CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

**CHANDUBHAI S PATEL INSTITUTE OF TECHNOLOGY**

**Name:- Patel Vraj**

**ID:- 21CE105**

**CSPIT – CE**

**GitHub Link:- https://github.com/PatelVraj10/java-practical-file-2/upload**

|  |  |
| --- | --- |
|  | **Practical-3** |
| **Practical 3.1** | Create an abstract class GeometricObject as the superclass for Circle and Rectangle. GeometricObject models common features of geometric objects. Both Circle and Rectangle contain the getArea() and getPerimeter() methods for computing the area and perimeter of a circle and a rectangle. Since you can compute areas and perimeters for all geometric objects, so define the getArea() and getPerimeter() methods in the GeometricObject class. Give implementation in the specific type of geometric object. Create TestGeometricObject class to display area and perimeter of Rectangle and Triangle, compare area of both and display results. Design of all classes are given in the  following UML diagram. |
| **CODE** | // this program is prepared by 21ce105\_patelvraj  // Create an abstract class GeometricObject as the superclass for Circle and Rectangle.  //GeometricObject models common features of geometric objects.  //Both Circle and Rectangle contain the getArea() and getPerimeter() methods for computing the area and perimeter of a circle and a rectangle.  //Since you can compute areas and perimeters for all geometric objects,  //so define the getArea() and getPerimeter() methods in the GeometricObject class.  //Give implementation in the specific type of geometric object. Create TestGeometricObject class to display area and perimeter of Rectangle and Triangle,  //compare area of both and display results.  //Design of all classes are given in the  //following UML diagram.  // GITHUB LINK : https://github.com/PatelVraj10/java-practical-file-2/upload  import java.util.\*;  abstract class Geometricobject  {  abstract void getArea(); abstract void getPerimeter();  }  class circle extends Geometricobject  {  Scanner sc=new Scanner(System.in);  float radius;  void getArea()  {  System.out.println("Enter radius of circle :");  radius=sc.nextInt();  System.out.println("Area ofcircle: "+3.14\*radius\*radius);  }  void getPerimeter()  {  System.out.println("perimeter of circle:"+2\*3.14\*radius);  }  }  class rectangle extends Geometricobject  {  Scanner sc=new Scanner(System.in);  int l,b;  void getArea()  {  System.out.println("Enter length and breadth of rectangle ");  l=sc.nextInt();  b=sc.nextInt();  System.out.println("Area of rectangle: "+l\*b);  }  void getPerimeter()  {  System.out.println("Perimeter of rectangle : "+2\*(l+b));  }  }  class TestGeometricObject  {  void getArea(int a, int b, int c,int d)  {  int s1=a;  int s2=b;  int s3=c;  int h1= d;  System.out.println("Area of triangle is :"+ 0.5\*s2\*h1);  }  void getPerimeter(int p, int q, int r)  {      int a=p;      int b=q;      int c=r;      System.out.println("Perimeter of triangle is :"+a+b+c);      }      } |
| **MAIN PROGRAM** | import java.util.\*;  public class practical\_1 {        public static void main(String[] args)      {          Geometricobject OC=new circle();          Geometricobject OR = new rectangle();          TestGeometricObject OT=new TestGeometricObject();          OC.getArea();          OC.getPerimeter();            OR.getArea();          OR.getPerimeter();      System.out.println("FOR TRIANGLE ");      OT.getArea(4,5,6,7);      OT.getPerimeter(4,5,6);        }      } |

|  |  |
| --- | --- |
| **output** |  |
| **Practical 3.2** | **Write a program to create a default method in an interface IPrinter. Create an interface IPrinter and IScanner. You can assume variables and methods for both interfaces. Create a concrete class to implement both the interfaces.**  **Create 5 objects of the class, store it in Vector and display the result of the vector** |
| **CODE** | // this program is prepared by 21ce105\_patelvraj  //Write a program to create a default method in an interface IPrinter.  //Create an interface IPrinter and IScanner. You can assume variables and methods for both interfaces.  //Create a concrete class to implement both the interfaces.  //Create 5 objects of the class, store it in Vector and display  //the result of the vector  // GITHUB LINK : https://github.com/PatelVraj10/java-practical-file-2/upload  import java.util.Vector;  interface iprinter  {  String ip();  default void show()  {  System.out.println("Default iprinter");  }  }  interface iscanner  {      String isc();      default void show()  {  System.out.println("Default testinterface2");  }  }  class defaultmethod implements iprinter,iscanner  { @Override  public String ip()  {      return "iprinter";  }  @Override  public String isc() {      return "iscanner";  }       public void show()  {     iprinter.super.show();  iscanner.super.show();  }  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **Main program** |  | import java.util.\*;  public class Practical\_2 {      public static void main(String[] args)      {          Vector<String> s = new Vector<>();          defaultmethod d = new defaultmethod();          s.add(d.ip());      s.add(d.isc());      s.add(d.ip());      s.add(d.isc());      s.add(d.ip());        d.show();      for (int i = 0; i < s.size(); i++)      {          System.out.println(s.get(i));      }        }      } |  |

|  |  |
| --- | --- |
| **Output** |  |
| **Practical 3.3** | **WAP that illustrate the interface inheritance. Interface P is extended by P1 and P2 interfaces. 1,2 Interface P12 extends both P1 and P2.**  **Each interface declares one method and one constant. Create one class that implements P12. By using the object of the class invokes each of its method and displays constant.** |
| **CODE** | // this program is prepared by 21ce105\_patelvraj  //WAP that illustrate the interface inheritance. Interface P is extended by P1 and P2 interfaces.  //1,2 Interface P12 extends both P1 and P2.  //Each interface declares one method and one constant.  //Create one class that implements P12. By using the object of the class invokes each of its method and displays constant.  // GITHUB LINK : https://github.com/PatelVraj10/java-practical-file-2/upload  interface P  {      int vP=2;      void methodP();  }  interface P1 extends P  {      int vP1=3;      void methodP1();  }  interface P2 extends P  {      int vP2=4;      void methodP2();  }  interface P12 extends P1,P2  {      int vP12=5;      void methodP12();  }  class InterfaceInheritance implements P12  {  public void methodP()  {  System.out.println("Interface method P called-");  }  public void methodP1()  {  System.out.println("Interface method P1 called-");  }  public void methodP2()  {  System.out.println("Interface method P2 called-");  }  public void methodP12()  {  System.out.println("Interface method called-");  }  } |
| **MAIN PROGRAM** | public class practical\_3  {  public static void main(String[] args)  {  InterfaceInheritance Intf=new InterfaceInheritance();  Intf.methodP();  System.out.println("Interface P constant:"+Intf.vP+"\n");  Intf.methodP1(); System.out.println("Interface P constant:"+Intf.vP1+"\n");  Intf.methodP2(); System.out.println("Interface P constant:"+Intf.vP2+"\n");  Intf.methodP12(); System.out.println("Interface P constant:"+Intf.vP12+"\n");  }  } |
| **OUTPUT** |  |
| **Practical**  **3.4** | **Develop a Program that illustrate method overriding concept** |
| **CODE** | // this program is prepared by 21ce105\_patelvraj  //Develop a Program that illustrate method overriding concept  // GITHUB LINK : https://github.com/PatelVraj10/java-practical-file-2/upload  class Vehicle {      // defining a method      void run() {          System.out.println("Vehicle is running");      }  }  // Creating a child class  class Bike2 extends Vehicle {      // defining the same method as in the parent class      void run() {          System.out.println("Bike is running safely");      }    } |
| **MAIN PROGRAM** | public class part3\_pr\_4 {      public static void main(String[] args) {          Bike2 obj = new Bike2();// creating object          obj.run();// calling method      }  } |
| **OUTPUT** |  |
| **Practical**  **3.5** | Write a java program which shows importing of classes from other user define packages. |
| **CODE** | // this program is prepared by 21ce105\_patelvraj  //Write a java program which shows importing of classes from other user define packages.  // GITHUB LINK : https://github.com/PatelVraj10/java-practical-file-2/upload  package mypackage;  public class part3\_pr\_5 {      public void msg() {          System.out.println("Hello");      }  } |
| **MAIN PROGRAM** | import mypackage.part3\_pr\_5;  public class part3\_pr\_5\_2 {      public static void main(String[] args) {          part3\_pr\_5 a=new part3\_pr\_5();          a.msg();      }  } |
| **Practical**  **3.6** | Write a program that demonstrates use of packages & import statements. |
| **CODE** | // this program is prepared by 21ce105\_patelvraj  //Write a program that demonstrates use of packages & import statements.  // GITHUB LINK : https://github.com/PatelVraj10/java-practical-file-2/upload  package part;  public class part3\_pr\_6 {      public static String getFormattedDollar (double value){          return String.format("$%.2f", value);      }  } |
| **MAIN PROGRAM** | import part.part3\_pr\_6;  public class part3\_pr\_6\_2 {      public static void main(String[] args) {          double value = 99.5;          String formattedValue = part3\_pr\_6.getFormattedDollar(value);          System.out.println("formattedValue = " + formattedValue);      }  } |
| **OUTPUT** |  |

|  |  |
| --- | --- |
| **Practical**  **3.7** | Write a program that illustrates the significance of interface default method. |
| **CODE** | // this program is prepared by 21ce105\_patelvraj  //Write a program that illustrates the significance of interface default method.  // GITHUB LINK : https://github.com/PatelVraj10/java-practical-file-2/upload  interface Sayable{      // Default method      default void say(){          System.out.println("Hello, this is default method");      }      // Abstract method      void sayMore(String msg);  } |
| **MAIN PROGRAM** | public class DefaultMethods implements Sayable{      public void sayMore(String msg){        // implementing abstract method          System.out.println(msg);      }      public static void main(String[] args) {          DefaultMethods dm = new DefaultMethods();          dm.say();   // calling default method          dm.sayMore("Work is worship");  // calling abstract method        }  } |
| **OUTPUT** |  |