



# Object Oriented Programing

Name: Rafi Ullah

Department: BS(CS) 5th Semester

ID: 14283

Submitted To: Sir Ayub sahab

Date: 29/06/2020

Q1. a. Why access modifiers are used in java, explain in detail Private and Default access modifiers?

Answer:

The access modifiers in Java specifies the accessibility of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it. Access modifiers are utilized for encapsulation: they allow you to organize your code in packages and classes, and have just an open interface noticeable to the outside, while hiding the implementation details which you need to do, so you can later change it without telling anybody.

1) Private Access modifier:

The extent of private modifier is restricted to the class as it were.

- Private Data members and methods are just open inside the class.
- Class and Interface can't be declared as Private.
- If a class has private constructor, at that point you can't create the object of that class from outside of the class.

2) Default access modifiers:

When we do not mention any access modifier, it is called default access modifier. The scope of this modifier is constrained to the package as it were. This implies on the off chance that we have a class with the default access modifier in a package, just those classes that are in this package can access this class. No different class outside this package can access this class. Additionally, on the off chance that we have a default method or Data member in a class, it would not be noticeable in the class of another package.

Q1:b. Write a specific program of the above mentioned access modifiers in java?

Answer:

1) Program code for Default Access modifier:

```
package Default_Modifier;

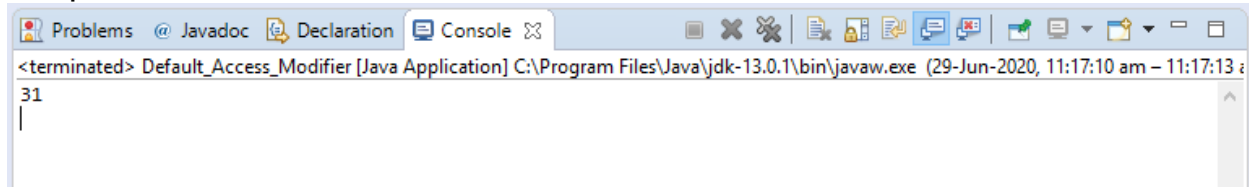
public class Default_Access_Modifier {

    public static void main(String[] args) {
        Addition obj = new Addition();
        obj.addTwoNumbers(10,21);
    }
}
```

```
package Default_Modifier;

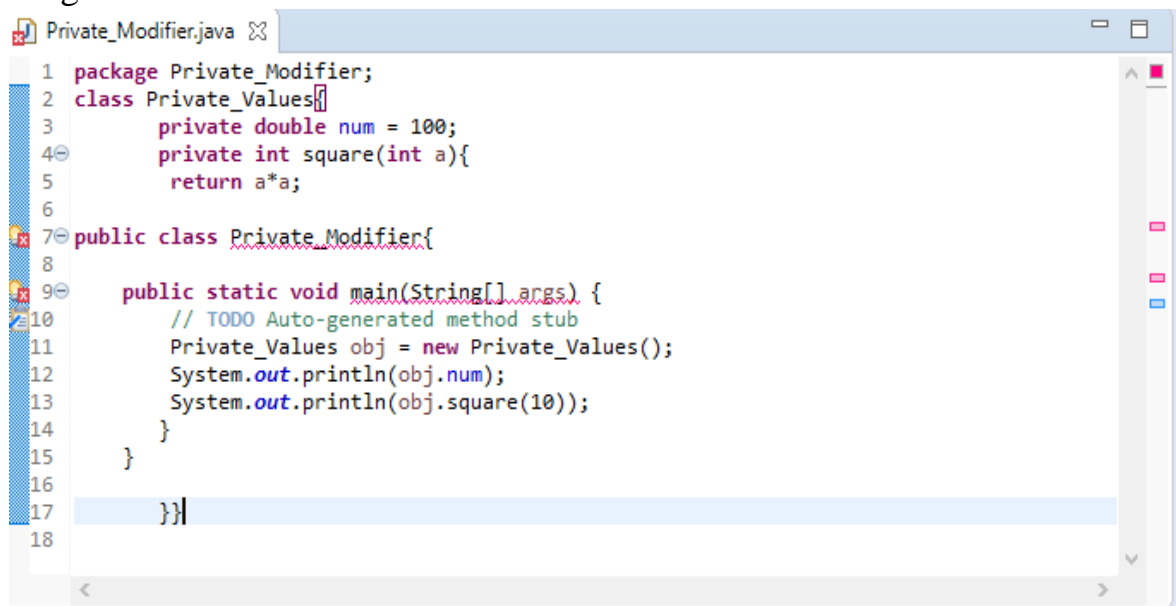
public class Addition {
    int addTwoNumbers(int a, int b){
        System.out.println(a+b);
        return a+b;
    }
}
```

Output:



```
<terminated> Default_Access_Modifier [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (29-Jun-2020, 11:17:10 am - 11:17:13 a
31
|
```

2) Program code for Private Access modifier:



```
1 package Private_Modifier;
2 class Private_Values {
3     private double num = 100;
4     private int square(int a){
5         return a*a;
6     }
7 }
8
9 public class Private_Modifier{
10
11     public static void main(String[] args) {
12         // TODO Auto-generated method stub
13         Private_Values obj = new Private_Values();
14         System.out.println(obj.num);
15         System.out.println(obj.square(10));
16     }
17 }
18 }
```

Output:

Illegal modifier for the local class Private\_Modifier; only abstract or final is permitted.  
Error Accoured.

Q2. a. Explain in detail Public and Protected access modifiers?

Answer:

1) Public access modifier:

The members, methods and classes that are declared public can be accessed from anyplace. It can be accessed from within the class, outside the class, within the package and outside the package. This modifier doesn't put any limitation on the access.

2) Protected access modifier:

Protected data member and method are just accessible by the classes of the same package and the sub-classes present in any package. You can likewise say that the protected access modifier is like default access modifier with one exception that it has visibility in sub classes. Protected access modifier is commonly utilized in a parent child relationship.

Q2. b. Write a specific program of the above mentioned access modifiers in java?

Answer:

1) Program code for Public access modifier:

```
package Public_Modifier;
```

```
import Public.ABC;
```

```
public class Public_Class_Modifier {
```

```
public static void main(String[] args) {
```

```
// TODO Auto-generated method stub
```

```
ABC obj= new ABC();
```

```
obj.msg();
```

```
}
```

```
}
```

```
// Class which we can access with Object
```

```
package Public;
```

```
public class ABC {
```

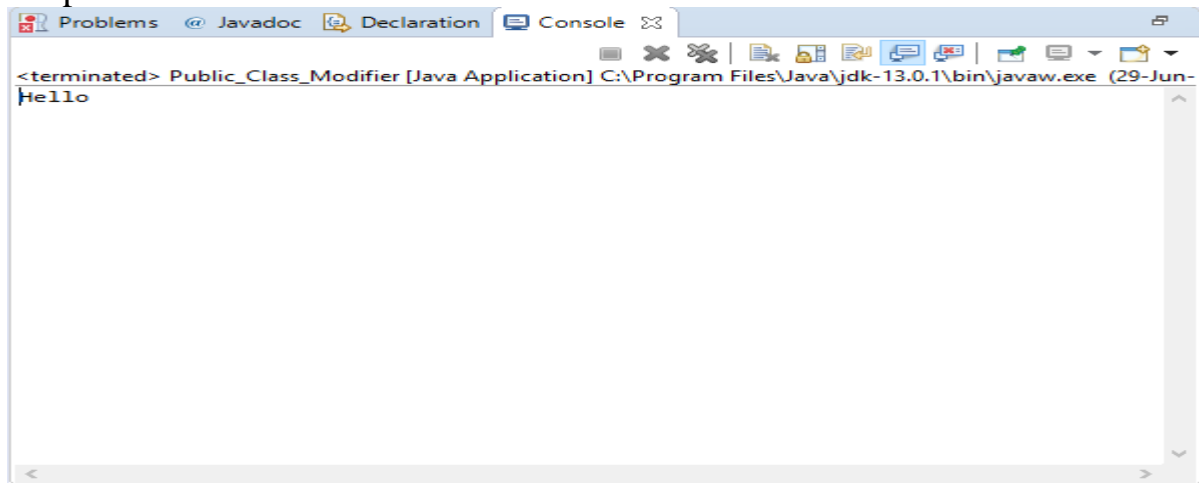
```
public void msg(){
```

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

```
}
```

```
}
```

Output:



```
<terminated> Public_Class_Modifier [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (29-Jun-2020)
Hello
```

2) Program code for Protected access modifier:

```
package Protected;

import Public_Modifier.Protcted;

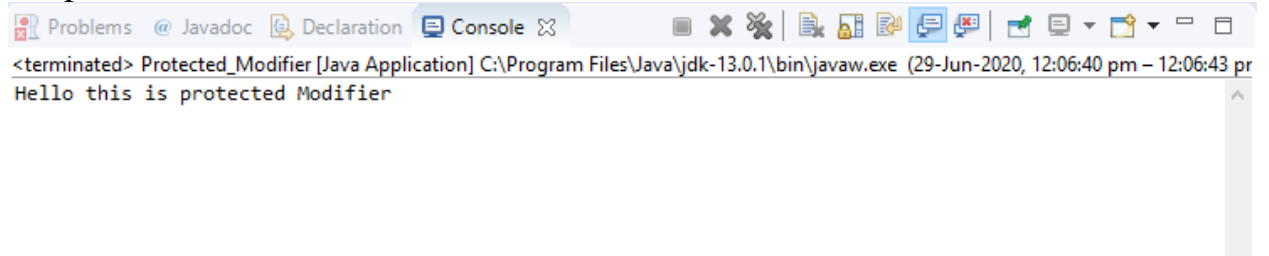
class Protected_Modifier extends Protcted {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Protected_Modifier obj = new Protected_Modifier();
        obj.msg();
    }

    //Class we are accessing
    package Public_Modifier;

    public class Protcted {
        protected void msg(){
            System.out.println("Hello this is protected Modifier");
        }
    }
}
```

Output:



```
Problems @ Javadoc Declaration Console
<terminated> Protected_Modifier [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (29-Jun-2020, 12:06:40 pm - 12:06:43 pr
Hello this is protected Modifier
```

Q3. a. What is inheritance and why it is used, discuss in detail?

Answer:

Inheritance in Java is the method to create a hierarchy between classes by inheriting from other classes.

The process by which one class gets the properties (data members) and functionalities (methods) of another class is called inheritance. The point of inheritance is to give the re-usability of code with the goal that a class needs to compose just the one of a kind highlights and rest of the basic properties and functionalities can be extended from the another class.

Inheritance allows us to reuse of code, it improves reusability in your java application.

The biggest advantage of Inheritance is that the code that is already present in base class need not be rewritten in the child class. This means that the data members (instance variables) and methods of the parent class can be used in the child class as.

Q3:b. Write a program using Inheritance class on Animal in java?

Answer:

Animals Class code:

```
package Animals;

public class Animals {
    private boolean vegetarian;

    private String eats;

    private int noOfLegs;

    public Animals(boolean veg, String food, int legs){
        this.vegetarian = veg;
        this.eats = food;
        this.noOfLegs = legs;
    }

    public boolean isVegetarian() {
        return vegetarian;
    }
}
```

```

}

public void setVegetarian(boolean vegetarian) {
    this.vegetarian = vegetarian;
}

public String getEats() {
    return eats;
}

public void setEats(String eats) {
    this.eats = eats;
}

public int getNoOfLegs() {
    return noOfLegs;
}

public void setNoOfLegs(int noOfLegs) {
    this.noOfLegs = noOfLegs;
}
}

```

Dogs class code:

```

package Animals;

public class Dogs extends Animals {
    private String color;

    public Dogs(boolean veg, String food, int legs) {
        super(veg, food, legs);
        this.color="White";
    }

    public Dogs(boolean veg, String food, int legs, String color){
        super(veg, food, legs);
        this.color=color;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }
}

```

```
    public static void main(String[] args) {  
    }  
}}
```

Cat class code:

```
package Animals;
```

```
public class Cat extends Animals {  
    private String color;  
  
    public Cat(boolean veg, String food, int legs) {  
        super(veg, food, legs);  
        this.color="White";  
    }  
  
    public Cat(boolean veg, String food, int legs, String color){  
        super(veg, food, legs);  
        this.color=color;  
    }  
  
    public String getColor() {  
        return color;  
    }  
  
    public void setColor(String color) {  
        this.color = color;  
    }  
  
    public static void main(String[] args) {  
  
    }  
  
}
```

```
}
```

Main Class code:

```
    package Animals;  
    public class Animals_Inheritance {  
  
        public static void main(String[] args) {  
            Cat cat = new Cat(false, "milk", 4, "black");  
  
            System.out.println("Cat is Vegetarian?" +  
cat.isVegetarian());  
            System.out.println("Cat eats " + cat.getEats());  
        }  
    }  
}
```



```

        System.out.println("Cat has " + cat.getNoOfLegs() + "
legs.");
        System.out.println("Cat color is " + cat.getColor());

System.out.println("
");
        Dogs obj = new Dogs(false, "bisket", 4, "brown");

        System.out.println("dogs is Vegetarian?" +
obj.isVegetarian());
        System.out.println("dogs eats " + obj.getEats());
        System.out.println("dogs has " + obj.getNoOfLegs() + "
legs.");
        System.out.println("dogs color is " + obj.getColor());

    }}}

```

## Output:

```

<terminated> Animals_Inheritance [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (29-Jun-2020, 8:20:27 pm - 8:20:36 pm)
Cat is Vegetarian?false
Cat eats milk
Cat has 4 legs.
Cat color is black

-----
dogs is Vegetarian?false
dogs eats bisket
dogs has 4 legs.
dogs color is brown

```

## Q4. a. What is polymorphism and why it is used, discuss in detail?

Answer:

The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. Polymorphism is the ability of an object to take on many forms.

The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object. Any java object that can pass more than one IS-A test is considered to be polymorphic. In Java, all java objects are polymorphic since any object will pass the IS-A test for their own type and for the class Object. It is important to know that the only possible way to access an object is through a reference variable. A reference variable can be of only one type. Once declared the type of a reference variable cannot be changed.

Q4: b. Write a program using polymorphism in a class on Employee in java?

Answer:

Employee class code:

```
package Polymorphism;

public class Employee {
    private String name;
    private String address;
    private int number;

    public Employee(String name, String address, int number) {
        this.name = name;
        this.address = address;
        this.number = number;
    }

    public void mailCheck() {
        System.out.println("Mailing a check to " + this.name + " "
+ this.address);
    }

    public String toString() {
        return name + " " + address + " " + number;
    }

    public String getName() {
        return name;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String newAddress) {
        address = newAddress;
    }

    public int getNumber() {
        return number;
    }
}
```

Salary Class code:

```
package Polymorphism;
```

```

public class Salary extends Employee {
    private double salary;

    public Salary(String name, String address, int number, double
salary) {
        super(name, address, number);
        setSalary(salary);
    }

    public void mailCheck() {
        System.out.println("Mailing check to " + getName()
+ " with salary " + salary);
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double newSalary) {
        if(newSalary >= 0.0) {
            salary = newSalary;
        }
    }

    public double computePay() {
        System.out.println("Computing salary pay for " +
getName());
        return salary/52;
    }
}

```

Main polymorphism class code:

```

package Polymorphism;

public class Polymorphism {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Salary s = new Salary("Rafi Ullah", "Mianwali, Punjab", 4,
360000);
        Employee e = new Salary("Khan", "Karachi, Sindh", 2,
240000);
    }
}

```

```

        System.out.println("Call mailCheck using Salary reference -
-");
        s.mailCheck();
        System.out.println("\n Call mailCheck using Employee
reference--");
        e.mailCheck();
    }
}

```

Output:

```

<terminated> Polymorphism [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (29-Jun-2020, 9:46:04)
Call mailCheck using Salary reference --
Mailing check to Rafi Ullah with salary 360000.0

Call mailCheck using Employee reference--
Mailing check to Khan with salary 240000.0

```

Q5. a. Why abstraction is used in OOP, discuss in detail?

Answer:

In Object-oriented programming, abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user. In other words, the user will have the information on what the object does instead of how it does it.

- The main benefit of using an abstract class is that it allows you to group several related classes as siblings.
- Abstraction helps to reduce the complexity of the design and implementation process of software.

Q5:b. Write a program on abstraction in java?

Answer:

Animal class code:

```

package Abstraction;
abstract class Animals {
    public abstract void animalSound();
    public void sleep() {
        System.out.println("Zzz");
    }
}

```

Cat class code:

```
package Abstraction;

public class Cat extends Animals {
    public void animalSound() {
        System.out.println("The cat sound: Meow Meow");
    }
}
```

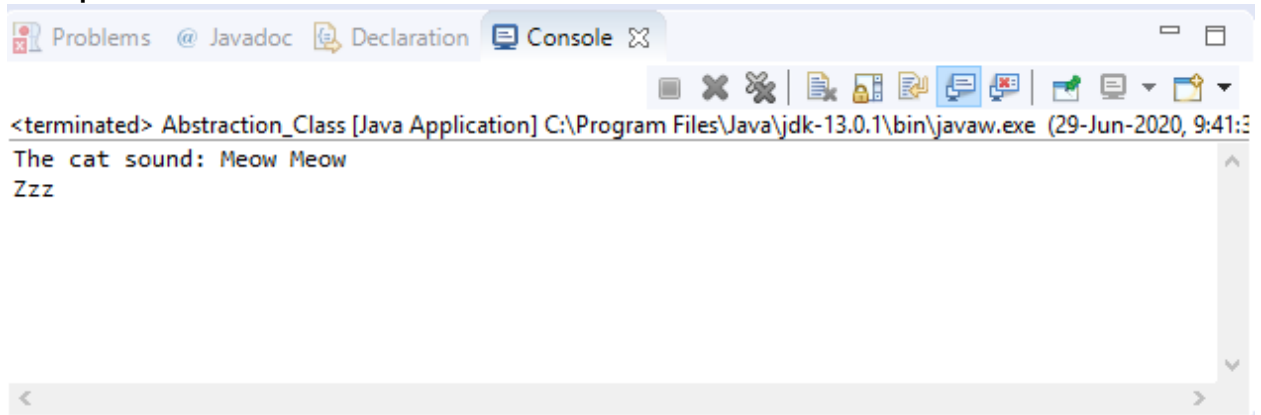
Main Abstract class code:

```
package Abstraction;

public class Abstraction_Class {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Cat obj = new Cat(); // Create a Pig object
        obj.animalSound();
        obj.sleep();
    }
}
```

Output:



```
<terminated> Abstraction_Class [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (29-Jun-2020, 9:41:3)
The cat sound: Meow Meow
Zzz
```