OBJECT ORIENTED PROGRAMMING USING JAVA

OUTLINE

- Modifiers IN JAVA
- Class level access modifiers
- Member level access modifiers

JAVA MODIFIERS

- A modifier is a keyword placed in a class, method or variable declaration that changes how it operates.

- There are two types of java modifiers, they are

➢ **Java Access Modifiers**

- ❑ public
- ❑ private
- ❑ protected
- ❑ Default (No modifiers

➢ **Non-Access Modifiers**

- ❑ final
- ❑ static
- ❑ abstract
- ❑ synchronized
- ❑ strictfp
- ❑ native
- ❑ transient
- ❑ volatile

JAVA ACCESS MODIFIER

- Usage of these access modifiers is restricted to **two levels**. The two levels are **class level access modifiers** and **member level access modifiers.**

1. **Class level access modifiers (java classes only):** Only two access modifiers is allowed, public and no modifier
   - If a class is ‘public’, then it CAN be accessed from ANYWHERE.
   - If a class has ‘no modifier’, then it CAN ONLY be accessed from ‘same package’.

2. **Member level access modifiers :** For the inner classes all the four access modifiers are allowed
   - public, private, protected and no modifier is allowed

PREDICT THE OUTPUT:

```
public class Main
{
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

```
class Main
{
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

```
private class Main
{
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

```
protected class Main
{
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

PREDICT THE OUTPUT:

```java
public class Main
{
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

```
Hello World
```

```java
class Main
{
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

```
Hello World
```

```java
private class Main
{
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

```
Main.java:11: error: modifier private not allowed here
private class Main
        ^
1 error
```

```java
protected class Main
{
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

```
Main.java:11: error: modifier protected not allowed here
protected class Main
          ^
1 error
```

DIFFERENT PACKAGES WITH PUBLIC ACCESSIBILITY

```
package pack1;
public class Test
{
        public void methodOne( )
        {
                System.out.println("Public Access Modifier");
        }
}
```

```
package pack2;
import pack1.Test;
class Test1
{
        public static void main(String args[])
        {
                Test t=new Test();
                t.methodOne();
        }
}
```

SAME PACKAGE WITH DEFAULT ACCESSIBILITY

```
package pack1;
class Test
{
        public void methodOne( )
        {
                System.out.println("Default Access Modifier");
        }
}
```

```
package pack1;
import pack1.Test;
class Test1
{
        public static void main(String args[])
        {
                Test t=new Test();
                t.methodOne();
        }
}
```

DIFFERENT PACKAGES WITH DEFAULT ACCESSIBILITY

```
package pack1;
class Test
{
        public void methodOne( )
        {
                System.out.println("Default Access Modifier");
        }
}
```

```
package pack2;
import pack1.Test;
class Test1
{
        public static void main(String args[])
        {
                Test t=new Test();
                t.methodOne();
        }
}
```

MEMBER LEVEL ACCESS MODIFIERS

- • public and no modifier – the same way as used in class level.
- private – members CAN ONLY access.
- protected – CAN be accessed from ‘same package’ and a subclass existing in any package can access**. (outside package we can access protected members only in child classes and should be by child reference only that is we can't use parent reference to call protected members from outside package.)**

ACCESSIBILITY WHILE USING DIFFERENT MODIFIERS

	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

- **The least accessible modifier is private.**
- **The most accessible modifier is public.**

DIFFERENT PACKAGES WITH DEFAULT ACCESSIBILITY

```
package pack1;
class Test
{
        public void methodOne( )
        {
                System.out.println("Public  Member Access Modifier");
        }
}
```

```
package pack2;
import pack1.Test;
class Test1
{
        public static void main(String args[])
        {
                Test t=new Test();     // Cannot Access
                t.methodOne();       //Cannot access
        }
}
```

SAME PACKAGE WITH DEFAULT ACCESSIBILITY

```
package pack1;
class Test
{
        void methodOne( )
        {
                System.out.println("Public  Member Access Modifier");
        }
}
```

```
package pack1;
import pack1.Test;        //This is optional
class Test1
{
        public static void main(String args[])
        {
                Test t=new Test();      //Allow to access
                t.methodOne();       //Allow to access
        }
}
```

DIFFERENT PACKAGES WITH DEFAULT ACCESSIBILITY

```
package pack1;
class Test
{
        void methodOne( )
        {
                System.out.println("Public  Member Access Modifier");
        }
}
```

```
package pack2;
import pack1.Test;
class Test1
{
        public static void main(String args[])
        {
                Test t=new Test();      //Cannot access
                t.methodOne();          //Cannot access
        }
}
```

SAME PACKAGE WITH PUBLIC ACCESSIBILITY AND EXTENDS

```
package pack1;
public class Test
{
        protected void methodOne( )
        {
                System.out.println("Protected Member Access Modifier");
        }
}
```

```
package pack1;
import pack1.Test; // Optional
class Test1  extends Test
{
        public static void main(String args[])
        {
                Test t=new Test();
                t.methodOne();     //Can access
                Test1  b = new Test1();
                b.methodOne();    //Can access
                Test c = new Test1( );
                c.methodOne(); //Can access due to Same Package
        }
}
```

DIFFERENT PACKAGES WITH PUBLIC ACCESSIBILITY AND EXTENDS

```
package pack1;
public class Test
{
        protected void methodOne( )
        {
                System.out.println("Protected Member Access Modifier");
        }
}
```

```
package pack2;
import pack1.Test;
class Test1 extends Test
{
        public static void main(String args[])
        {
                Test t=new Test();
                t.methodOne(); // Cannot Access
                Test1 b = new Test1();
                b.methodOne();  // Can Access
                Test c = new Test1();
                c.methodOne();  // Cannot Access due to parent Reference
        }
}
```

IMPORTANT POINTS:

- public is the least restrictive access modifier in Java programming language and its bad practice to declare variable, method or class by default public

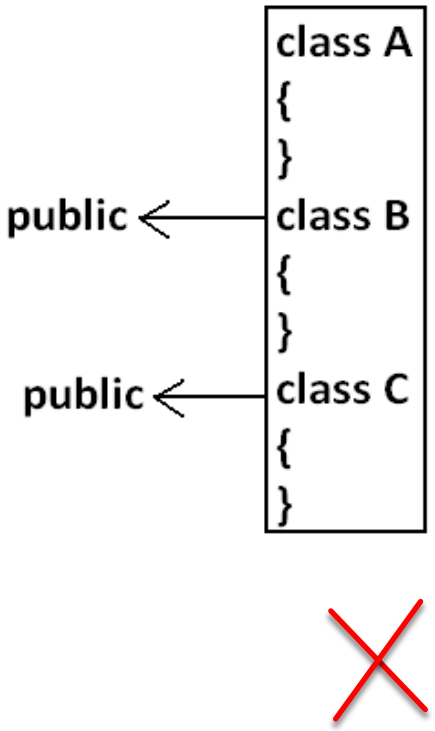
- Making class or instance variable public is also violated principle of Encapsulation which is not good at all and affects maintenance badly.

- Instead of making variable public you should make it private and provided public getter and setter.

- In Java name of file must be same with public class declared in the file.

IMPORTANT POINTS:

- A java Program can contain any no. of classes but at most one class can be declared as public.
- If there is a public class the name of the Program and name of the public class must be matched otherwise we will get compile time error.
- If there is no public class then any name we gives for java source file.
- If there is no public class then we can use any name for java source file there are no restrictions



IMPORTANT POINTS:

- The main () method of an application has to be public. Otherwise, it could not be called by a Java interpreter (such as java) to run the class.

- private keyword in Java: private keyword or modifier in java can be applied to member field, method or nested class in Java.

- You cannot use private modifier on top level class. private variables, methods and class are only accessible on the class on which they are declared.

- Private is highest form of Encapsulation Java API provides and should be used as much as possible.

PREDICT THE OUTPUT:

```
class access
    {
        public int x;
    private int y;
        void cal(int a, int b)
        {
            x =  a + 1;
            y =  b;
        }
    }
    public class Main
    {
        public static void main(String args[])
        {
            access obj = new access();
            obj.cal(2, 3);
            System.out.println(obj.x + " " + obj.y);
        }
   }
```

PREDICT THE OUTPUT:

```
class access
    {
        public int x;
    private int y;
        void cal(int a, int b)
        {
            x =  a + 1;
            y =  b;
        }
    }
    public class Main
    {
        public static void main(String args[])
        {
            access obj = new access();
            obj.cal(2, 3);
            System.out.println(obj.x + " " + obj.y);
        }
   }
```

Output

```
Main.java:17: error: y has private access in access
            System.out.println(obj.x + " " + obj.y);
                                             ^
```

CAN YOU CREATE A SUB CLASS TO THE FOLLOWING CLASS?

```
class A
{
    private A()
    {
        //First Constructor
    }

    private A(int i)
    {
        //Second Constructor
    }
}
```

CAN YOU CREATE A SUB CLASS TO THE FOLLOWING CLASS?

```
class A
{
    private A()
    {
        //First Constructor
    }

    private A(int i)
    {
        //Second Constructor
    }
}
```

Answer: No, you can't create sub classes to that class which has only private constructors.

ENCAPSULATION EXAMPLE:

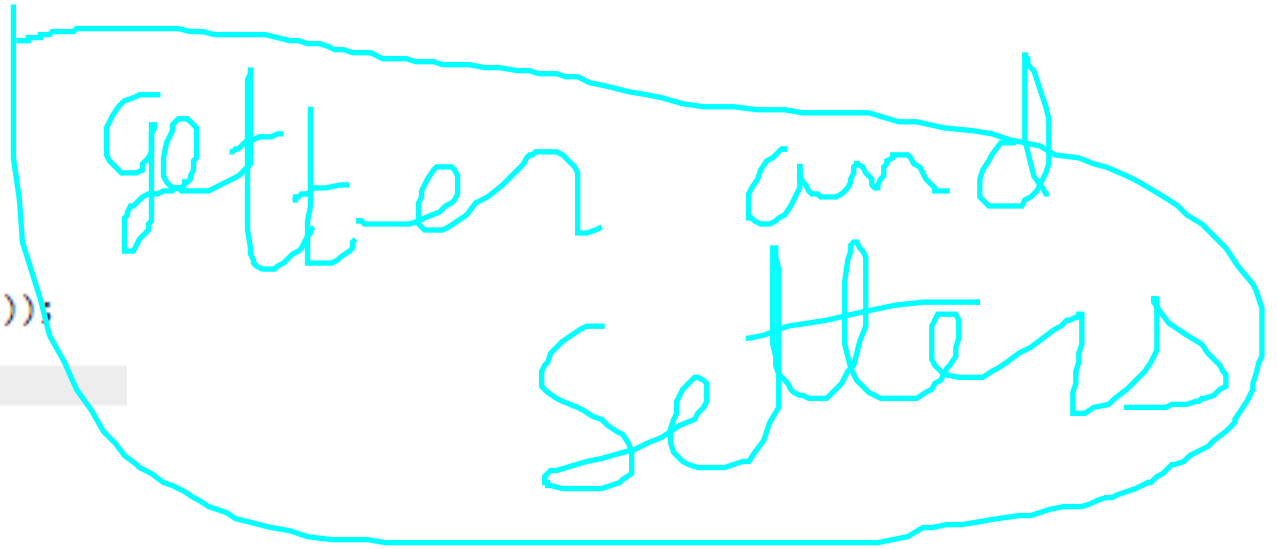
```
class Person {
   private int age;

   public int getAge() {
      return age;
   }

   public void setAge(int age) {
      this.age = age;
   }
}

class Main {
   public static void main(String[] args) {
      Person p1 = new Person();
      p1.setAge(24);
     System.out.println("My age is " + Age());
      System.out.println("My age is " + p1.getAge());
   }
}
```

- The whole idea behind encapsulation is to hide the implementation details from users. If a data member is private it means it can only be accessed within the same class. No outside class can access private data member (variable) of other class.



WHY WE CAN'T INSTANTIATE CLASS-A IN THE BELOW CODE OUTSIDE THE PACKAGE EVEN THOUGH IT HAS PUBLIC CONSTRUCTOR?

```
package pack1;

class A
{
    public A()
    {
        //public constructor
    }
}

package pack2;

import pack1.*;

class B
{
    A a = new A();        //Compile Time Error
}
```

THANK YOU

?