Department of Computer Science

Final Term Exam Spring 2020

Subject: Object Oriented Programming

BS (CS,SE) Instructor: M.Ayub Khan

There are total **5** questions in this paper. Max Marks: 50

NAME	**=**	**FAYYAZ MUHMMAD**
ID	**=**	**14294**
SUBJECT	**=**	**Object Oriented Programming**
DEPARTMENT	**=**	**BS(CS) 5TH SEMESTER**
TEACHER	**=**	**SIR AYUB KHAN**

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

ANS:- A Java access modifier specifies which classes can access a given class and its fields, constructors and methods. Access modifiers can be specified separately for a class, its constructors, fields and methods. Java access modifiers are also sometimes referred to in daily speech as Java access specifiers, but the correct name is Java access modifiers. Classes, fields, constructors and methods can have one of four different Java access modifiers:

- private
- default (package)
- protected
- Public

PRIVATE ACCESS MODIFIERS:- In the event that a technique or variable is set apart as private (has the private access modifier relegated to it), at that point just code inside a similar class can get to the variable, or call the strategy. Code inside sub classes can't get to the variable or technique, nor can code from any outside class.

Classes can't be set apart with the private access modifier. Denoting a class with the private access modifier would imply that no different class could get to it, which implies that you couldn't generally utilize the class by any stretch of the imagination. In this way the private access modifier isn't took into account classes.

DEFAULT ACCESS MODIFIERS:- The default Java get to modifier is pronounced by not composing any entrance modifier whatsoever. The default get to modifier implies that code inside the class itself just as code inside classes in a similar bundle as this class, can get to the class, field, constructor or strategy which the default get to modifier is appointed to. Hence, the default get to modifier is likewise now and then alluded to as the bundle get to modifier. In the event that you don't have the foggiest idea what a Java bundle is, I have clarified that in my Java bundles instructional exercise.

Sub-classes can't get to techniques and part factors (fields) in the super class, in the event that they these strategies and fields are set apart with the default get to modifier, except if the subclass is situated in a similar bundle as the super class.

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

Q2. **a. Explain in detail Public and Protected access modifiers?**
<u>**ANS:-**</u>

PROTECTED ACCESS MODIFIERS:- The protected access modifier provides the same access as the default access modifier, with the addition that sub-classes can access protected methods and member variables (fields) of the super class. This is true even if the subclass is not located in the same package as the super class.

PUBLIC ACCESS MODIFIERS:- The Java access modifier public means that all code can access the class, field, constructor or method, regardless of where the accessing code is located. The accessing code can be in a different class and different package.

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

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

ANS:- Java inheritance refers to the ability in Java for one class to inherit from another class. In Java this is also called extending a class. One class can extend another class and thereby inherit from that class.

When one class inherits from another class in Java, the two classes take on certain roles. The class that extends (inherits from another class) is the subclass and the class that is being extended (the class being inherited from) is the super-class . In other words, the subclass extends the super-class. Or, the subclass inherits from the super-class.

Another commonly used term for inheritance is specialization and generalization. A subclass is a specialization of a super-class, and a super-class is a generalization of one or more sub-classes.

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

ANS:-
```
class Animal{
   public void eat(){
      System.out.println("I can eat");
   }

   public void sleep(){
      System.out.println("I Can sleep");
   }
}
class Dog extends Animal{
   public void bark(){
      System.out.println("I can bark");
   }
}
class Main{
   public static void main(String[]args){

      Dog dog1=new Dog();
      dog1.eat();
      dog1.sleep();
      dog1.bark();
   }
```

}

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

ANS:-

POLYMORPHISM:- Polymorphism is the capacity of an item to take on numerous structures. The most widely recognized utilization of polymorphism in OOP happens when a parent class reference is utilized to allude to a kid class object.

Any Java object that can pass more than one IS-A test is viewed as polymorphic. In Java, all Java objects are polymorphic since any article will breeze through the IS-An assessment for their own sort and for the class Item.

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

ANS:- class person

```
{
void walk()
{
   System.out.println("Can Run....");
}
}
class Employee extends Person
{
   void walk()
{
   System.out.println("Running Fast...");
}
  public static void main(String arg[]){
{
   Person p=new Employee(); //upcasting
   p.walk();
}
}
```

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

ANS:- Deliberation is choosing information from a bigger pool to show just the important subtleties of the article to the client. Reflection "appears" just the basic properties and "stows away" superfluous data. It assists with lessening programming intricacy and exertion. It is one of the most significant ideas of OOPs.

b. Write a program on abstraction in java.

ANS:-
```
abstract class Person{
   public String fname="john";
   public int age=24;
   public abstract void study();//abstract method
}
//Subclass (inherit from Person)
class Student extends Person{
   public int graduation Year=2018;
   public void study(){// the body of the abstract method is provided here
   System.out.println("Studying all day long");
}}
class Myclass {
   public static void main(String[] args){
      //create an object of the Student class
      (which inherits attributes and methods from Person)
      Student myObj = new student();
      System.out.println("Name:"+ myObj.fname);
      System.out.println("Age:"+ myObj.age);
      System.out.println("Graduation Year:"+ myObj.graduationYear);
      myObj.study();//call abstract method

   }}
```