Day 7

Coding / Naming Conventions

```
1. Pascal Case Coding Convention
2. Camel Case Coding Convention
```

Pascal Case Coding Convention

- Example
    1. System
    2. StringBuilder
    3. NullPointerException
    4. IndexOutOfBoundsException
- In this case, Including first word, first character of each word must be in upper case.
- In Java, we should use this convention for:
    1. Type Name(Interface, Class, Enum, Annotation)
    2. File Name.

```
File Name : Program.java
class Program{
}
```

Camel Case Coding Convention

- Example
    1. main( )
    2. parseInt( )
    3. showInputDialog( )
    4. addNumberOfDays( )
- In this case, excluding first word, first character of each word must be in upper case.
- We should use this convention for:
    1. Method local variable
    2. Method parameter
    3. Field
    4. Method
    5. Object referece / reference

```
File Name : Complex.java
class Complex{
    private int real;
    private int imag;
```

```
    public int getReal( );
    public void setReal( int real );
    public int getImag( );
    public void setImag( int imag );
}
```

Coding convention for final variable and enum constant

- Final Variable

```
class Math{
    public static final double PI = 3.142;
}
```

- Name of the final variable / field must be in upper case.
- enum

```
enum Color{
    RED, GREEN, BLUE
}
```

- Name of the enum constant must be un upper case.

Naming convention for package

- Example
    1. java
    2. java.lang
    3. java.lang.reflect
    4. java.util
    5. java.io
    6. com.mysql.cj.jdbc
    7. org.cdac.sunbeam.dac
- Name of the package should be in lower case.

Path

- Example: C:\CDAC\Sunbeam\dac\java\Day6\Day_6.1\src\Program.java
- Path contains:
    1. Root directory
    2. Path seperator character.
    3. Sub directories
    4. File Name

Absolute path

- A path of a file from root directory is called absolute path.
- Example: C:/CDAC/Sunbeam/dac/java/Day6/Day_6.1/src/Program.java

Relative path

- A path of a file from current directory is called relative path. cd c: cd CDAC cd SunBeam cd dac cd java cd Day6 cd Day_6.1
- How to refere Program.java
    1. C:/CDAC/Sunbeam/dac/java/Day6/Day_6.1/src/Program.java
    2. .\src\Program.java

Package

- It is java language feature that we can use to group functionally equivalent types together and to avoid name ambiguity.
- Package can contain sub package, interface, class, enum, error, exception and type annotation.
- package is a keyword in Java.
- How to add type inside package?
    - File Name : Complex.java

```
class TComplex{

}
//Output : TComplex.class
```

```
- File Name : Complex.java
```

```
package p1; //OK
class TComplex{ //OK
}
```

- If we define any type/class inside package then it is called packaged type/class.
    - File Name : Complex.java

```
class TComplex{ //OK
}
package p1; //NOT OK
```

```
- File Name : Complex.java
```

```
import java.util.Scanner; //OK
package p1; //NOT OK
class TComplex{ //OK
}
```

```
- File Name : Complex.java
```

```
package p1; //OK
package p2; //NOT OK
class TComplex{ //OK
}
```

- Package declaration statement must be first statement inside .java file.
    - File Name : Complex.java

```
package p1,p2; //NOT OK
class TComplex{ //OK
}
```

- We can define class inside one package only.
    - File Name : Complex.java

```
package p1; //OK
class TComplex{ //OK
}
```

- Steps:
    1. set path(optional : open jdk)
    2. Compile Complex.java
        - javac -d ./bin ./src/Complex.java
        - Output : p1/TComplex.class
- Package name is pysically mapped to the folder.
    - File Name : Complex.java

```
package p1; //OK
class TComplex{ //OK
}
```

```
- File Name : Program.java
```

```
import p1.TComplex;
class Program{
    public static void main(String[] args) {
        //Step No 1 : Use F.Q. Class/Type Name
        //p1.TComplex c1 = new p1.TComplex( );

        //Step No 2 : Use import statement
        TComplex c1 = new TComplex( );
    }
}
```

- If we want to use packaged type inside different package then
    1. Either we should use F.Q. Type name
    2. Or we should use import statement.
- Default access modifier of any type is package level private/default.

```
package p1; //OK
??? class TComplex{
}
```

- Here "???" means package level private / default.
- If access modifier of any type is package level private the we can not use it in different package.
- If we want to use any type outside package then access modifier of type must be public.
    - File Name : Complex.java

```
package p1; //OK
public class TComplex{  //OK
}
```

```
- File Name : Complex.java
```

```
package p1; //OK
private class TComplex{  //NOT OK
}
```

```
- File Name : Complex.java
```

```
package p1; //OK
protected class TComplex{  //NOT OK
}
```

- Access modifier of type can be either package level private(default) or public only.
- According to Java Language Specification, name of the public type and name of the .java file must be same.
    - File Name : Complex.java

```
package p1; //OK
public class Complex{ //Packaged Class
}
```

```
- File Name : Program.java
```

```
import p1.Complex;
class Program{  //Unpackaged Class
    public static void main(String[] args) {
        //Step No 1 : Use F.Q. Class/Type Name
        //p1.Complex c1 = new p1.Complex( );

        //Step No 2 : Use import statement
        Complex c1 = new Complex( );
    }
}
```

- Compilation Steps

```
    javac -d ./bin ./src/Complex.java   =>  Complex.class
    set classpath=./bin;
    javac -d ./bin ./src/Program.java   =>  Program.class
    java Program
```

- We can use packaged types inside unpackaged type.
- Consider unpackaged class
    - Complex.java

```
public class Complex{   //Unpackaged class=> part of default package
    public void print( ){
```

```
        System.out.println("Hello World");
    }
}
```

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

- If we define any type without package then it is considered as a member of default package.
- We can not import default package.
- Since we can not import default package, it is not possible to use unpackaged type from packaged type.
- Consider packaged class

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

```
javac -d ./bin ./src/Program.java --Error
```

- Consider packaged class
    - Complex.java

```
package p1;
public class Complex{   //packaged class
    public void print( ){
        System.out.println("Hello World");
    }
}
```

- Consider packaged class

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

```
        c1.print( );
    }
}
```

- Steps for compilation
    1. javac -d ./bin ./src/Complex.java //p1/Complex.class
    2. export CLASSPATH=./bin
    3. javac -d ./bin ./src/Program.java //p2/Program.class
    4. java p2.Program
- Consider packaged class
    - Complex.java

```
package p1;
public class Complex{   //packaged class
    public void print( ){
        System.out.println("Hello World");
    }
}
```

- Consider packaged class

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

- Steps for compilation
    1. javac -d ./bin ./src/Complex.java //p1/Complex.class
    2. export CLASSPATH=./bin
    3. javac -d ./bin ./src/Program.java //p1/Program.class
    4. java p1.Program
- Consider packaged class
    - Complex.java

```
package p1.p2;
public class Complex{   //packaged class
    public void print( ){
```

```
        System.out.println("Hello World");
    }
}
```

- Consider packaged class

```
package p1.p3;
import p1.p2.Complex;  //Optional
public class Program {  //Packaged class
    public static void main(String[] args) {
        Complex c1 = new Complex( );
        c1.print( );
    }
}
```

- Steps for compilation
    1. javac -d ./bin ./src/Complex.java //p1.p2/Complex.class
    2. export CLASSPATH=./bin
    3. javac -d ./bin ./src/Program.java //p1.p3/Program.class
    4. java p1.p3.Program