**Java Basics**

**Date: 1st Dec 2017**

* **JDK , JRE and JVM**

**JDK**: To develop, compile and run program.

**JRE**: JDK contains JRE (provides run time environment for program)

* **Path and Class path**

**Path:** Once you installed Java on your machine, it is required to Set the PATH environment variable to conveniently run the executable (javac.exe, java.exe, javadoc.exe, and so on).

Exemple: C:\Java\jdk1.7.0\bin\javac TestClass.java

**Class path:** Class path is system environment variable used by the Java compiler and JVM.

Java compiler and JVM is used Class path to determine the location of required class files.

Example: C:\Program Files\Java\jdk1.6.0\bin

* JVM (Java Virtual Machine) is an abstract machine (Cannot touch since no physical existence). It provides runtime environment in which java bytecode can be executed. JVM is the one that actually calls the main method present in a java code.
* **JVM Architecture:**

Class Loader Subsystem:

Loading: stores info of loaded class, parent class, variable, methods and modifiers info etc.

Linking: Verification, Preparation and resolution.

Initialization: Bootstrap, Extension and application class loader and delegation Hierarchy algorithm.

**JVM Memory:**

**Method area:** In method area, all class level information like class name, immediate parent class name, methods and variables information etc.

**Heap area:** Information of all objects

**Stack area:** For every thread, JVM create one run-time stack, all local variables of that method are stored in their corresponding frame.

**PC Registers:** Store address of current execution instruction of a thread. Obviously each thread has separate PC Registers.

**Native method stacks:** For every thread, separate native stack is created. It stores native method information.

**Execution engine** (To execute Byte code (.class file))

**Interpreter:** It interprets the bytecode line by line and then executes.

**Just-In-Time Compiler (JIT)**: JIT provide direct native code for repeated method calls

**Garbage Collector:** It destroys un-referenced objects.

**Date: 9th Jan 2018**

* Write a java program to print Fibonacci series without using recursion and using recursion.
* Write a java program to check prime number.

**OOPS Concepts:**

* **Object-based programming language:** All the features of OOPs except Inheritance example JavaScript, VBScript etc.
* **Simula -> Smalltalk -> Java**

**Object:** Any real world entity that has state (colour, size) and behaviour (functionality like writing). For example: chair, pen, table, keyboard, bike etc.

**Class:** Collection of objects and template or blueprint from which objects are created.

**Inheritance:** one object acquires all the properties and behaviours of parent object (**Reusability**)

**Polymorphism:** When one task is performed by different ways Example: Method overloading, Overriding

**Abstraction:** Hiding internal details and showing functionality Example TV remote.

**Encapsulation:** Binding (or wrapping) code and data together into a single unit Example Capsule.

**Java Constructer & Constructer Overloading:**

* It is used to construct data for object and it is invoked during object creation.

**Date: 11th Jan 2018**

**Static Keyword:**

* It is used for memory management.
* It gets memory while class loading.
* We have Static Variables, methods, blocks etc.
* Example College name

We can define static variable using static keyword.

Static methods can access others static methods and variables only.

**We don’t need object (class instance) to call static methods.**

Example: **Classname.MethodName ()**

**This Keyword:**

It is used to resolve the ambiguity between local variables and instance variables.

It is used:

1. When **local and instance variables have same names**
2. To call current class method Example **this. Method()**
3. To call current class constructer Example **this()**
4. **To reuse constructor data** Example this(parameter1, parameter2)

**Inheritance :( IS-A)**

To acquire properties and behaviour of existing class Example Single, Multilevel, and Hierarchical etc.

Example: Programmer is an Emp.