Day 7

Path and Classpath

- A path of a file/directory from root directory is called absolute path.
  - Example:
    - /Users/sandeepkulange/Desktop/cjonline/Day_7/Day_7.1/src
- A path of a file/directory from current directory is called relative path.
  - Example
    - ./src/Complex.java
- Path is OS platforms environment variable which is used to located java language tools.
  - In Windows
    - set path="c:\program files(x86)\java\jdk1.8\bin";
  - In Linux
    - export PATH=/usr/bin
- Classpath is Java platforms environment variable which is used to locate .class / .jar file.
  - In Windows
    - set classpath=.\bin;
  - In Linux
    - export CLASSPATH=./bin/;
- By default, classpath is set to current directory.

Final

- final is keyword in Java.
- We can use final to declare local variable / Method parameter, field, method, class, reference.
- If we dont want to modify value of variable then we should use final keyword.
- If we dont want to modify state of any field inside any method of the class then we should delare that field final.
- We can declare refernce final but we can not declare instance final.

```java
final Complex c1 = new Complex(10, 20);
c1.setReal(100);
c1.setImag(200);
//c1 = new Complex(50,60); //Not OK
```

Static

- If we want to share value of any field in all the instances of same class the we should declare it static.
- Only non static fields get space inside instance.
- Static field do not get space inside instance rather all the instances of same class share single copy of it.
- Non static field is also called instance variable. It is designed to access using object reference.
- Static field is also called as class level variable. It is designed to access using class name and dot operator.
- Instance variable get space once per instance and class level variable get space once per class.

- Class level variable get space once per class, during class loading, on method area.
- Method area is a part of JVM memory.

Static Initializer Block

- If we want to intialize static field then we should define static initializer block inside class.
- JVM execute it during class loading.
- We can define multiple static block inside class. In this case JVM executes it sequentially.

Static Method

- If we want to access non static members of the class then we should define non static member function inside class and if we want to access static members of the class then we should define static method inside class.
- A method of a class which is designed to call on instance is called instance method. In other words, non static method is also called instance method.
- A method of a class which is designed to call on class is called class level method. In other words, static method is also called class level method.
- Instance members are designed to call using object reference whereas class level members are designed to access using class name.
- 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 designe to call on instance, it doesn't get this reference.
- Since static method do not get this reference, we can not access, non static members inside static method. In other words, static method can access only static members of the class.
- Using instance, we can access, non static members inside static method.

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

- Consider following syntax:

```java
class A
{
    public static void showRecord(){    }
}
class B
{
    public static void displayRecord(){    }
    public static void printRecord( )
    {
        displayRecord( );   //OK
        B.displayRecord( );   //OK

        showRecord( );  //Not OK
        A.showRecord( );  //OK
    }
}
class Program
{
    public static void main(String[] args)
    {
        B.printRecord();
    }
}
```

Write a program to count number of instances created from the class.

Write a program to implement singleton class.

- A class from which we can create single instance is called singleton class.