UNIT – 1

1. **INTRODUCTION TO OOP.**
2. **NEED OF OOP.**
3. **PRINCIPLES OF OBJECT ORIENTED LANGUAGES.**
4. **PROCEDURAL LANGUAGES Vs OOP.**
5. **APPLICATIONS OF OOP.**
6. **HISTORY OF JAVA.**
7. **JAVA VIRTUAL MACHINE.**
8. **JAVA FEATURES.**
9. **PROGRAM STRUCTURES.**
10. **INSTALLATION OF JDK1.6.**

1. INTRODUCTION TO OOP.

- **Object-oriented programming** (**OOP**) is a programming Paradigm based on the concept of "objects", which are data structures that contain data, in the form of fields, often known as *attributes;* and code, in the form of procedures, often known as *methods*.
- A distinguishing feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated (objects have a notion of "this").
- In object-oriented programming, computer programs are designed by making them out of objects that interact with one another. There is significant diversity in object-oriented programming, but most popular languages are class-based, meaning that objects are instances of classes, which typically also determines their type.
- Many of the most widely used programming languages are multi-paradigm programming languages that support object-oriented programming to a greater or lesser degree, typically in combination with imperative, procedural programming. Significant object-oriented languages include C++, Objective-C, Smalltalk, Delphi, Java, C#, Perl, Python, Ruby and PHP.
- Objects sometimes correspond to things found in the real world. For example, a graphics program may have objects such as "circle," "square," "menu."
- An online shopping system will have objects such as "shopping cart," "customer," and "product." The shopping system will support behaviors such as "place order," "make payment," and "offer discount."
- Objects are designed in class hierarchies. For example, with the shopping system there might be high level classes such as "electronics product," "kitchen product," and "book."
- There may be further refinements for example under "electronic products": "CD Player," "DVD player," etc.

2. NEED OF OOP

- When we are going to work with classes and objects we always think why there is need of classes and object although without classes and objects our code works well. But after some work on classes and objects we think, it's better to work with classes and objects.
- And then we can understand that Object oriented Programming is better than procedural Programming. Object oriented programming requires a different way of thinking about how can we construct our application.

- **Let’s take a short example:**
- When we are going to build a house .We distribute the works needed to make a house we can divide the works in different part and deliver to different persons like plumber for water supply, electrician for electricity etc. The electrician doesn’t need to know that the work of plumber and vice versa.
- In the same way in Object Oriented Programming we can divide our applications in different modules called classes. And one class is separate from other classes .It has its own functionality and features.

- So what advantages we can get from Object Oriented Programming**:**

1: Reusability Of Code: In object oriented programming one class can easily copied to another application if we want to use its functionality,

EX: When we work with hospital application and railway reservation application .In both application we need person class so we have to write only one time the person class and it can easily use in other application.

2: Easily Discover a bug: When we work with procedural programming it take a lot of time to discover a bug and resolve it .But in object Oriented Programming due to modularity of classes we can easily discover the bug and we have to change in one class only and this change make in all the application only by changing it one class only.

3. PRINCIPLES OF OBJECT ORIENTED LANGUAGES.

The Objects Oriented Programming (OOP) is constructed over four major principles:

1. Encapsulation,
2. Data Abstraction,
3. Polymorphism
4. Inheritance.

1.Encapsulation:

Encapsulation means that the internal representation of an object is generally hidden from view outside of the object’s definition. Typically, only the object’s own methods can directly inspect or manipulate its fields.

Encapsulation is the hiding of data implementation by restricting access to **accessors** and **mutators**.

An **accessor** is a method that is used to ask an object about itself. In OOP, these are usually in the form of properties, which have a ***get*** method, which is an accessor method. However, accessor methods are not restricted to properties and can be any public method that gives information about the state of the object.

A **Mutator** is public method that is used to modify the state of an object, while hiding the implementation of exactly how the data gets modified. It's the ***set*** method that lets the caller modify the member data behind the scenes.

Hiding the internals of the object protects its integrity by preventing users from setting the internal data of the component into an invalid or inconsistent state. This type of data protection and implementation protection is called ***Encapsulation*.**
A benefit of encapsulation is that it can reduce system complexity.

- **Abstraction**

  Data abstraction and encapsulation are closely tied together, because a simple definition of data abstraction is the development of classes, objects, types in terms of their interfaces and functionality, instead of their implementation details. Abstraction denotes a model, a view, or some other focused representation for an actual item.
- According to **Graddy Booch,** "An abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of object and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer."

- In short, data abstraction is nothing more than the implementation of an object that contains the same essential properties and actions we can find in the original object we are representing.

3. Inheritance is a way to reuse code of existing objects, or to establish a subtype from an existing object, or both, depending upon programming language support.

In classical inheritance where objects are defined by classes, classes can inherit attributes and behaviour from pre-existing classes called base classes, super classes, parent classes or ancestor classes.
The resulting classes are known as derived classes, subclasses or child classes. The relationships of classes through inheritance give rise to a hierarchy.

4.Polymorphism

Polymorphism means one name, many forms. Polymorphism manifests itself by having multiple methods all with the same name, but slightly different functionality.
There are 2 basic types of polymorphism.
Overriding, also called run-time polymorphism. For method overloading, the compiler determines which method will be executed, and this decision is made when the code gets compiled.
Overloading, which is referred to as compile-time polymorphism. Method will be used for method overriding is determined at runtime based on the dynamic type of an object.

4. PROCEDURAL LANGUAGES Vs OOP

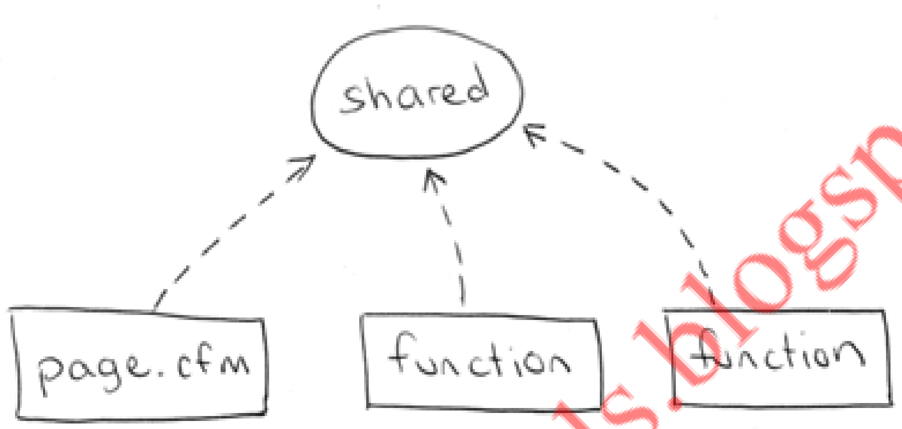
When programming any system you are essentially dealing with data and the code that change that data. These two fundamental aspects of programming are handled quite differently in procedural systems compared with object oriented systems, and these differences require different strategies in how we think about writing code.

Procedural programming

- In procedural programming our code is organized into small procedures that use and change our data. We write our procedures as either custom tags or functions.
- These functions typically take some input, do something, and then produce some output. Ideally your functions would behave as "black boxes" where input data goes in and output data comes out.

- The key idea here is that our functions have no intrinsic relationship with the data they operate on. As long as you provide the correct number and type of arguments, the function will do its work and faithfully return its output.

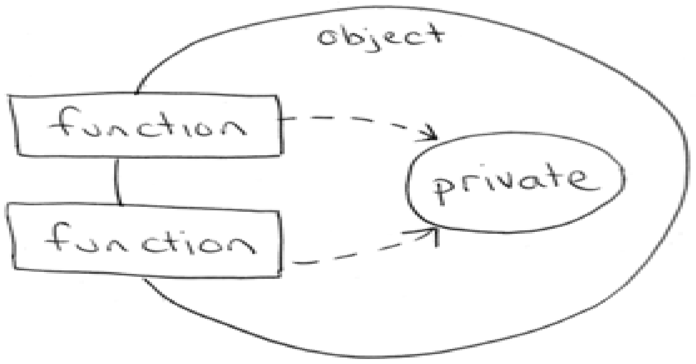
- Sometimes our functions need to access data that is not provided as a parameter, i.e., we need access data that is outside the function. Data accessed in this way is considered "global" or "shared" data.



So in a procedural system our functions use data they are "given" (as parameters) but also directly access any shared data they need.

Object oriented programming

- In object oriented programming, the data and related functions are bundled together into an "object". Ideally, the data inside an object can only be manipulated by calling the object's functions.
- This means that your data is locked away inside your objects and your functions provide the only means of doing something with that data. In a well designed object oriented system objects never access shared or global data, they are only permitted to use the data they have, or data they are given.

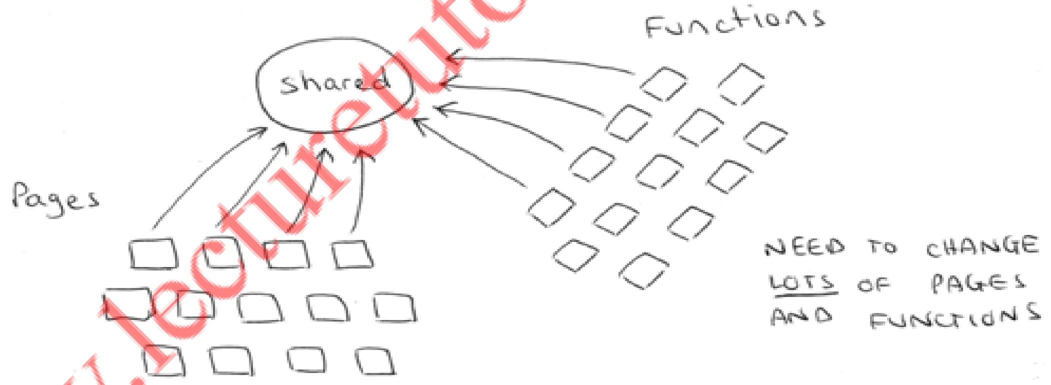


Global and shared data

We can see that one of the principle differences is that procedural systems make use of shared and global data, while object oriented systems lock their data privately away in objects.

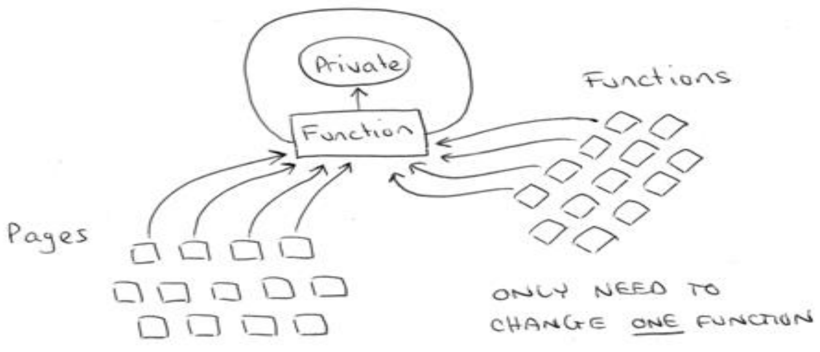
Let's consider a scenario where you need to change a shared variable in a procedural system. Perhaps you need to rename it, change it from a string to a numeric, change it from a struct to an array, or even remove it completely.

In a procedural application you would need to find and change each place in the code where that variable is referenced. In a large system this can be a widespread and difficult change to make.



In an object oriented system we know that all variables are inside objects and that only functions within those objects can access or change those variables. When a variable needs to be changed then we only need to change the functions that access those variables.

As long as we take care that the functions' input arguments and output types are not changed, then we don't need to change any other part of the system.



5. APPLICATIONS OF OOP.

Applications of OOP are beginning to gain importance in many areas. The most important application is user interface design. Real business systems are more complex and contain many attributes and methods, but OOP applications can simplify a complex problem.

Application areas:

- Computer animation
- To design Compiler
- To access relational data base
- To develop administrative tools and system tools
- Simulation and modeling
- To develop computer games

-

OOP Provides many applications:

Real Time Systems : A real time system is a system that give output at given instant and its parameters changes at every time. A real time system is nothing but a dynamic system. Dynamic means the system that changes every moment based on input to the system.

OOP approach is very useful for Real time system because code changing is very easy in OOP system and it leads toward dynamic behavior of OOP codes thus more suitable to real timeSystem

SimulationAndModeling:-

System modeling is another area where criteria for OOP approach is countable. Representing a system is very easy in OOP approach because OOP codes are very easy to understand and thus is preferred to represent a system in simpler form.

HypertextAndHypermedia:-

Hypertext and hypermedia is another area where OOP approach is spreading its legs. Its ease of using OOP codes that makes it suitable for various media approaches.

DecisionSupportSystem:-

Decision support system is an example of Real time system that too very advance and complex system. More details is explained in real time system

CAM/CAE/CAD System:-

Computer has wide use of OOP approach. This is due to time saving in writing OOP codes and dynamic behavior of OOP codes.

Office AutomationSystem:-

Automation system is just a part or type of real time system. Embedded systems make it easy to use OOP for automated system.

AIAndExpertSystem:-

It is mixed system having both hypermedia and real time system.

6 .HISTORY OF JAVA.

- Brief history of Java
- Java Version History

Java history is interesting to know. The history of java starts from Green Team. Java team members (also known as **Green Team**), initiated a revolutionary task to develop a language for digital devices such as set-top boxes, televisions etc.

For the green team members, it was an advance concept at that time. But, it was suited for internet programming. Later, Java technology as incorporated by Netscape.

James Gosling

Currently, Java is used in internet programming, mobile devices, games, e-business solutions etc. There are given the major points that describe the history of java.

1) **James Gosling**, **Mike Sheridan**, and **Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called **Green Team**.

2) Originally designed for small, embedded systems in electronic appliances like set-top boxes.

3) Firstly, it was called **"Greentalk"** by James Gosling and file extension was .gt.

4) After that, it was called **Oak** and was developed as a part of the Green project.

Why Oak name for java language?

5) **Why Oak?** Oak is a symbol of strength and chosen as a national tree of many countries like U.S.A., France, Germany, Romania etc.

6) In 1995, Oak was renamed as **"Java"** because it was already a trademark by Oak Technologies.

Why Java name for java language?

7) **Why they choose java name for java language?** The team gathered to choose a new name. The suggested words were "dynamic", "revolutionary", "Silk", "jolt", "DNA" etc. They wanted something that reflected the essence of the technology: revolutionary, dynamic, lively, cool, unique, and easy to spell and fun to say.

According to James Gosling "Java was one of the top choices along with **Silk**". Since java was so unique, most of the team members preferred java.

8) Java is an island of Indonesia where first coffee was produced (called java coffee).

9) Notice that Java is just a name not an acronym.

10) Originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995.

11) In 1995, Time magazine called **Java one of the Ten Best Products of 1995**.

12) JDK 1.0 released in (January 23, 1996).

Java Version History

There are many java versions that have been released. Current stable release of Java is Java SE 8.

1. JDK Alpha and Beta (1995)
2. JDK 1.0 (23rd Jan, 1996)
3. JDK 1.1 (19th Feb, 1997)
4. J2SE 1.2 (8th Dec, 1998)
5. J2SE 1.3 (8th May, 2000)
6. J2SE 1.4 (6th Feb, 2002)
7. J2SE 5.0 (30th Sep, 2004)
8. Java SE 6 (11th Dec, 2006)
9. Java SE 7 (28th July, 2011)
10. Java SE 8 (18th March, 2014)

7. JAVA VIRTUAL MACHINE.

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java byte code can be executed.

JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).

What is JVM?

It is:

1. **A specification** where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Sun and other companies.
2. **An implementation** Its implementation is known as JRE (Java Runtime Environment).
3. **Runtime Instance** Whenever you write java command on the command prompt to run the java class, and instance of JVM is created.

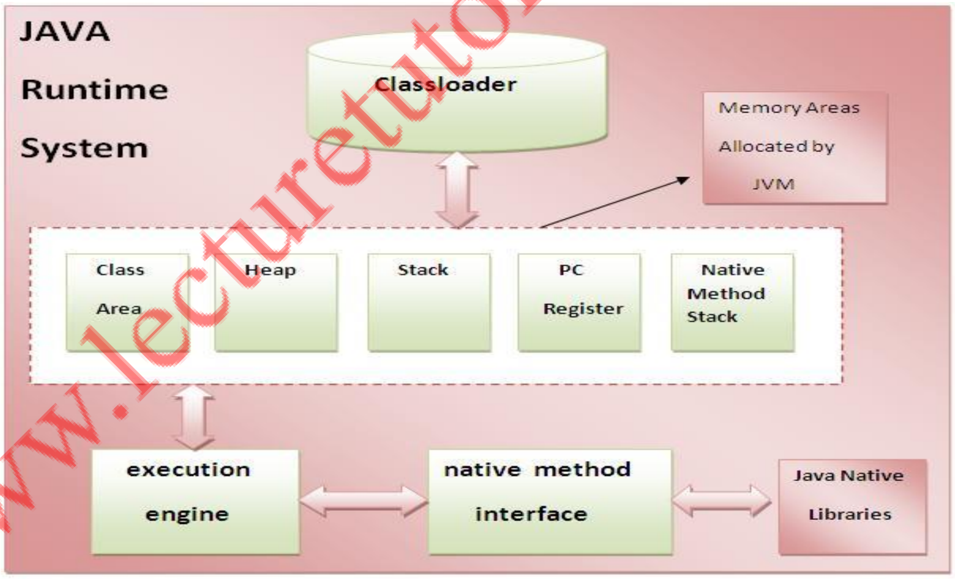
What it does?

The JVM performs following operation:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

Internal Architecture of JVM

Let's understand the internal architecture of JVM. It contains class loader, memory area, execution engine etc.



1) **Class loader**:

Class loader is a subsystem of JVM that is used to load class files.

2) **Class (Method) Area**: Class (Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

3) Heap:

It is the runtime data area in which objects are allocated.

4) Stack:

Java Stack stores frames .It holds local variables and partial results, and plays a part in method invocation and return.

Each thread has a private JVM stack, created at the same time as thread.

A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

5) Program Counter Register:

PC (program counter) register. It contains the address of the Java virtual machine instruction currently being executed.

6) Native Method Stack:

It contains all the native methods used in the application.

7) Execution Engine:

It contains:

1) Virtual processor

. **2) Interpreter:** Read byte code stream then execute the instructions.

3) Just-In-Time (JIT) compiler:

It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation.

Here the term? Compiler? Refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

8. **JAVA FEATURES.**

There is given many features of java. They are also known as java buzzwords. The Java Features given below are simple and easy to understand.

1. Simple
2. Object-Oriented
3. Platform independent
4. Secured
5. Robust
6. Architecture neutral
7. Portable
8. Dynamic
9. Interpreted
10. High Performance
11. Multithreaded
12. Distributed

Simple

- According to Sun, Java language is simple because:
- syntax is based on C++ (so easier for programmers to learn it after C++).
- removed many confusing and/or rarely-used features e.g., explicit pointers, operator overloading etc.
- No need to remove unreferenced objects because there is Automatic Garbage Collection in java.

Object-oriented

Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior.
Object-oriented programming (OOPs) is a methodology that simplify software development and maintenance by providing some rules

Basic concepts of OOPs are:

1. Object
2. Class
3. Inheritance
4. Polymorphism

5. Abstraction
6. Encapsulation

Platform Independent

A platform is the hardware or software environment in which a program runs. There are two types of platforms software-based and hardware-based. Java provides software-based platform.

The Java platform differs from most other platforms in the sense that it's a software-based platform that runs on top of other hardware-based platforms .It has two components:

1. Runtime Environment.
2. API (Application Programming Interface).

Java code can be run on multiple platforms e.g. Windows, Linux, Sun Solaris, and Mac/OS etc. Java code is compiled by the compiler and converted into byte code.

This byte code is a platform independent code because it can be run on multiple platforms i.e. Write Once and Run Anywhere (WORA).

Secured

Java is secured because:

- No explicit pointer.
- Programs run inside virtual machine sandbox.

Robust

Robust simply means strong. Java uses strong memory management. There is lack of pointers that avoids security problem.

There is automatic garbage collection in java. There is exception handling and type checking mechanism in java. All these points make java robust.

Architecture-neutral

There is no implementation dependent feature e.g. size of primitive types is set.

Portable

We may carry the java byte code to any platform.

High-performance

Java is faster than traditional interpretation since byte code is "close" to native code still somewhat slower than a compiled language (e.g., C++).

Distributed

We can create distributed applications in java. RMI and EJB are used for creating distributed applications. We may access files by calling the methods from any machine on the internet.

Multi-threaded

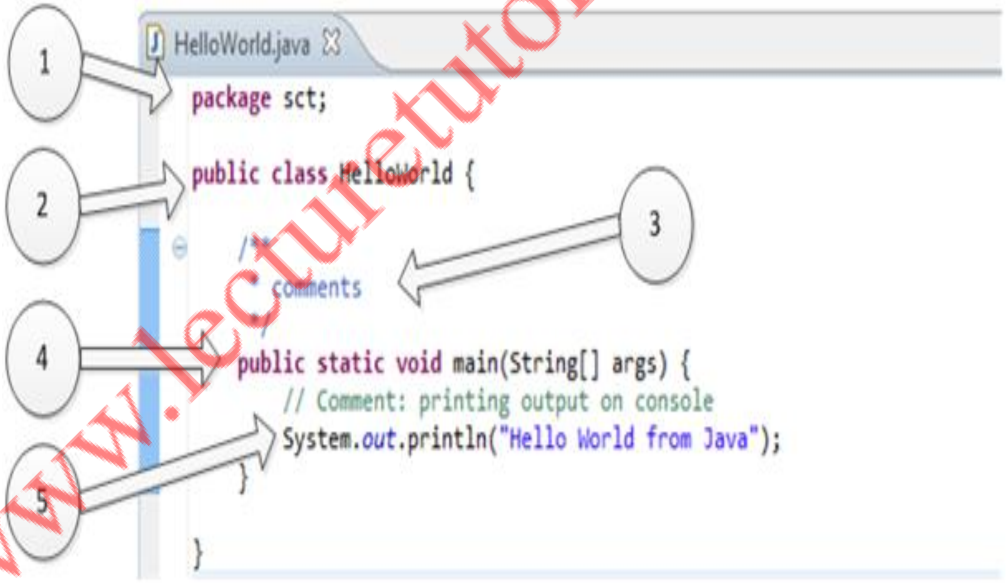
A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it shares the same memory. Threads are important for multi-media, Web applications etc.

9.PROGRAM STRUCTURES.

Description:

Let's use example of HelloWorld Java program to understand structure and features of class. This program is written on few lines, and its only task is to print "Hello World from Java" on the screen.

Refer the following picture:



```
package sct;

public class HelloWorld {

    /**
     * comments
     */
    public static void main(String[] args) {
        // Comment: printing output on console
        System.out.println("Hello World from Java");
    }

}
```

1. "package sct":

It is package declaration statement. The package statement defines a name space in which classes are stored. Package is used to organize the classes based on functionality. If you omit the package statement, the class names are put into the default package, which has no name. Package statement cannot appear anywhere in program. It must be first line of your program or you can omit it.

2. "public class HelloWorld":

This line has various aspects of java programming.

a. public: This is access modifier keyword which tells compiler access to class. Various values of access modifiers can be public, protected, and private or default (no value).
b. class: This keyword used to declare class. Name of class (Hello World) followed by this keyword.

3. Comments section:

We can write comments in java in two ways.

a. Line comments: It start with two forward slashes (//) and continue to the end of the current line. Line comments do not require an ending symbol.
b. Block comments start with a forward slash and an asterisk (/*) and end with an asterisk and a forward slash (*/).Block comments can also extend across as many lines as needed.

4. "public static void main (String [ ] args)":

Its method (Function) named main with string array as argument.

a. public : Access Modifier
b. static: static is reserved keyword which means that a method is accessible and usable even though no objects of the class exist.
c. void: This keyword declares nothing would be returned from method. Method can return any primitive or object.
d. Method content inside curly braces. { }

5. System.out.println("Hello World from Java") :

a. System:It is name of Java utility class.
b. out:It is an object which belongs to System class.
c. println:It is utility method name which is used to send any String to console.
d. "Hello World from Java": It is String literal set as argument to println method.

10. INSTALLATION OF JDK1.6.

If you do not already have the JDK software installed or if JAVA_HOME is not set, the Java CAPS installation will not be successful.

The following tasks provide the information you need to install JDK software and set JAVA_HOME on UNIX or Windows systems.

Microsoft Windows

JDK5: At least release 1.5.0_14

JDK6: At least release 1.6.0_03

Caution –

It is not recommended to user JDK 1.6.0_13 or 1.6.0_14 with Java CAPS due to issues with several of the wizards used to develop applications. In addition, the installation fails on Windows when using JDK 1.6.0_13 or 1.6.0_14. The Java CAPS Installer does not support JDK release 1.6.0_04 in the 64–bit version on Solaris SPARC or AMD 64–bit environments. The installer also does not support JDK 1.6.0 or later on AIX 5.3.

•

To Install the JDK Software and Set JAVA_HOME on a Windows System

1. Install the JDK software.
   a. Go to http://java.sun.com/javase/downloads/index.jsp.
   b. Select the appropriate JDK software and click Download.

   The JDK software is installed on your computer, for example, at C:\Program Files\Java\jdk1.6.0_02. You can move the JDK software to another location if desired.

2. To set JAVA_HOME:
   a. Right click My Computer and select Properties.
   b. On the Advanced tab, select Environment Variables, and then edit JAVA_HOME to point to where the JDK software is located, for example, C:\Program Files\Java\jdk1.6.0_02.