



# Kristu Jayanti College

**AUTONOMOUS**

**Bengaluru**

Reaccredited A++ Grade by NAAC | Affiliated to Bengaluru North University

**Department of Computer Science (PG)**

**I Semester**

**Practical Record on**

**JAVA PROGRAMMING PRACTICAL (MCC2L2C11)**

**For the academic year 2024 -2025**

**Submitted by**

**NAME : NATALIYA IQBAL**

**ROLL. NO : 24MCAA42**



# Kristu Jayanti College

**AUTONOMOUS** Bengaluru

Reaccredited A++ Grade by NAAC | Affiliated to Bengaluru North University

**DEPARTMENT OF COMPTUER SCIENCE (PG)**

**Master of Computer Applications**

**CERTIFICATE**

This is to certify that **Ms. Nataliya Iqbal** bearing roll number **24MCAA42** has satisfactorily completed the practical programs for the course **JAVA PROGRAMMING PRACTICAL, MCC2L2C11** during the academic year 2024 – 2025.

Date:

Signature of the Course Teacher

Head of the Department

Examiners (Name & Signature)

1. \_\_\_\_\_

Center : Kristu Jayanti College

Date :

2. \_\_\_\_\_



# Kristu Jayanti College

**AUTONOMOUS** Bengaluru  
Reaccredited A++ Grade by NAAC | Affiliated to Bengaluru North University

## Table of Contents

Sl. No	Date	Program title	Page No.
01	19.08.2024	Java program to convert decimal to binary, octal and hexadecimal	1-5
02	21.08.2024	Java program to explain the concept of constructor overloading	6-9
03	26.08.2024	Java program to explain the concept of passing object as parameters by adding two distance and inches	10-13
04	28.08.2024	Java Program to implement the concept of inheritance	14-19
05	02.09.2024	Java program to explain the concept of runtime polymorphism	20-23
06	04.09.2024	Java program to implement producer consumer problem using thread concept	24-29
07	05.10.2024	Java program to create object for Tree Set and Stack and use all methods	30-35
08	18.09.2024	Java program to implement Exception Handling in Java	36-45
09	06.10.2024	Java program to implement the concept of Interface	46-50
10	07.10.2024	Java program to get file name at runtime and display no. of lines and words	51-54
11	09.10.2024	Java program to list files in current working directory depending upon a given pattern	55-57
12	16.10.2024	Java program to create frame for Student Registration containing all the fields.	58-63
13	22.10.2024	Java program to create Student Database Application using java Swing	64-70
14	06.11.2024	Java program using Java Database Connectivity (JDBC) for storing employee details	71-76



**PROGRAM 1:****Source Code:**

```
import java.util.Scanner;

public class DecimalConversion{

    public static String toBinary(int decimal){

        if(decimal==0){

            return "";

        }

        return toBinary(decimal/2)+(decimal%2);

    }

    public static String toOctal(int decimal){

        if(decimal==0){

            return "";

        }

        return toOctal(decimal/8)+(decimal%8);

    }

    public static String toHexa(int decimal){

        if(decimal==0){

            return "";

        }

        int reminder=decimal%16;

        String

        hexDigit=(reminder<10)?Integer.toString(reminder):String.valueOf((char)('A'+(reminder-

        10)));

        return toHexa(decimal/16)+(decimal%16);

    }

    public static void main(String[] args){

        Scanner scanner=new Scanner(System.in);

        System.out.print("Enter a decimal no.:");

        int decimalNum=scanner.nextInt();
```

```
String binary=toBinary(decimalNum);
System.out.println("Binary:"+(binary.isEmpty()? "0":binary))
String octal=toOctal(decimalNum);
System.out.println("Octal:"+(octal.isEmpty()? "0":octal));
String hexa=toHexa(decimalNum);
System.out.println("Hexa:"+(hexa.isEmpty()? "0":hexa))
scanner.close();
}
}
```

**OUTPUT :**

```
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>javac DecimalConversion.java
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>java DecimalConversion
Enter a decimal no.:45
Binary:101101
Octal:55
Hexa:213
```





**PROGRAM 2:****Source Code:**

```
import java.util.Scanner;

class Rectangle{

    private double length;

    private double width;

    public Rectangle(){

        this.length=1.0;

        this.width=1.0;

    }

    public Rectangle(double side){

        this.length=side;

        this.width=side;

    }

    public Rectangle(double length,double width){

        this.length=length;

        this.width=width;

    }

    public double calculateArea(){

        return length*width;

    }

    public void display(){

        System.out.println("Length:"+length+",Width:"+width);

    }

    public static void main(String[] args){

        Scanner scan=new Scanner(System.in);

        System.out.println("Chose the type of rectangle you want to create:");

        System.out.println("1.Default rectangle(1.0x1.0)");

        System.out.println("2.Square(Enter side length)");
```

```
System.out.println("3.Rectangle(Enter Length and width)");
System.out.println("Enter your choice(1,2 or 3):");
int choice=scan.nextInt();

Rectangle rect;
switch(choice){
    case 1:
        rect=new Rectangle();
        break;
    case 2:
        System.out.println("Enter the side length of square:");
        double side=scan.nextDouble();
        rect=new Rectangle(side);
        break;
    case 3:
        System.out.println("Enter the length of Rectangle:");
        double length= scan.nextDouble();
        System.out.println("Enter The width of rectangle:");
        double width=scan.nextDouble();
        rect=new Rectangle(length,width);
        break;
    default:
        System.out.println("Invalid choice. Using default rectangle.");
        rect=new Rectangle();
}

rect.display();
System.out.println("Area:"+rect.calculateArea());
scan.close();
}
```

**OUTPUT:**

```
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>java Rectangle
Chose the type of rectangle you want to create:
1.Default rectangle(1.0x1.0)
2.Square(Enter side length)
3.Rectangle(Enter Length and width)
Enter your choice(1,2 or 3):
3
Enter the length of Rectangle:
5
Enter The width of rectangle:
9
Length:5.0,Width:9.0
Area:45.0

C:\Program Files (x86)\Java\jdk1.7.0_65\bin>_
```

```
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>javac Rectangle.java

C:\Program Files (x86)\Java\jdk1.7.0_65\bin>java Rectangle
Chose the type of rectangle you want to create:
1.Default rectangle(1.0x1.0)
2.Square(Enter side length)
3.Rectangle(Enter Length and width)
Enter your choice(1,2 or 3):
1
Length:1.0,Width:1.0
Area:1.0

C:\Program Files (x86)\Java\jdk1.7.0_65\bin>java Rectangle
Chose the type of rectangle you want to create:
1.Default rectangle(1.0x1.0)
2.Square(Enter side length)
3.Rectangle(Enter Length and width)
Enter your choice(1,2 or 3):
2
Enter the side length of square:
6
Length:6.0,Width:6.0
Area:36.0
```



**PROGRAM 3:****Source Code:**

```
import java.util.Scanner;

class Distance{

    private int feet;

    private int inches;

    public Distance(int feet,int inches){

        this.feet=feet;

        this.inches=inches;

    }

    public static Distance addDistances(Distance d1,Distance d2){

        int totalFeet=d1.feet+d2.feet;

        int totalInches=d1.inches+d2.inches;

        if(totalInches>=12){

            totalFeet+=totalInches/12;

            totalInches=totalInches%12;

        }

        return new Distance(totalFeet,totalInches);

    }

    public void display(){

        System.out.println(feet+" feet\t"+inches+" inches");

    }

    public static void main(String[]args){

        Scanner scan=new Scanner(System.in);

        System.out.println("Enter the feet for the first distance:");

        int feet1=scan.nextInt();

        System.out.println("Enter the inches for the first distance:");

        int inches1=scan.nextInt();

        System.out.println("Enter the feet for the second distance:");
```

```
int feet2=scan.nextInt();  
System.out.println("Enter the inches for the second distance:");  
int inches2=scan.nextInt();  
  
Distance d1=new Distance(feet1,inches1);  
Distance d2=new Distance(feet2,inches2);  
System.out.println("Distance 1:");  
d1.display();  
System.out.println("Distance 2:");  
d2.display();  
Distance sum=Distance.addDistances(d1,d2);  
System.out.println("Sum of Distances:");  
sum.display();  
scan.close();  
}  
}
```

**OUTPUT:**

```
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>java Distance
Enter the feet for the first distance:
3
Enter the inches for the first distance:
6
Enter the feet for the second distance:
5
Enter the inches for the second distance:
7
Distance 1:
3feet6inches
Distance 2:
5feet7inches
Sum of Distances:
9feet1inches
```





**PROGRAM 4:****Source Code:**

```
import java.util.Scanner;

class Fruit {
    void eat() {
        System.out.println("This fruit can be eaten.");
    }
}

class Citrus extends Fruit {
    void vitaminC() {
        System.out.println("Citrus fruits are rich in vitamin C.");
    }
}

class Orange extends Citrus {
    void orangeSpecific() {
        System.out.println("This is Orange.");
    }
}

class Berry extends Fruit {
    void berryType() {
        System.out.println("This is a berry.");
    }
}

interface Tropical {
    void tropicalFruit();
}

interface Edible {
    void isEdible();
}
```

```
class Mango extends Fruit implements Tropical, Edible {  
    @Override  
    public void tropicalFruit() {  
  
        System.out.println("Mango is a tropical fruit.");  
    }  
    @Override  
    public void isEdible() {  
        System.out.println("Mango can be eaten.");  
    }  
    void mangoSpecific() {  
        System.out.println("This is a Mango.");  
    }  
}  
  
public class FruitExample1 {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        int choice;  
        do {  
            System.out.println("\nSelect a fruit type or exit:");  
            System.out.println("1. Citrus (Single Inheritance)");  
            System.out.println("2. Orange (Multilevel Inheritance)");  
            System.out.println("3. Berry (Hierarchical Inheritance)");  
            System.out.println("4. Mango (Multiple Inheritance)");  
            System.out.println("5. Exit");  
            System.out.println("\nEnter your choice (1-5):");  
            choice = scanner.nextInt();  
            switch (choice) {  
                case 1:
```

```
System.out.println("Single Inheritance:");  
Citrus citrus = new Citrus();  
citrus.eat();  
citrus.vitaminC();  
break;  
case 2:  
System.out.println("Multilevel Inheritance:");  
Orange orange = new Orange();  
orange.eat();  
orange.vitaminC();  
orange.orangeSpecific();  
break;  
case 3:  
System.out.println("Hierarchical Inheritance:");  
Berry berry = new Berry();  
berry.eat();  
berry.berryType();  
break;  
case 4:  
System.out.println("Multiple Inheritance:");  
Mango mango = new Mango();  
mango.eat();  
mango.tropicalFruit();  
mango.isEdible();  
mango.mangoSpecific();  
break;  
case 5:  
System.out.println("Exiting...");  
break;
```

```
        default:
            System.out.println("Invalid choice! Please select a number between 1 and 5.");
            break;
    }
} while (choice != 5);
scanner.close();
}
}
```

## OUTPUT:

```
C:\Windows\System32\cmd.exe - java FruitExample1
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>java FruitExample1

Select a fruit type or exit:
1. Citrus (Single Inheritance)
2. Orange (Multilevel Inheritance)
3. Berry (Hierarchical Inheritance)
4. Mango (Multiple Inheritance)
5. Exit

Enter your choice (1-5):
1
Single Inheritance:
This fruit can be eaten.
Citrus fruits are rich in vitamin C.

Select a fruit type or exit:
1. Citrus (Single Inheritance)
2. Orange (Multilevel Inheritance)
3. Berry (Hierarchical Inheritance)
4. Mango (Multiple Inheritance)
5. Exit

Enter your choice (1-5):
3
Hierarchical Inheritance:
This fruit can be eaten.
This is a berry.

Select a fruit type or exit:
1. Citrus (Single Inheritance)
2. Orange (Multilevel Inheritance)
3. Berry (Hierarchical Inheritance)
4. Mango (Multiple Inheritance)
5. Exit

Enter your choice (1-5):
2
Multilevel Inheritance:
This fruit can be eaten.
Citrus fruits are rich in vitamin C.
This is Orange.

Select a fruit type or exit:
1. Citrus (Single Inheritance)
2. Orange (Multilevel Inheritance)
3. Berry (Hierarchical Inheritance)
4. Mango (Multiple Inheritance)
5. Exit

Enter your choice (1-5):
4
Multiple Inheritance:
This fruit can be eaten.
Mango is a tropical fruit.
Mango can be eaten.
This is a Mango.
```



## PROGRAM 5

### Source Code:

```
import java.io.*;

class Fruit {
    void describe() {
        System.out.println("This is a fruit");
    }
}

class Apple extends Fruit {
    @Override
    void describe() {
        System.out.println("This is an Apple. It is sweet and juicy.");
    }
}

class Orange extends Fruit {
    @Override
    void describe() {
        System.out.println("This is an Orange. It is sour and tangy.");
    }
}


class Banana extends Fruit {
    @Override
    void describe() {
        System.out.println("This is a Banana. It is soft and sweet.");
    }
}

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

```
Fruit myFruit;  
myFruit = new Apple();  
myFruit.describe();  
myFruit = new Orange();  
  
myFruit.describe();  
myFruit = new Banana();  
myFruit.describe();  
    }  
}
```



### **OUTPUT:**

 C:\Windows\System32\cmd.exe

```
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>javac Poly.java
```

```
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>java Poly
```

```
This is an Apple. It is sweet and juicy.
```

```
This is an Orange. It is sour and tangy.
```

```
This is a Banana. It is soft and sweet.
```





**PRORGAM 6:****Source Code:**

```
import java.util.LinkedList;
import java.util.Queue;

public class RestaurantOrderProcessing2 {
    public static void main(String[] args) {
        OrderQueue orderQueue = new OrderQueue();
        Thread producer = new Thread(new Customer(orderQueue));
        Thread consumer = new Thread(new Kitchen(orderQueue));
        producer.start();
        consumer.start();
    }
}

class OrderQueue {
    private final Queue<String> orders = new LinkedList<>();
    private final int capacity = 5;
    public synchronized void placeOrder(String order) {
        while (orders.size() == capacity) {
            try {
                wait();
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
        orders.add(order);
        System.out.println("Order placed: " + order);
        notify();
    }
}
```

```
public synchronized String processOrder() {
    while (orders.isEmpty()) {
        try {
            wait();

        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
    String order = orders.poll();
    notify();
    return order;
}

class Customer implements Runnable {
    private final OrderQueue orderQueue;
    public Customer(OrderQueue orderQueue) {
        this.orderQueue = orderQueue;
    }
    @Override
    public void run() {
        String[] orders = {"Pizza", "Burger", "Pasta", "Salad", "Soda"};
        for (String order : orders) {
            orderQueue.placeOrder(order);
            try {
                Thread.sleep(500); // Fixed typo here
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }
}
```

```
    }  
}  
class Kitchen implements Runnable {  
    private final OrderQueue orderQueue;  
    public Kitchen(OrderQueue orderQueue) {  
  
        this.orderQueue = orderQueue;  
    }  
    @Override  
    public void run() {  
        while (true) {  
            String order = orderQueue.processOrder();  
            System.out.println("Processing order: " + order);  
            try {  
                Thread.sleep(1000);  
            } catch (InterruptedException e) {  
                Thread.currentThread().interrupt();  
            }  
        }  
    }  
}
```

**OUTPUT:**

```
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>java RestaurantOrderProcessing2
Order placed: Pizza
Processing order: Pizza
Order placed: Burger
Processing order: Burger
Order placed: Pasta
Order placed: Salad
Processing order: Pasta
Order placed: Soda
Processing order: Salad
Processing order: Soda
```







## PROGRAM 7

### Source Code:

```
import java.util.Scanner;
import java.util.InputMismatchException;
class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
}
public class ExceptionTypesDemo { public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in); try {
        System.out.print("Enter your age (0-120): "); int age = scanner.nextInt();
        checkAge(age);
    } catch (InvalidAgeException e) {
        System.out.println("Checked Exception: " + e.getMessage());
    } catch (InputMismatchException e) {
        System.out.println("Checked Exception: Please enter a valid integer.");
        scanner.next();
    }
    try {
        System.out.println("Enter two numbers to divide (numerator denominator): ");
        int numerator = scanner.nextInt();
        int denominator = scanner.nextInt();
        int result = numerator / denominator;
        System.out.println("Result of division: " + result);
    } catch (ArithmeticException e) {
        System.out.println("Unchecked Exception: Cannot divide by zero.");
    } catch (InputMismatchException e) {
```

```
System.out.println("Unchecked Exception: Please enter valid integers.");
scanner.next();

}

try {
    int[] numbers = {1, 2, 3};

    System.out.print("Enter an index to access (0-2): ");

    int index = scanner.nextInt();

    System.out.println("Number at index " + index + ": " + numbers[index]);
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Unchecked Exception: Array index out of bounds.");
} catch (InputMismatchException e) {
    System.out.println("Unchecked Exception: Please enter a valid integer.");
    scanner.next();
}

try {
    System.out.print("Enter a number to find its square root: ");

    double num = scanner.nextDouble();

    if (num < 0) {
        throw new IllegalArgumentException("Cannot calculate square root of a negative
number.");
    }

    System.out.println("Square root: " + Math.sqrt(num));
} catch (IllegalArgumentException e) {
    System.out.println("Unchecked Exception: " + e.getMessage());
} catch (InputMismatchException e) {
    System.out.println("Unchecked Exception: Please enter a valid number.");
    scanner.next();
} finally {
    System.out.println("End of exception demonstration.");

    scanner.close();
}
```

```
}  
public static void checkAge(int age) throws InvalidAgeException {  
    if (age < 0 || age > 120) {  
        throw new InvalidAgeException("Age must be between 0 and 120.");  
    }  
    System.out.println("Valid age entered: " + age);  
}  
}
```

**OUTPUT:**

```
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>javac ExceptionTypesDemo.java
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>java ExceptionTypesDemo
Enter your age (0-120): 23
Valid age entered: 23
Enter two numbers to divide (numerator denominator):
9 5
Result of division: 1
Enter an index to access (0-2): 1
Number at index 1: 2
Enter a number to find its square root: 56
Square root: 7.483314773547883
End of exception demonstration.
```





**PROGRAM 8:****Source Code:**

```
import java.util.*;

public class ExNo7 {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        TreeSet<Integer> treeSet = new TreeSet<>();
        Stack<String> stack = new Stack<>();
        while (true) {
            System.out.println("\n----- Menu ---- ");
            System.out.println("1. TreeSet Operations");
            System.out.println("2. Stack Operations");
            System.out.println("3. Exit");
            System.out.print("Choose an option: ");
            int choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    performTreeSetOperations(scanner, treeSet);
                    break;
                case 2:
                    performStackOperations(scanner, stack);
                    break;
                case 3:
                    System.out.println("Exiting.. ");
                    scanner.close();
                    return;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }
    }
}
```



```
    }  
}  
private static void performTreeSetOperations(Scanner scanner, TreeSet<Integer> treeSet)  
{  
  
    while (true) {  
        System.out.println("\n----- TreeSet Operations -----");  
        System.out.println("1. Add Element");  
        System.out.println("2. Check Element");  
        System.out.println("3. Remove Element");  
        System.out.println("4. Display First and Last");  
        System.out.println("5. Create Subset");  
        System.out.println("6. Go Back");  
        System.out.print("Choose an option: ");  
        int choice = scanner.nextInt();  
        switch (choice) {  
            case 1:  
                System.out.print("Enter an integer to add: ");  
                int num = scanner.nextInt();  
                treeSet.add(num);  
                System.out.println("Added " + num + " to TreeSet.");  
                System.out.println("Current TreeSet: " + treeSet);  
                break;  
            case 2:  
                System.out.print("Enter an integer to check: ");  
                int checkNum = scanner.nextInt();  
                System.out.println("TreeSet contains " + checkNum + "? " +  
treeSet.contains(checkNum));  
                break;  
            case 3:  
                System.out.print("Enter an integer to remove: ");  
                int removeNum = scanner.nextInt();
```

```
        treeSet.remove(removeNum);

        System.out.println("Removed " + removeNum + " from TreeSet. Current
TreeSet: " + treeSet);

        break;
    case 4:

        System.out.println("First element: " + (treeSet.isEmpty() ? "N/A" :
treeSet.first()));

        System.out.println("Last element: " + (treeSet.isEmpty() ? "N/A" :
treeSet.last()));

        break;
    case 5:

        System.out.print("Enter lower bound for subset: ");

        int lowerBound = scanner.nextInt();

        System.out.print("Enter upper bound for subset: ");

        int upperBound = scanner.nextInt();

        System.out.println("Subset from " + lowerBound + " to " + upperBound + ": " +
treeSet.subSet(lowerBound, upperBound));

        break;
    case 6:

        return;
    default:

        System.out.println("Invalid choice. Please try again.");
    }
}

}

private static void performStackOperations(Scanner scanner, Stack<String> stack) {
    while (true) {

        System.out.println("\n----- Stack Operations----- ");

        System.out.println("1. Push Element");

        System.out.println("2. Pop Element");

        System.out.println("3. Peek Element");

        System.out.println("4. Check if Stack is Empty");
```

```
System.out.println("5. Search Element");
System.out.println("6. Clear Stack");
System.out.println("7. Go Back");
System.out.print("Choose an option: ");
int choice = scanner.nextInt();
scanner.nextLine(); // Consume newline

switch (choice) {
    case 1:
        System.out.print("Enter a string to push: ");
        String str = scanner.nextLine();
        stack.push(str);
        System.out.println("Pushed " + str + " onto the stack.");
        System.out.println("Current Stack: " + stack);
        break;
    case 2:
        if (!stack.isEmpty()) {
            String poppedElement = stack.pop();
            System.out.println("Popped element: " + poppedElement);
        } else {
            System.out.println("Stack is empty, cannot pop.");
        }
        System.out.println("Current Stack: " + stack);
        break;
    case 3:
        if (!stack.isEmpty()) {
            String topElement = stack.peek();
            System.out.println("Top element: " + topElement);
        } else {
```

```
        System.out.println("Stack is empty, cannot peek.");
    }
    System.out.println("Current Stack: " + stack);
    break;

case 4:
    System.out.println("Is the stack empty? " + stack.isEmpty());
    break;
case 5:
    System.out.println("Current Stack: " + stack);
    System.out.print("Enter a string to search: ");
    String searchStr = scanner.nextLine();
    int position = stack.search(searchStr);
    System.out.println("Position of '" + searchStr + "' in stack: " + (position == -1 ?
    "Not found" : position));
    break;
case 6:
    stack.clear();
    System.out.println("Stack has been cleared.");
    break;
case 7:
    return;
default:
    System.out.println("Invalid choice. Please try again.");
}
}
}
}
```

**OUTPUT:**

```

C:\Windows\System32\cmd.exe - java ExNo7

C:\Program Files (x86)\Java\jdk1.7.0_65\bin>javac ExNo7.java
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>java ExNo7

----- Menu -----
1. TreeSet Operations
2. Stack Operations
3. Exit
Choose an option: 1

----- TreeSet Operations -----
1. Add Element
2. Check Element
3. Remove Element
4. Display First and Last
5. Create Subset
6. Go Back
Choose an option: 1
Enter an integer to add: 23
Added 23 to TreeSet.
Current TreeSet: [23]

----- TreeSet Operations -----
1. Add Element
2. Check Element
3. Remove Element
4. Display First and Last
5. Create Subset
6. Go Back
Choose an option: 1
Enter an integer to add: 45
Added 45 to TreeSet.
Current TreeSet: [23, 45]

----- TreeSet Operations -----
1. Add Element
2. Check Element
3. Remove Element
4. Display First and Last
5. Create Subset
6. Go Back
Choose an option: 1
Enter an integer to add: 10
Added 10 to TreeSet.
Current TreeSet: [10, 23, 45]

----- TreeSet Operations -----
1. Add Element
2. Check Element
3. Remove Element
4. Display First and Last
5. Create Subset
6. Go Back
Choose an option: 2
Enter an integer to check: 23
TreeSet contains 23? true

```

```

C:\Windows\System32\cmd.exe - java ExNo7

Choose an option: 1
Enter an integer to add: 10
Added 10 to TreeSet.
Current TreeSet: [10, 23, 45]

----- TreeSet Operations -----
1. Add Element
2. Check Element
3. Remove Element
4. Display First and Last
5. Create Subset
6. Go Back
Choose an option: 2
Enter an integer to check: 23
TreeSet contains 23? true

----- TreeSet Operations -----
1. Add Element
2. Check Element
3. Remove Element
4. Display First and Last
5. Create Subset
6. Go Back
Choose an option: 4
First element: 10
Last element: 45

----- TreeSet Operations -----
1. Add Element
2. Check Element
3. Remove Element
4. Display First and Last
5. Create Subset
6. Go Back
Choose an option: 5
Enter lower bound for subset: 56
Enter upper bound for subset: 89
Subset from 56 to 89: []

----- TreeSet Operations -----
1. Add Element
2. Check Element
3. Remove Element
4. Display First and Last
5. Create Subset
6. Go Back
Choose an option: 3
Enter an integer to remove: 45
Removed 45 from TreeSet. Current TreeSet: [10, 23]

----- TreeSet Operations -----
1. Add Element
2. Check Element
3. Remove Element
4. Display First and Last
5. Create Subset
6. Go Back
Choose an option: 4
First element: 10
Last element: 23

```

**STACK OPERATION:**

```
cmd C:\Windows\System32\cmd.exe - java ExNo7
```

```
----- Menu -----
1. TreeSet Operations
2. Stack Operations
3. Exit
Choose an option: 2

----- Stack Operations -----
1. Push Element
2. Pop Element
3. Peek Element
4. Check if Stack is Empty
5. Search Element
6. Clear Stack
7. Go Back
Choose an option: 1
Enter a string to push: 45
Pushed 45 onto the stack.
Current Stack: [45]

----- Stack Operations -----
1. Push Element
2. Pop Element
3. Peek Element
4. Check if Stack is Empty
5. Search Element
6. Clear Stack
7. Go Back
Choose an option: 1
Enter a string to push: 23
Pushed 23 onto the stack.
Current Stack: [45, 23]

----- Stack Operations -----
1. Push Element
2. Pop Element
3. Peek Element
4. Check if Stack is Empty
5. Search Element
6. Clear Stack
7. Go Back
Choose an option: 1
Enter a string to push: 45
Pushed 45 onto the stack.
Current Stack: [45, 23, 45]

----- Stack Operations -----
1. Push Element
2. Pop Element
3. Peek Element
4. Check if Stack is Empty
5. Search Element
6. Clear Stack
7. Go Back
Choose an option: 3
Top element: 45
Current Stack: [45, 23, 45]
```

C:\Windows\System32\cmd.exe - java ExNo7

Pushed 45 onto the stack.

Current Stack: [45, 23, 45]

----- Stack Operations -----

1. Push Element
2. Pop Element
3. Peek Element
4. Check if Stack is Empty
5. Search Element
6. Clear Stack
7. Go Back

Choose an option: 3

Top element: 45

Current Stack: [45, 23, 45]

----- Stack Operations -----

1. Push Element
2. Pop Element
3. Peek Element
4. Check if Stack is Empty
5. Search Element
6. Clear Stack
7. Go Back

Choose an option: 4

Is the stack empty? false

----- Stack Operations -----

1. Push Element
2. Pop Element
3. Peek Element
4. Check if Stack is Empty
5. Search Element
6. Clear Stack
7. Go Back

Choose an option: 5

Current Stack: [45, 23, 45]

Enter a string to search: 23

Position of '23' in stack: 2

----- Stack Operations -----

1. Push Element
2. Pop Element
3. Peek Element
4. Check if Stack is Empty
5. Search Element
6. Clear Stack
7. Go Back

Choose an option: 2

Popped element: 45

Current Stack: [45, 23]

----- Stack Operations -----

1. Push Element
2. Pop Element
3. Peek Element
4. Check if Stack is Empty
5. Search Element
6. Clear Stack
7. Go Back

Choose an option: 6

Stack has been cleared.





**PROGRAM 9:****Source Code:**

```
import java.util.ArrayList;
import java.util.List;
interface LibraryItem {
    String getTitle();
    boolean isAvailable();
}
class Book implements LibraryItem {
    private String title;
    private boolean available;
    public Book(String title, boolean available) {
        this.title = title;
        this.available = available;
    }
    @Override
    public String getTitle() {
        return title;
    }
    @Override
    public boolean isAvailable() {
        return available;
    }
}
class Magazine implements LibraryItem {
    private String title;
    private boolean available;
    public Magazine(String title, boolean available) {
        this.title = title;
```

```
        this.available = available;
    }

    @Override

    public String getTitle() {
        return title;
    }

    @Override
    public boolean isAvailable() {
        return available;
    }
}

class Library {
    private List<LibraryItem> items;

    public Library() {
        items = new ArrayList<>();
    }

    public void addItem(LibraryItem item) {
        items.add(item);
    }

    public void displayItems() {
        for (LibraryItem item : items) {
            System.out.println("Title: " + item.getTitle() + ", Available: " + item.isAvailable());
        }
    }
}

public class LibraryManagement {
    public static void main(String[] args) {
        Library library = new Library();

        LibraryItem book1 = new Book("The Great Gatsby", true);

        LibraryItem book2 = new Book("1984", false);
    }
}
```

```
LibraryItem magazine1 = new Magazine("National Geographic", true);
LibraryItem magazine2 = new Magazine("TIME", false);
library.addItem(book1);

library.addItem(book2);
library.addItem(magazine1);
library.addItem(magazine2);
System.out.println("Library Items:");
library.displayItems();
}
}
```

**OUTPUT:**

```
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>javac LibraryManagement.java  
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>java LibraryManagement  
Library Items:  
Title: The Great Gatsby, Available: true  
Title: 1984, Available: false  
Title: National Geographic, Available: true  
Title: TIME, Available: false
```



**PROGRAM 10:****Source Code:**

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class WordCount {
    public static void main(String[] args) {
        BufferedReader reader = null;
        Scanner scanner = new Scanner(System.in);
        int charCount = 0;
        int wordCount = 0;
        int lineCount = 0;
        System.out.print("Enter the path of the file: ");
        String filePath = scanner.nextLine();
        try {
            reader = new BufferedReader(new FileReader(filePath))
            String currentLine = reader.readLine();
            while (currentLine != null) {
                lineCount++;
                String[] words = currentLine.split("\\s+"); // Split by whitespace
                wordCount += words.length;
                for (String word : words) {
                    charCount += word.length();
                }
                currentLine = reader.readLine();
            }
            System.out.println("\nNumber Of Chars In A File: " + charCount);
            System.out.println("Number Of Words In A File: " + wordCount);
        }
```

```
        System.out.println("Number Of Lines In A File: " + lineCount);
    }
    catch (IOException e) {

        e.printStackTrace();
    }
    finally {
        try {
            if (reader != null) {
                reader.close();
            }
            scanner.close();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

**OUTPUT:**

```
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>javac WordCount.java  
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>java WordCount  
Enter the path of the file: C:\Program Files (x86)\Java\jdk1.7.0_65\bin\WordCount.txt  
Number Of Chars In A File: 1318  
Number Of Words In A File: 274  
Number Of Lines In A File: 67
```





**PROGRAM 11:****Source Code:**

```
import java.io.IOException;
import java.nio.file.DirectoryStream;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Scanner;

public class FileLister {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the file pattern (e.g., *.txt): ");
        String pattern = scanner.nextLine();
        Path currentDir = Paths.get(System.getProperty("user.dir"));
        try (DirectoryStream<Path> stream = Files.newDirectoryStream(currentDir, pattern)) {
            System.out.println("Files matching pattern " + pattern + ":");
            for (Path entry : stream) {
                System.out.println(entry.getFileName());
            }
        } catch (IOException e) {
            System.err.println("Error reading directory: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

**OUTPUT:**

```
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>javac FileLister.java
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>java FileLister
Enter the file pattern (e.g., *.txt): *.exe
Files matching pattern *.exe:
apt.exe
extcheck.exe
FruitExample1.exe
idlj.exe
jabswitch.exe
jar.exe
jarsigner.exe
java-rmi.exe
java.exe
javac.exe
javadoc.exe
javafxpackager.exe
javah.exe
javap.exe
javaw.exe
javaws.exe
jcmd.exe
jconsole.exe
jdb.exe
jhat.exe
jinfo.exe
jmap.exe
jmc.exe
jps.exe
jrunscript.exe
jsadebugd.exe
jstack.exe
jstat.exe
jstatd.exe
jvisualvm.exe
keytool.exe
kinit.exe
klist.exe
ktab.exe
native2ascii.exe
orbd.exe
pack200.exe
packager.exe
policytool.exe
rmic.exe
rmid.exe
rmiregistry.exe
schemagen.exe
serialver.exe
servertool.exe
tnameserv.exe
unpack200.exe
wsagen.exe
wsimport.exe
xjc.exe
```



**PROGRAM 12:****Source Code:**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.regex.Pattern;

public class StudentRegistrationForm extends JFrame implements ActionListener {
    private JTextField nameField, ageField, emailField, phoneField;
    private JComboBox<String> genderBox, courseBox;
    private JTextArea addressArea;
    private JButton submitButton;
    private JLabel resultLabel;

    public StudentRegistrationForm() {
        setTitle("Student Registration Form");
        setSize(500, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new GridLayout(8, 2, 10, 10));
        add(new JLabel("Name:"));
        nameField = new JTextField();
        add(nameField);
        add(new JLabel("Age:"));
        ageField = new JTextField();
        add(ageField);
        add(new JLabel("Gender:"));
        String[] genders = {"Select", "Male", "Female", "Other"};
        genderBox = new JComboBox<>(genders);
        add(genderBox);
        add(new JLabel("Address:"));
        addressArea = new JTextArea(3, 5);
```

```
add(new JScrollPane(addressArea));
add(new JLabel("Email:"));
emailField = new JTextField();

add(emailField);
add(new JLabel("Phone:"));
phoneField = new JTextField();
add(phoneField);
add(new JLabel("Course:"));
String[] courses = {"Select", "Computer Science", "Business", "Engineering", "Arts",
"Law"};
courseBox = new JComboBox<>(courses);
add(courseBox);
submitButton = new JButton("Submit");
submitButton.addActionListener(this);
add(submitButton);
resultLabel = new JLabel();
add(resultLabel);
setVisible(true);
}
@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == submitButton) {
        if (validateFields()) {
            resultLabel.setText("Registration successful!");
        } else {
            resultLabel.setText("Please correct the errors.");
        }
    }
}
```

```
private boolean validateFields() {  
    String name = nameField.getText().trim();  
    if (name.isEmpty()) {  
        JOptionPane.showMessageDialog(this, "Name cannot be empty.");  
        return false;  
    }  
    String ageText = ageField.getText().trim();  
    if (!ageText.matches("\\d+")) {  
        JOptionPane.showMessageDialog(this, "Age must be a number.");  
        return false;  
    }  
    if (genderBox.getSelectedIndex() == 0) {  
        JOptionPane.showMessageDialog(this, "Please select your gender.");  
        return false;  
    }  
    String address = addressArea.getText().trim();  
    if (address.isEmpty()) {  
        JOptionPane.showMessageDialog(this, "Address cannot be empty.");  
        return false;  
    }  
    String email = emailField.getText().trim();  
    if (!isValidEmail(email)) {  
        JOptionPane.showMessageDialog(this, "Invalid email format.");  
        return false;  
    }  
    String phone = phoneField.getText().trim();  
    if (!phone.matches("\\d{10}")) {  
        JOptionPane.showMessageDialog(this, "Phone number must be 10 digits.");  
        return false;  
    }  
}
```

```
    }  
    if (courseBox.getSelectedIndex() == 0) {  
        JOptionPane.showMessageDialog(this, "Please select a course.");  
        return false;  
    }  
    return true;  
  
}  
private boolean isValidEmail(String email) {  
    String emailRegex = "^[a-zA-Z0-9_+&*-.](?:\\.[a-zA-Z0-9_+&*-.])*@" +  
        "(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,7}$";  
    Pattern pattern = Pattern.compile(emailRegex);  
    return pattern.matcher(email).matches();  
}  
public static void main(String[] args) {  
    new StudentRegistrationForm();  
}  
}
```



**OUTPUT:**

The screenshot shows a Java Swing window titled "Student Registration Form". The window contains a registration form with the following fields and values:

- Name: abc
- Age: 12
- Gender: Female (selected from a dropdown menu)
- Address: ghjkhgfgfh
- Email: edc@gmail.com
- Phone: 7457828234
- Course: Business (selected from a dropdown menu)

At the bottom left of the form is a "Submit" button. At the bottom right, a message "Registration successful!" is displayed.





**PROGRAM 13:****Source Code:**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class StudentDatabaseApp {
    private static JTextField nameField, rollNoField, classField, mobileField, addressField;
    private static JRadioButton maleButton, femaleButton;
    private static JCheckBox[] languageCheckBoxes;
    private static JCheckBox[] subjectCheckBoxes;
    private static JTextArea outputArea;
    public static void main(String[] args) {
        JFrame frame = new JFrame("Student Database");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(500, 600);
        JPanel inputPanel = new JPanel();
        inputPanel.setLayout(new GridLayout(10, 2));
        inputPanel.add(new JLabel("Name:"));
        nameField = new JTextField();
        inputPanel.add(nameField);
        inputPanel.add(new JLabel("Roll No:"));
        rollNoField = new JTextField();
        inputPanel.add(rollNoField);
        inputPanel.add(new JLabel("Class:"));
        classField = new JTextField();
        inputPanel.add(classField);
        inputPanel.add(new JLabel("Mobile No:"));
        mobileField = new JTextField();
```

```
inputPanel.add(mobileField);

inputPanel.add(new JLabel("Address:"));

addressField = new JTextField();
inputPanel.add(addressField); inputPanel.add(new
JLabel("Gender:")); maleButton = new
JRadioButton("Male"); femaleButton = new
JRadioButton("Female"); ButtonGroup genderGroup =
new ButtonGroup(); genderGroup.add(maleButton);
genderGroup.add(femaleButton);
inputPanel.add(maleButton);
inputPanel.add(femaleButton);

inputPanel.add(new JLabel("Date of Birth (DD/MM/YYYY:"));
JTextField dobField = new JTextField();
inputPanel.add(dobField);

inputPanel.add(new JLabel("Languages Known:"));
String[] languages = { "English", "Hindi", "Spanish", "French", "German" };
languageCheckBoxes = new JCheckBox[languages.length];
for (int i = 0; i < languages.length; i++) {
    languageCheckBoxes[i] = new JCheckBox(languages[i]);
    inputPanel.add(languageCheckBoxes[i]);
}
inputPanel.add(new JLabel("Subjects Studied:"));
String[] subjects = { "Math", "Science", "History", "Geography", "Computer Science" };
subjectCheckBoxes = new JCheckBox[subjects.length];
for (int i = 0; i < subjects.length; i++) {
    subjectCheckBoxes[i] = new JCheckBox(subjects[i]);
    inputPanel.add(subjectCheckBoxes[i]);
}

JButton submitButton = new JButton("Submit");
inputPanel.add(submitButton);
```

```
outputArea = new JTextArea(10, 40);

outputArea.setEditable(false);
JScrollPane outputScrollPane = new JScrollPane(outputArea);
inputPanel.add(outputScrollPane);
submitButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        submitForm();
    }
});
frame.add(inputPanel);
frame.setVisible(true);
}

private static void submitForm() {
    String name = nameField.getText().trim();
    String rollNo = rollNoField.getText().trim();
    String className = classField.getText().trim();
    String mobile = mobileField.getText().trim();
    String address = addressField.getText().trim();
    String gender = maleButton.isSelected() ? "Male" : "Female";
    StringBuilder languagesKnown = new StringBuilder();
    for (JCheckBox cb : languageCheckBoxes) {
        if (cb.isSelected()) {
            languagesKnown.append(cb.getText()).append(" ");
        }
    }
    StringBuilder subjectsStudied = new StringBuilder();
    for (JCheckBox cb : subjectCheckBoxes) {
        if (cb.isSelected()) {
            subjectsStudied.append(cb.getText()).append(" ");
        }
    }
}
```

```
    }

    }
    outputArea.setText("Student Information:\n");
    outputArea.append("Name: " + name + "\n");
    outputArea.append("Roll No: " + rollNo + "\n");
    outputArea.append("Class: " + className + "\n");
    outputArea.append("Mobile No: " + mobile + "\n");
    outputArea.append("Address: " + address + "\n");
    outputArea.append("Gender: " + gender + "\n");
    outputArea.append("Languages Known: " + languagesKnown.toString().trim() + "\n");
    outputArea.append("Subjects Studied: " + subjectsStudied.toString().trim() + "\n");
    }
}
```

## OUTPUT:

Student Database

Name:	abc	Roll No:
34	Class:	A
Mobile No:	2345678910	Address:
#24,Narasimhpura	Gender:	<input type="radio"/> Male
<input type="radio"/> Female	Date of Birth (DD/MM/YYYY):	13/12/1998
Languages Known:	<input checked="" type="checkbox"/> English	<input type="checkbox"/> Hindi
<input type="checkbox"/> Spanish	<input checked="" type="checkbox"/> French	<input type="checkbox"/> German
Subjects Studied:	<input checked="" type="checkbox"/> Math	<input type="checkbox"/> Science
<input type="checkbox"/> History	<input type="checkbox"/> Geography	<input type="checkbox"/> Computer Science
Submit	Student Information: Name: abc Roll No: 34 Class: A Mobile No: 2345678910 Address: #24,Narasimhpura	







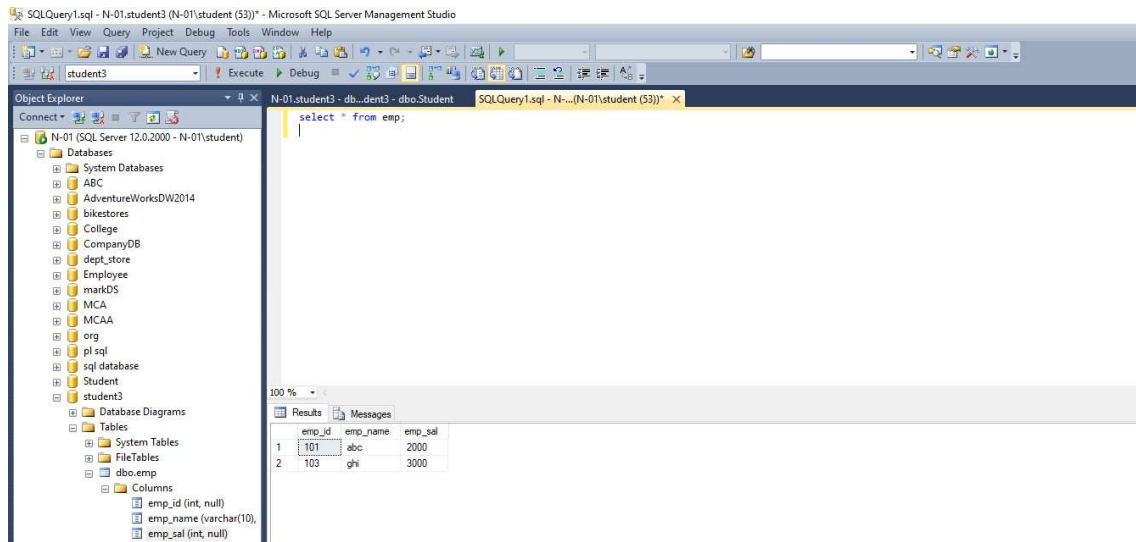


**PROGRAM 14:****Source Code:**

```
import java.sql.*; public
class Prgm14 {
    public static void main(String[] args) {
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection con = DriverManager.getConnection("jdbc:odbc:student3", "", "");
            System.out.println("Connected");
            Statement st = con.createStatement();
            //st.executeUpdate("drop table emp");
            st.executeUpdate("create table emp(emp_id int, emp_name varchar(10), emp_sal
int)");
            System.out.println("Table created");
            st.executeUpdate("insert into emp values(101,'abc',1000)");
            st.executeUpdate("insert into emp values(102,'def',2000)");
            st.executeUpdate("insert into emp values(103,'ghi',3000)");
            System.out.println("Inserted 3 rows");
            ResultSet rs = st.executeQuery("select * from emp");
            while(rs.next()){
                System.out.println(rs.getInt(1));
                System.out.println(rs.getString(2));
                System.out.println(rs.getInt(3));
            }
            st.executeUpdate("update emp set emp_sal = 2000 where emp_name = 'abc'");
            System.out.println("\n After Update \n");
            ResultSet rs1 = st.executeQuery("select * from emp");
            while(rs1.next()){
                System.out.println(rs1.getInt(1));
                System.out.println(rs1.getString(2));
```

```
        System.out.println(rs1.getInt(3));
    }
    st.executeUpdate("Delete from emp where emp_name = 'def'");

    System.out.println("\nAfter Delete\n");
    ResultSet rs2 = st.executeQuery("select * from emp");
    while(rs2.next()){
        System.out.println(rs2.getInt(1));
        System.out.println(rs2.getString(2));
        System.out.println(rs2.getInt(3));
    }
    st.close();
    con.close();
}
catch(Exception e){
    System.out.println(e.getMessage());
}
}
}
```

**OUTPUT:**

```
C:\Program Files (x86)\Java\jdk1.7.0_65\bin>java Prgm15
Connected
Table created
Inserted 3 rows
101
abc
1000
102
def
2000
103
ghi
3000

After Update

101
abc
2000
102
def
2000
103
ghi
3000

After Delete

101
abc
2000
103
ghi
3000
```

