Day 8

Object Class

- In java parent class is called super class and child class is called sub class.
- Consider C++ Code:

```
class Person //Parent/Base class
{   };
class Employee : public Person //Child/Derived class
{   };
```

- Consider Java Code:

```
class Person //Parent/Super class
{   };
class Employee extends Person //Child/Sub class
{   };
```

- During inheritance, all the members (except constructor) of super class inherit into sub class.

```
class A
{
    public void print( )
    {
        System.out.println("Hello World");
    }
}
class B extends A
{   }
class Program
{
    public static void main(String[] args)
    {
       A a = new A();
       a.print();   //OK

       B b = new B();
       b.print( );  //OK
    }
}
```

```
class A{   }
//Super class of B is class A
```

```
class B extends A{   }
//B is direct super class and A is indirect super class of class C
class C extends B{   }
```

- Object is non final concrete class which is declared in java.lang package.
- It is ultimate base class / super cosmic base class / root class in java class hierarchy.
- In other words, there is no super class of Object class but every class is extending Object class as a super class.

```
class Program
{
    public static void main(String[] args)
    {
        //Integer(int value)
        Integer i1 = new Integer( 125 ); //OK
        //Integer(String s)
        Integer i2 = new Integer( "125" );//OK
        Integer i3 = new Integer( );//Not OK
    }
}
```

- java.lang.Object class do not contain Nested types(interface/class/enum).
- java.lang.Object class do not contain field.
- It contains only default constructor.

```
Object o1 = new Object("SunBeam");  //Not OK
Object o2 = new Object( );  //OK
```

- There 11 methods in java.lang.Object class[5 non final methods + 6 final methods].
- We can not override/redefine final method in sub class but we can override non final method in sub class.
- Native method is a such method which is implemeted in C++ and designed to call from java language.
- Following are 5 non final methods of java.lang.Object class
    1. public String toString();
    2. public boolean equals(Object obj);
    3. public native int hashCode();
    4. protected native Object clone() throws CloneNotSupportedException;
    5. protected void finalize() throws Throwable;
- Following are 6 final methods of java.lang.Object class 6. public final native Class<?> getClass(); 7. public final void wait() throws InterruptedException; 8. public final native void wait(long) throws InterruptedException; 9. public final void wait(long, int) throws InterruptedException; 10. public final native void notify(); 11. public final native void notifyAll();

toString() implementation

- toString is non final method of java.lang.Object class.

- Syntax:

```
    public String toString( );
```

- If we want to return state of current instance in string format the we should use toString() method.
- If we do not define toString() method inside class then super class's toString() method gets called.
- toString() method of java.lang.Object class returns string in following format:

```
this.getClass().getName()+'@'+Integer.toHexString(this.hashCode())
//F.Q.TypeName@Hashcode
```

- According to client's requiremnt, if implementation of super class method is partially complete then we should override/redefine method in sub class.
- The result in toString method should be a concise but informative that is easy for a person to read.
- It is recommended that all subclasses override this method.

Package

- It is a java language feature which is used:
    1. To avoid name clashing/collision/ambiguity.
    2. To group/organize, functionally related/equivalent types together.
- package is a keyword in java.
- If we want to define any type inside package then we should use package delaration statament.
- Package declaration statement must be first statement in .java file.
- We can declare following elements inside package:
    1. Sub package
    2. Interface
    3. Class
    4. Enum
    5. Exception
    6. Error
    7. Annotation Types.
- Package name is physically mapped to the folder/directory,
- If we define any type inside package then it is called packaged type otherwise unpackaged type.
- If we want to use packaged type from unpackaged type then either we should use F.Q.Typename or import statement.
- If we define any type then its default access modifier is package level private.
- Example1

```
??? class Complex   //???=>Package level private
{   }
```

- Example2

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

- If we want to use packaged type from unpackaged type then we must declare packaged type public.

```
package p1;
public class Complex   //???=>Package level private
{   }
```

- According to Java Language Specification(JLS), name of public type and .java file must be same.
- If we define any type without package then it is considered as a member of default package. Since we can not import default package it is impossible to access unpackaged type from packaged type.