DAY 1

1. Structure of Java program
2. Basic data types
3. Variables

- **HISTORY OF JAVA**

Java is an object-oriented programming language. Java was developed by James Gosling and colleagues at Sun Microsystems in the 1990s.

Java was started as a project called “Oak” by James Gosling in June 1991. The goal was to implement a virtual machine that is much simpler than C/C++.

Java was developed with the goal to implement “Write Once, Run Anywhere” programming model.

The Java programming language was built on the following five philosophies.

- The same program should be executable on multiple operating systems.
- Built-in support for using computer networks.
- Designed to execute code from the remote sources securely.
- It should be easy to use, take the good features of Object-oriented

STRUCTURE OF JAVA PROGRAM

Java Program

Let us look at a simple code that will print the words **Hello World**.

```
public  class myfirstjavaprogram  {


    /* This my first java program
     *  This will print Hello World
     */


     public static void main(String args[])  {
     System.out.println("Hello World"); // prints Hello World
     }
}
```

Let's look at how to save the file, compile, and run the program.
Please follow the subsequent steps:

- Open notepad and add the code as above.
- Save the file as: MyFirstJavaProgram.java.
- Open a command prompt window and go to the directory where you saved the class. Assume it's C:\.

Type 'javac MyFirstJavaProgram.java' and press enter to compile your code. If there are no errors in your code, the command prompt will take you to the next line (Assumption : The path variable is set).

- Now, type ' java MyFirstJavaProgram ' to run your program.
- You will be able to see ' Hello World ' printed on the window.

```
C:\> javac MyFirstJavaProgram.java
C:\> java MyFirstJavaProgram
Hello World
```

Basic Syntax

About Java programs, it is very important to keep in mind the following points.

Case Sensitivity - Java is case sensitive,which **Hello** and **hello** would have different meaning in Java.

Class Names - For all class names the first letter should be in Upper Case.

If several words are used to form a name of the class, each inner word's first letter should be in Upper Case.

Example: class MyFirstJavaClass

Method Names - All method names should start with a Lower Case letter.

If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case.

Example: public void myMethodName()

Program File Name - Name of the program file should exactly match the class name.

When saving the file, you should save it using the class name (Remember Java is case sensitive) and append '.java' to the end of the name (if the file name and the class name do not match, your program will not compile).

Example: Assume 'MyFirstJavaProgram' is the class name. Then the file should be saved as 'MyFirstJavaProgram.java'

public static void main(String args[]) - Java program processing starts from the main() method which is a mandatory

Java Identifiers

All Java components require names. Names used for classes, variables, and methods are called identifiers.

In Java, there are several points to remember about identifiers. They are as follows:

1. All identifiers should begin with a letter (A to Z or a to z), currency character ($) or an underscore (_).

2. After the first character, identifiers can have any combination of characters.

3. A key word cannot be used as an identifier.

4. Most importantly, identifiers are case sensitive.

Examples of legal identifiers: age, $salary, _value, 1_value.

Examples of illegal identifiers: 123abc, -salary.

Java Modifiers

Like other languages, it is possible to modify classes, methods, etc., by using modifiers. There are two categories of modifiers:

1. **Access Modifiers:** default, public , protected, private

2. **Non-access Modifiers:** final, abstract, strictfp

We will be looking into more details about modifiers in the next section.

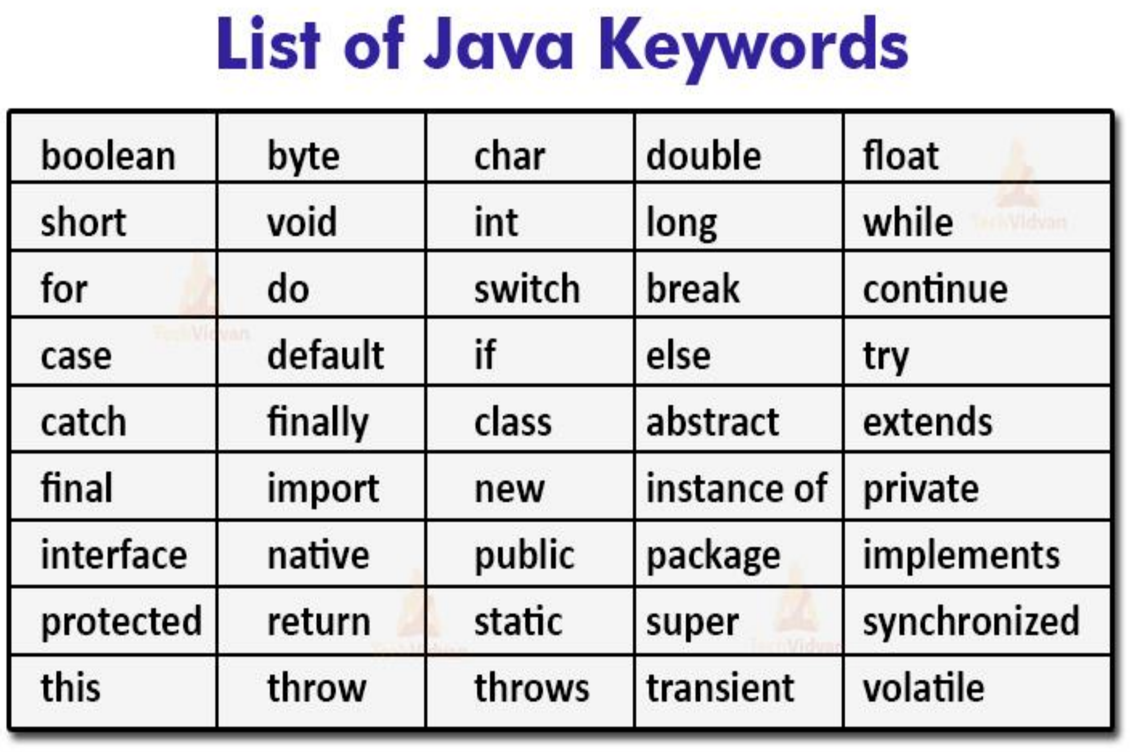
Java Variables

Following are the types of variables in Java:

1. Local Variables
2. Class Variables (Static Variables)
3. Instance Variables (Non-static Variables)

Java Keywords

The following list shows the reserved words in Java. These reserved words may not be used as constant or variable or any other identifier names.

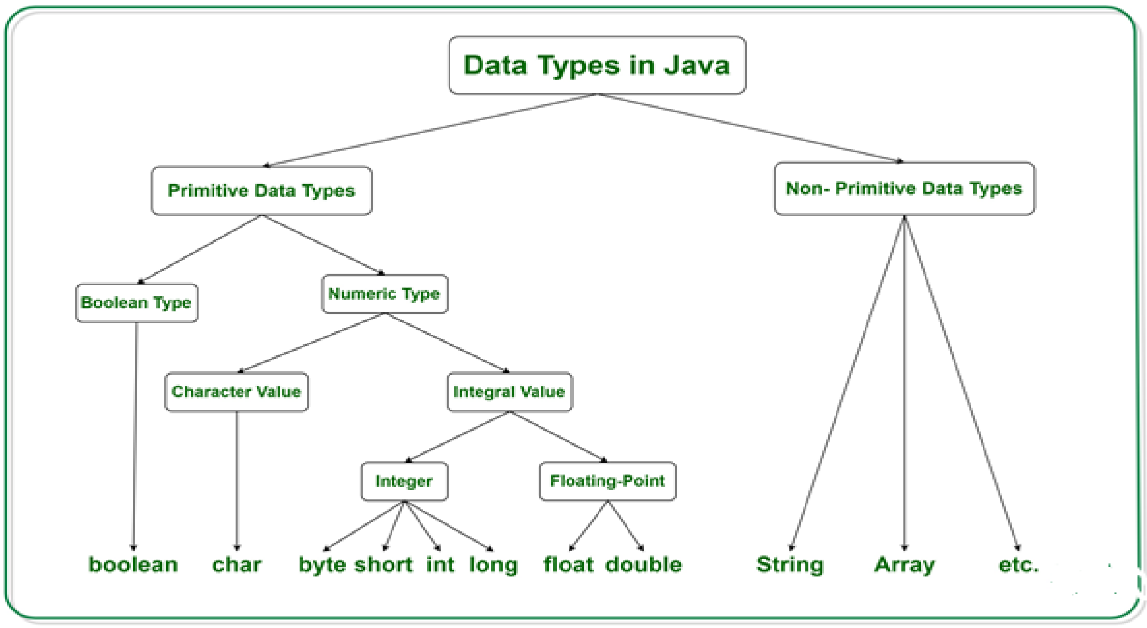


List of Java Keywords

boolean	byte	char	double	float
short	void	int	long	while
for	do	switch	break	continue
case	default	if	else	try
catch	finally	class	abstract	extends
final	import	new	instance of	private
interface	native	public	package	implements
protected	return	static	super	synchronized
this	throw	throws	transient	volatile

Basic Datatypes

Java is **statically typed and also a strongly typed language** because in Java, each type of data (such as integer, character, hexadecimal, packed decimal, and so forth) is predefined as part of the programming language and all constants or variables defined for a given program must be described with one of the data types.



Java has two categories of data:

- **Primitive Data Type:** such as boolean, char, int, short, byte, long, float and double
- **Non-Primitive Data Type or Object Data type:** such as String, Array, etc.

1. **<u>boolean:</u>** boolean data type represents only one bit of information **either true or false**
   Syntax:
   ```
   boolean booleanVar;
   ```

```
    class A{
   public static void main(String args[])
   {
      boolean b = true;
      if (b == true)
         System.out.println("Hi");
   }
}
```

   Output-Hi

2. **<u>byte:</u>** The byte data type is an 8-bit signed two's complement integer.
3. **<u>short:</u>** The short data type is a 16-bit signed two's complement integer.
4. **<u>int</u>**: It is a 32-bit signed two's complement integer.
5. **<u>long:</u>** The long data type is a 64-bit two's complement integer.
6. **<u>float:</u>** The float data type is a single-precision 32-bit IEEE 754 floating point

7. **double:** The double data type is a double-precision 64-bit IEEE 754 floating point.
8. **char**: The char data type is a single 16-bit Unicode character.

```
Class B{
public static void main(String args[])
   {
      // declaring character
      char a = 'G';

      // Integer data type is generally
      // used for numeric values
      int i = 89;

      // use byte and short
      // if memory is a constraint
      byte b = 4;

      // this will give error as number is
      // larger than byte range
      // byte b1 = 7888888955;

      short s = 56;

      // this will give error as number is
      // larger than short range
      // short s1 = 87878787878;

      // by default fraction value
      // is double in java
      double d = 4.355453532;

      // for float use 'f' as suffix
      float f = 4.7333434f;

      System.out.println("char: " + a);
      System.out.println("integer: " + i);
      System.out.println("byte: " + b);
      System.out.println("short: " + s);
      System.out.println("float: " + f);
      System.out.println("double: " + d);
   }
}
```

Output:

```
char: G

integer: 89

byte: 4

short: 56

float: 4.7333436

double: 4.355453532
```

Non-Primitive Data Type or Reference Data Types

The **Reference Data Types** will contain a memory address of variable value because the reference types won't store the variable value directly in memory. They are **strings**, **objects**, arrays, etc.

- **String**:
  Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character '\0'.
  Below is the basic syntax for declaring a string in Java programming language.

  Syntax:

  ```
  <String_Type> <string_variable> = "<sequence_of_string>";
  ```

  Example:

  ```
  // Declare String without using new operator

  String s = "helloeveryone";


  // Declare String using new operator

  String s1 = new String("helloeveryone");
  ```

- **Class**:
  A class is a user-defined blueprint or prototype from which objects are created.
  1. **Modifiers** : A class can be public or has default access
  2. **Class name:** The name should begin with a initial letter (capitalized by convention).
  3. **Superclass(if any):** The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
  4. **Interfaces(if any):** A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
  5. **Body:** The class body surrounded by braces, { }.

- **Object**:
  It is a basic unit of Object-Oriented Programming and represents the real-life entities. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of :
  1. **State** : It is represented by attributes of an object. It also reflects the properties of an object.
  2. **Behavior** : It is represented by methods of an object. It also reflects the response of an object with other objects.
  3. **Identity** : It gives a unique name to an object and enables one object to interact with other objects.
- **Interface:** Like a class, an interface can have methods and variables, but the methods declared in an interface are by default abstract (only method signature, no body).
  1. Interfaces specify what a class must do and not how. It is the blueprint of the class.
  2. An Interface is about capabilities like a Player may be an interface and any class implementing Player must be able to (or must implement) move(). So it specifies a set of methods that the class has to implement.
  3. If a class implements an interface and does not provide method bodies for all functions specified in the interface, then class must be declared abstract.
  4. A Java library example is, Comparator Interface. If a class implements this interface, then it can be used to sort a collection.
- **Array:**
  An array is a group of like-typed variables that are referred to by a common name.Arrays in Java work differently than they do in C/C++. Following are some important point about Java arrays.
  1. In Java all arrays are dynamically allocated.(discussed below)
  2. Since arrays are objects in Java, we can find their length using member length. This is different from C/C++ where we find length using sizeof.
  3. A Java array variable can also be declared like other variables with [] after the data type.
  4. The variables in the array are ordered and each have an index beginning from 0.

Variables

Variable is name *of* reserved area allocated in memory. In other words, it is a name of memory location. It is a combination of "vary + able" that means its value can be change

There are three types of variables in Java:

- local variable
- instance variable
- static variable

1) Local Variable

A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.

2) Instance Variable

A variable declared inside the class but outside the body of the method, is called instance variable. It is not declared as static.

It is called instance variable because its value is instance specific and is not shared among instances.

3) Static variable

A variable which is declared as static is called static variable. It cannot be local. You can create a single copy of static variable and share among all the instances of the class. Memory allocation for static variable happens only once when the class is loaded in the memory.

Example to understand the types of variables in java

```
class A{
int data=50;//instance variable
static int m=100;//static variable
void method(){
int n=90;//local variable
}
}//end of class
```