Name : Bahrooz Alam Sir : M Ayub Id:16578

Department : Bs(cs)2nd semester Final Term Exam 2020

Subject : Object Oriented Programming

QNO 1: A . Why access modifiers are used in java, explain in detail Private and Default

 access modifiers?

### Ans : Access Modifiers

Access modifiers deal with the visibility of class members. They control whether other classes can see or change certain variables/methods of our class.

These types of modifiers are closely related to an important part of Object Oriented Programming called **encapsulation**. As a reminder, encapsulation is an idea that links data with the code that manipulates it. By controlling access, you can prevent misuse.

For example, by making sure that certain variables can only be accessed through well-defined methods (the typical get/set combination of methods) we make sure that we won't encounter any unexpected values or deny outside access to certain variables/methods altogether.

As previously mentioned, there are three access modifiers: public, private, and protected. Java also provides default access control (when no modifier is specified), which behaves similarly to protected.

## 1: Private Access Modifier - Private

Methods, variables, and constructors that are declared private can only be accessed within the declared class itself.

Private access modifier is the most restrictive access level. Class and interfaces cannot be private.

Variables that are declared private can be accessed outside the class, if public getter methods are present in the class.

Using the private modifier is the main way that an object encapsulates itself and hides data from the outside world.

### Example

The following class uses private access control –

public class Logger {

 private String format;

 public String getFormat() {

 return this.format;

 }

 public void setFormat(String format) {

 this.format = format;

 }

}

Here, the *format* variable of the Logger class is private, so there's no way for other classes to retrieve or set its value directly.

So, to make this variable available to the outside world, we defined two public methods: *getFormat()*, which returns the value of format, and *setFormat(String)*, which sets its value

**2:** **Default Access Modifier**

When we do not mention any access modifier, it is called default access modifier. The scope of this modifier is limited to the package only. This means that if we have a class with the default access modifier in a package, only those classes that are in this package can access this class. No other class outside this package can access this class. Similarly, if we have a default method or data member in a class, it would not be visible in the class of another package. Lets see an example to understand this:

### Default Access Modifier Example in Java

To understand this example, you must have the knowledge of [packages in java](https://beginnersbook.com/2013/03/packages-in-java/).

In this example we have two classes, Test class is trying to access the default method of Addition class, since class Test belongs to a different package, this program would throw compilation error, because the scope of default modifier is limited to the same package in which it is declared.

package abcpackage;

public class Addition {

 /\* Since we didn't mention any access modifier here, it would

 \* be considered as default.

 \*/

 int addTwoNumbers(int a, int b){

 return a+b;

 }

}

Test.java

package xyzpackage;

import abcpackage.\*;

public class Test {

 public static void main(String args[]){

 Addition obj = new Addition();

 obj.addTwoNumbers(10, 21);

 }

}

Output:

Exception in thread "main" java.lang.Error: Unresolved compilation problem:

The method addTwoNumbers(int, int) from the type Addition is not visible

at xyzpackage.Test.main(Test.java:12)

QNO 1 : B. Write a specific program of the above mentioned access modifiers in java.

Ans: The programming is in picture form



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

## Ans : 1: Public access modifier

The members, methods and classes that are declared public can be accessed from anywhere. This modifier doesn’t put any restriction on the access.

### public access modifier example in java

Lets take the same example that we have seen above but this time the method addTwoNumbers() has public modifier and class Test is able to access this method without even extending the Addition class. This is because public modifier has visibility everywhere.
Addition.java

package abcpackage;

public class Addition {

 public int addTwoNumbers(int a, int b){

 return a+b;

 }

}

## 2: Protected Access Modifier

Protected data member and method are only accessible by the classes of the same package and the subclasses present in any package. You can also say that the protected access modifier is similar to default access modifier with one exception that it has visibility in sub classes.
Classes cannot be declared protected. This access modifier is generally used in a parent child relationship.

### Protected access modifier example in Java

In this example the class Test which is present in another package is able to call the addTwoNumbers() method, which is declared protected. This is because the Test class extends class Addition and the protected modifier allows the access of protected members in subclasses (in any packages).
**Addition.java**

package abcpackage;

public class Addition {

 protected int addTwoNumbers(int a, int b){

 return a+b;

 }

}

QNo 2: Write a specific program of the above mentioned access modifiers in java?

Ans : A program using protected access modifier is as follow

package bahrooz;

import alam.\*;

class Test extends Addition{

 public static void main(String args[]){

 Test obj = new Test();

 System.out.println(obj.addTwoNumbers(11, 22));

 }

}

Output:

33

**A program by using public access modifier**

package bahrooz;

import alam.\*;

class FinalExam{

 public static void main(String args[]){

 Addition obj = new Addition();

 System.out.println(obj.addTwoNumbers(100, 1));

 }

}

Output:

101

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

Ans : Inheritane

The process by which one class acquires the properties(data members) and functionalities(methods) of another class is called inheritance. The aim of inheritance is to provide the reusability of code so that a class has to write only the unique features and rest of the common properties and functionalities can be extended from the another class.
Child Class:
The class that extends the features of another class is known as child class, sub class or derived class.

Parent Class:
The class whose properties and functionalities are used(inherited) by another class is known as parent class, super class or Base class.

Inheritance is a process of defining a new class based on an existing class by extending its common data members and methods.
Inheritance allows us to reuse of code, it improves reusability in your java application.
Note: 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.

If you are finding it difficult to understand what is class and object then refer the guide that I have shared on object oriented programming: [OOPs Concepts](https://beginnersbook.com/2013/04/oops-concepts/)

Lets back to the topic:

**Syntax: Inheritance in Java**

To inherit a class we use extends keyword. Here class XYZ is child class and class ABC is parent class. The class XYZ is inheriting the properties and methods of ABC class.

class XYZ extends ABC

{

}

QNO 3: B. Write a program using Inheritance class on Animal in java.

ANS: The program is as follow

class Teacher {

 String eatBanana = "monkey";

 String forestAnimal = "elephant";

 void does(){

 System.out.println("Animals");

 }

}

public class PetCat extends cat{

 String kingOfForest = "lion";

 public static void main(String args[]){

 PetCat obj = new PetCat();

 System.out.println(obj.forestanimal);

 System.out.println(obj.eatBanana);

 System.out.println(obj.kingOfForest);

 obj.does();

 }

}

Out Put

Animals

elephant

cat

lion

monkey

QNO4. A. What is polymorphism and why it is used, discuss in detail ?

ANS: Definition Of Polymorphism

Polymorphism is one of the [OOPs](https://beginnersbook.com/2013/04/oops-concepts/) feature that allows us to perform a single action in different ways. For example, lets say we have a class Animal that has a method sound(). Since this is a generic class so we can’t give it a implementation like: Roar, Meow, Oink etc. We had to give a generic message.

public class Animal{

 ...

 public void sound(){

 System.out.println("Animal is making a sound");

 }

Uses of Polymorphism in java

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.

The reference variable can be reassigned to other objects provided that it is not declared final. The type of the reference variable would determine the methods that it can invoke on the object.

A reference variable can refer to any object of its declared type or any subtype of its declared type. A reference variable can be declared as a class or interface type

QNO4. B. Write a program using polymorphism in a class on Employee in java

Ans : The program is as follow

class Employ extends AyubSir{

 public void sound(){

 System.out.println("Ayub sir salary is 1 lac..");

 }

 public static void main(String args[]){

 AyubSir obj = new Employ();

 obj.sound();

 }

}

Output:

Ayub sir salary is 1 lac..

QNO 5 . A. Why abstraction is used in OOP, discuss in detail ?

## Ans : Abstraction in OOP

Objects in an OOP language provide an abstraction that hides the internal implementation details. Similar to the coffee machine in your kitchen, you just need to know which methods of the object are available to call and which input parameters are needed to trigger a specific operation. But you don’t need to understand how this method is implemented and which kinds of actions it has to perform to create the expected result.

Let’s implement the coffee machine example in Java. You do the same in any other object-oriented programming language. The syntax might be a little bit different, but the general concept is the same

### Use abstraction to implement a coffee machine

Modern coffee machines have become pretty complex. Depending on your choice of coffee, they decide which of the available coffee beans to use and how to grind them. They also use the right amount of water and heat it to the required temperature to brew a huge cup of filter coffee or a small and strong espresso.

### Implementing the CoffeeMachine abstraction

Using the concept of abstraction, you can hide all these decisions and processing steps within your CoffeeMachine class. If you want to keep it as simple as possible, you just need a constructor method that takes a Map of CoffeeBean objects to create a new CoffeeMachine object and a brewCoffee method that expects your CoffeeSelection and returns a Coffee object.

import org.thoughts.on.java.coffee.CoffeeException;

import java.utils.Map;

public class CoffeeMachine {

 private Map<CoffeeSelection, CoffeeBean> beans;

 public CoffeeMachine(Map<CoffeeSelection, CoffeeBean> beans) {

 this.beans = beans

 }

 public Coffee brewCoffee(CoffeeSelection selection) throws CoffeeException {

 Coffee coffee = new Coffee();

 System.out.println(“Making coffee ...”);

 return coffee;

 }

}

CoffeeSelection is a simple enum providing a set of predefined values for the different kinds of coffees.

public enum CoffeeSelection {

 FILTER\_COFFEE, ESPRESSO, CAPPUCCINO;

}

And the classes CoffeeBean and Coffee are simple POJOs (plain old Java objects) that only store a set of attributes without providing any logic

public class CoffeeBean {

###  private String name;

###  private double quantity;

###

###  public CoffeeBean(String name, double quantity) {

###  this.name = name;

###  this.quantity;

###  }

### }

### public class Coffee {

###  private CoffeeSelection selection;

###  private double quantity;

###

###  public Coffee(CoffeeSelection, double quantity) {

###  this.selection = selection;

###  this. quantity = quantity;

###  }

### }Using the CoffeeMachine abstraction

Using the CoffeeMachine class is almost as easy as making your morning coffee. You just need to prepare a Map of the available CoffeeBeans, instantiate a new CoffeeMachine object, and call the brewCoffee method with your preferred CoffeeSelection.

import org.thoughts.on.java.coffee.CoffeeException;

import java.util.HashMap;

import java.util.Map;

public class CoffeeApp {

 public static void main(String[] args) {

 // create a Map of available coffee beans

 Map<CoffeeSelection, CoffeeBean> beans = new HashMap<CoffeeSelection, CoffeeBean>();

 beans.put(CoffeeSelection.ESPRESSO,

 new CoffeeBean("My favorite espresso bean", 1000));

 beans.put(CoffeeSelection.FILTER\_COFFEE,

 new CoffeeBean("My favorite filter coffee bean", 1000));

 // get a new CoffeeMachine object

 CoffeeMachine machine = new CoffeeMachine(beans);

 // brew a fresh coffee

 try {

 Coffee espresso = machine.brewCoffee(CoffeeSelection.ESPRESSO);

 } catch(CoffeeException e) {

 e.printStackTrace();

 }

 } // end main

} // end CoffeeApp

You can see in this example that the abstraction provided by the CoffeeMachine class hides all the details of the brewing process. That makes it easy to use and allows each developer to focus on a specific class.

If you implement the CoffeeMachine, you don’t need to worry about any external tasks, like providing cups, accepting orders or serving the coffee. Someone else will work on that. Your job is to create a CoffeeMachine that makes good coffee.

And if you implement a client that uses the CoffeeMachine, you don’t need to know anything about its internal processes. Someone else already implemented it so that you can rely on its abstraction to use it within your application or system.

That makes the implementation of a complex application a lot easier. And this concept is not limited to the public methods of your class. Each system, component, class, and method provides a different level of abstraction. You can use that on all levels of your system to implement software that’s highly reusable and easy to understand.

 QNO 5: B. Write a program on abstraction in java.

Ans: The program is as follow

public Interface Sport{

void play();

}

//Now, we will create class named "Badminton"

public class Badminton implements Sport {

@Override

public void play() {

System.out.println("Playing badminton");

}

}

//Next let's create our last class “Football”

public class Football implements Sport {

public void play() {

System.out.println("Playing football");

}

Out put

Playing badminton

Playing football