 *DEPARTMENT OF INFORMATION TECHNOLOGY*

Experiment No.13

|  |  |  |
| --- | --- | --- |
| Experiment No | 13 | |
| Experiment Title | A] Write a program to demonstrate abstract class and abstract methods in java B] Write a program to demonstrate interface in java. | |
| Resources / Apparatus Required | Java SE(JDK)8u102 ,  gedit text editor | PC |
| Objectives | The objective of this experiment is to learn programs based on programs based on abstract class and abstract methods and interface in java.. | |
| Theory | Abstraction: 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. In Java, abstraction is achieved using Abstract classes and interfaces. Abstract class in Java: A class that is declared with abstract keyword, is known as abstract class in java. It can have abstract and non-abstract methods (method with body).It needs to be extended and its method implemented. It cannot be instantiated. Example abstract class:  1. abstract class A{}  Abstract method:  |  | | --- | | A method that is declared as abstract and does not have implementation is known as abstract method. |  Example abstract method:  1. **abstract** **void** printStatus();//no body and abstract  Example of abstract class that has abstract method: In this example, Bike the abstract class that contains only one abstract method run. It implementation is provided by the Honda class.   1. abstract class Bike 2. { 3. abstract void run(); 4. } 5. class Honda4 extends Bike 6. { 7. void run() 8. {System.out.println("running safely.."); 9. }   public static void main(String args[])  {   Bike obj = new Honda4();   obj.run();  }  }  **Output:**  running safely.. Interface in Java: An interface in java is a blueprint of a class. It has static constants and abstract methods only. he interface in java is a mechanism to achieve fully abstraction. There can be only abstract methods in the java interface not method body. It is used to achieve fully abstraction and multiple inheritance in Java. It cannot be instantiated just like abstract class.  **Why use Java interface?**  There are mainly three reasons to use interface. They are given below.   * It is used to achieve fully abstraction. * By interface, we can support the functionality of multiple inheritance. * It can be used to achieve loose coupling.  The java compiler adds public and abstract keywords before the interface method and public, static and final keywords before data members. In other words, Interface fields are public, static and final by default, and methods are public and abstract.  **Abstract Classes Compared to Interfaces:**  Abstract classes are similar to interfaces. You cannot instantiate them, and they may contain a mix of methods declared with or without an implementation. However, with abstract classes, you can declare fields that are not static and final, and define public, protected, and private concrete methods. With interfaces, all fields are automatically public, static, and final, and all methods that you declare or define (as default methods) are public. In addition, you can extend only one class, whether or not it is abstract, whereas you can implement any number of interfaces. Difference between abstract class and interface: Abstract class and interface both are used to achieve abstraction where we can declare the abstract methods. Abstract class and interface both can't be instantiated.  But there are many differences between abstract class and interface that are given below:   |  |  |  | | --- | --- | --- | | **Abstract class** | | **Interface** | | Abstract class can **have abstract and non-abstract** methods. | Interface can have **only**  **abstract** methods. | | | Abstract class **doesn't support multiple inheritance**. | Interface **supports multiple**  **inheritance**. | | | Abstract class **can have final, non-final, static and non-static variables**. | Interface has **only static**  **and final variables**. | | | Abstract class **can have static methods, main method and constructor**. | Interface **can't have**  **static methods, main**  **method or constructor**. | | | Abstract class **can provide the implementation of interface**. | Interface **can't provide**  **the implementation of**  **abstract class**. | | | The **abstract keyword** is used to declare abstract class. | The **interface keyword**  is used to declare interface. | |   Simply, abstract class achieves partial abstraction (0 to 100%) whereas interface achieves fully abstraction (100%). | |
| Program & output | A]  **/\* W.A.P to display to calculate Volume of Sphere & Hemisphere using Abstract Concept \*/**  import java.util.\*;abstract class Base { protected float r,volume; public void getData(float x) { r=x; } public abstract void Calculate();  final public void Display()  { System.out.println(" Volume = "+volume); }}  class Sphere extends Base  { public void Calculate() { volume=3.14f\*r\*r\*r\*4/3; }}  class Hemisphere extends Base  { public void Calculate()  { volume=3.14f\*r\*r\*r\*2/3; }}  class AbstractDemo  { public static void main(String args[])  { float x; Scanner sc=new Scanner(System.in); System.out.println(" Enter The Radius : "); x=sc.nextFloat(); Sphere s=new Sphere();  s.getData(x); s.Calculate();  System.out.println(" Sphere :: "); s.Display();  Hemisphere h=new Hemisphere();  h.getData(x); h.Calculate(); System.out.println(" HemiSphere :: "); h.Display(); }}  C:\Users\Mahesh\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\Untitled.png  B]  **/\* W.A.P to demonstrate use of Interface \*/**  interface Area //Interface Area defined  { float pi=3.14F; //by default :: public and static float compute(float x, float y); //by default :: public,abstract } //end of area interface class Rectangle implements Area  { public float compute (float x, float y)  { return(x\*y); } } //end of class rectangle  class Circle implements Area  { public float compute(float x, float y)  { return(pi\*x\*x); } } //end of class circle class InterfaceArea  { public static void main(String args[])  { Rectangle rect=new Rectangle(); Circle cir=new Circle(); Area area; //Interface object area=rect; //area refers to rect object  System.out.println("Area of rectangle : "+area.compute(10,20));  area=cir; //area now refers to cir object  System.out.println("Area of circle : "+area.compute(5,10)); } //end of main()} //end of main() class  C:\Users\Mahesh\Desktop\New folder (2)\2.png | |
| Conclusion | Thus, we have learnt programs based on abstract classes, abstract methods and interface in java. | |