Experiment 01:

```java
/*
1.Classes and objects: Design a class ‘Complex ‘with data members for real and
imaginary parts. Provide default and Parameterized constructors. Write a program to
perform arithmetic operations of two complex numbers.
*/

import java.text.DecimalFormat;
import java.util.*;

class Complex {
   double real;
   double imaginary;

   Complex(){
       this(0,0);
   }

   Complex(double real, double imaginary){
       this.real = real;
       this.imaginary = imaginary;
   }

   static Complex addition(Complex num1, Complex num2){
       return new Complex(num1.real + num2.real, num1.imaginary + num2.imaginary);
   }

   static Complex subtractions(Complex num1, Complex num2){
       return new Complex(num1.real - num2.real, num1.imaginary - num2.imaginary);
   }

   static Complex multiplication(Complex num1, Complex num2){
       double realPart = (num1.real * num2.real) - (num1.imaginary *
num2.imaginary);
       double imaginaryPart = (num1.real * num2.imaginary) + (num1.imaginary *
num2.real);

       return new Complex(realPart, imaginaryPart);
   }

   static Complex division(Complex num1, Complex num2) {
       double realNum = ((num1.real*num2.real) + (num1.imaginary*num2.imaginary));
       double realDen = (num2.real*num2.real + num2.imaginary*num2.imaginary);
       double realPart = realNum / realDen;
```

```
        double imagNum = ((num1.imaginary*num2.real)- (num1.real * num2.imaginary));
        double imagDen = (num2.real*num2.real + num2.imaginary*num2.imaginary);
        double imagPart = imagNum / imagDen;

        return new Complex(realPart, imagPart);
    }

    static Complex takeInput(String num) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter " + num + " Real part: ");
        double real = sc.nextDouble();
        System.out.print("Enter " + num + " Imaginary part: ");
        double imag = sc.nextDouble();
        return new Complex(real, imag);
    }

    static void display(Complex num){
        DecimalFormat numberFormat = new DecimalFormat("#.##");
        System.out.print("(" + numberFormat.format(num.real) + " + " +
numberFormat.format(num.imaginary) + "i" + ")");
    }

    static void displayResult(Complex num1, String opra, Complex num2, Complex
result){
        System.out.println("\n");
        System.out.println("The Answer is: ");
        Complex.display(num1);
        System.out.print(" " + opra + " ");
        Complex.display(num2);
        System.out.print(" = ");
        Complex.display(result);
        System.out.println('\n');
    }
}

public class Assignment_01 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Complex number1 = Complex.takeInput("Num1");
        System.out.print("Enter A: Addition, S: Subtraction, M: Multiplication, D:
Division, E: Exit: ");
        String input = sc.nextLine().trim().toUpperCase();
        while(!"E".equals(input)){
            switch (input){
                case "A" -> {
                    Complex number2 = Complex.takeInput("Num2");
                    Complex addition = Complex.addition(number1, number2);
                    Complex.displayResult(number1, "+", number2, addition);
                }
                case "S" -> {
                    Complex number2 = Complex.takeInput("Num2");
                    Complex subtraction = Complex.subtractions(number1, number2);
                    Complex.displayResult(number1, "-", number2, subtraction);
                }
                case "M" -> {
                    Complex number2 = Complex.takeInput("Num2");
                    Complex multiplication = Complex.multiplication(number1,
number2);
                    Complex.displayResult(number1, "*", number2, multiplication);
                }
                case "D" -> {
                    Complex number2 = Complex.takeInput("Num2");
                    Complex division = Complex.division(number1, number2);
                    Complex.displayResult(number1, "/", number2, division);
                }
                default -> System.out.println("Incorrect operator!");
            }
            System.out.println("Would you like to try again(Y/N): ");
            String input2 = sc.nextLine().trim().toUpperCase();
            if("N".equals(input2)){
                break;
            }else{
                number1 = Complex.takeInput("Num1");
                System.out.print("Enter A: Addition, S: Subtraction, M:
Multiplication, D: Division, E: Exit: ");
                input = sc.nextLine().trim().toUpperCase();
            }
        }
        System.out.println("See you later!");
        sc.close();
    }
}
```

Output:

```
Enter Num1 Real part: 12
Enter Num1 Imaginary part: 7
Enter A: Addition, S: Subtraction, M: Multiplication, D: Division, E:
Exit: S
Enter Num2 Real part: 22
Enter Num2 Imaginary part: 13


The Answer is:
(12 + 7i) - (22 + 13i) = (-10 + -6i)

Would you like to try again(Y/N):
Y
Enter Num1 Real part: 33
Enter Num1 Imaginary part: 22
Enter A: Addition, S: Subtraction, M: Multiplication, D: Division, E:
Exit: M
Enter Num2 Real part: 10
Enter Num2 Imaginary part: 7


The Answer is:
(33 + 22i) * (10 + 7i) = (176 + 451i)

Would you like to try again(Y/N):
N
See you later!
```

Experiment 02:

```
/*
2. Polymorphism. Identify commonalities and differences between
Publication, Book and Magazine classes. Title, Price, Copies are common
instance variables and saleCopy is a common method. The differences are,
Bookclass has author and orderCopies(). Magazine Class has methods
orderQty, Current issue, receiveissue().Write a program to find how many
copies of the given books are ordered and display total sale of publication
*/

class Publication {
    protected String title;
    protected double price;
    protected int copies;

    public Publication(String title, double price, int copies) {
        this.title = title;
        this.price = price;
        this.copies = copies;
    }

    public void saleCopy(int numCopies) {
        System.out.println(numCopies + " copies of \"" + title + "\" sold
at $" + price + " each.");
        System.out.println("Total sale: $" + (numCopies * price));
    }
}

class Book extends Publication {
    private String author;

    public Book(String title, double price, int copies, String author) {
        super(title, price, copies);
        this.author = author;
    }

    public void orderCopies(int numCopies) {
        copies += numCopies;
        System.out.println("Ordered " + numCopies + " more copies of the
book \"" + title + "\" by " + author + ".");
```

```java
    }

    @Override
    public void saleCopy(int numCopies) {
        super.saleCopy(numCopies);
    }
}

class Magazine extends Publication {
    private int orderQty;
    private String currentIssue;

    public Magazine(String title, double price, int copies, int orderQty,
String currentIssue) {
        super(title, price, copies);
        this.orderQty = orderQty;
        this.currentIssue = currentIssue;
    }

    public void receiveIssue(String newIssue) {
        currentIssue = newIssue;
        System.out.println("New issue of \"" + title + "\" received: " +
currentIssue);
    }

    public void orderQty(int qty) {
        orderQty = qty;
        System.out.println("Ordered " + orderQty + " copies of the magazine
\"" + title + "\".");
    }

    @Override
    public void saleCopy(int numCopies) {
        super.saleCopy(numCopies);
    }
}

public class PublicationSystem {
    public static void main(String[] args) {
        Book book1 = new Book("Java Programming", 29.99, 100, "John Doe");
        Magazine magazine1 = new Magazine("Tech Monthly", 5.99, 200, 50,
"January 2024");
```

```java
        book1.orderCopies(20);
        magazine1.orderQty(100);

        book1.saleCopy(10);
        magazine1.saleCopy(30);

        magazine1.receiveIssue("February 2024");
    }
}
```

Output:

```
Ordered 20 more copies of the book "Java Programming" by John Doe.
Ordered 100 copies of the magazine "Tech Monthly".
10 copies of "Java Programming" sold at $29.99 each.
Total sale: $299.9
30 copies of "Tech Monthly" sold at $5.99 each.
Total sale: $179.7
New issue of "Tech Monthly" received: February 2024
```

Experiments 03:

```
/*
3.Inheritance: Design and develop inheritance for a given case study, identify
objects and relationships and implement inheritance wherever applicable. Employee
class hasEmp_name, Emp_id, Address, Curriculum for Second Year of Information
Technology (2019 Course), Savitribai Phule Pune University SE (Information
Technology) Syllabus (2019 Course) 37 Mail_id, and Mobile_noas members. Inherit the
classes: Programmer, Team Lead, Assistant Project Manager and Project Manager from
employee class. Add Basic Pay (BP) as the member of all the inherited classes with
97% of BP as DA, 10 % of BP as HRA, 12% of BP as PF, 0.1% of BP for staff club
fund. Generate pay slips for the employees with their gross and net salary.
*/

import java.util.*;

class Employee{
   String empName;
   int empId;
   String empAddress;
   String empMailId;
   String empMobileNo;

   Employee(String empName, int empId, String empAddress, String empMailId, String
empMobileNo){
       this.empName = empName;
       this.empId = empId;
       this.empAddress = empAddress;
       this.empMailId = empMailId;
       this.empMobileNo = empMobileNo;
   }

   void makePaySlip(double basicPay){
       double DA = ((double) 97 / 100) * basicPay;
       double HRA = ((double) 10 / 100) * basicPay;
       double PF = ((double) 12 / 100) * basicPay;
       double staffClubFund  = ((double) 0.1 / 100) * basicPay;
       double grossSalary = basicPay + DA + HRA;

       System.out.println("################ Start ################");
       System.out.println("Name: " + empName + " | Employee id: " + empId + " |
Email id: " + empMailId);
       System.out.println("Employee address: " + empAddress + " | Mobile no: " +
empMobileNo + "\n");
       System.out.println("Basic Pay: $" + basicPay);
```

```
        System.out.println("Gross Salary: $" + grossSalary);
        System.out.println("Net Salary: $" + (grossSalary - (PF + staffClubFund)));
        System.out.println("################ End ################\n");
    }
}

class Programmer extends Employee{
    public double basicPay;
    Programmer(String empName, int empId, String empAddress, String empMailId,
String empMobileNo, double basicPay){
        super(empName, empId, empAddress, empMailId, empMobileNo);
        this.basicPay = basicPay;
    }

    public void showPaySlip(){
        makePaySlip(basicPay);
    }
}

class TeamLead extends Employee{
    double basicPay;
    TeamLead(String empName, int empId, String empAddress, String empMailId, String
empMobileNo, double basicPay){
        super(empName, empId, empAddress, empMailId, empMobileNo);
        this.basicPay = basicPay;
    }

    public void showPaySlip(){
        makePaySlip(basicPay);
    }
}

class AssistantProjectManager extends Employee{
    double basicPay;
    AssistantProjectManager(String empName, int empId, String empAddress, String
empMailId, String empMobileNo, double basicPay){
        super(empName, empId, empAddress, empMailId, empMobileNo);
        this.basicPay = basicPay;
    }

    public void showPaySlip(){
        makePaySlip(basicPay);
    }
}

class ProjectManager extends Employee{
    double basicPay;
```

```
   ProjectManager(String empName, int empId, String empAddress, String empMailId,
String empMobileNo, double basicPay){
       super(empName, empId, empAddress, empMailId, empMobileNo);
       this.basicPay = basicPay;
   }

   public void showPaySlip(){
       makePaySlip(basicPay);
   }
}

public class Assignment_03 {
   public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       boolean exit = false;

       while (!exit) {
           System.out.println("Menu:");
           System.out.println("1. Generate Pay Slip for Programmer");
           System.out.println("2. Generate Pay Slip for Team Lead");
           System.out.println("3. Generate Pay Slip for Assistant Project
Manager");
           System.out.println("4. Generate Pay Slip for Project Manager");
           System.out.println("5. Exit");
           System.out.print("Enter your choice: ");
           int choice = sc.nextInt();

           switch (choice) {
               case 1:
                   System.out.print("Enter Name: ");
                   String pname = sc.next();
                   System.out.print("Enter ID: ");
                   int pid = sc.nextInt();
                   System.out.print("Enter Address: ");
                   String paddress = sc.next();
                   System.out.print("Enter Email: ");
                   String pmail = sc.next();
                   System.out.print("Enter Mobile No: ");
                   String pmobile = sc.next();
                   System.out.print("Enter Basic Pay: ");
                   double ppay = sc.nextDouble();
                   Programmer prog = new Programmer(pname, pid, paddress, pmail,
pmobile, ppay);
                   prog.showPaySlip();
                   break;
               case 2:
                   System.out.print("Enter Name: ");
```

```
                    String tlname = sc.next();
                    System.out.print("Enter ID: ");
                    int tlid = sc.nextInt();
                    System.out.print("Enter Address: ");
                    String tladdress = sc.next();
                    System.out.print("Enter Email: ");
                    String tlmail = sc.next();
                    System.out.print("Enter Mobile No: ");
                    String tlmobile = sc.next();
                    System.out.print("Enter Basic Pay: ");
                    double tlpay = sc.nextDouble();
                    TeamLead lead = new TeamLead(tlname, tlid, tladdress, tlmail,
tlmobile, tlpay);
                    lead.showPaySlip();
                    break;
                case 3:
                    System.out.print("Enter Name: ");
                    String apname = sc.next();
                    System.out.print("Enter ID: ");
                    int apid = sc.nextInt();
                    System.out.print("Enter Address: ");
                    String apaddress = sc.next();
                    System.out.print("Enter Email: ");
                    String apmail = sc.next();
                    System.out.print("Enter Mobile No: ");
                    String apmobile = sc.next();
                    System.out.print("Enter Basic Pay: ");
                    double appay = sc.nextDouble();
                    AssistantProjectManager apm = new
AssistantProjectManager(apname, apid, apaddress, apmail, apmobile, appay);
                    apm.showPaySlip();
                    break;
                case 4:
                    System.out.print("Enter Name: ");
                    String pmname = sc.next();
                    System.out.print("Enter ID: ");
                    int pmid = sc.nextInt();
                    System.out.print("Enter Address: ");
                    String pmaddress = sc.next();
                    System.out.print("Enter Email: ");
                    String pmmail = sc.next();
                    System.out.print("Enter Mobile No: ");
                    String pmmobile = sc.next();
                    System.out.print("Enter Basic Pay: ");
                    double pmpay = sc.nextDouble();
                    ProjectManager pm = new ProjectManager(pmname, pmid, pmaddress,
pmmail, pmmobile, pmpay);
```

```
                    pm.showPaySlip();
                    break;
                case 5:
                    exit = true;
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice!");
            }
        }
        sc.close();
    }
}
```

Output:

```
1. Generate Pay Slip for Programmer
2. Generate Pay Slip for Team Lead
3. Generate Pay Slip for Assistant Project Manager
4. Generate Pay Slip for Project Manager
5. Exit
Enter your choice: 3
Enter Name: Ali
Enter ID: 1227
Enter Address: Pune
Enter Email: example@gmail.com
Enter Mobile No: 1234567890
Enter Basic Pay: 79666

################ Start ################
Name: Ali | Employee id: 1227 | Email id: example@gmail.com
Employee address: Pune | Mobile no: 1234567890

Basic Pay: $79666.0
Gross Salary: $164908.62000000002
Net Salary: $155269.034
################ End ################

Menu:
1. Generate Pay Slip for Programmer
```

```
2. Generate Pay Slip for Team Lead
3. Generate Pay Slip for Assistant Project Manager
4. Generate Pay Slip for Project Manager
5. Exit
Enter your choice: 5
Exiting...
```

Experiment 04:

```java
/*
4.Dynamic Binding: Design a base class shape with two double type values and member
functions to input the data and compute_area() for calculating area of shape.
Derive two classes: triangle and rectangle. Make compute_area() as an abstract
function and redefine this function in the derived class to suit their
requirements. Write a program that accepts dimensions of triangle/rectangle and
displays calculated area. Implement dynamic binding for given case study
*/

import java.util.*;

abstract class Shape{
   double length;
   double breath;

   Shape(double length, double breath){
       this.length = length;
       this.breath = breath;
   }

   abstract void calc_area();
}

class Rectangle extends Shape{
   Rectangle(double length, double breath){
       super(length, breath);
   }

   void calc_area() {
       System.out.println("The area of rectangle is: " + length * breath);
   }
}

class Triangle extends Shape{
   Triangle(double length, double breath){
       super(length, breath);
   }

   void calc_area() {
       System.out.println("The area of triangle is: " + (0.5 * length * breath));
   }
}
```

```java
public class Assignment_04 {

   public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       System.out.print("Enter R: Rectagle, T: Triangle, E: Exit : ");
       String res = sc.next();

       do {
           switch(res) {
               case "R" -> {
                   System.out.print("Enter lenght: ");
                   double length = sc.nextDouble();
                   System.out.print("Enter breath: ");
                   double breath = sc.nextDouble();
                   Rectangle R1 = new Rectangle(length, breath);
                   R1.calc_area();
               }
               case "T" -> {
                   System.out.print("Enter lenght: ");
                   double length = sc.nextDouble();
                   System.out.print("Enter breath: ");
                   double breath = sc.nextDouble();
                   Triangle T1 = new Triangle(length, breath);
                   T1.calc_area();
               }
               default -> {
                   System.out.println("Invalid input!");
               }
           }
           System.out.print("Do you want continue: Y/N?: ");
           String input = sc.next();
           if("N".equals(input)) {
               break;
           }else {
               System.out.print("Enter R: Rectagle, T: Triangle, E: Exit : ");
               res = sc.next();
           }
       }while(!"E".equals(res));
   }
}
```

Output:

```
Enter R: Rectangle, T: Triangle, E: Exit : T
Enter length: 44
Enter breath: 17
```

```
The area of triangle is: 374.0
Do you want continue: Y/N?: y
Enter R: Rectangle, T: Triangle, E: Exit : R
Enter length: 19
Enter breath: 11
The area of rectangle is: 209.0
Do you want continue: Y/N?: N
```

Experiment 05:

```
/*
5.Interface: Design and develop a context for given case study and
implement an interface for Vehicles Consider the example of vehicles like
bicycle, car and bike. All Vehicles have common functionalities such as
Gear Change, Speed up and apply brakes. Make an interface and put all these
common functionalities. Bicycle, Bike, Car classes should be implemented
for all these functionalities in their own class in their own way
*/

interface Vehicle {
    void changeGear(int gear);
    void speedUp(int increment);
    void applyBrakes(int decrement);
}

class Bicycle implements Vehicle {
    private int speed;
    private int gear;

    public Bicycle() {
        this.speed = 0;
        this.gear = 1;
    }

    @Override
    public void changeGear(int gear) {
        this.gear = gear;
        System.out.println("Bicycle gear changed to: " + this.gear);
    }

    @Override
    public void speedUp(int increment) {
        speed += increment;
        System.out.println("Bicycle speed increased to: " + speed + "
km/h");
    }

    @Override
    public void applyBrakes(int decrement) {
```

```java
        speed -= decrement;
        if (speed < 0) speed = 0;
        System.out.println("Bicycle speed after applying brakes: " + speed
+ " km/h");
    }
}

class Bike implements Vehicle {
    private int speed;
    private int gear;

    public Bike() {
        this.speed = 0;
        this.gear = 1;
    }

    @Override
    public void changeGear(int gear) {
        this.gear = gear;
        System.out.println("Bike gear changed to: " + this.gear);
    }

    @Override
    public void speedUp(int increment) {
        speed += increment;
        System.out.println("Bike speed increased to: " + speed + " km/h");
    }

    @Override
    public void applyBrakes(int decrement) {
        speed -= decrement;
        if (speed < 0) speed = 0;
        System.out.println("Bike speed after applying brakes: " + speed + "
km/h");
    }
}

class Car implements Vehicle {
    private int speed;
    private int gear;

    public Car() {
        this.speed = 0;
        this.gear = 1;
    }

    @Override
    public void changeGear(int gear) {
        this.gear = gear;
        System.out.println("Car gear changed to: " + this.gear);
    }

    @Override
    public void speedUp(int increment) {
        speed += increment;
        System.out.println("Car speed increased to: " + speed + " km/h");
    }

    @Override
    public void applyBrakes(int decrement) {
        speed -= decrement;
        if (speed < 0) speed = 0;
        System.out.println("Car speed after applying brakes: " + speed + "
km/h");
    }
}

public class Main {
    public static void main(String[] args) {
        Bicycle bicycle = new Bicycle();
        Bike bike = new Bike();
        Car car = new Car();

        System.out.println("Testing Bicycle:");
        bicycle.changeGear(2);
        bicycle.speedUp(10);
        bicycle.applyBrakes(3);

        System.out.println("\nTesting Bike:");
        bike.changeGear(3);
        bike.speedUp(20);
        bike.applyBrakes(5);

        System.out.println("\nTesting Car:");
        car.changeGear(4);
        car.speedUp(30);
        car.applyBrakes(10);
    }
}
```

Output

```
Testing Bicycle:
Bicycle gear changed to: 2
Bicycle speed increased to: 10 km/h
Bicycle speed after applying brakes: 7 km/h

Testing Bike:
Bike gear changed to: 3
Bike speed increased to: 20 km/h
Bike speed after applying brakes: 15 km/h

Testing Car:
Car gear changed to: 4
Car speed increased to: 30 km/h
Car speed after applying brakes: 20 km/h
```

Experiment 06

```java
/*
6.Exception handling: Implement a program to handle Arithmetic exception,
Array Index Out of Bounds. The user enters two numbers Num1 and Num2. The
division of Num1 and Num2 is displayed. If Num1 and Num2 are not integers,
the program would throw a Number Format Exception. If Num2 were zero, the
program would throw an Arithmetic Exception. Display the exception
*/

import java.util.Scanner;

public class ExceptionHandlingExample {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        try {
            // Prompt the user to enter two numbers
            System.out.print("Enter the first number (Num1): ");
            String input1 = scanner.nextLine();
            System.out.print("Enter the second number (Num2): ");
            String input2 = scanner.nextLine();

            // Attempt to parse the inputs as integers
            int num1 = Integer.parseInt(input1);
            int num2 = Integer.parseInt(input2);

            // Perform division
            int result = num1 / num2;
            System.out.println("Result of division (Num1 / Num2): " +
result);

            // Simulate Array Index Out of Bounds for demonstration
            int[] array = new int[5];
            System.out.println("Accessing an out-of-bounds index: " +
array[10]);

        } catch (ArithmeticException e) {
            System.out.println("Arithmetic Exception: Division by zero is
not allowed.");
        } catch (NumberFormatException e) {
            System.out.println("Number Format Exception: Please enter valid
integers.");
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array Index Out of Bounds Exception: Tried
to access an invalid array index.");
        } finally {
            scanner.close();
            System.out.println("Program execution completed.");
        }
    }
}
```

Output:

```
//Test 1
Enter the first number (Num1): 10
Enter the second number (Num2): 0
Arithmetic Exception: Division by zero is not allowed. Program
execution completed.

//Test 2
Enter the first number (Num1): ten
Enter the second number (Num2): five
Number Format Exception: Please enter valid integers. Program
execution completed.

//Test 3
Enter the first number (Num1): 10
Enter the second number (Num2): 2
Result of division (Num1 / Num2): 5
Array Index Out of Bounds Exception: Tried to access an invalid array
index. Program execution completed.
```

Experiment 07:

```
/*
7.Template: Implement a generic program using any collection class to count
the number of elements in a collection that have a specific property such
as even numbers, odd number, prime number and palindromes.
*/

import java.util.ArrayList;
import java.util.List;
import java.util.function.Predicate;

public class GenericCollectionCounter {

    public static <T> int countIf(List<T> collection, Predicate<T>
condition) {
        int count = 0;
        for (T element : collection) {
            if (condition.test(element)) {
                count++;
            }
        }
        return count;
    }

    public static Predicate<Integer> isEven() {
        return num -> num % 2 == 0;
    }

    public static Predicate<Integer> isOdd() {
        return num -> num % 2 != 0;
    }

    public static Predicate<Integer> isPrime() {
        return num -> {
            if (num <= 1) return false;
            for (int i = 2; i <= Math.sqrt(num); i++) {
                if (num % i == 0) return false;
            }
            return true;
        };
```

```java
    }

    public static Predicate<Integer> isPalindrome() {
        return num -> {
            String original = Integer.toString(num);
            String reversed = new
StringBuilder(original).reverse().toString();
            return original.equals(reversed);
        };
    }

    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<>();
        numbers.add(121);
        numbers.add(44);
        numbers.add(37);
        numbers.add(22);
        numbers.add(101);
        numbers.add(7);
        numbers.add(56);
        numbers.add(13);

        int evenCount = countIf(numbers, isEven());
        int oddCount = countIf(numbers, isOdd());
        int primeCount = countIf(numbers, isPrime());
        int palindromeCount = countIf(numbers, isPalindrome());

        System.out.println("Total even numbers: " + evenCount);
        System.out.println("Total odd numbers: " + oddCount);
        System.out.println("Total prime numbers: " + primeCount);
        System.out.println("Total palindrome numbers: " + palindromeCount);
    }
}
```

Output:

```
Total even numbers: 3
Total odd numbers: 5
Total prime numbers: 3
Total palindrome numbers: 3
```

Experiment 08:

```java
/*
8.File Handling: Implement a program for maintaining a database of student
records using Files. Students have Student_id,name, Roll_no, Class, marks
and address. Display the data for a few students.
1. Create Database
2. Display Database
3. Delete Records
4. Update Record
5. Search Record
*/

import java.io.*;
import java.util.*;

class Student {
    String studentId;
    String name;
    String rollNo;
    String className;
    int marks;
    String address;

    public Student(String studentId, String name, String rollNo, String
className, int marks, String address) {
        this.studentId = studentId;
        this.name = name;
        this.rollNo = rollNo;
        this.className = className;
        this.marks = marks;
        this.address = address;
    }

    @Override
    public String toString() {
        return studentId + "," + name + "," + rollNo + "," + className +
"," + marks + "," + address;
    }

    public static Student fromString(String str) {
        String[] parts = str.split(",");
        return new Student(parts[0], parts[1], parts[2], parts[3],
Integer.parseInt(parts[4]), parts[5]);
    }
}

public class StudentDatabase {
    private static final String FILE_NAME = "students.txt";

    public static void createStudent(Student student) throws IOException {
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(FILE_NAME, true))) {
            writer.write(student.toString());
            writer.newLine();
        }
    }

    public static void displayDatabase() throws IOException {
        try (BufferedReader reader = new BufferedReader(new
FileReader(FILE_NAME))) {
            String line;
            System.out.println("Student Records:");
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
        }
    }

    public static void deleteRecord(String studentId) throws IOException {
        File inputFile = new File(FILE_NAME);
        File tempFile = new File("temp.txt");

        try (BufferedReader reader = new BufferedReader(new
FileReader(inputFile));
             BufferedWriter writer = new BufferedWriter(new
FileWriter(tempFile))) {
            String line;
            while ((line = reader.readLine()) != null) {
                if (!line.startsWith(studentId + ",")) {
                    writer.write(line);
                    writer.newLine();
                }
            }
        }

        if (!inputFile.delete()) {
            System.out.println("Could not delete original file");
        }
        if (!tempFile.renameTo(inputFile)) {
            System.out.println("Could not rename temp file");
        }
    }

    public static void updateRecord(String studentId, Student
updatedStudent) throws IOException {
        deleteRecord(studentId);
        createStudent(updatedStudent);
    }

    public static void searchRecord(String studentId) throws IOException {
        try (BufferedReader reader = new BufferedReader(new
FileReader(FILE_NAME))) {
            String line;
            while ((line = reader.readLine()) != null) {
                if (line.startsWith(studentId + ",")) {
                    System.out.println("Student Found: " + line);
                    return;
                }
            }
            System.out.println("Student with ID " + studentId + " not
found.");
        }
    }

    public static void main(String[] args) throws IOException {
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("\nStudent Database Management System");
            System.out.println("1. Create Database");
            System.out.println("2. Display Database");
            System.out.println("3. Delete Record");
            System.out.println("4. Update Record");
            System.out.println("5. Search Record");
            System.out.println("6. Exit");
            System.out.print("Choose an option: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch (choice) {
                case 1:
                    System.out.print("Enter Student ID: ");
                    String id = scanner.nextLine();
                    System.out.print("Enter Name: ");
                    String name = scanner.nextLine();
                    System.out.print("Enter Roll No: ");
                    String rollNo = scanner.nextLine();
                    System.out.print("Enter Class: ");
                    String className = scanner.nextLine();
                    System.out.print("Enter Marks: ");
                    int marks = scanner.nextInt();
                    scanner.nextLine(); // Consume newline
                    System.out.print("Enter Address: ");
                    String address = scanner.nextLine();
                    Student student = new Student(id, name, rollNo,
className, marks, address);
                    createStudent(student);
                    System.out.println("Student record created.");
                    break;

                case 2:
                    displayDatabase();
                    break;

                case 3:
                    System.out.print("Enter Student ID to delete: ");
                    String deleteId = scanner.nextLine();
                    deleteRecord(deleteId);
                    System.out.println("Student record deleted if it
existed.");
                    break;

                case 4:
                    System.out.print("Enter Student ID to update: ");
                    String updateId = scanner.nextLine();
                    System.out.print("Enter new Name: ");
                    String newName = scanner.nextLine();
                    System.out.print("Enter new Roll No: ");
                    String newRollNo = scanner.nextLine();
                    System.out.print("Enter new Class: ");
                    String newClassName = scanner.nextLine();
                    System.out.print("Enter new Marks: ");
                    int newMarks = scanner.nextInt();
                    scanner.nextLine(); // Consume newline
                    System.out.print("Enter new Address: ");
                    String newAddress = scanner.nextLine();
                    Student updatedStudent = new Student(updateId, newName,
newRollNo, newClassName, newMarks, newAddress);
                    updateRecord(updateId, updatedStudent);
                    System.out.println("Student record updated if it
existed.");
                    break;

                case 5:
                    System.out.print("Enter Student ID to search: ");
                    String searchId = scanner.nextLine();
                    searchRecord(searchId);
                    break;

                case 6:
                    System.out.println("Exiting the system.");
                    scanner.close();
                    System.exit(0);
                    break;

                default:
                    System.out.println("Invalid option. Please try
again.");
            }
        }
    }
}
```

Output:

```
Student Database Management System
1. Create Database
2. Display Database
3. Delete Record
4. Update Record
5. Search Record
```

```
6. Exit
Choose an option: 1
Enter Student ID: 1332
Enter Name: Sam
Enter Roll No: 23
Enter Class: FE
Enter Marks: 86
Enter Address: Mumbai
Student record created.

Student Database Management System
1. Create Database
2. Display Database
3. Delete Record
4. Update Record
5. Search Record
6. Exit
Choose an option: 2
Student Records:
1332,Sam,23,FE,86,Mumbai

Student Database Management System
1. Create Database
2. Display Database
3. Delete Record
4. Update Record
5. Search Record
6. Exit
Choose an option: 6
Exiting the system.
```

Experiment 09:

```
/*
9.Case Study: Using concepts of Object-Oriented programming develop
solution for any one application
1) Banking system having following operations :
1. Create an account
2. Deposit money
3. Withdraw money
4. Honour daily withdrawal limit
5. Check the balance
6. Display Account information.
2) Inventory management system having following operations :
1. List of all products
2. Display individual product
*/

import java.util.Scanner;

class BankAccount {
    private String accountNumber;
    private String accountHolderName;
    private double balance;
    private double dailyWithdrawalLimit;
    private double dailyWithdrawnAmount;

    public BankAccount(String accountNumber, String accountHolderName,
double initialBalance, double dailyWithdrawalLimit) {
        this.accountNumber = accountNumber;
        this.accountHolderName = accountHolderName;
        this.balance = initialBalance;
        this.dailyWithdrawalLimit = dailyWithdrawalLimit;
        this.dailyWithdrawnAmount = 0;
    }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited: $" + amount);
        } else {
            System.out.println("Invalid deposit amount!");
        }
    }

    public void withdraw(double amount) {
        if (amount > balance) {
            System.out.println("Insufficient balance!");
        } else if (dailyWithdrawnAmount + amount > dailyWithdrawalLimit) {
            System.out.println("Withdrawal exceeds daily limit!");
        } else if (amount > 0) {
            balance -= amount;
            dailyWithdrawnAmount += amount;
            System.out.println("Withdrawn: $" + amount);
        } else {
            System.out.println("Invalid withdrawal amount!");
        }
    }

    public void resetDailyWithdrawal() {
        dailyWithdrawnAmount = 0;
    }

    public double checkBalance() {
        return balance;
    }

    public void displayAccountInfo() {
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Account Holder Name: " + accountHolderName);
        System.out.println("Balance: $" + balance);
        System.out.println("Daily Withdrawal Limit: $" +
dailyWithdrawalLimit);
        System.out.println("Amount Withdrawn Today: $" +
dailyWithdrawnAmount);
    }
}

public class BankingSystem {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        BankAccount account = new BankAccount("123456", "John Doe", 500.0,
200.0);
```

```
        while (true) {
            System.out.println("\nBanking System Menu:");
            System.out.println("1. Create Account (Account already created
in this example)");
            System.out.println("2. Deposit Money");
            System.out.println("3. Withdraw Money");
            System.out.println("4. Check Balance");
            System.out.println("5. Display Account Information");
            System.out.println("6. Exit");
            System.out.print("Choose an option: ");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.println("Account creation not implemented in
this example.");
                    break;

                case 2:
                    System.out.print("Enter amount to deposit: ");
                    double depositAmount = scanner.nextDouble();
                    account.deposit(depositAmount);
                    break;

                case 3:
                    System.out.print("Enter amount to withdraw: ");
                    double withdrawAmount = scanner.nextDouble();
                    account.withdraw(withdrawAmount);
                    break;

                case 4:
                    System.out.println("Current Balance: $" +
account.checkBalance());
                    break;

                case 5:
                    account.displayAccountInfo();
                    break;

                case 6:
                    System.out.println("Exiting the system. Have a nice
day!");
                    scanner.close();
                    System.exit(0);

                default:
                    System.out.println("Invalid option. Please choose
again.");
            }
        }
    }
}
```

Output:

```
Banking System Menu:
1. Create Account (Account already created in this example)
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. Display Account Information
6. Exit
Choose an option: 2
Enter amount to deposit: 200
Deposited: $200.0
Banking Syste
Menu: 1. Create Account (Account already created in this example)
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. Display Account Information
6. Exit
Choose an option: 3
Enter amount to withdraw: 150
Withdrawn: $150.0
```

Experiment 10:

```
/*
10. Factory Design Pattern: Implement Factory design pattern for the given
context. Consider Car building process, which requires many steps from
allocating accessories to final makeup. These steps should be written as
methods and should be called while creating an instance of a specific car
type. Hatchback, Sedan, SUV could be the subclasses of Car class. Car class
and its subclasses, CarFactory and Test Factory Pattern should be
implemented.
*/

abstract class Car {
    public Car() {
        assemble();
    }

    protected abstract void addAccessories();
    protected abstract void paint();
    protected abstract void finalMakeup();

    private void assemble() {
        allocateChassis();
        allocateEngine();
        allocateWheels();
        addAccessories();
        paint();
        finalMakeup();
    }

    private void allocateChassis() {
        System.out.println("Chassis allocated.");
    }

    private void allocateEngine() {
        System.out.println("Engine allocated.");
    }

    private void allocateWheels() {
        System.out.println("Wheels allocated.");
    }
```

```java
}

class Hatchback extends Car {
    @Override
    protected void addAccessories() {
        System.out.println("Adding basic accessories for Hatchback.");
    }

    @Override
    protected void paint() {
        System.out.println("Painting Hatchback with basic color.");
    }

    @Override
    protected void finalMakeup() {
        System.out.println("Final makeup for Hatchback.");
    }
}

class Sedan extends Car {
    @Override
    protected void addAccessories() {
        System.out.println("Adding premium accessories for Sedan.");
    }

    @Override
    protected void paint() {
        System.out.println("Painting Sedan with luxury color.");
    }

    @Override
    protected void finalMakeup() {
        System.out.println("Final makeup for Sedan.");
    }
}

class SUV extends Car {
    @Override
    protected void addAccessories() {
        System.out.println("Adding rugged accessories for SUV.");
    }

    @Override
    protected void paint() {
        System.out.println("Painting SUV with durable color.");
    }

    @Override
    protected void finalMakeup() {
        System.out.println("Final makeup for SUV.");
    }
}

class CarFactory {
    public static Car createCar(String type) {
        switch (type.toLowerCase()) {
            case "hatchback":
                return new Hatchback();
            case "sedan":
                return new Sedan();
            case "suv":
                return new SUV();
            default:
                throw new IllegalArgumentException("Unknown car type: " +
type);
        }
    }
}

public class TestFactoryPattern {
    public static void main(String[] args) {
        System.out.println("Creating a Hatchback:");
        Car hatchback = CarFactory.createCar("hatchback");

        System.out.println("\nCreating a Sedan:");
        Car sedan = CarFactory.createCar("sedan");

        System.out.println("\nCreating an SUV:");
        Car suv = CarFactory.createCar("suv");
    }
}
```

Output:

```
Creating a Hatchback:
Chassis allocated.
Engine allocated.
Wheels allocated.
Adding basic accessories for Hatchback.
Painting Hatchback with basic color.
Final makeup for Hatchback.

Creating a Sedan:
Chassis allocated.
Engine allocated.
Wheels allocated.
Adding premium accessories for Sedan.
Painting Sedan with luxury color.
Final makeup for Sedan.

Creating an SUV:
Chassis allocated.
Engine allocated.
Wheels allocated.
Adding rugged accessories for SUV.
Painting SUV with durable color.
Final makeup for SUV.
```

Experiment 10:

```
/*
11. Strategy Design Pattern: Implement and apply Strategy Design pattern
for simple Shopping Cart where three payment strategies are used such as
Credit Card, PayPal, Bitcoin. Create an interface for strategy patterns and
give concrete implementation for payment.
*/

// PaymentStrategy.java
public interface PaymentStrategy {
    void pay(double amount);
}
```

```
// CreditCardPayment.java
public class CreditCardPayment implements PaymentStrategy {
    private String cardNumber;
    private String name;
    private String cvv;

    public CreditCardPayment(String cardNumber, String name, String cvv) {
        this.cardNumber = cardNumber;
        this.name = name;
        this.cvv = cvv;
    }

    @Override
    public void pay(double amount) {
        System.out.println("Paid $" + amount + " using Credit Card.");
    }
}
```

```
// PayPalPayment.java
public class PayPalPayment implements PaymentStrategy {
    private String email;

    public PayPalPayment(String email) {
        this.email = email;
    }

```

```java
    @Override
    public void pay(double amount) {
        System.out.println("Paid $" + amount + " using PayPal.");
    }
}
```

```java
// BitcoinPayment.java
public class BitcoinPayment implements PaymentStrategy {
    private String walletAddress;

    public BitcoinPayment(String walletAddress) {
        this.walletAddress = walletAddress;
    }

    @Override
    public void pay(double amount) {
        System.out.println("Paid $" + amount + " using Bitcoin.");
    }
}
```

```java
// ShoppingCart.java
import java.util.ArrayList;
import java.util.List;

public class ShoppingCart {
    private List<Double> items;

    public ShoppingCart() {
        this.items = new ArrayList<>();
    }

    public void addItem(double price) {
        items.add(price);
    }

    public void removeItem(double price) {
        items.remove(price);
    }

    public double calculateTotal() {
        return items.stream().mapToDouble(Double::doubleValue).sum();
    }

    public void pay(PaymentStrategy paymentStrategy) {
        double amount = calculateTotal();
        paymentStrategy.pay(amount);
    } }
```

```java
// Main.java
public class Main {
    public static void main(String[] args) {
        ShoppingCart cart = new ShoppingCart();

        cart.addItem(100.00);
        cart.addItem(200.50);

        PaymentStrategy creditCardPayment = new
CreditCardPayment("1234-5678-9101-1121", "John Doe", "123");
        cart.pay(creditCardPayment);

        PaymentStrategy paypalPayment = new
PayPalPayment("johndoe@example.com");
        cart.pay(paypalPayment);

        PaymentStrategy bitcoinPayment = new
BitcoinPayment("1A2b3C4d5E6f7G8h9I0j");
        cart.pay(bitcoinPayment);
    }
}
```

Output:

```
Paid $300.5 using Credit Card.
Paid $300.5 using PayPal.
Paid $300.5 using Bitcoin.
```