Day 5

Components of Java Platform

```
1. Java API
2. Java Virtual Machine
```

Components of JVM

```
1. Class Loader Subsystem
2. Runtime Data Areas
3. Execution Engine
```

Class Loaders

```
1. BootStrap ClassLoader
2. Extension ClassLoader
3. Application ClassLoader
```

Runtime Data Areas

```
1. Method Area
2. Heap
3. Java Stack
4. PC register
5. Native Method Stack
```

Execution Engine

```
1. Interpreter
2. Just In Time Compiler
```

Value Type

- Primitive type is called as value type.
- boolean, byte, char, short, int, float, double, long are primitive / value types.
- We can not create variable/instance of value type using new operator.

```
int num1 = 10;  //OK
int num1 = new int( 10 );//NOT OK
```

- Variable of value type get space on Java stack.
- A variable of value type contains value.
- If we try to initalize/assign variable of value type then value gets copied.

```
int num1 = 10;    //OK
int num2 = num1;    //OK
```

- We can not store null value inside variable of value type.

```
int num1 = null;    //NOT OK
```

- Field of value type, by default contains 0.

```
class Test{
    private boolean b;  //false -> 0
    private int i;      //0
    private float f;    //0.0
    private double d;   //0.0
    private long l;     //0
}
```

Reference Type

- Non primitive type is called as reference type.
- Interface, Class, Enum, Array are non pritive type / reference type.
- To create instance of reference type it is mandatory to use new operator.

```
Complex c1(10,20);  //NOT OK
Complex c1 = new Complex( 10, 20 ); //OK
```

- A variable of reference type contains reference.
- Instance of reference type get space on heap.
- If we try to initalize/assign variable of reference type then reference gets copied.

```
Complex c1 = new Complex(10,20);
Complex c2 = c1;    //OK
```

- We can store null value inside variable of reference type.

```
Complex c1 = null;  //OK
```

- Field of reference type by default contains null value.

```
class Date{
}
class Employee{
    private Date joinDate;  //null
}
```

Characteristics of Instance

- Only Fields get space once per instance and according to order of their declaration inside class.
- Method do not get space inside instance rather all the instances of same class share single copy it.
- Instances share single copy of method by passing this reference.

State

- Data/value stored inside instance is called as state.

```
int num1 = 10;  //state of num1 is 10
Employee emp = new Employee("Sandeep",33,15000);
//State of instance is "Sandeep",33,15000
```

- value of of the field represents state of instance.

Behavior

- Set of operations which are allowed to perorm on instance is called behavior of instance.

```
Employee emp = new Employee();
emp.acceptRecord( );
emp.printRecord( );
//behavior of instance is acceptRecord and printRecord
```

- Method defined inside class represent behavior of instance.

Identity

- It is a property of instance which is used to identify instance uniquely.
- A value of any field which is used to identify instance uniquely represents it identity.

Class

- Class is collection of fields and methods.
- Structrue(skeleton) and behavior of instance depends on class hence class is considered as a template/model/blueprint for an instance.
- Class is collection/set of such instances which is having common structure and common behavior.
- Tata NANO, Maruti 800, Ford Figo -> class -> Car
- Akash, Yogesh, Nitin, Nilesh, Sandeep -> class -> Person
- Let US C, Mastering DBT, Core Java Vol-1 -> class -> Book
- NOKIA 1100, One Plus 5T, iPhone6 -> class -> MobilePhone
- MacBook Air, Lenovo Thinkpad, HP Envy, Dell Inspiron -> class -> Laptop
- Class is a imaginary / logical entity.
- Class implementation represents encapsulation.

Instance

- Any entity which has physical existance is called object.
- In Java object is called as instance.
- An entity which has State, Behavior, Idenitity is called instance.
- Instance is real time / physical entity.

Instantiation

- Process of creating instance from a class is called instantiation.
- It represents abstraction.
- Field of a class which get space inside instance is called instance variable. In short, non static field declared inside class is called instance variable.
- Instance variable get space inside instance during instantiation once per instance according to order of their declaration inside class.
- A method of a class, which is designed to proces state of instance / designed to call on instance is called instance method. In short, non static method of a class is called instance method.
- If we want to access instance members then we should use object reference(this, t1, t2, t3).

Class Level variable

- If we want to share value of any field inside all the instances of same class then we should declare such field static.
- Static field do not get space inside instance rather all the instances of same class share single copy of it hence size of instance depends on size of non static field.
- Field of a class which do not get space inside instance is called class level variable. In short static field is also called as class level variable.
- To access class level variable we should use classname and dot opetator.
- Static field / class level variable get space once per class during class loading on method area.

Constructor

- If we want to initialize non static field/instance variable then we should use constructor.
- Constructor gets called once per instance

- We can call constructor from another constructor. it is called constructor chaining.

Static Initialization Block

- If we want to initialize static field/class level variable then we should use Static Initialization Block.
- Static Initialization Block gets called once per class.
- We can not call Static Initialization Block from another Static Initialization Block.
- We can write multiple static block inside class. In this case JVM execute them sequentially.

Static Method

- To access non static members of the class method should be non static and to access static members of the class method should be static.
- Static method of a class is also called as class level method and class level method is designed to call on class name.
- Why static method do not get this reference?
    - If we call non static method on instance then method get this reference.
    - Static method is designed to call on class name.
    - Since static method is not designed to call on instance it doesnt get this reference.
- Since static method do not get this reference, hence we can not access non static membrs inside it. In other words static method can access only static members of the class.
- Using instance, we can access non static members inside static method.

```
public class Program {
    private int num1 = 10;
    private static int num2 = 20;
    public static void main(String[] args) {
        //System.out.println("Num1  :   "+num1);    //Not OK
        Program p = new Program();
        System.out.println("Num1    :   "+p.num1);  //OK : 10
        System.out.println("Num2    :   "+num2);    //OK : 20
    }
}
```

- Inside method, if there is a requirement to use this reference then, we should delcare that method non static otherwise it should be static.