.html>

<http://www.gcreddy.com/2017/02/java-user-defined-methods.html>