Practical - 01

Object:- WAP to create a simple class to find out the area and perimeter of rectangle and box using super and this keyword.

Source Code:-

```
class Rectangle {
   int length;
   int breadth;
   Rectangle(int length, int breadth) {
      this.length = length;
      this.breadth = breadth;
   }
   double getArea() {
      return length * breadth;
   }
   double getPerimeter() {
      return 2 * (length + breadth);
   }
}
class Area extends Rectangle {
   Area() {
      super(10, 20);
   }
   void Barea() {
      System.out.println("Area = " + super.length * super.breadth);
   }
}
class Perimeter extends Rectangle {
   Perimeter() {
      super(10, 20);
   }
   void Bperimeter() {
      System.out.println("Perimeter = " + 2 * (super.length + super.breadth));
   }
}
class Practical {
   public static void main(String[] args) {
      Area a = new Area();
      a.Barea();
      Perimeter p = new Perimeter();
      p.Bperimeter();
   }
}
```

Practical – 02

Object:- WAP to design a classs account using inheritance and static that show all function of bank(withdrawal, deposit).

Source Code:-

```
class Account {
   public double balance;
   public double interest;
}
class SavingAccounts extends Account {
   public SavingAccounts() {
      balance = 0;
      interest = 0;
   }
   public SavingAccounts(double initialBalance, double initialInterest) {
      super(); // Call the parent class (Account) constructor
      balance = initialBalance;
      interest = initialInterest;
   }
   public void deposit(double amount) {
      balance = balance + amount;
   }
   public void withdraw(double amount) {
      balance = balance - amount;
   }
   public void addInterest() {
      balance = balance + balance * interest;
   }
   public double getBalance() {
      return balance;
   }
}
class Practical3 {
   public static void main(String[] args) {
      SavingAccounts al = new SavingAccounts(5000, 1.2);
      al.withdraw(500);
      al.deposit(800);
      al.addInterest();
      System.out.println(al.getBalance());
   }
}
```

Practical - 03

Object:- WAP to design a class using abstract methods and classes.

Source Code:-

```
abstract class Shape
{
   abstract void show();
}
class OOP extends Shape
{
   void show()
   {
      System.out.println("This is a Java Program");
   }
}
class Java extends Shape
{
   void show()
   {
      System.out.println("Java is a Object Oriented Programming language" );
   }
}
class Practical4
{
   public static void main(String[] args)
   {
      Shape s=new Java();
      Shape s1=new OOP();
      s.show();
      s1.show();
   }
}
```

Practical - 04

Object:- WAP to design a string calss that perform string method (eqaul, reverse the string, change case).

Source Code:-

```
class MyString {
   private String str;
   public MyString(String str) {
      this.str = str;
   }
   public boolean equals(String other) {
      return str.equals(other);
   }
   public String reverse() {
      StringBuilder sb = new StringBuilder(str);
      return sb.reverse().toString();
   }
   public String changeCase() {
      StringBuilder sb = new StringBuilder(str.length());
      for (char c : str.toCharArray()) {
         if (Character.isLowerCase(c)) {
            sb.append(Character.toUpperCase(c));
         } else if (Character.isUpperCase(c)) {
            sb.append(Character.toLowerCase(c));
         } else {
            sb.append(c);
         }
      }
      return sb.toString();
   }
   public static void main(String[] args) {
      MyString myString = new MyString("Hello, World!");
      // Equality checking
      System.out.println(myString.equals("Hello, World!")); // true
      System.out.println(myString.equals("HELLO, WORLD!")); // false
      // Reversing the string
      String reversedString = myString.reverse();
      System.out.println(reversedString); // !dlroW ,olleH
      // Changing case
      String changedCaseString = myString.changeCase();
      System.out.println(changedCaseString); // hELLO, wORLD!
   }
}
```

Practical - 05

Object:- Consider we have a Class of Cars under which Santro Xing, Alto and Wagon Rrepresents individual Objects. In this context each Car Object will have its own,Model, Year of Manufacture, Colour, Top Speed, etc. which form Properties of theCar class and the associated actions i.e., object functions like Create(), Sold(),display() form the Methods of Car Class.

Source Code:-

```
class Car {
   String model;
   int yearOfManufacture;
   String color;
   double topSpeed;
   public Car(String model, int yearOfManufacture, String color, double topSpeed) {
      this.model = model;
      this.yearOfManufacture = yearOfManufacture;
      this.color = color;
      this.topSpeed = topSpeed;
   }
   public void create() {
      System.out.println("Car Name: " + model);
   }
   public void sold() {
      System.out.println("Car sold: " + model);
   }
   public void display() {
      System.out.println("Car Details:");
      System.out.println("Model: " + model);
      System.out.println("Year of Manufacture: " + yearOfManufacture);
      System.out.println("Color: " + color);
      System.out.println("Top Speed: " + topSpeed);
   }
}
class Practical5 {
   public static void main(String[] args) {
      Car santroXing = new Car("Santro Xing", 2010, "Red", 140.5);
      Car alto = new Car("Alto", 2015, "Blue", 130.2);
      Car wagonR = new Car("Wagon R", 2018, "Silver", 135.8);

      santroXing.create();
      santroXing.display();
      santroXing.sold();
      System.out.println();
      alto.create();
      alto.display();
      alto.sold();
      System.out.println();
      wagonR.create();
      wagonR.display();
      wagonR.sold();
  }
 }
```

Practical - 06

Object:- In a software company Software Engineers, Sr. Software Engineers, Module Lead, Project Manager are the employee of company but their role, perk, responsibilities differs. Create the Employee base class would provide common behavior.

Source Code:-

```
class Employee
{
void Display()
{
}
}
// software engineer class start
class Soft_Eng extends Employee
{
int Salary=20000;
String Department="IT Department";
String Role="Engineer";
String Company="Oracle";
void Display()
{
System.out.println("Software Engineer");
System.out.println("Salary is "+Salary);
System.out.println("Department is "+Department);
System.out.println("Role is "+Role);
System.out.println("Company is "+Company);
}
}
// senior software engineer class start
class Sr_Soft_Eng extends Employee
{
int Salary=50000;
String Department="IT Department";
String Role="Senior Engineer";
String Company="Oracle";
void Display()
{
System.out.println("Senior Software Engineer");
System.out.println("Salary is "+Salary);
System.out.println("Department is "+Department);
System.out.println("Role is "+Role);
System.out.println("Company is "+Company);
}
}
//module lead class start
class Mod_Lead extends Employee
{
int Salary=25000;
String Department="Technical Department";
String Role="Module Lead";
String Company="Oracle";
void Display()
{
System.out.println("Software Engineer");
System.out.println("Salary is "+Salary);
System.out.println("Department is "+Department);
System.out.println("Role is "+Role);
System.out.println("Company is "+Company);
}
}
// project manager class start
class Pro_Mang extends Employee
{
int Salary=70000;
String Department="Project Analysis Department";
String Role="Project Manager";
String Company="Oracle";
void Display()
{
System.out.println("Software Engineer");
System.out.println("Salary is "+Salary);
System.out.println("Department is "+Department);
System.out.println("Role is "+Role);
System.out.println("Company is "+Company);
}
}
class Practical6
{
public static void main(String[]args)
{
Soft_Eng SE=new Soft_Eng();
SE.Display();
System.out.println("\n");
Sr_Soft_Eng SSE=new Sr_Soft_Eng();
SSE.Display();
System.out.println("\n");
Mod_Lead ML=new Mod_Lead();
ML.Display();
System.out.println("\n");
Pro_Mang PM= new Pro_Mang();
PM.Display();
}
}
```

Practical - 07

Object:- Using the concept of multiple Inheritance create Classes Shape, Circle, Square, Cube, Sphere, Cylinder. Your classes may have only class variable specified in the table below and the method area and/or volume to output their Area and/or volume.

Source Code:-

```
class Shape
{
String name="Shape";
Shape()
{
System.out.println(name+"constructor created");
}
}
class Circle extends Shape
{
public double radius;
public Circle(double r,String n)
{
this.radius=r;
String m=n;
float Pi=22/7;
double a=Pi*radius*radius;
System.out.println(m);
System.out.println("Area is "+a);
System.out.println(" \n ");
}
}
class Square extends Shape
{
double side;
Square(double s,String n)
{
this.side=s;
String m=n;
double a=side*side;
System.out.println(m);
System.out.println("Area is "+a);
System.out.println(" \n ");
}
}
class Cylinder extends Circle
{
double height;
Cylinder(double h,double r,String n)
{
super(7,"Circle");
this.height=h;
this.radius=r;
String m=n;
double v=22/7*radius*radius*height;
System.out.println(m);
System.out.println("Volume is "+v);
System.out.println(" \n ");
}
}
class Sphere extends Circle
{
Sphere(double r,String n)
{
super(7,"Circle");
this.radius=r;
String m=n;
double Pi=3.14;
double v=(3*Pi*radius*radius*radius)/4;
System.out.println(m);
System.out.println("Volume is "+v);
System.out.println(" \n ");
}
}
class Cube extends Square
{
Cube(double s,String n)
{
super(7,"Square");
this.side=s;
String m=n;
double v=side*side*side;
System.out.println(m);
System.out.println("Volume is "+v);
System.out.println(" \n ");
}
}
//main class
class Practical7
{
public static void main(String[]args)
{
Cylinder cy=new Cylinder(7,7,"Cylinder");
Sphere sp=new Sphere(7,"Sphere");
Cube cu=new Cube(7,"Cube");
}
}
```

Practical - 08

Object:- WAP to handle the exception using try multiple catch block.

Source Code:-

```
import java.util.Scanner;
class TryAndMultipleCatch {
  public static void main(String[] args)
  {
    Scanner scanner = new Scanner(System.in);
    try
    {
      System.out.print("Enter a number: ");
      int num1 = scanner.nextInt();
      System.out.print("Enter another number: ");
      int num2 = scanner.nextInt();
      int result = divisionNumbers(num1, num2);
      System.out.println("Result: " + result);
    }
    catch (ArithmeticException e)
    {
      System.out.println("Error: Division by zero is not allowed.");
    }
    catch (Exception e)
    {
      System.out.println("Invalid input. Please enter a valid integer.");
      scanner.close();
    }
  }
  public static int divisionNumbers(int a, int b)
  {
    return a / b;
  }
}
```

Practical - 09

Object:-WAP that implement the Nested try statement.

Souce Code:-

```
import java.util.Scanner;
class NestedTry
{
   public static void main(String args[])
   {
      Scanner sc=new Scanner(System.in);
      {
      try
      {
         int a[] = { 30, 45, 60, 75, 90, 105, 120, 140, 160};
         // displaying element at index 8
         System.out.println("Element array index number:");
         int index= sc.nextInt();
         // another try block
         try
         {
            System.out.println("Division");
            int result = 100 / 0;
         }
         catch (ArithmeticException ex2)
         {
            System.out.println("Sorry! Division by zero in invalid");
         }
      }
         catch (ArrayIndexOutOfBoundsException ex1)
         {
            System.out.println("ArrayIndexOutOfBoundsException");
         }
   }
}
}
```

Practical - 10

Object:- WAP to create a thread that implement the Runnable interface.

Source Code:-

```
class NewThread implements Runnable
{
   public void run()
   {
      System.out.println("Concurrent Thread Start Running");
   }
}
class NewThreadExample
{
   public static void main(String[] args)
   {
      NewThread rn=new NewThread();
      Thread n=new Thread(rn);
      n.start();
   }
}
```