

Experiment No: 2.1

Student Name: Rajdeep Majumder
Branch: CSE
Semester: 6th
Subject Name: Java LAB

UID: 21BCS5895
Section: 21BCS CC FL 602 - B
Date of Performance: 06.02.24
Subject Code: 21CSH - 319

1. Aim: Write a program to collect and store all the cards to assist the users in finding all the cards in a given symbol.

2. Objective: This cards game consists of N number of cards. Get N number of cards details from the user and store the values in Card object with the attributes symbol and number. Store all the cards in a map with symbol as its key and list of cards as its value.

Map is used here to easily group all the cards based on their symbol. Once all the details are captured print all the distinct symbols in alphabetical order from the Map. For each symbol print all the card details, number of cards and their sum respectively.

3. Code:

```
import java.util.*;
//import java.util.ArrayList;
//import java.util.Map;
//import java.util.Scanner;
//import java.util.TreeMap;

public class Cards
{
    public static void main(String[] args)
    {
        Map<String, ArrayList<Integer>> m = new TreeMap<>();
```

```
Scanner sc = new Scanner(System.in);
System.out.print("Enter Number of Cards: ");
int n = sc.nextInt();
String s;
for(int i=0; i<n; i++)
{
    System.out.println("Enter Card " + (i+1));
    s = sc.next();
    if (!m.containsKey(s))
    {
        m.put(s, new ArrayList<>());
    }
    m.get(s).add(sc.nextInt());
}
System.out.println("Distinct Symbols are: ");
for(String k: m.keySet())
{
    System.out.print(k + " ");
}
int sum=0;
System.out.println();
for(Map.Entry<String, ArrayList<Integer>> mv: m.entrySet())
{
    System.out.println("Cards in " + mv.getKey() + " Symbol");
    for (Integer integer : mv.getValue())
    {
        System.out.println(mv.getKey() + " " + integer);
        sum += integer;
    }
    System.out.println("Number of cards: " + mv.getValue().size());
    System.out.println("Sum of numbers: " + sum);
    sum=0;
}
sc.close();
}
```

4. Result/Output:

```
3
Enter Card 5
c
4
Enter Card 6
d
5
Enter Card 7
h
6
Enter Card 8
s
7
Enter Card 9
c
8
Enter Card 10
d
9
Distinct Symbols are:
cdhs
Cards in c Symbol
c0
c4
c8
Number of cards: 3
Sum of numbers: 12
Cards in d Symbol
d1
d5
d9
Number of cards: 3
Sum of numbers: 15
Cards in h Symbol
h2
h6
Number of cards: 2
Sum of numbers: 8
Cards in s Symbol
s3
s7
Number of cards: 2
Sum of numbers: 10

...Program finished with exit code 0
Press ENTER to exit console.█
```

Experiment No: 2.2

Student Name: Rajdeep Majumder
Branch: CSE
Semester: 6th
Subject Name: Java LAB

UID: 21BCS5895
Section: 21BCS CC FL 602 - B
Date of Performance: 20.02.24
Subject Code: 21CSH - 319

1. Aim: Write a program to create Card class with attributes symbol and number.

2. Objective: Playing cards during travel is a fun filled experience. For this game they wanted to collect all four unique symbols. Can you help these guys to collect unique symbols from a set of cards?

Create Card class with attributes symbol and number. From our main method collect each card details (symbol and number) from the user. Collect all these cards in a set, since set is used to store unique values or objects. Once we collect all four different symbols display the first occurrence of card details in alphabetical order

3. Code:

```
import java.util.*;
class Card implements Comparable<Card>
{
    private char symbol;
    private int number;
    public Card() {}
    public Card(char symbol, int number)
    {
        super();
        this.symbol = symbol;
        this.number = number;
    }
    public char getSymbol()
```

```
{
    return symbol;
}
public void setSymbol(char symbol)
{
    this.symbol = symbol;
}
public int getNumber()
{
    return number;
}
public void setNumber(int number)
{
    this.number = number;
}
public String toString()
{
    return "Card [symbol=" + symbol + ", number=" + number + "];"
}
public int compareTo(Card o)
{
    if (this.symbol > o.symbol) return -1;
    else if (this.symbol < o.symbol) return 1;
    else return 0;
}
public int hashCode()
{
    return String.valueOf(symbol).hashCode();
}
public boolean equals(Object obj)
{
    if (obj instanceof Card)
    {
        Card card = (Card) obj;
        return (card.symbol == this.symbol);
    }
}
```

```
    }
    else
    {
        return false;
    }
}
}

public class Exp22
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        Set<Card> set = new HashSet<Card>();
        for (int i=0; i<8; i++)
        {
            System.out.println("Enter a card:");
            Card card = new Card();
            card.setSymbol(sc.nextLine().charAt(0));
            card.setNumber(sc.nextInt());
            sc.nextLine();
            set.add(card);
        }
        System.out.println("Four symbols gathered in eight cards.");
        System.out.println("Cards in Set are:");
        for (Card card : set)
            System.out.println(card.getSymbol() + " " +card.getNumber());
        sc.close();
    }
}
```

4. Result/Output:

```
Enter a card:
a
1
Enter a card:
a
2
Enter a card:
b
3
Enter a card:
b
4
Enter a card:
c
5
Enter a card:
c
6
Enter a card:
d
7
Enter a card:
d
8
Four symbols gathered in eight cards.
Cards in Set are:
a 1
b 3
c 5
d 7

...Program finished with exit code 0
Press ENTER to exit console.
```



DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment No: 2.3

Name: Rajdeep Majumder

UID: 21BCS5895

Branch: BE CSE

Section: 21BCS CC FL 602 B

Semester: 6th

Date: 05.03.2024

Subject: PBLJ With LAB

Subject Code: 21CSP 351

Aim:

Write a Program to perform the basic operations like insert, delete, display and search in list. List contains String object items where these operations are to be performed.

Code:

```
import java.util.*;

public class experiment {

    private static void insert(ArrayList<String> list, String item){
        list.add(item);
        System.out.println(item + " inserted successfully.");
    }

    private static void delete(ArrayList<String> list, String item){
        if (list.remove(item))
            System.out.println(item + " deleted successfully.");
        else
            System.out.println(item + " not found in the list.");
    }

    private static void display(ArrayList<String> list){
        if (list.isEmpty()){
            System.out.println("List is empty.");
        } else {
            System.out.println("List contents:");
            for (String item : list)
                System.out.println(item);
        }
    }

    private static void search(ArrayList<String> list, String item){
```




DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        if (list.contains(item))
            System.out.println(item + " found in the list.");
        else
            System.out.println(item + " not found in the list.");
    }
    public static void main(String[] args){
        ArrayList<String> stringList = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);
        boolean running = true;
        while (running){
            System.out.println("\nSelect an option:");
            System.out.println("1. Insert");
            System.out.println("2. Delete");
            System.out.println("3. Display");
            System.out.println("4. Search");
            System.out.println("5. Exit");
            int choice = scanner.nextInt();
            scanner.nextLine();
            switch (choice){
                case 1:
                    System.out.println("Enter the string to insert:");

                    String insertString = scanner.nextLine();
                    insert(stringList, insertString);
                    break;
                case 2:
                    System.out.println("Enter the string to delete:");

                    String deleteString = scanner.nextLine();
                    delete(stringList, deleteString);
                    break;
                case 3:
                    display(stringList);
                    break;
```

```
        case 4:
            System.out.println("Enter the string to search:");

            String searchString = scanner.nextLine();
            search(stringList, searchString);
            break;
        case 5:
            running = false;
            System.out.println("Exiting...");
            break;
        default:
            System.out.println("Invalid choice!"); }
    }
    scanner.close(); }
```

Output:

```
Select an option:
1. Insert
2. Delete
3. Display
4. Search
5. Exit
1
Enter the string to insert:
Hello
Hello inserted successfully.
```

```
Select an option:
1. Insert
2. Delete
3. Display
4. Search
5. Exit
1
Enter the string to insert:
Rajdeep
Rajdeep inserted successfully.
```

```
Select an option:
1. Insert
2. Delete
3. Display
4. Search
5. Exit
3
List contents:
Hello
Rajdeep
```

```
Select an option:
1. Insert
2. Delete
3. Display
4. Search
5. Exit
4
Enter the string to search:
Rajdeep
Rajdeep found in the list.
```

```
Select an option:
1. Insert
2. Delete
3. Display
4. Search
5. Exit
2
Enter the string to delete:
Rajdeep
Rajdeep deleted successfully.
```

```
Select an option:
1. Insert
2. Delete
3. Display
4. Search
5. Exit
3
List contents:
Hello
```

```
Select an option:
1. Insert
2. Delete
3. Display
4. Search
5. Exit
5
Exiting...
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```

Experiment No: 2.4

Student Name: Rajdeep Majumder

Branch: CSE

Semester: 6th

Subject: Project Based Learning in Java

UID: 21BCS5895

Section: 21BCS CC FL 602 B

Date of Performance: 05-03-24

Subject Code: 21CSH - 319

- **Aim:** Write a menu-based program to create a Java Application 'Employee Management System'.
- **Objective:** To create an application that should gather details of the employee like employee name, employee id, designation and salary and store it in a file. The application should display all the employee details
- **Input/Apparatus Used:** Virtual Studio Code.
- **Procedure/Algorithm/Pseudocode:**
 1. **Define the Employee Class:**
 - Define a class named Employee with private fields id, name, age, and salary.
 - Implement a constructor to initialize the Employee object with the provided values.
 - Override the toString() method to return a string representation of the Employee object.
 2. **Create the EmployeeManagementSystem Class:**
 - Declare a constant FILE_NAME to store the name of the file where employee details will be stored.
 - Create a static Scanner object named scanner to read user input from the console.
 3. **Implement the Main Method:**
 - Declare a list to store Employee objects named employees.
 - Initialize an integer variable choice to store the user's menu choice.
 - Enter a do-while loop to display the main menu and handle user input until the user chooses to exit.

- Inside the loop, display the main menu options:
 - Option 1: Add an Employee
 - Option 2: Display All Employees
 - Option 3: Exit
- Read the user's choice from the console and execute the corresponding action.

4. Implement the addEmployee Method:

- Prompt the user to enter the employee's ID, name, age, and salary.
- Read the input values using the scanner object and create a new Employee object.
- Add the new employee to the employees list.
- Open the file specified by FILE_NAME in append mode using a PrintWriter.
- Write the details of the new employee to the file.
- Close the PrintWriter and handle any IOException that may occur.

5. Implement the displayAllEmployees Method:

- Display the header for the employee report.
- Open the file specified by FILE_NAME using a Scanner.
- Read each line from the file and display it on the console.
- Close the Scanner and handle any FileNotFoundException that may occur.

6. Handle User Input:

- Consume newline characters after reading integer values using `in.nextLine()` to avoid skipping input.
- Display appropriate error messages for invalid input.

7. Exit the Program:

- Display a farewell message when the user chooses to exit the program.

8. Compile and Run the Program:

- Compile the Java file containing the Employee and EmployeeManagementSystem classes.
- Run the compiled program to execute the Employee Management System.

- **Code:**

```
import java.io.*;
import java.util.*;
class Employee {
    private int id;
    private String name;
    private int age;
    private double salary;

    public Employee(int id, String name, int age, double salary) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.salary = salary;
    }
    @Override
    public String toString() {
        return id + " " + name + " " + age + " " + salary;
    }
}
public class EmployeeManagementSystem {
    private static final String FILE_NAME = "employees.txt";
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        List<Employee> employees = new ArrayList<>();
        int choice;
        do {
            System.out.println("Main Menu");
            System.out.println("1. Add an Employee");
            System.out.println("2. Display All");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch (choice) {
                case 1:
                    addEmployee(employees);
                    break;
                case 2:
                    displayAllEmployees(employees);
                    break;
                case 3:
                    System.out.println("Exiting the System");
                    break;
                default:
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        System.out.println("Invalid choice. Please enter
again.");
    }
} while (choice != 3);
}
private static void addEmployee(List<Employee> employees) {
    System.out.print("Enter Employee ID: ");
    int id = scanner.nextInt();
    scanner.nextLine(); // Consume newline

    System.out.print("Enter Employee Name: ");
    String name = scanner.nextLine();

    System.out.print("Enter Employee Age: ");
    int age = scanner.nextInt();
    scanner.nextLine(); // Consume newline

    System.out.print("Enter Employee Salary: ");
    double salary = scanner.nextDouble();
    scanner.nextLine(); // Consume newline

    Employee employee = new Employee(id, name, age, salary);
    employees.add(employee);

    // Save the employee to file
    try (PrintWriter writer = new PrintWriter(new
FileWriter(FILE_NAME, true))) {
        writer.println(employee);
    } catch (IOException e) {
        System.out.println("Error occurred while writing to file: "
e.getMessage());
    }
}
private static void displayAllEmployees(List<Employee> employees) {
    System.out.println("----Report----");
    try (Scanner fileScanner = new Scanner(new File(FILE_NAME))) {
        while (fileScanner.hasNextLine()) {
            System.out.println(fileScanner.nextLine());
        }
    } catch (FileNotFoundException e) {
        System.out.println("File not found: " + e.getMessage());
    }
    System.out.println("----End of Report----");
}
}
```



- **Result/Output:**

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! <https://aka.ms/PSWindows>

```
PS D:\Vscode\JavaCodes> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '--enable-preview' '-e\AppData\Roaming\Code\User\workspaceStorage\261149df62e83f7ec16771a9659ed960\redhat.java\jdMain Menu
```

```
1. Add an Employee
2. Display All
3. Exit
```

Enter your choice: 1

Enter Employee ID: 5895

Enter Employee Name: Rajdeep

Enter Employee Age: 21

Enter Employee Salary: 10000

Main Menu

```
1. Add an Employee
2. Display All
3. Exit
```

Enter your choice: 1

Enter Employee ID: 5886

Enter Employee Name: Shashank

Enter Employee Age: 22

Enter Employee Salary: 15000

Main Menu

```
1. Add an Employee
2. Display All
3. Exit
```

Enter your choice: 2

```
----Report-----
5895 Rajdeep 21 10000.0
5886 Shashank 22 15000.0
----End of Report-----
```

Main Menu

```
1. Add an Employee
2. Display All
3. Exit
```

Enter your choice: 3

Exiting the System

PS D:\Vscode\JavaCodes> █

Experiment 3.1

Student Name: Rajdeep Majumder

Branch: BE CSE

Semester: 6th

Subject Name: Java Lab

UID: 21BCS5895

Section: FL 602 B

Date: 19.03.2024

Subject Code: 21CSH 319

1. Aim: Create a palindrome creator application for making a largest possible palindrome string out of given input string.

2. Objective:

- To learn about concept of HashMap in java.
- To learn about concept of String in java.

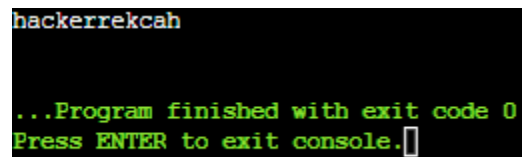
3. Code:

```
class Largest_Palindromic_Substring{
    public static void longestPalindrome(String str)
    {
        int n = str.length();
        int max = 1, start = 0;
        for (int i = 0; i < str.length(); i++) {
            for (int j = i; j < str.length(); j++) {
                boolean flag = true;
                for (int k = 0; k < (j - i + 1) / 2; k++)
                    if (str.charAt(i + k) != str.charAt(j - k))
                        flag = false;
                if (flag && (j - i + 1) > max) {
                    start = i;
                    max = j - i + 1;
                }
            }
        }
    }
}
```



```
System.out.println(str.substring(start,start+max));
}
public static void main(String[] args)
{
    String str = "hackerrekcahbahh";
    longestPalindrome(str);
}
}
```

4. Output:

A screenshot of a console window showing the output of the program. The first line is 'hackerrekcah' in red text. The second line is '...Program finished with exit code 0' in green text. The third line is 'Press ENTER to exit console.' in green text, followed by a cursor icon.

Experiment 3.2

Student Name: Rajdeep Majumder
Branch: BE CSE
Semester: 6th
Subject Name: Java Lab

UID: 21BCS5895
Section: FL 602 B
Date: 26.04.2024
Subject Code: 21CSH 319

- 1. Aim:** Write a program to write a JSP application to demonstrate the expression tag by using mathematical operation.
- 2. Objective:** There are multiple checkbox in html page to perform different mathematical operations. By default, Addition is checked and its add the given number in text box. At the time of division operation when we enter any invalid data in text box, it generates error message from "error.jsp" page.

3. Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Mathematical Operation</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="hero">
    <h4> Rajdeep Majumder</h4>
    <form action="index.jsp" method="post">
      <label for="operation">Select Operation:</label><br>
      <div class="checkbox-container">
        <input type="radio" name="operation" value="add" checked>Addition<br>
        <input type="radio" name="operation" value="subtract">Subtraction<br>
        <input type="radio" name="operation" value="multiply">Multiplication<br>
        <input type="radio" name="operation" value="divide">Division<br><br>
      </div>
    </form>
  </div>
</body>
</html>
```

```

Enter first Value: <input type="number" name="num1" class="input-field"><br>
Enter second Value: <input type="number" name="num2" class="input-field"><br><br>
<input type="submit" value="Check Result!" class="submit-button">
</form>
</div>
<script>
function handleFormSubmit(event) {
  event.preventDefault(); // Prevent default form submission
  const operation = document.querySelector('input[name="operation"]:checked').value;
  const num1 = parseInt(document.querySelector('input[name="num1"]').value);
  const num2 = parseInt(document.querySelector('input[name="num2"]').value);
  let result;
  try {
    switch (operation)
    {
      case "add":
        result = num1 + num2;
        break;
      case "subtract":
        result = num1 - num2;
        break;
      case "multiply":
        result = num1 * num2;
        break;
      case "divide":
        if (num2 === 0) {
          throw new Error("Division by zero");
        }
        result = num1 / num2;
        break;
    }
    alert("The result is: " + result);
  } catch (error)
  {
    alert("Error: " + error.message);
  }
}

const form = document.querySelector('form');
form.addEventListener('submit', handleFormSubmit);
</script>
</body>
</html>

body {
  font-family: sans-serif;
}

```

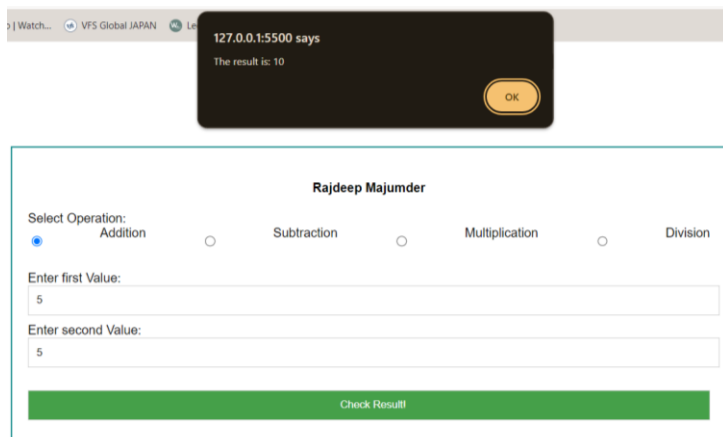
```

.hero{
  padding: 20px;
  margin: 300px;
  border: 2px solid teal;
}
h3,h4 {
  text-align: center;
  margin-bottom: 20px;
}
.checkbox-container {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
  margin-bottom: 20px;
}

}
.input-field,
.submit-button {
  padding: 10px;
  padding-right: 0px;
  border: 1px solid #ccc;
  margin-bottom: 10px;
  width: 100%;
}
.submit-button {
  background-color: #4CAF50;
  color: white;
  cursor: pointer;
}
.submit-button:hover {
  background-color: #45A049;
}

```

4. Output:



The image shows a web browser window with a terminal at the top and a calculator application below it. The terminal displays the command `127.0.0.1:5500 says` and the output `The result is: 10`, with an `OK` button. The calculator application, titled "Rajdeep Majumder", has a "Select Operation:" section with radio buttons for Addition (selected), Subtraction, Multiplication, and Division. Below this are two input fields: "Enter first Value:" with the value `5` and "Enter second Value:" with the value `5`. At the bottom is a green button labeled "Check Result".

Experiment No: 3.3

Student Name: Rajdeep Majumder

Branch: CSE

Semester: 6th

Subject Name: Project Based Learning with Java

Subject Code: 21CSH - 319

UID: 21BCS5895

Section/Group: FL 602 - B

Date of Performance: 16-03-24

- **Aim:** To create an application for online auction using Servlet and JSP.
- **Objective:** Create a HTML page to store the items available at the Auction using the Java Servlets and JSP. The user will be able to add the item ID, price, description as the item details.
- **Procedure/Algorithm/Pseudocode:**
 1. **User Registration:**

User fills out registration form with username and password.
Servlet receives registration form data.
Servlet validates input data and inserts user into the database.
Servlet redirects user to a registration success page.
 2. **User Login:**

User enters username and password on the login form.
Servlet receives login form data.
Servlet verifies credentials against the database.
If credentials are valid, Servlet creates a session and stores user information.
Servlet redirects user to the home/dashboard page.
 3. **Auction Listing:**

Servlet fetches active auctions from the database.
Servlet sends auction data to the JSP page for display.
 4. **Auction Details:**

User clicks on an auction to view its details.
Servlet retrieves auction details from the database.
Servlet sends auction data to the JSP page for display.
 5. **Placing a Bid:**

User submits a bid for an auction.
Servlet receives bid data.
Servlet verifies bid against auction rules (e.g., minimum bid

increment).

If bid is valid, Servlet updates auction with new bid information in the database.

6. Managing Auctions:

Registered users can create new auctions.

Servlet receives auction creation form data.

Servlet validates input data and inserts new auction into the database.

Users can also edit or delete their own auctions.

Servlet handles edit and delete requests accordingly, updating or removing auctions from the database.

7. Logout:

User clicks on the logout button.

Servlet invalidates the session.

Servlet redirects user to the login page.

- **Code:**

For User Registration

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class UserRegistrationServlet extends HttpServlet {
    public void doPost(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        response.sendRedirect("registration_success.jsp");
    }
}
```

UserLoginServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class UserLoginServlet extends HttpServlet {
    public void doPost(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");
    }
}
```

```
HttpSession session = request.getSession();  
session.setAttribute("username", username);  
response.sendRedirect("home.jsp");  
} }
```

UserRegistration.jsp

```
<!-- registration.jsp -->  
<html>  
<body>  
    <form action="user_registration" method="post">  
        Username: <input type="text" name="username"><br>  
        Password: <input type="password" name="password"><br>  
        <input type="submit" value="Register">  
    </form>  
</body>  
</html>
```

UserLogin.jsp

```
<!-- login.jsp -->  
<html>  
<body>  
    <form action="user_login" method="post">  
        Username: <input type="text" name="username"><br>  
        Password: <input type="password" name="password"><br>  
        <input type="submit" value="Login">  
    </form>  
</body>  
</html>
```

SuccessfulRegistration.jsp

```
<!-- registration_success.jsp -->  
<html>  
<body>  
    <h2>Registration Successful!</h2>  
    <p>You can now login to your account.</p>  
    <a href="login.jsp">Login</a>  
</body>  
</html>
```

Jsp for Home/Dashboard

```
<!-- home.jsp -->  
<html>  
<body>
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
<h2>Welcome, <%= session.getAttribute("username") %>!/h2>
<!-- Your home/dashboard content goes here -->
</body>
</html>
```

- **Result/Output:**

Online Auction

Title:

Premium

Item ID:

9782

Title:

Nissan GTR Skyline

Description:

The 10th generation Skyline R34 was rolled out in 1998. The body had high rigidity to improve motion performance while securing enough interior space in a smaller body. Its highest-level performance model "GT-R" was released in 1999.

Starting Bid:

\$ 65000

End Date:

18-04-2024

Submit

Copyright © 2024 Online Auction



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Java Intermediate Problem

Student Name: Rajdeep Majumder

UID: 21BCS5895

Branch: Computer Science Engineering

Section/Group: 21BCS FL 602 B

Semester: 6th

Date of Performance: 09.04.2024

Subject Name: Java LAB

Subject Code: 21CSP-351

Question 1

The next greater element of some element x in an array is the first greater element that is to the right of x in the same array. You are given two distinct 0-indexed integer arrays `nums1` and `nums2`, where `nums1` is a subset of `nums2`. For each $0 \leq i < \text{nums1.length}$, find the index j such that `nums1[i] == nums2[j]` and determine the next greater element of `nums2[j]` in `nums2`. If there is no next greater element, then the answer for this query is `-1`. Return an array `ans` of length `nums1.length` such that `ans[i]` is the next greater element as described above.

CODE:

```
class Solution {
    public int[] nextGreaterElement(int[] nums1, int[] nums2) {
        if (nums2.length == 0 || nums1.length == 0)
            return new int[0];
        Map<Integer, Integer> numberNGE = new HashMap<>();
        Stack<Integer> numStack = new Stack<>();
        numStack.push(nums2[nums2.length - 1]);
        numberNGE.put(nums2[nums2.length - 1], -1);
        for (int i = nums2.length - 2; i >= 0; i--) {
            if (nums2[i] < numStack.peek()) {
                numberNGE.put(nums2[i], numStack.peek());
                numStack.push(nums2[i]);
                continue;
            }
            while (!numStack.isEmpty() && numStack.peek() < nums2[i])
                numStack.pop();
            if (numStack.isEmpty()) {
                numStack.push(nums2[i]);
                numberNGE.put(nums2[i], -1);
            } else {
                numberNGE.put(nums2[i], numStack.peek());
            }
        }
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        numStack.push(nums2[i]);
    }
}
for (int i = 0; i < nums1.length; i++)
    nums1[i] = numberNGE.get(nums1[i]);
return nums1;
}
```

OUTPUT:

Input

```
nums1 =
[4, 1, 2]
```

```
nums2 =
[1, 3, 4, 2]
```

Output

```
[-1, 3, -1]
```

Expected

```
[-1, 3, -1]
```

Question 2

Develop a Java program showcasing the concept of inheritance. Create a base class and a derived class with appropriate methods and fields.

CODE:

```
class Animal {

    String name;

    public void eat() {

        System.out.println("I can eat");

    }

}

class Dog extends Animal {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
public void bark() {  
  
    System.out.println("I can bark");  
  
} }  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Dog labrador = new Dog();  
  
        labrador.name = "Yuvraj";  
  
        labrador.eat();  
  
        labrador.bark();  
  
    } } }
```

OUTPUT:

A screenshot of a Java program's output in a console window. The background is black, and the text is white and green. The output shows two lines of text: "I can eat" and "I can bark". Below these, there is a green line indicating the program finished with exit code 0, and a prompt to press ENTER to exit the console.

```
I can eat  
I can bark  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Question 3:

Implement a Java program that uses method overloading to perform different mathematical operations.

CODE:

```
public class MathOperations {  
  
    public static int add(int a, int b) {  
  
        return a + b;  
  
    } } }
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
}
```

```
public static double add(double a, double b) {
```

```
    return a + b;
```

```
}
```

```
public static int subtract(int a, int b) {
```

```
    return a - b;
```

```
}
```

```
public static double subtract(double a, double b) {
```

```
    return a - b;
```

```
}
```

```
public static int multiply(int a, int b) {
```

```
    return a * b;
```

```
}
```

```
public static double multiply(double a, double b) {
```

```
    return a * b;
```

```
}
```

```
public static int divide(int a, int b) {
```

```
    return a / b;
```

```
}
```

```
public static double divide(double a, double b) {
```

```
    return a / b;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
}  
  
public static void main(String[] args) {  
  
    System.out.println(add(1, 2));  
  
    System.out.println(add(1.5, 2.5));  
  
    System.out.println(subtract(5, 3));  
  
    System.out.println(subtract(5.5, 3.5));  
  
    System.out.println(multiply(3, 4));  
  
    System.out.println(multiply(3.5, 4.5));  
  
    System.out.println(divide(10, 2));  
  
    System.out.println(divide(10.0, 2.0));  
  
} }
```

OUTPUT:

A screenshot of a Java program's output in a console window. The output shows the results of the arithmetic operations defined in the code: 3, 4.0, 2, 2.0, 12, 15.75, 5, and 5.0. At the bottom, it states '...Program finished with exit code 0' and 'Press ENTER to exit console.' with a cursor.

```
3  
4.0  
2  
2.0  
12  
15.75  
5  
5.0  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Question 4

Define an interface in Java and create a class that implements it, demonstrating the concept of abstraction.

Ans: -



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

An interface is declared by using the interface keyword. It provides total abstraction; means all the methods in an interface are declared with the empty body, and all the fields are public, static and final by default. A class that implements an interface must implement all the methods declared in the interface.

Here is an example of an interface in Java:

```
interface Animal {  
    void animalSound();  
    void run(); }  

```

This interface defines two methods, animalSound() and run(), which are both abstract. This means that they do not have any implementation, and any class that implements this interface must provide its own implementation for these methods.

Question 5:

Explain the difference between the throw and throws keywords in Java. Provide examples illustrating their usage.

Ans: -

In Java, the throw and throws keywords are used in exception handling. However, they serve different purposes and are used in different scenarios.

The throw keyword is used to explicitly throw an exception from within a block of code or a method. This is useful when you want to handle an exception explicitly, rather than letting it propagate up the call stack.

The throws keyword is used in the method signature to declare which exceptions can be thrown from a method. This is useful for informing the caller of the method that they need to handle these exceptions.

| SL No. | <u>throw</u> | <u>throws</u> |
|--------|--|--|
| 1. | The throw keyword is used inside a function. It is used when it is required to throw an Exception logically. | The throws keyword is used in the function signature. It is used when the function has some statements that can lead to exceptions. |
| 2. | The throw keyword is used to throw an exception explicitly. It can throw only one exception at a time. | The throws keyword can be used to declare multiple exceptions, separated by a comma. Whichever exception occurs, if matched with the declared ones, is thrown automatically then. |
| 3. | Syntax of throw keyword includes the instance of the Exception to be thrown. Syntax wise throw keyword is followed by the instance variable. | Syntax of throws keyword includes the class names of the Exceptions to be thrown. Syntax wise throws keyword is followed by exception class names. |
| 4. | throw keyword cannot propagate checked exceptions. It is only used to propagate the unchecked Exceptions that are not checked using the throws keyword. | throws keyword is used to propagate the checked Exceptions only. |

Experiment 1.1

Student Name: Rajdeep Majumder
Branch: BE-CSE
Semester: 6
Subject Name: Java Lab

UID: 21BCS5895
Section: FL-602 B
Date: 16.01.2024
Subject Code: 21CSH-319

- 1. Aim:** Create an application to save the employee information using arrays
- 2. Objective:** Given the following table containing information about employees of an organization, develop a small java application, which accepts employee id from the command prompt and displays the details

3. Algo. /Approach and output:

```
import java.util.*;

class Project1
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        int m,n;
        String[][] employees = {
            {"11", "Rajdeep", "R&D", "e", "20000", "8000", "3000"},
            {"22", "Shashank", "PM", "C", "30000", "12000", "9000"},
            {"33", "Soumik", "Acct", "k", "10000", "8000", "1000"},
            {"44", "Akhand", "Front Desk", "r", "12000", "6000", "2000"},
        }
```



```
{ "55", "Arnab", "Engg", "m", "50000", "20000", "20000"},
{ "66", "Sumit", "Manufacturing", "e", "23000", "9000", "4400"},
{ "77", "Aryan", "PM", "C", "29000", "12000", "10000"} };

System.out.print("Enter employee ID: ");

String empId = in.nextLine();

boolean employeeFound = false;

for (String[] employee : employees)
{
    if (employee[0].equals(empId))
    {
        employeeFound = true;

        String designation = "";

        int da = 0;

        switch (employee[3])
        {
            case "e":

                designation = "Engineer";

                da = 20000;

                break;

            case "C":

                designation = "Consultant";

                da = 32000;

                break;

            case "k":
```

```
        designation = "Clerk";  
        da = 12000;  
        break;  
        case "r":  
            designation = "Receptionist";  
            da = 15000;  
            break;  
        case "m":  
            designation = "Manager";  
            da = 40000;  
            break;  
    }  
    int salary = Integer.parseInt(employee[4]);  
    int hra = Integer.parseInt(employee[5]);  
    int it = Integer.parseInt(employee[6]);  
    int totalSalary = salary + hra + it + da;  
    System.out.println("Emp No: " + employee[0]);  
    System.out.println("Emp Name: " + employee[1]);  
    System.out.println("Department: " + employee[2]);  
    System.out.println("Designation: " + designation);  
    System.out.println("Salary: " + totalSalary);  
    break;  
}  
}
```

```
        if (!employeeFound)
        {
            System.out.println("Employee not found.");

        }
    }
}
```

Output:

```
Enter employee ID: 11
Emp No: 11
Emp Name: Rajdeep
Department: R&D
Designation: Engineer
Salary: 51000

...Program finished with exit code 0
Press ENTER to exit console.█
```

Experiment No: 1.2

Student Name: Rajdeep Majumder
Branch: CSE
Semester: 6th
Subject Name: Java LAB

UID: 21BCS5895
Section/Group: FL_602 - B
Date of Performance: 23-01-24
Subject Code: 21CSH - 319

1. Aim: Design a simple inventory control system for a small video rental store.

2. Objective: The goal of this project is to design and implement a simple inventory control system for a small video rental store. Define least two classes: a class Video to model a video and a class VideoStore to model the actual store.

- **Input/Apparatus Used:** Virtual Studio Code.
- **Procedure/Algorithm/Pseudocode:**

1. Define the Video class:

- Declare private fields for title, rating, and checkedOut.
- Implement a constructor to initialize the title, rating, and checkedOut status.
- Define getter and setter methods for the title, rating, and checkedOut status.
- Implement methods to check out and return videos.

2. Define the InventoryControlSystem class:

- Declare an ArrayList to store video objects.
- Implement methods to add videos to the inventory, checkout videos, rate videos, list the inventory, and return videos.
- Ensure proper error handling for cases where videos are not found or are already checked out.

3. Define the Main class:

- Create an instance of the InventoryControlSystem class.
- Display a menu to the user with options to add, checkout, rate, list, and return videos.
- Prompt the user for input and invoke corresponding methods based on their choice.
- Repeat the menu display until the user chooses to exit the program.

3. Code:

```
import java.util.ArrayList;
import java.util.Scanner;
class Video {
    private String title;
    private int rating;
    private boolean checkedOut;
    public Video(String title) {
        this.title = title;
        this.rating = 0;
        this.checkedOut = false; }
    public String getTitle() {
        return title; }
    public int getRating() {
        return rating; }
    public void setRating(int rating) {
        this.rating = rating; }
    public boolean isCheckedOut() {
        return checkedOut; }
    public void checkOut() {
        checkedOut = true; }

    public void returnVideo() {
        checkedOut = false; } }
class InventoryControlSystem {
```

```
private ArrayList<Video> inventory;
public InventoryControlSystem() {
    inventory = new ArrayList<>(); }
    public void addVideo(String title) {
        inventory.add(new Video(title));
System.out.println("Video " + title + " added to inventory."); }
public void checkoutVideo(String title) {
    for (Video video : inventory) {
        if (video.getTitle().equalsIgnoreCase(title) && !video.isCheckedOut()) {
            video.checkOut();
            System.out.println("Video " + title + " checked out successfully.");
            return; }
}

    System.out.println("Video " + title + " not found or already checked out."); }
public void rateVideo(String title, int rating) {
    for (Video video : inventory) {
        if (video.getTitle().equalsIgnoreCase(title)) {
            video.setRating(rating);
            System.out.println("Rating for video " + title + " set to " + rating);
            return; } }
    System.out.println("Video " + title + " not found."); }
public void listInventory() {
    if (inventory.isEmpty()) {
        System.out.println("Inventory is empty.");
    } else {
        System.out.println("Video Inventory:");
        for (Video video : inventory) {
            System.out.println("Title: " + video.getTitle() + ", Rating: " +
video.getRating() + ", Checked Out: " + (video.isCheckedOut() ? "Yes" : "No"));
        }
    }
}

    public boolean returnVideo(String title) {
        for (Video video : inventory) {
            if (video.getTitle().equalsIgnoreCase(title) && video.isCheckedOut()) {
```

```
        video.returnVideo();
        System.out.println("Video " + title + " returned successfully.");
        return true; } }
System.out.println("Video " + title + " not found or not checked out.");
return false; }
}

public class Main {
public static void main(String[] args) {
    InventoryControlSystem inventorySystem = new InventoryControlSystem();
    Scanner scanner = new Scanner(System.in);
    int choice;
    do {
        System.out.println("\nInventory Control System Menu:");
        System.out.println("1. Add Video");
        System.out.println("2. Checkout Video");
        System.out.println("3. Rate Video");
        System.out.println("4. List Inventory");
        System.out.println("5. Return Video");
        System.out.println("6. Exit");
        System.out.print("\nEnter your choice: ");
        choice = scanner.nextInt();
        scanner.nextLine();
        switch (choice) {
            case 1:
                System.out.print("Enter video title: ");
                String addTitle = scanner.nextLine();
                inventorySystem.addVideo(addTitle);
                break;
            case 2:
                System.out.print("Enter video title to checkout: ");
                String checkoutTitle = scanner.nextLine();
                inventorySystem.checkoutVideo(checkoutTitle);
                break;
            case 3:
                System.out.print("Enter video title to rate: ");
```

```
String rateTitle = scanner.nextLine();
System.out.print("Enter rating (1-5): ");
int rating = scanner.nextInt();
inventorySystem.rateVideo(rateTitle, rating);
break;
case 4:
inventorySystem.listInventory();
break;
case 5:
System.out.print("Enter video title to return: ");
String returnTitle = scanner.nextLine();
inventorySystem.returnVideo(returnTitle);
break;
case 6:
System.out.println("Exiting Inventory Control System.Goodbye!");
break;
default:
System.out.println("Invalid choice. Please enter a number between 1
and 6.");
}
    } while (choice != 6);
    scanner.close();
}
}
```


4. Result/Output:

```
Inventory Control System Menu:
1. Add Video
2. Checkout Video
3. Rate Video
4. List Inventory
5. Return Video
6. Exit

Enter your choice: 1
Enter video title: The Usual Suspects
Video 'The Usual Suspects' added to inventory.

Inventory Control System Menu:
1. Add Video
2. Checkout Video
3. Rate Video
4. List Inventory
5. Return Video
6. Exit

Enter your choice: 1
Enter video title: Pulp Fiction
Video 'Pulp Fiction' added to inventory.

Inventory Control System Menu:
1. Add Video
2. Checkout Video
3. Rate Video
4. List Inventory
5. Return Video
6. Exit

Enter your choice: 1
Enter video title: Silence of The Lambs
Video 'Silence of The Lambs' added to inventory.

Inventory Control System Menu:
1. Add Video
2. Checkout Video
3. Rate Video
4. List Inventory
5. Return Video
6. Exit

Enter your choice: 4
Video Inventory:
Title: The Usual Suspects, Rating: 0, Checked Out: No
Title: Pulp Fiction, Rating: 0, Checked Out: No
Title: Silence of The Lambs, Rating: 0, Checked Out: No

Inventory Control System Menu:
1. Add Video
2. Checkout Video
3. Rate Video
4. List Inventory
5. Return Video
6. Exit
```



Experiment 1.3

Student Name: Rajdeep Majumder

UID: 21BCS5895

Branch: BE-CSE

Section/Group: FL-602-B

Semester: 6th

Date of Performance: 29-01-2024

Subject Name: Project Based Learning in Java with Lab

Subject Code: 21CSP-319

1. Aim:

Create an application to calculate interest for FDs, RDs based on certain conditions using inheritance.

2. Objective:

Write a program to create an application to make an Account holders list and calculate interest for FDs, RDs based on certain conditions using inheritance.

3. Algo. /Approach and output:

```
import java.util.Scanner;
class InvalidAgeException extends Exception {}
class InvalidAmountException extends Exception {}
class InvalidDaysException extends Exception {}
class InvalidMonthsException extends Exception {}
abstract class Account {
    double interestRate;
    double amount;

    abstract double calculateInterest(double amount) throws InvalidMonthsException,
InvalidAgeException, InvalidAmountException, InvalidDaysException;
}

class FDaccount extends Account {
    double FDinterestRate;
    double FDAmount;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
int noOfDays;
int ageOfACHolder;
double General, SCitizen;
Scanner FDScanner = new Scanner(System.in);

double calculateInterest(double amount) throws InvalidAgeException,
InvalidAmountException, InvalidDaysException {
    this.FDAmount = amount;
    System.out.println("Enter FD days");
    noOfDays = FDScanner.nextInt();
    System.out.println("Enter FD age holder ");
    ageOfACHolder = FDScanner.nextInt();

    if (amount < 0) {
        throw new InvalidAmountException();
    }
    if (noOfDays < 0) {
        throw new InvalidDaysException();
    }
    if (ageOfACHolder < 0) {
        throw new InvalidAgeException();
    }

    if (amount < 10000000) {
        if (noOfDays >= 7 && noOfDays <= 14) {
            General = 0.0450;
            SCitizen = 0.0500;
        } else if (noOfDays >= 15 && noOfDays <= 29) {
            General = 0.0470;
            SCitizen = 0.0525;
        } else if (noOfDays >= 30 && noOfDays <= 45) {
            General = 0.0550;
            SCitizen = 0.0600;
        } else if (noOfDays >= 45 && noOfDays <= 60) {
            General = 0.0700;
            SCitizen = 0.0750;
        } else if (noOfDays >= 61 && noOfDays <= 184) {
            General = 0.0750;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        SCitizen = 0.0800;
    } else if (noOfDays >= 185 && noOfDays <= 365) {
        General = 0.0800;
        SCitizen = 0.0850;
    }
    FDinterestRate = (ageOfACHolder < 50) ? General : SCitizen;
} else {
    if (noOfDays >= 7 && noOfDays <= 14) {
        interestRate = 0.065;
    } else if (noOfDays >= 15 && noOfDays <= 29) {
        interestRate = 0.0675;
    } else if (noOfDays >= 30 && noOfDays <= 45) {
        interestRate = 0.0675;
    } else if (noOfDays >= 45 && noOfDays <= 60) {
        interestRate = 0.080;
    } else if (noOfDays >= 61 && noOfDays <= 184) {
        interestRate = 0.0850;
    } else if (noOfDays >= 185 && noOfDays <= 365) {
        interestRate = 0.10;
    }
}
return FDAmount * FDinterestRate;
}
}
```



```
class RDaccount extends Account {
    double RDInterestRate;
    double RDAmount;
    int noOfMonths;
    double monthlyAmount;
    double General, SCitizen;
    Scanner RDScanner = new Scanner(System.in);

    double calculateInterest(double Ramount) throws InvalidMonthsException,
InvalidAmountException, InvalidAgeException {
        this.RDAmount = Ramount;
        System.out.println("Enter RD months");
        noOfMonths = RDScanner.nextInt();
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
System.out.println("Enter RD holder age");
int age = RDScanner.nextInt();

if (RDamount < 0) {
    throw new InvalidAmountException();
}
if (noOfMonths < 0) {
    throw new InvalidMonthsException();
}
if (age < 0) {
    throw new InvalidAgeException();
}

if (noOfMonths >= 0 && noOfMonths <= 6) {
    General = 0.0750;
    SCitizen = 0.080;
} else if (noOfMonths >= 7 && noOfMonths <= 9) {
    General = 0.0775;
    SCitizen = 0.0825;
} else if (noOfMonths >= 10 && noOfMonths <= 12) {
    General = 0.0800;
    SCitizen = 0.0850;
} else if (noOfMonths >= 13 && noOfMonths <= 15) {
    General = 0.0825;
    SCitizen = 0.0875;
} else if (noOfMonths >= 16 && noOfMonths <= 18) {
    General = 0.0850;
    SCitizen = 0.0900;
} else if (noOfMonths >= 22) {
    General = 0.0875;
    SCitizen = 0.0925;
}
RDInterestRate = (age < 50) ? General : SCitizen;
return RDamount * RDInterestRate;
}
}
```

```
class SBaccount extends Account {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
double sbAmount, sbInterestRate, interest;
Scanner SBScanner = new Scanner(System.in);

double calculateInterest(double amount) throws InvalidAmountException {
    this.sbAmount = amount;
    if (sbAmount < 0) {
        throw new InvalidAmountException();
    }
    System.out.println("Select account type \n1. NRI \n2. Normal ");
    int accountChoice = SBScanner.nextInt();
    switch (accountChoice) {
        case 1:
            sbInterestRate = 0.06;
            break;
        case 2:
            sbInterestRate = 0.04;
            break;
        default:
            System.out.println("Please choose right account again");
            break;
    }
    return amount * sbInterestRate;
}

public class Main {
    public static void main(String[] args) {
        boolean val = true;
        Scanner sc = new Scanner(System.in);
        while (val) {
            System.out.println("SELECT THE OPTIONS " + "\n1." + " Interest Calculator-SB" +
                "\n2." +
                " Interest Calculator-FD" + "\n3." + " Interest Calculator-RD" + "\n4 " + " Exit");
            int choice = sc.nextInt();
            switch (choice) {
                case 1:
                    SBaccount sb = new SBaccount();
                    try {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        System.out.println("Enter the Average SB amount ");
        double amount = sc.nextDouble();
        System.out.println("Interest gained is : Rs " + sb.calculateInterest(amount));
    } catch (InvalidAmountException e) {
        System.out.println("Exception: Invalid amount entered.");
    }
    break;
case 2:
    try {
        FDaccount fd = new FDaccount();
        System.out.println("Enter the FD Amount");
        double fAmount = sc.nextDouble();
        System.out.println("Interest gained is: Rs " + fd.calculateInterest(fAmount));
    } catch (InvalidAgeException e) {
        System.out.println("Invalid Age Entered");
    } catch (InvalidAmountException e) {
        System.out.println("Invalid Amount Entered");
    } catch (InvalidDaysException e) {
        System.out.println("Invalid Days Entered");
    }
    break;
case 3:
    try {
        RDaccount rd = new RDaccount();
        System.out.println("Enter the RD amount");
        double Ramount = sc.nextDouble();
        System.out.println("Interest gained is: Rs " + rd.calculateInterest(Ramount));
    } catch (InvalidAgeException e) {
        System.out.println("Invalid Age Entered");
    } catch (InvalidAmountException e) {
        System.out.println("Invalid Amount Entered");
    } catch (InvalidMonthsException e) {
        System.out.println("Invalid Months Entered");
    }
    break;
case 4:
    val = false;
    System.out.println("Exiting the program.");
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        break;
    default:
        System.out.println("Wrong choice");
        break;
    }
}
sc.close();
}
```

Output:

```
<terminated> InterestCalculator [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (04-Feb-2024, 6:00:58 p
SELECT THE OPTIONS
1. Interest Calculator-SB
2. Interest Calculator-FD
3. Interest Calculator-RD
4 Exit
1
Enter the Average SB amount
10000
Select account type
1. NRI
2. Normal
2
Interest gained is : Rs 400.0
SELECT THE OPTIONS
1. Interest Calculator-SB
2. Interest Calculator-FD
3. Interest Calculator-RD
4 Exit
2
Enter the FD Amount
10000
Enter FD days
91
Enter FD age holder
65
Interest gained is: Rs 800.0
SELECT THE OPTIONS
1. Interest Calculator-SB
2. Interest Calculator-FD
3. Interest Calculator-RD
4 Exit
4
Exiting the program.
```