JAVA BASICS

4KCJ001 - LECTURE 1

By
Sudha Agarwal
sudha.agarwal@4kitsolutions.com

http://www.4kitsolutions.com/

INDEX

- Java Overview
- Features of Different versions of Java
- Java Environment Setup
- JDK
- JRE
- JVM
- JVM Architecture
- Java Features
- ByteCode
- Java Data Types
- Type Casting
- Java Classes

What is Java?

❖Java is a **high-level programming language** originally developed by Sun Microsystems and released in 1995.

❖Java is a robust, object-oriented and secure programming language.

❖Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

APPLICATION

According to Sun, 3 billion devices run Java. There are many devices where Java is currently used. Some of them are as follows:

- Desktop Applications such as acrobat reader, media player, antivirus, etc.
- Web Applications such as irctc.co.in, javatpoint.com, etc.
- Enterprise Applications such as banking applications.
- Mobile
- Embedded System
- Smart Card
- Robotics
- Games, etc.

TYPES OF JAVA APPLICATIONS

There are mainly 4 types of applications that can be created using Java programming:

- ❖Standalone Application
- ❖Web Application
- ❖Enterprise Application
- ❖Mobile Application

JAVA PLATFORMS / EDITIONS

There are 4 platforms or editions of Java:

- Java SE (Java Standard Edition)
- Java EE (Java Enterprise Edition)
- Java ME (Java Micro Edition)
- JavaFX

JAVA5 - FEATURES

- Generics
- Annotations
- Autoboxing/unboxing
- Enumerations
- Varargs
- Enhanced for each loop
- Static imports
- New concurrency utilities in java.util.concurrent
- Scanner class for parsing data from various input streams and buffers.

JAVA6 - FEATURES

- Scripting Language Support
- Performance improvements
- JAX-WS
- JDBC 4.0
- Java Compiler API
- JAXB 2.0 and StAX parser
- Pluggable annotations
- New GC algorithms

JAVA7 - FEATURES

- Now String can be used to control Switch statement.
- Multi Catch Exception
- try-with-resource statement
- Binary Integer Literals
- Underscore in numeric literal

JAVA8 - FEATURES

- Lambda Expressions
- New Collection Package java.util.stream to provide Stream API.
- Enhanced Security
- Nashorn Javascript Engine included
- Parallel Array Sorting
- The JDBC-ODBC Bridge has been removed

JAVA ENVIRONMENT SETUP

❖Java SE is freely available from the link https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html.

❖Can download a version based on your operating system.

❖Follow the instructions to download Java and run the **.exe** to install Java on your machine.

❖Once you installed Java on your machine, you will need to set environment variables to point to correct installation directories −

JAVA ENVIRONMENT SETUP

Assuming you have installed Java in c:\Program Files\java\jdk directory −

- Right-click on 'My Computer' and select 'Properties'.
- Click the 'Environment variables' button under the 'Advanced' tab.
- Now, alter the 'Path' variable so that it also contains the path to the Java executable. Example, if the path is currently set to *'C:\WINDOWS\SYSTEM32', then change your path to read 'C:\WINDOWS\SYSTEM32;c:\Program Files\java\jdk\bin'.*

JDK (Java Development Kit)

- JDK is a software development environment used for developing Java applications.

- It includes:
  - the Java Runtime Environment (JRE).
  - an interpreter/loader (java)
  - a compiler (javac)
  - an archiver (jar)
  - a documentation generator (javadoc) and other tools needed in Java development.

- JDK consists of the core Java API and some utilities for building, testing and documenting Java programs.

JRE (Java Runtime Environment)

- JRE contains everything required to run Java application which has already been compiled.
- It doesn't contain the code library required to develop Java application.
- It provides the libraries, the **Java Virtual Machine**, and other components to run applications written in the Java programming language.
- The JRE does not contain tools and utilities such as compilers or debuggers for developing applications.

JVM (Java Virtual Machine)

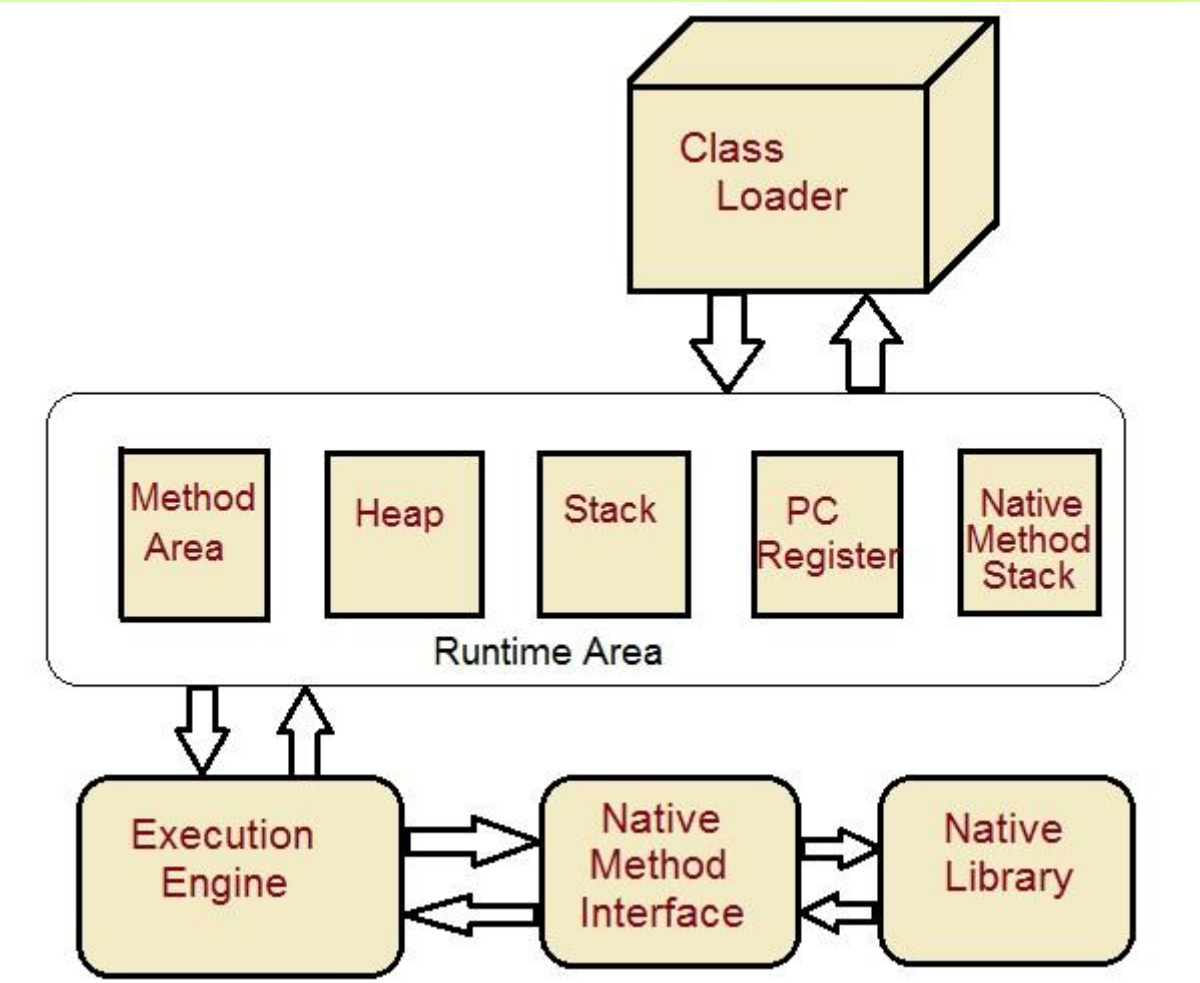
❖The Java Virtual machine (JVM) is the virtual machine that runs the Java bytecodes.

❖ JVM only works with bytecode. Hence you need to compile your Java application(.java) so that it can be converted to bytecode format (also known as the .class file).

❖Bytecode then will be used by JVM to run application.

❖There are specific implementations of the JVM for different systems (Windows, Linux, MacOS etc.).

JVM ARCHITECTURE



JVM ARCHITECTURE

- **Class Loader :**
  - Class loader loads the Class for execution.
- **Method area** :
  - Stores pre-class structure as constant pool.
- **Heap** :
  - Heap is in which objects are allocated.
- **Stack** :
  - Local variables and partial results are store here. Each thread has a private JVM stack created when the thread is created.
- **Program register** :
  - Program register holds the address of JVM instruction currently being executed.

JVM ARCHITECTURE

- **Native method stack :**
  - It contains all native used in application.
- **Executive Engine :**
  - Execution engine controls the execute of instructions contained in the methods of the classes.
- **Native Method Interface :**
  - Native method interface gives an interface between java code and native code during execution.
- **Native Method Libraries :**
  - Native Libraries consist of files required for the execution of native code.

JDK Vs. JRE Vs. JVM

- JRE => JVM + Required Library to run Application.
- JDK => JRE + Required Library to develop Java Application.
- The **JDK** is a **superset** of the **JRE**, and contains everything that is in the JRE, plus tools such as the compilers and debuggers necessary for developing applications.

HOW TO SET PATH

- To set the PATH variable permanently, add the full path of the jdk1.8.0\bin directory to the PATH variable. Typically, this full path looks something like C:\Program Files\Java\jdk1.8.0\bin. Set the PATH variable as follows on Microsoft Windows:

  - Click Start, then Control Panel, then System.

  - Click Advanced, then Environment Variables.

  - Add the location of the bin folder of the JDK installation to the PATH variable in System Variables. The following is a typical value for the PATH variable:

  - C:\WINDOWS\system32;C:\WINDOWS;C:\Program Files\Java\jdk1.8.0\bin

Why do we set path in java?

❖To execute java console based programs in windows environment we have to use java and javac commands.

❖Since, java and javac commands are unknown for windows till we do not specify explicitly where those executable resides.

❖This is the reason while setting the path we specify path of bin folder(bin contains all the binary executable).

What is JAVA_HOME?

❖**JAVA_HOME** is a system environment variable which represent JDK installation directory.

❖When we install JDK in your machine (windows, Linux or unix) it creates a home directory and puts all its binary (bin), library(lib) and other tools.

❖In order to compile java program "javac" tool should be in your PATH and in order to get that in PATH, we use JAVA_HOME environment variable.

❖Many tools like ANT and web servers like tomcat use JAVA_HOME to find java binaries.

JAVA FEATURES

❖**Object Oriented –** In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

❖**Platform Independent** – Unlike other programming languages such as C, C++ etc which are compiled into platform specific machines. Java is guaranteed to be write-once, run-anywhere language.

❖On compilation Java program is compiled into bytecode. This bytecode is platform independent and can be run on any machine, plus this bytecode format also provide security. Any machine with Java Runtime Environment can run Java Programs.

❖**Simple** – Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.

JAVA FEATURES

❖**Secure** – With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

❖**Architecture neutral** – Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.

❖**Portable** – Java Byte code can be carried to any platform. No implementation dependent features.

❖**Robust** – Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.

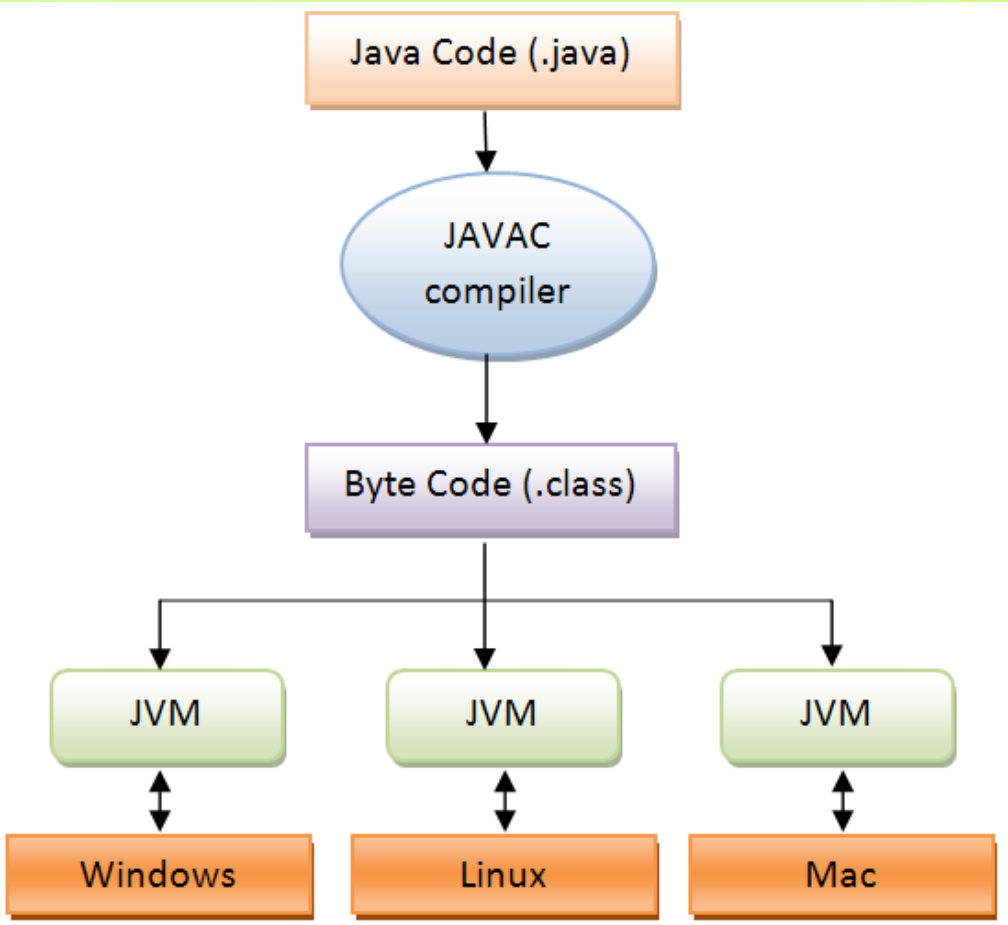
JAVA FEATURES

❖**Multithreaded** – Java multithreading feature makes it possible to write program that can do many tasks simultaneously.

❖**Interpreted** – Java byte code is translated on the fly to native machine instructions and is not stored anywhere.

❖**High Performance** – With the use of Just-In-Time compilers, Java enables high performance.

BYTE CODE



- Java class is written in Unicode characters.
- Java compiler converts these Unicode characters into Byte code.
- Java Byte code can only be understandable by JVM

IDE - ECLIPSE

Release	Main Release	Platform Version	Projects
Oxygen	June 2017		
Neon	22 June 2016	4.6	
Mars	24 June 2015	4.5	Mars Projects
Luna	24 June 2014	4.4	Luna Projects
Kepler	26 June 2013	4.3	Kepler Projects
Juno	27 June 2012	4.2	Juno Projects
Indigo	22 June 2011	3.7	Indigo Projects

JAVA DATA TYPES

In java, data types are classified into two categories :

- **Primitive Data type** – byte, short, int, long, float, double, char, boolean.
- **Non-Primitive Data type**

PRIMITIVE DATA TYPES

The primitive types are also commonly referred to as simple types which can be put in four groups

❖**Integers:** This group includes **byte, short, int, and long**, which are for whole-valued signed numbers.

❖**Floating-point numbers:** This group includes **float** and **double**, which represent numbers with fractional precision.

❖**Characters:** This group includes **char**, which represents symbols in a character set, like letters and numbers.

❖**Boolean:** This group includes **boolean**, which is a special type for representing true/false values.

INTEGERS

❖**byte :** It is 1 byte(8-bits) integer data type. Value range from -128 to 127. Default value zero. example: byte b=10;

❖**short :** It is 2 bytes(16-bits) integer data type. Value range from -32768 to 32767. Default value zero. example: short s=11;

❖**int :** It is 4 bytes(32-bits) integer data type. Value range from -2147483648 to 2147483647. Default value zero. example: int i=10;

❖**long :** It is 8 bytes(64-bits) integer data type. Value range from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. Default value zero. example: long l=100012;

FLOATING POINT NUMBER

❖**float** : It is 4 bytes(32-bits) float data type. Default value 0.0f. example: float ff=10.3f;

❖**double** : It is 8 bytes(64-bits) float data type. Default value 0.0d. example: double db=11.123;

❖**Boolean**

❖The boolean data type has only two possible values: true and false. Example:

```
boolean  flag = true;
booleanval  = false;
```

CHARACTERS

❖**char** : It is 2 bytes(16-bits) unsigned unicode character.

❖It has a minimum value of '\u0000' (or 0) and a maximum value of '\uffff' (or 65,535 inclusive). There are no negative chars.

```
char ch1 = 88; // code for X
char  ch2 = 'Y';
```

NON-PRIMITIVE(REFERENCE) DATA TYPE

A reference data type is used to refer to an object. A reference variable is declare to be of specific and that type can never be changed.

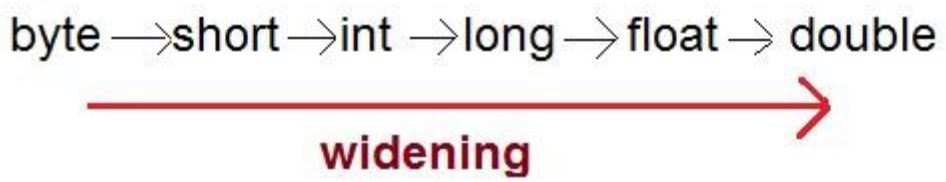
TYPE CASTING

❖Assigning a value of one type to a variable of another type is known as **Type Casting**.

```
int x = 10;
byte y = (byte)x;
```

❖In Java, type casting is classified into two types:

❖Widening casting or upcasting(implicit)



❖Narrowing casting or downcasting (explicit)

TYPE CASTING

- Automatic Type casting take place when,
  - the two types are compatible
  - the target type is larger than the source type

```
int i = 100;
      long l = i; //no explicit type casting required
      float f = l //no explicit type casting required
```

- When you are assigning a larger type value to a variable of smaller type, then you need to perform **explicit type casting**.

```
double d = 100.04;
      long l = (long)d; //explicit type casting
required
      int i = (int)l;   //explicit type casting
required
```

MAIN METHOD

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

- The main method must be **public** so it can be found by the JVM when the class is loaded.

- Similarly, it must be **static** so that it can be called after loading the class, without having to create an instance of it.

- All methods must have a **return type**, which in this case is void.

- The list of **String arguments** is there to allow to pass parameters when executing a Java program from the command line.

CLASS

❖A class can be defined as a template/blueprint that describes the behaviour/state that the object of its type supports.

❖A class defines new data type. Once defined this new type can be used to create object of that type.

❖Object is an instance of class.

❖A class is declared by '**class**' keyword. A class contains both **data** and the **code** that operates on that data.

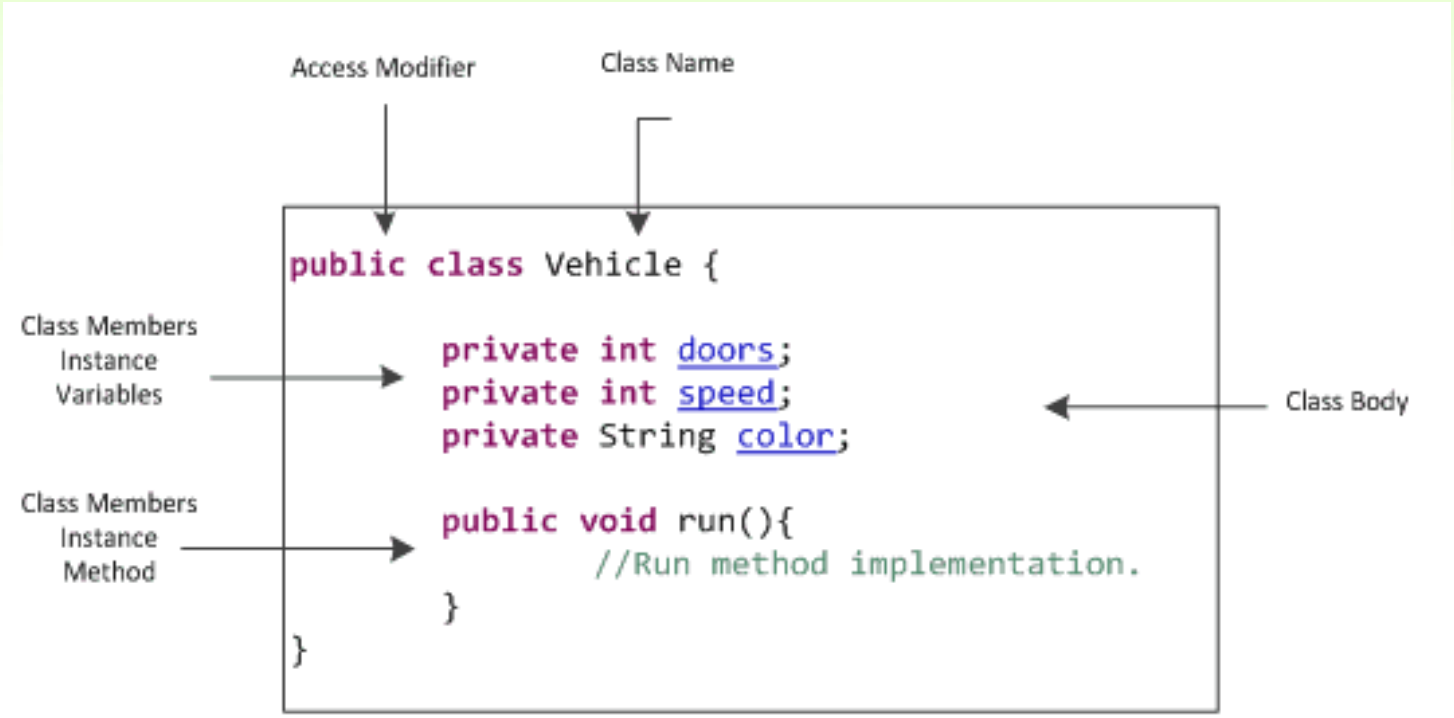
❖The data or variables defined within a class are called **instance variables** and the code that operates on this data is known as **methods**.

❖The instance variables and methods are known as **class members**.

RULES FOR JAVA CLASS

- A class can have only public or default(no modifier) access specifier.
- It can be either abstract, final or concrete (normal class).
- It must have the class keyword, and class must be followed by a legal identifier.
- It may optionally extend one parent class. By default, it will extend java.lang.Object.
- It may optionally implement any number of comma-separated interfaces.
- The class's variables and methods are declared within a set of curly braces {}.
- Each .java source file may contain only one public class. A source file may contain any number of default visible classes.
- Finally, the source file name must match the public class name and it must have a .java suffix.

CLASS



http://www.4kitsolutions.com/

CLASS EXAMPLE

- Lets create a **Student** class

```
class Student
{
 String name;
 int rollno;
 int age;
}
```

- Now create a reference variable

```
Student std
```

- Now create an object and assign it to reference variable

```
std= new Student();
```

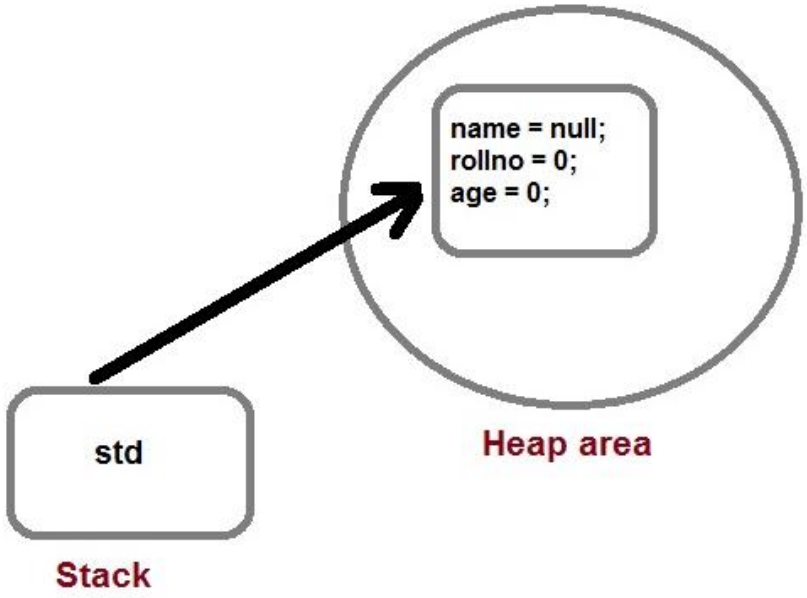
CLASS EXAMPLE

❖After the above statement std is instance/object of Student class.

❖Here the new keyword creates an actual physical copy of the object and assign it to the **std** variable.

❖It will have physical existence and get memory in heap area.

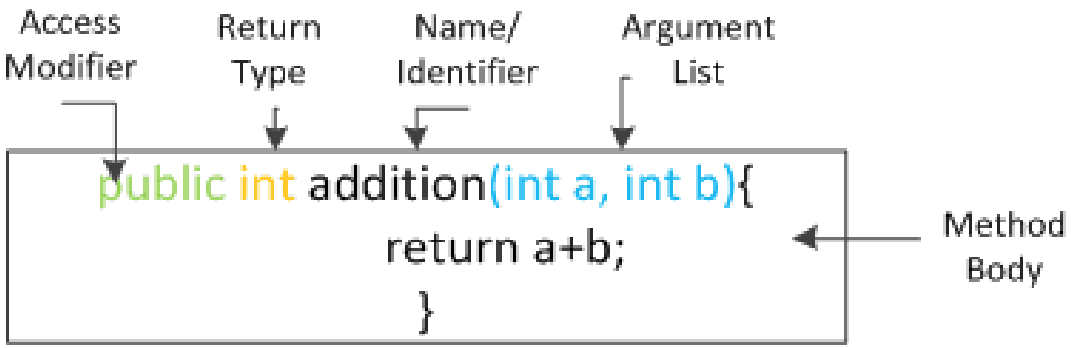
❖ The new operator dynamically allocates memory for an object



METHODS

❖A method is a program module that contains a series of statements that carry out a task.

❖To execute a method, you invoke or call it from another method.



LAB 1

- Q1. Install multiple JDK versions and you should be able to point to a particular version. Check the version on command prompt.

- Q2. Create a new workspace and then different projects in that workspace in Eclipse.

- Q3. Write a Java program to print the sum of two numbers.

- Q4. Write a Java program that takes number 5 as input and prints its multiplication table upto 10.

- Q5. Write a Java program to print the sum (addition), multiply, subtract, divide and remainder of two numbers.

- Q6. Create a class student.
  - Create two class members: **Name** and **Roll No**.
  - Add a method **print()** which prints student details
  - Create an object of the class and invoke print() method.

THANK YOU

http://www.4kitsolutions.com/

By
Sudha Agarwal
sudha.agarwal@4kitsolutions.com