Day 4

Package

- It is a java language feature that is used to
    1. To avoid name collision/clashing/ambiguity.
    2. To group functionally equivalent/related types together.
- It can contain following types:
    1. Sub package
    2. Interface
    3. Class
    4. Enum
    5. Error
    6. Exception
    7. Annotation.
- "rt.jar" contains following main packages:
    1. sun
    2. org
    3. com
    4. java
    5. javax
- java main package contains following sub packagaes:
    1. applet
    2. awt
    3. beans
    4. io
    5. lang
    6. math
    7. net
    8. nio
    9. rmi
    10. security
    11. sql
    12. text
    13. time
    14. util
- How to declare namespace in C++

```
namespace std
{
    class Stack
    {

    };
}
```

- How to declare package

```
package p1;
class Stack
{
    //TODO : Declare member
}
```

- Package declaration statement must be first statement in .java file.

```
package p1; //Ok
package p2; //Not Ok
class Stack
{
    //TODO : Declare member
}
```

- We can not declare any type inside multiple packages.
- If we declare any type inside package then it is called packaged type otherwise it is called as unpackaged type.
- package name is physically mapped to folder/directory.
- If we want to access type from different package then we should use F.Q. Type name or import statement.

```
class Program
{
    public static void main(String[] args)
    {
       //Stack s1 =  new Stack( );//Not OK
       p1.Stack s1 = new p1.Stack(); //Ok
    }
}
```

- Using import

```
//import p1.*;    //OK
import p1.Stack;    //OK
class Program
{
    public static void main(String[] args)
    {
       p1.Stack s1 = new p1.Stack(); //Ok
       Stack s2 = new Stack(); //Ok
    }
}
```

- In java, default access modifier of any type is package level private

```
package p1;
??? class Complex
{   }
// ??? -> package level private
```

- If we want to access type outside package then it must be public.
- Access modifier of a type can be either package level private or public Only. In other words, we can declare type private or protected.

```
package p1;
public class Complex
{   }
```

- According java language specification, name of public class and .java file must be same.
- Commands to compile and execute program

```
javac -d ./bin/ ./src/Complex.java
export CLASSPATH=./bin/
javac -d ./bin/ ./src/Program.java
java Program
```

- java.lang package is by default imported in every .java file hence importing lang package is optional.
- If we define any type without package then it is considered as member of default package.
- Since we can not import default package, it is impossible to access unpackaged type from packaged type. But we can access packaged type from unpackaged type.
- we can define types in different package

```
package p1;
public class Complex
{   }
```

```
package p2;
import p1.Complex;
public class Program
{
    public static void main(String[] args)
    {
        Complex c1 = new Complex();
```

```
    }
}
```

- We can define types in same package. In this case use of import statement is optional.

```
package p1;
public class Complex
{   }
```

```
package p1;
//import p1.Complex;  //OK
public class Program
{
    public static void main(String[] args)
    {
        Complex c1 = new Complex();
    }
}
```

- Math is a final class declared in java.lang package.
- It contains fields and methods for performing numeric operations.
- All the members of Math class are static.
- without typename, if we want to access static members then we should use static import statement

```
import static java.lang.System.out;
//import static java.lang.Math.*;
import static java.lang.Math.PI;
import static java.lang.Math.pow;
class Program
{
    public static void main( String[] args )
    {
        float radius = 10.0f;
        //float area = ( float )( Math.PI * Math.pow(radius, 2) );
        float area = ( float )( PI * pow(radius, 2) );
        out.println("Area    :   "+area);
    }
}
```

Static in Java

- According oops, static variable is called class level variable and it must exist at class scope. Hence we can not declare local variable static.
- In java, we can not declare local variable static but we can declare field static.

```
class Test
{
        private static int count; //OK
        public void print( )
        {
        //static int count; //Not Ok
                ++ count;
                System.out.println("Count       :       "+count);
        }
}
```

- If we want to share, value of any field in all the instances of same class then we should declare that field static.
- In java, static field get space during class loading, once per class on method area.
- Non static field is also called as instance variable. It gets space once per instance.
- static field is also called as class level variable. It gets space once per class.
- To access instance members we should use instance whereas to accss class level members we should use class name and dot operator.

```
class Test
{
        int x = 10;     //Instance variable
        static int y = 20;      //Class level variable
}
public class Program
{
        public static void main(String[] args)
        {
                Test t = new Test();
                //System.out.println("X :       "+Test.x );     //Not OK
                System.out.println("X   :       "+t.x );        //OK

                System.out.println("Y   :       "+Test.y);      //Ok
                System.out.println("Y   :       "+t.y); //Ok
        }
}
```

- Non static method is also called as instance method and static method is also called as class level method.
- Instance method is designed to call on instance whereas class level method is designed to call on class name.
- since non static field get space inside instace, we should initialize it inside constructor. JVM invoke constructor, once per instance.
- To intialize static field, we should use static initializer block. JVM invoke it once per class.
- we can write multiple static initializer block inside class.

```
class Test
{
        private int num1;
        private int num2;
        private static int num3;
        static  //static initializer block
        {
                Test.num3 = 0;
        }
        public Test( )
        {
                this( 0, 0);
        }
        public Test( int num1, int num2 )
        {
                this.num1 = num1;
                this.num2 = num2;
    }
}
```

- If we want to access non static members of the class then we should define non static method / instance method inside class.
- instance method is designed to call on instance.
- If we want to access static members of the class then we should define static method/class level method inside class.
- class level method is designed to call on class name.
- Static method do not get this reference.
- this reference is considered as link between non static field and non static method.
- Since static method do not get this reference, we can not access non static members inside static method directly.
- In other words, static method can access static members of the class only.
- 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
                System.out.println("Num2        :       "+num2); //OK
    }
}
```

- We can not declate local variable and top level class static.

Singleton class

- It is a creational design pattern
- It allows us to create only one instance.

```
class Singleton
{
        private Singleton( )
        {       }
        static Singleton instance;
        public static Singleton getInstance( )
        {
                if( instance == null )
                        instance = new Singleton();
                return instance;
        }
}
```

Final

- After storing value, if we dont want to modify state/value of the variable then we should use final keyword.

```
final int number = 10;
++ number;  //Not OK
System.out.println( number );
```

```
final int number = 10;  //OK
System.out.println( number ); //OK
```

```
final int number;  //OK
number = 10;    //OK
System.out.println( number ); //OK
```

```
final int number = 10;  //OK
number = 20;//Not OK
System.out.println( number );
```

- We can provide value to the final variable at runtime.

```
final int number;
System.out.print("Number        :       ");
```

```
number = sc.nextInt();
```

- If we dont want to modify state field inside any method of the class(including ctor) then we should declare field final.
- Note : If we want to declare field final then we should declare it static also.

```
class Math
{
        public static final double PI = 3.14;
}
```

- In java, we can not delare instance final but we can declare reference final.

```
final Complex c1 = new Complex(10, 20); //OK
c1.setReal(11); //Ok
c1.setImag(22); //Ok
//c1 = new Complex(50, 60); //Not OK
System.out.println(c1.toString());
```

Array

- It is linear data structure which is used to store elements of same type in continous memory location.
- If we want to access elements of array then it is necessary to use index and subscript operator.
- Array index always begins with 0.
- In java, array is reference type i.e array instance get space on heap section.
- Types of array
    1. Single dimensional
    2. Multi dimensional
    3. Ragged array
- Types loop
    1. do-while
    2. while
    3. for
    4. foreach (it is also called iterator)
- If we want to manipulate elements of array then we should use java.util.Arrays class.