QUESTION 1 : Main Program:In the main program, create instances of Vehicle, Car, and SportsCar to represent different types of vehicles.Demonstrate the use of methods and properties of these classes by starting the engine, honking the horn, and activating the turbo (for sports cars) .Display relevant information about each vehicle, such as its name, manufacturer, number of doors (for cars), and top speed (for sports cars).

CODE :

```
class Vehical{
   public String name ;
   public String manufacturer;
   public Vehical(String name ,String  manufacturer)
   {
      this.name = name ;
      this.manufacturer  = manufacturer;
   }
   public void Display()
   {
      System.out.println("The Name of the vehicle is:" + name );
      System.out.println("The manufacturer of this vehicle is:" + manufacturer);
   }
   public void startEngine()
   {
      System.out.println("The car Engine has been Started wHuorm whuourm whourm... ");
   }
}
class Car extends Vehical{
   public int numberofdoors;
```

```
    public Car(String name ,String manufacturer ,  int numberofdoors)
    {
       super(name , manufacturer);
       this.numberofdoors  = numberofdoors;
    }
    public void Display()
    {
        System.out.println("The Name of the car is:" +  name );
        System.out.println("The manufacturer of the Car is : " + manufacturer);
          System.out.println("The number of doors of this car are :" + numberofdoors);
    }
    public void honkHorn()
    {
       System.out.println("Honking : Peeep Peep peeeeeeeeeeeee.");
    }

}

class SportsCar extends Car{
   public int speed;
   public SportsCar(String name ,String manufacturer ,  int numberofdoors,int speed)
   {
    super(name , manufacturer , numberofdoors);
    this.speed = speed;
   }
```

```
public void startEngine()
{
   System.out.println("The Sports car engine has started");
}
   public void honkHorn(){
       System.out.println("side please ");
   }

  public void topSpeed()
  {
   System.out.println("The Top spped of the  car is :" + speed);
  }
  public void activateTurbo()
  {
   System.out.println("Activating the Turbomode");
  }
}

class Question1{
   public static void main(String args[])
   {
       System.out.println("Bharat Prajapati \n Roll no : 3058" );
       Vehical v1 = new Vehical("Car" ,"Maruti Suzuki");
      v1.Display();
      v1.startEngine();
```

```
System.out.println();
        Car funti = new Car("Alto800" ,"Maruti Suzuki", 4);
        funti.Display();
        funti.honkHorn();
System.out.println();
        SportsCar supra = new SportsCar("Supra" , "Toyota" , 2 , 400);
        supra.startEngine();
         supra.honkHorn();
        supra.topSpeed();
        supra.activateTurbo();



    }
}
```

OUTPUT:

```
Bharat Prajapati
 Roll no : 3058
The Name of the vehicle is:Car
The manufacturer of this vehicle is:Maruti Suzuki
The car Engine has been Started wHuorm whuourm whourm...

The Name of the car is:Alto800
The manufacturer of the Car is : Maruti Suzuki
```

The number of doors of this car are :4

Honking : Peeep Peep peeeeeeeeeeeee.

The Sports car engine has started

side please

The Top spped of the car is :400

Activating the Turbomode

QUESTION 2 : this program demonstrates the Factory design pattern by creating different types of Thalis (Gujarati and Punjabi) with customizable components and prices. It encapsulates the creation logic within the ThaliRestaurant class, providing a consistent interface for creating Thalis while allowing for flexibility and customization of each Thali type.

CODE:

```
abstract class Thali{


    private double price = 0.0 ;

    abstract void addSabji(double price);

    abstract void addDal(double price);

    abstract void addRice(double price);

     abstract void addRoti(double price);
```

```
    void makeThali()
    {
        System.out.print("Veg Thali will be ready in 30 minutes");
        System.out.println("We appericiate your patience");
    }
    public double getPrice()
    {
        return price ;
    }
    protected void  setPrice(double price)
    {
            this.price += price;
    }
}
class GujaratiThali extends Thali {
    // private Thali t1;
        void addSabji(double price)
        {
            System.out.println("The price for Gujarati thali's the Sabji is :" + price);
            setPrice(price);
        }
            void addDal(double price)
```

```
    {
        System.out.println("The price for Gujarati thali's  the Dal is :" + price);
        setPrice(price);
    }
        void addRice(double price)
    {
        System.out.println("The price for Gujarati thali's  the Rice is :" + price);
        setPrice(price);
    }
        void addRoti(double price)
    {
        System.out.println("The price for Gujarati thali's the Roti is :" + price);
        setPrice(price);
    }

  //    super.getPrice();



}
class PunjabiThali extends Thali {
    void addSabji(double price)
```

```
    {
        System.out.println("The price for Punjabi thali's the Sabji is :" + price);
            setPrice(price);
    }
        void addDal(double price)
    {
        System.out.println("The price for Punjabi thali's  the Dal is :" + price);
            setPrice(price);
    }
        void addRice(double price)
    {
        System.out.println("The price for Punjabi thali's  the Rice is :" + price);
            setPrice(price);
    }
        void addRoti(double price)
    {
        System.out.println("The price for Punjabi thali's the Roti is :" + price);
            setPrice(price);
    }
}

abstract class BaseThali{
```

```
   GujaratiThali ganesh  = new GujaratiThali();
   PunjabiThali harman = new PunjabiThali();
   abstract void createGujaratiThali();
   abstract void createPunjabiThali();

}

class ThaliRestaurant extends BaseThali{
      public void createGujaratiThali()
      {
         ganesh.addDal(50);
         ganesh.addRice(40);
         ganesh.addRoti(40);
         ganesh.addSabji(60);
      }
      public void totalOfGujarati()
      {
         System.out.println("The total Price of the Gujarati Thadi is:" +
ganesh.getPrice());
      }
      public void createPunjabiThali()
      {
```

```
        harman.addDal(70);
        harman.addRice(60);
        harman.addRoti(40);
        harman.addSabji(30);

    }
    public void totalOfPunjabi()
    {
        System.out.println("The total Price for the Punjabi Thali is :" +
harman.getPrice());
    }
 }

public class Question2 {
  public static void main(String args[])
  {
    System.out.println("\t Name : BHARAT PRAJAPATI \n Roll no : 3058");

            ThaliRestaurant baseThali = new ThaliRestaurant();
            baseThali.createGujaratiThali();
            baseThali.ganesh.makeThali();
            double gtprice =  baseThali.ganesh.getPrice();
```

```
        System.out.println("The total price for your gujarati thali is : "+ gtprice + "only.");
        System.out.println();
            baseThali.createPunjabiThali();
            baseThali.harman.makeThali();
            double ptprice = baseThali.harman.getPrice();
            System.out.println("The total price for your punjabi thali is :" + ptprice + "only.");



  }
}
```

OUTPUT:

```
      Name : BHARAT PRAJAPATI
 Roll no : 3058
The price for Gujarati thali's  the Dal is :50.0
The price for Gujarati thali's  the Rice is :40.0
The price for Gujarati thali's the Roti is :40.0
The price for Gujarati thali's the Sabji is :60.0
Veg Thali will be ready in 30 minutesWe appericiate your patience
```

The total price for your gujarati thali is : 190.0only.

The price for Punjabi thali's the Dal is :70.0

The price for Punjabi thali's the Rice is :60.0

The price for Punjabi thali's the Roti is :40.0

The price for Punjabi thali's the Sabji is :30.0

Veg Thali will be ready in 30 minutesWe appericiate your patience

The total price for your punjabi thali is :200.0only.

QUESTION 3: This program demonstrates how the PizzaOrderSystem interface defines a contract for pizza ordering operations, and the PizzaOrderProcessor class provides the concrete implementation of those operations. The main method showcases the usage of these methods within a simplified pizza ordering system.

CODE:

```
import java.util.*;


interface  PizzaOrderSystem{
   abstract void placeOrder(String pizzaType , int quantity);
   abstract String checkOrderStatus(int orderId);
   abstract boolean cancelOrder(int orderId);
```

```
    abstract double calculateOrderCos(int orderId);
    void listAvailablePizza();
}

class pizzaOrderProcessor implements PizzaOrderSystem{
    private List<Order> orders = new ArrayList<>();
    private int nextOrderId = 1;
    public void placeOrder(String pizzaType , int quantity)
    {
        int orderId = nextOrderId++;
        Order order = new Order(orderId , pizzaType , quantity);
        orders.add(order);
        System.out.println("Order with ID" + orderId  + " and type " + pizzaType +
"with Quantity " + quantity + "places successfully");
    }

    public String checkOrderStatus(int orderId)
    {
            Order  order = findOrderById(orderId);
            if(order!= null)
            {
                return "Your order with orderID"  + orderId +  "is in progress";
```

```
        }
        return  " Order not found ";
    }
    public boolean cancelOrder(int orderID)
    {
        Order order = findOrderById(orderID);
        if(order != null)
        {   orders.remove(order);
            System.out.println("Your order with orderId" +orderID + "hase been
cancelled");
            return true;


        }
        else{
            System.out.println("Order not found , could not be canceled");
            return false;
        }
    }
    public double calculateOrderCos(int orderID)
    {
        Order order = findOrderById(orderID);
        if(order!= null)
```

```
    {
        return 100 * order.getQuantity();
    }
    return 0.0;

}
public void listAvailablePizza()
{
          System.out.println("Available Pizza Options:");
    System.out.println("1. Margherita Pizza");
    System.out.println("2. Pepperoni Pizza");
    System.out.println("3. Veggie Pizza");

}

private Order findOrderById(int orderId)
{
    for(Order order : orders)
    {
        if(order.getOrderId() == orderId)
        {
            return order;
```

```
        }
      }
      return null;

   }

}

class Order{
   private int orderId;
   private String pizzaType;
   private int quantity;
   public Order(int orderId , String pizzaType , int quantity)
   {
      this.orderId = orderId;
      this.pizzaType = pizzaType;
      this.quantity = quantity;
   }
   public int getOrderId()
   {
      return orderId;
   }
```

```
    public String getpizzaType()
    {
        return pizzaType;
    }
    public int getQuantity()
    {
        return quantity;
    }
}



public class Question3{
    public static void main(String args[])
    {
      System.out.println("Bharat Prajapati \n Roll no : 3058" );
PizzaOrderSystem pizza = new pizzaOrderProcessor();
        pizza.listAvailablePizza();

        Scanner scanner = new Scanner(System.in);
        int numberOfOrders;

        do {
```

```
System.out.print("Enter the number of orders (0 to exit): ");
numberOfOrders = scanner.nextInt();

for (int i = 1; i <= numberOfOrders; i++) {
    System.out.print("Enter pizza type for order " + i + ": ");
    String pizzaType = scanner.next();
    System.out.print("Enter quantity for order " + i + ": ");
    int quantity = scanner.nextInt();

    pizza.placeOrder(pizzaType, quantity);
}

for (int i = 1; i <= numberOfOrders; i++) {
    String orderStatus = pizza.checkOrderStatus(i);
    System.out.println(orderStatus);

    double orderCost = pizza.calculateOrderCos(i);
    System.out.println("Order " + i + " Cost: " + orderCost);

    boolean isOrderCanceled = pizza.cancelOrder(i);
    if (isOrderCanceled) {
        System.out.println("Order " + i + " canceled successfully.");
```

```
            } else {
                System.out.println("Order " + i + " could not be canceled.");
            }
        }
    } while (numberOfOrders > 0);

    scanner.close();
  }
}
```

OUTPUT :

Bharat Prajapati

Roll no : 3058

Available Pizza Options:

1. Margherita Pizza

2. Pepperoni Pizza

3. Veggie Pizza

Enter the number of orders (0 to exit): 2

Enter pizza type for order 1: Margherita

Enter quantity for order 1: 1

Order with ID1 and type Margheritawith Quantity 1places successfully

Enter pizza type for order 2:

Veggie

Enter quantity for order 2: 1

Order with ID2 and type Veggiewith Quantity 1places successfully

Your order with orderID1is in progress

Order 1 Cost: 100.0

Your order with orderId1hase been cancelled

Order 1 canceled successfully.

Your order with orderID2is in progress

Order 2 Cost: 100.0

Your order with orderId2hase been cancelled

Order 2 canceled successfully.

QUESTION 4: Overall, this program demonstrates the principles of inheritance, encapsulation, and method overriding in a school system context, providing a clear separation of concerns and a structured object-oriented design.

CODE:

```
class Person{
```

```
private String name ;
private int age;
public Person(String name , int age)
{
   this.name = name;
   this.age = age;
}
public void setName(String name)
{
   this.name = name;
}
public void setAge(int age)
{
   this.age = age;
}
public String getName()
{
   return name;
}
public int getAge()
{
   if(age <0)
```

```
        {
            System.out.println("Your age must be positive");
        }
        return age;

    }
    public void introduce()
    {
        System.out.println("The Name of the Person is: " + name);
        System.out.println("The Age of the Person is:" + age);
    }


}

class Student extends Person{
    private int studentId;
    public Student(String name , int age)
    {
        super(name, age);
    }
    public void setId(int studentId)
```

```
    {
        this.studentId = studentId;
    }
    public int getId()
    {
        return studentId;
    }
    public void introduce()
    {
        // super.introduce();
        System.out.println("The Name of the student is :" + super.getName());
        System.out.println("The age of the student is :" + super.getAge());
        System.out.println("Id of the Student is :" + studentId);
    }
    public void study()
    {
        System.out.println("The student is currently studying in our collage");
    }
}

class Teacher  extends Person{
    private String Subject;
```

```
public Teacher(String name , int age , String Subject)
{
   super(name, age);
   this.Subject = Subject;

}
public void setSubject(String subject)
{
       this.Subject = subject;
}
public String getSubject()
{
   return Subject;
}
 public void introduce()
{
   // super.introduce();
   System.out.println("The Name of the Teacher is :" + super.getName());
   System.out.println("The age of the Teacher is :" + super.getAge());
   System.out.println("Subject of the Teacher is :" + this.Subject);
}
```

```
    public void teach()
    {
        System.out.println("The teacher had been teaching in our collage for last 20
years");
    }
}
public class Question4 {
    public static void main(String[] args) {
        System.out.println("Bharat Prajapati \n Roll no : 3058" );
        Student s1 = new Student("Bharat", 20);
   System.out.println("*******   *******");
        s1.setId(3058);
        s1.introduce();
        s1.study();
System.out.println("*******   *******");
        Teacher T1 = new Teacher("kanu", 54, "Statics");
        T1.introduce();
        T1.teach();
    }
}
```

OUTPUT:

Bharat Prajapati

Roll no : 3058

******* *******

The Name of the student is :Bharat

The age of the student is :20

Id of the Student is :3058

The student is currently studying in our collage

******* *******

The Name of the Teacher is :kanu

The age of the Teacher is :54

Subject of the Teacher is :Statics

The teacher had been teaching in our collage for last 20 years

QUESTION 5: This Java program illustrates concepts of inheritance, where the Professor class inherits properties and methods from the User class, and composition, where the Department class can contain instances of other classes (a professor and a course) to model relationships between different entities in a university department system. It provides a basic structure for managing and displaying information related to professors, courses, and departments within a university context.

CODE :

```
class User{
    private String userName;
    private String email;
       public User(String userName , String email)
       {
        this.userName = userName;
        this.email = email;
       }
       public String getuserName()
       {
        return userName;
       }
       public String getEmail()
       {
        return email;
       }
}

class Professor extends User{
   private String Department;
   public Professor(String userName , String email , String Department)
   {
```

```
        super(userName, email);
        this.Department = Department;

    }
    public String getDepartment()
    {
     return Department;

    }
}
class Course{
     private int code;
     private String name;
     private int CreditHours;
    public Course(int code , String name , int CreditHours)
    {
     this.code = code;
     this.name = name;
     this.CreditHours = CreditHours;

    }
     public int getcode()
```

```
    {
        return code;
    }
    public String  getName()
    {
        return name;
    }
    public int getCreditHours()
    {
        return CreditHours;
    }

}
class Department {
    private String department_name;
    private Professor professor1;
    private Course course1;
    public Department(String department_name  )
    {
        this.department_name =department_name ;
    }
    public void setprofessor1(Professor professor)
```

```
{
   this.professor1 = professor;
}
public void  setcourse1(Course course)
{
   this.course1 = course;
}
public String getDepartment()
{
   return department_name;
}
public String getProfessorUsernmae()
{
   return professor1.getuserName();
}
public String getProfessorEmaill()
{
   return professor1.getEmail();
}
public int getCourseCode()
{
   return course1.getcode();
```

```
    }
    public String getCourseName()
    {
      return course1.getName();
    }
    public int getCourseHours()
    {
      return course1.getCreditHours();
    }



}



public class Question5 {
   public static void main(String[] args) {
      System.out.println("Bharat Prajapati \n Roll no : 3058" );
      Professor p1 = new Professor("ksfac120", "ks123@gmail.com",
"Computer_science");
      Course c1 = new Course(105, "MSC.IT", 5);
      Department d1 = new Department("Computer_science");
      d1.setprofessor1(p1);
```

```
        d1.setcourse1(c1);


            System.out.println("Department Name is : " + d1.getDepartment());

            System.out.println("Professor's user  name is:
"+d1.getProfessorUsernmae());

            System.out.println( "Professor's email is: "+  d1.getProfessorEmaill());

            System.out.println("Course code is:  "+ d1.getCourseCode());

            System.out.println("Course Name is: "+ d1.getCourseName());

            System.out.println("Course credit hours are: "+ d1.getCourseHours());

    }

}
```

OUTPUT:

```
Bharat Prajapati

 Roll no : 3058

Department Name is : Computer_science

Professor's user  name is: ksfac120

Professor's email is: ks123@gmail.com

Course code is:  105

Course Name is: MSC.IT
```

Course credit hours are: 5

QUESTION 6: PATTERN

CODE :

```
public class Question6 {
   public static void main(String args[])
   {
      System.out.println("Bharat prajapati \n");
      System.out.println("roll no : 3058");
      int n = 7;
      int originaln = n;
      n = 2*n;
      for(int i = 0;i<=n;i++)
      {
         for(int k=0;k<=n;k++)
         {
            int Index = originaln - Math.min(Math.min(i, k) , Math.min(n-i , n-k));
            System.out.print(Index + " ");
         }
         System.out.println();
```

```
        }


    }
}
```

OUTPUT:

Bharat prajapati

roll no : 3058

```
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
7 6 6 6 6 6 6 6 6 6 6 6 6 6 7
7 6 5 5 5 5 5 5 5 5 5 5 5 6 7
7 6 5 4 4 4 4 4 4 4 4 4 5 6 7
7 6 5 4 3 3 3 3 3 3 3 4 5 6 7
7 6 5 4 3 2 2 2 2 2 3 4 5 6 7
7 6 5 4 3 2 1 1 1 2 3 4 5 6 7
7 6 5 4 3 2 1 0 1 2 3 4 5 6 7
7 6 5 4 3 2 1 1 1 2 3 4 5 6 7
7 6 5 4 3 2 2 2 2 2 3 4 5 6 7
7 6 5 4 3 3 3 3 3 3 3 4 5 6 7
7 6 5 4 4 4 4 4 4 4 4 4 5 6 7
7 6 5 5 5 5 5 5 5 5 5 5 5 6 7
7 6 6 6 6 6 6 6 6 6 6 6 6 6 7
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
```

QUESTION 7: Find GCD of two numbers using while loop and if else statement

CODE:

```
import java.util.Scanner;

public class Question7 {
   public static void main(String args[])
   {
      System.out.println("Bharat Prajapati \n Roll no : 3058" );
      Scanner sc = new Scanner(System.in);
System.out.println("Enter the first number");
      int a = sc.nextInt();
      System.out.println("Enter the second number");
      int b = sc.nextInt();
      a = Math.abs(a);
      b=Math.abs(b);

      int gcd = findGcd(a , b);
      System.out.println("The GCD of " + a + " and " + b + " is " + gcd);
      sc.close();
   }
   public static int findGcd(int a, int b)
   {
      while(b != 0)
      {
         int temp  =b;
         b = a%b;
         a= temp;

      }
      return a;

   }
}
```

OUTPUT :

Bharat Prajapati

Roll no : 3058
Enter the first number
65
Enter the second number
50
The GCD of 65 and 50 is 5

QUESTION 8 : Java Program to Add Two Matrix Using Multi-dimensional Arrays

CODE :

```
public class Question8 {
   public static void main(String args[])
   {
      System.out.println("Bharat Prajapati \n Roll no : 3058" );
      int[][] arr  =
      {
         {1,2,3},
         {1,2,3},
         {1,2,3}
      };
      int[][] arr2 = {
         {1,2,3},
         {1,2,3},
         {1,2,3}
      };
      int[][] finalans = new int[arr.length][arr2.length];
      for(int i =0;i < arr.length;i++)
      {
         for(int j= 0;j<arr2.length;j++)
         {
            finalans[i][j] = arr[i][j] + arr2[i][j];

         }
      }
      for(int i=0 ; i< arr.length;i++)
```

```
      {
         for(int j=0 ; j< arr2.length;j++)
         {
            System.out.print(finalans[i][j] + " ");
         }
         System.out.println();
      }

   }
}
```

OUTPUT :

```
Bharat Prajapati
 Roll no : 3058
2 4 6
2 4 6
2 4 6
```

QUESTION 9: program to check prime number using recursion

CODE:

```
import java.util.Scanner;
public class Question9 {
   public static void main(String[] args)
   {
      System.out.println("Bharat Prajapati \n Roll no : 3058" );
      int n, x;
      Scanner s = new Scanner(System.in);
      System.out.print("Enter any number:");
      n = s.nextInt();

      x = prime(n, 2);
      if(x == 1)
      {
         System.out.println(n+" is prime number");
```

```
        }
        else
        {
            System.out.println(n+" is not prime number");
        }
    }
    public static int prime(int y,int i)
    {
        if(i < y)
        {
            if(y % i != 0)
            {
                return(prime(y, ++i));
            }
            else
            {
                return 0;
            }
        }
        return 1;
    }
}
```

OUTPUT :

```
Bharat Prajapati
 Roll no : 3058
Enter any number:13
13 is prime number
```

QUESTION 10 : STRING BUFFER PRACTICE.

CODE:

```
public class Question10 {
    public static void main(String args[])
    {
```

```
        System.out.println("Bharat Prajapati \n Roll no : 3058" );
        StringBuffer string = new StringBuffer("Hello");
        string.append("World");
        System.out.println("After appending the text :" + string);
        string.insert(5 , "java");
        System.out.println("After Inserting the text : " + string);
        string.delete(5, 7);
        System.out.println("After deleting some text: " + string);
        string.replace(0, 4,"Bye Bye: ");
        System.out.println("After the replacing the string: " + string);
        string.reverse();
        System.out.println("After reversing the text : " + string);
        String res = string.toString();
        System.out.println("after converting the string :  " + res);
    }
}
```

OUTPUT :

```
Bharat Prajapati
 Roll no : 3058
After appending the text :HelloWorld
After Inserting the text : HellojavaWorld
After deleting some text: HellovaWorld
After the replacing the string: Bye Bye: ovaWorld
After reversing the text : dlroWavo :eyB eyB
after converting the string :  dlroWavo :eyB eyB
```