

Experiment1.1

Student Name: Aafreen Khan

Branch:BE-CSE

Semester: 6

Subject Name: Java Lab

Subject Code:21CSH-319

UID: 21BCS1397

Section/Group:CC-606-A

Date of Performance:18-01-2024

- 1. Aim:** Create a 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.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
class Employee {
    int empNo;
    String empName;
    Date joinDate;
    char desigCode;
    String department;
    double basic;
    double hra;
    double it;
    public Employee(int empNo, String empName, String joinDate, char
desigCode, String department, double basic, double hra, double it) throws
ParseException {
        this.empNo = empNo;
        this.empName = empName;
```

```
this.joinDate = new
SimpleDateFormat("dd/MM/yyyy").parse(joinDate);
this.desigCode = desigCode;
this.department = department;
this.basic = basic;
this.hra = hra;
this.it = it;
}
public double calculateSalary() {
    double da = getDA();
    return basic + hra + da - it;
}
public String getDesignation() {
    switch (desigCode) {
        case 'e':
            return "Engineer";
        case 'c':
            return "Consultant";
        case 'k':
            return "Clerk";
        case 'r':
            return "Receptionist";
        case 'm':
            return "Manager";
        default:
            return "Unknown";
    }
}
private double getDA() {
    switch (desigCode) {
        case 'e':
            return 20000;
        case 'c':
            return 32000;
        case 'k':
            return 12000;
        case 'r':
```

```
        return 15000;
    case 'm':
        return 40000;
    default:
        return 0;
    }
}
}

public class Project1 {
    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Usage: java Project1 <EmpNo>");
            return;
        }
        int empNo = Integer.parseInt(args[0]);
        Employee[] employees = new Employee[7];
        try {
            employees[0] = new Employee(1001, "Ashish", "01/04/2009", 'e',
                "R&D", 20000, 8000, 3000);
            employees[1] = new Employee(1002, "Shashi", "23/08/2012", 'c',
                "PM", 30000, 12000, 9000);
            employees[2] = new Employee(1003, "Rahul", "12/11/2008", 'k',
                "Acct", 10000, 8000, 1000);
            employees[3] = new Employee(1004, "Chahat", "29/01/2013", 'r',
                "Front Desk", 12000, 6000, 2000);
            employees[4] = new Employee(1005, "Ranjan", "16/07/2005", 'm',
                "Engg", 50000, 20000, 20000);
            employees[5] = new Employee(1006, "Suman", "01/01/2000", 'e',
                "Manufacturing", 23000, 9000, 4400);
            employees[6] = new Employee(1007, "Tanmay", "12/06/2006", 'c',
                "PM", 29000, 12000, 10000);
        } catch (ParseException e) {
            e.printStackTrace();
        }
        boolean found = false;
        for (Employee employee : employees) {
            if (employee != null && employee.empNo == empNo) {
```

```
        found = true;
        System.out.println("Emp No.\tEmp
Name\tDepartment\tDesignation\tSalary");
        System.out.println(employee.empNo + "\t" + employee.empName
+ "\t" + employee.department + "\t" + employee.getDesignation() + "\t" +
employee.calculateSalary());
        break;
    }
}
if (!found) {
    System.out.println("There is no employee with empid : " + empNo);
}
}
}
```

Output:

```
PS C:\Users\pavilion\Downloads\JavaLab> java Project1 1003
Emp No. Emp Name      Department      Designation      Salary
1003    Rahul    Acct    Clerk    29000.0
PS C:\Users\pavilion\Downloads\JavaLab> java Project1 1002
Emp No. Emp Name      Department      Designation      Salary
1002    Shashi  PM        Consultant    65000.0
PS C:\Users\pavilion\Downloads\JavaLab> java Project1 1302
There is no employee with empid : 1302
PS C:\Users\pavilion\Downloads\JavaLab>
```

Experiment-1.2

Student Name: Aafreen Khan

Branch: BE-CSE

Semester: 6

Subject Name: Java Lab

Subject Code:21CSH-319

UID: 21BCS1397

Section/Group:CC-606 A

Date of Performance:25-01-2024

- 1. Aim:** Design and implement 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.

3. Algo. /Approach and output:

```
class Video {  
    private String title;  
    private boolean checkedOut;  
    private double averageRating;  
    private int numberOfRatings;  
    public Video(String title) {  
        this.title = title;  
        this.checkedOut = false;  
        this.averageRating = 0.0;  
        this.numberOfRatings = 0;  
    }  
    public String getTitle() {  
        return title;  
    }  
}
```

```
public boolean isCheckedOut() {
    return checkedOut;
}

public void checkOut() {
    checkedOut = true;
}

public void returnVideo() {
    checkedOut = false;
}

public void receiveRating(int rating) {
    averageRating = (averageRating * numberOfRatings + rating) / (numberOfRatings +
1);
    numberOfRatings++;
}

public double getAverageRating() {
    return averageRating;
}
}

class VideoStore {
    private Video[] inventory;

    public VideoStore() {
        this.inventory = new Video[10];
    }

    public void addVideo(String title) {
        for (int i = 0; i < inventory.length; i++) {
            if (inventory[i] == null) {
                inventory[i] = new Video(title);
                System.out.println("Video " + title + " added to the inventory.");
            }
        }
    }
}
```

```
        return;
    }
}

System.out.println("Error: Inventory is full. Cannot add video '" + title + "'.");
}

public void checkOut(String title) {
    for (Video video : inventory) {
        if (video != null && video.getTitle().equals(title) && !video.isCheckedOut()) {
            video.checkOut();
            System.out.println("Video '" + title + "' checked out.");
            return;
        }
    }
    System.out.println("Error: Video '" + title + "' not found or already checked out.");
}

public void returnVideo(String title) {
    for (Video video : inventory) {
        if (video != null && video.getTitle().equals(title) && video.isCheckedOut()) {
            video.returnVideo();
            System.out.println("Video '" + title + "' returned.");
            return;
        }
    }
    System.out.println("Error: Video '" + title + "' not found or not checked out.");
}

public void receiveRating(String title, int rating) {
    for (Video video : inventory) {
        if (video != null && video.getTitle().equals(title)) {
```

```
        video.receiveRating(rating);

        System.out.println("Rating of " + rating + " received for video '" + title + "'.");

        return;
    }
}

System.out.println("Error: Video '" + title + "' not found.");
}

public void listInventory() {
    System.out.println("Current Inventory:");

    for (Video video : inventory) {
        if (video != null) {
            System.out.println("Title: " + video.getTitle() +
                                ", Checked Out: " + video.isCheckedOut() +
                                ", Average Rating: " + video.getAverageRating());
        }
    }
}

}

public class VideoStoreLauncher {
    public static void main(String[] args) {
        VideoStore videoStore = new VideoStore();
        videoStore.addVideo("The Matrix");
        videoStore.addVideo("Godfather II");
        videoStore.addVideo("Star Wars Episode IV: A New Hope");
        videoStore.receiveRating("The Matrix", 4);
        videoStore.receiveRating("The Matrix", 5);
        videoStore.receiveRating("Godfather II", 5);
        videoStore.receiveRating("Godfather II", 4);
    }
}
```



```
videoStore.receiveRating("Star Wars Episode IV: A New Hope", 3);  
videoStore.checkOut("The Matrix");  
videoStore.returnVideo("The Matrix");  
videoStore.checkOut("Godfather II");  
videoStore.returnVideo("Godfather II");  
videoStore.checkOut("Star Wars Episode IV: A New Hope");  
videoStore.returnVideo("Star Wars Episode IV: A New Hope");  
videoStore.listInventory();  
}  
}
```

Output:

```
PS C:\Users\pavilion\Downloads\JavaLab> cd C:\Users\pavilion\Downloads\JavaLab\ ; if ($?) { javac  
VideoStoreLauncher.java } ; if ($?) { java VideoStoreLauncher }  
Video 'The Matrix' added to the inventory.  
Video 'Godfather II' added to the inventory.  
Video 'Star Wars Episode IV: A New Hope' added to the inventory.  
Rating of 4 received for video 'The Matrix'.  
Rating of 5 received for video 'The Matrix'.  
Rating of 5 received for video 'Godfather II'.  
Rating of 4 received for video 'Godfather II'.  
Rating of 3 received for video 'Star Wars Episode IV: A New Hope'.  
Video 'The Matrix' checked out.  
Video 'The Matrix' returned.  
Video 'Godfather II' checked out.  
Video 'Godfather II' returned.  
Video 'Star Wars Episode IV: A New Hope' checked out.  
Video 'Star Wars Episode IV: A New Hope' returned.  
Current Inventory:  
Title: The Matrix, Checked Out: false, Average Rating: 4.5  
Title: Godfather II, Checked Out: false, Average Rating: 4.5  
Title: Star Wars Episode IV: A New Hope, Checked Out: false, Average Rating: 3.0  
PS C:\Users\pavilion\Downloads\JavaLab>
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment 1.3

Student Name: Aafreen Khan

UID: 21BCS1397

Branch: BE-CSE

Section/Group: CC-606-A

Semester: 6th

Date of Performance: 01-02-2024

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 = 0.00675;
    } 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.
```




DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment 2.1

Student Name: Aafreen Khan

UID: 21BCS1397

Branch: BE-CSE

Section/Group: 606-A

Semester: 6th

Date of Performance: 08-02-2024

Subject Name: Project Based Learning in Java with Lab

Subject Code: 21CSP-319

1. Aim:

Create a program to collect and store all the cards to assist the users in finding all the cards in a given symbol using collection

2. Objective:

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

3. Algo. /Approach and output:

- 1) Create a Scanner object (input) to take user input.
- 2) Initialize an ArrayList (valueList) to store card values.
- 3) Create a TreeMap (mapObj) to store symbols as keys and corresponding lists of card values as values.
- 4) Declare variables totalCards, index, value, sum, and count.
- 5) Use a loop to iterate from 1 to totalCards.
- 6) Check if the symbol already exists in the map (mapObj). If yes, retrieve the existing list of values and add the new value to the list. If no, create a new list, add the value, and put it in the map
- 7) Iterate over the entries in the map and print the distinct symbols.
- 8) Iterate over the entries in the map. For each symbol, print the symbol and its corresponding card values. Calculate the sum and count of card values for each symbol.
- 9) Print the number of cards and the sum of values for each symbol.

4. Code:

```
import java.util.*;

public class Solution {

    public static void main(String[] args){
        Scanner input = new Scanner(System.in);
        List<Integer> valueList = new ArrayList<Integer>();
        TreeMap<String, List<Integer>> mapObj = new TreeMap<String, List<Integer>>();
        int totalCards, index, value, sum = 0, count = 0;

        System.out.println("ENTER NUMBER OF CARDS : ");
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
totalCards = input.nextInt();
String symbol;

for(index = 1; index <= totalCards; index++){
    System.out.println("ENTER CARD" + " " + index);
    symbol = input.next();
    value = input.nextInt();

    if(mapObj.containsKey(symbol)){
        valueList = mapObj.get(symbol);
        valueList.add(value);
    }else{
        valueList = new ArrayList<Integer>();
        valueList.add(value);
        mapObj.put(symbol, valueList);
    }
}

System.out.println("DISTINCT SYMBOLS ARE :");
for(Map.Entry getData : mapObj.entrySet()){
    System.out.print(getData.getKey() + " ");
}
System.out.println();

for(Map.Entry getData : mapObj.entrySet()){
    System.out.println("\nCARDS IN " + getData.getKey() + " SYMBOL :");
    ArrayList<Integer> temp = (ArrayList<Integer>) getData.getValue();
    Iterator itr= temp.iterator();
    while(itr.hasNext())
    {
        count++;
        int val = (int) itr.next();
        System.out.print(getData.getKey());
        System.out.println(" " + val);
        sum += val;
    }

    System.out.println("NUMBER OF CARDS : " + count);
    System.out.println("SUM OF NUMBERS : " + sum);
    sum = 0;
    count=0;
}
}
```

5. Output :

```
java -cp /tmp/mBuxS1Wnhs Solution
ENTER NUMBER OF CARDS :
4
ENTER CARD 1
a
6
ENTER CARD 2
b
4
ENTER CARD 3
a
9
ENTER CARD 4
b
1
DISTINCT SYMBOLS ARE :
a b CARDS IN a SYMBOL :
a 6
a 9
NUMBER OF CARDS : 2
SUM OF NUMBERS : 15

CARDS IN b SYMBOL :
b 4
b 1
NUMBER OF CARDS : 2
SUM OF NUMBERS : 5
```

6. Learning outcomes:

- 1) The code is organized into functions/methods, which makes it modular and easy to understand. The main logic is contained within the main method.
- 2) The code uses generics (`List<Integer>`, `TreeMap<String, List<Integer>>`) to ensure type safety. It also performs type casting when retrieving values from the map.
- 3) The code demonstrates how to check if a symbol is already present in the `TreeMap`. If the symbol exists, it appends the value to the existing list; otherwise, it creates a new entry in the map.
- 4) The program uses the `TreeMap` to store symbols as keys and lists of integers as values. This allows for easy retrieval and organization of card values based on their symbols.
- 5) The program uses `Scanner` to take input from the user. It prompts the user to enter the number of cards, and then for each card, it takes the symbol and value.



Experiment No: 2.2

Name: Aafreen Khan

UID: 21BCS1397

Branch: CSE

Section: 21BCS_CC-606A

Semester: 6th

Date: 29/02/24

Subject: PBLJ

Subject Code: 21CSP-351

Aim:

Create a program to collect unique symbols from a set of cards using set interface.

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.

Code:

```
package project1;
import java.util.*;

public 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;
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
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;
}

@Override
public String toString() {
    return "Card[symbol="+symbol+",number="+number+"]";
}

@Override
public int compareTo(Card o) {
    if (this.symbol < o.symbol) return
        -1;
    else if (this.symbol > o.symbol)
        return 1;
    else
        return 0;
}

@Override
public int hashCode() {
    return String.valueOf(symbol).hashCode();
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

@Override

```
public boolean equals(Object obj) { if
    (obj instanceof card) {
        card card = (card) obj;
        return (card.symbol == this.symbol);
    }
    else {
        return false;
    }
}

}

class TestMain {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Set<card> set = new HashSet<>();
        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();
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Output:

The screenshot displays the Eclipse IDE interface. The left sidebar shows the Package Explorer with a project named 'project1' containing files 'card.java', 'Data.java', 'J3.java', and 'module-info.java'. The main editor window shows the code for 'card.java'.

```
1 package project1;
2 import java.util.HashSet;
3 import java.util.Scanner;
4 import java.util.Set;
5
6 public class card implements Comparable<card>{
7     private char symbol;
8     private int number;
9     public card() {}
10    public card(char symbol, int number){
11        super();
12        this.symbol = symbol;
13        this.number = number;
14    }
15    public char getSymbol() {
16        return symbol;
17    }
18    public void setSymbol(char symbol) {
19        this.symbol = symbol;
20    }
21    public int getNumber() {
22        return number;
23    }
24    public void setNumber(int number) {
25        this.number = number;
26    }
27    @Override
28    public String toString() {
29        return "Card [symbol=" + symbol + ", number=" + number + "]";
30    }
31    @Override
32    public int compareTo(card o) {
33        if (this.symbol < o.symbol)
```

The right sidebar shows the Console window with the following output:

```
<terminated> TestMain [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (16-Feb-2024, 12:36:00 pm)
Enter a card:
a
1
Enter a card:
a
2
Enter a card:
a
7
Enter a card:
d
6
Enter a card:
c
2
Enter a card:
d
1
Enter a card:
c
1
Enter a card:
b
2
Four symbols gathered in eight cards.
Cards in Set are:
a 1
b 2
c 2
d 6
```



Experiment No: 2.3

Student Name: Aafreen Khan
Branch: CSE
Semester: 6th
Subject Name: Java Lab

UID:21BCS1397
Section/Group: 606-A
Date :29-02-2024
Subject Code: 21CSH-319

1. 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.

2. Objective:

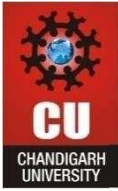
- To learn about various operations in list.

3. Input/Apparatus Used:

- Software Requirements: - Java SDK and VS Code

4. Code:

```
import java.util.*;
public class Main {
    public static void insertItem(ArrayList<String> list, String item) {
        list.add(item);
        System.out.println("Inserted successfully");
    }
    public static void searchItem(ArrayList<String> list, String item) {
        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 deleteItem(ArrayList<String> list, String item) {
        if (list.contains(item)) {
            list.remove(item);
            System.out.println("Deleted successfully");
        } else {
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        System.out.println("Item does not exist.");
    }
}

public static void displayItems(ArrayList<String> list) {
    System.out.println("The Items in the list are :");
    for (String item : list) {
        System.out.println(item);
    }
}

public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
    Scanner scanner = new Scanner(System.in);
    while (true) {
        System.out.println("1. Insert");
        System.out.println("2. Search");
        System.out.println("3. Delete");
        System.out.println("4. Display");
        System.out.println("5. Exit");
        System.out.print("Enter your choice : ");
        int choice = scanner.nextInt();
        scanner.nextLine();
        switch (choice) {
            case 1:
                System.out.print("Enter the item to be inserted: ");
                String itemToInsert = scanner.nextLine();
                insertItem(list, itemToInsert);
                break;
            case 2:
                System.out.print("Enter the item to search : ");
                String itemToSearch = scanner.nextLine();
                searchItem(list, itemToSearch);
                break;
            case 3:
                System.out.print("Enter the item to delete : ");
                String itemToDelete = scanner.nextLine();
                deleteItem(list, itemToDelete);
                break;
            case 4:
                displayItems(list);
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        break;
    case 5:
        scanner.close();
        return;
    default:
        System.out.println("Invalid choice");
    }
    }}}
}
```

5. Result/Output:

```
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice : 1
Enter the item to be inserted: apple
Inserted successfully
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice : 1
Enter the item to be inserted: orange
Inserted successfully
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice : 1
Enter the item to be inserted: melon
Inserted successfully
1. Insert
2. Search
3. Delete
4. Display
5. Exit
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
Enter your choice : 1
Enter the item to be inserted: melon
Inserted successfully
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice : 4
The Items in the list are :
apple
orange
melon
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice : 2
Enter the item to search : apple
Item found in the list.
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice : 
```

6. Learning Outcomes:

- Learned about the concept of classes and how to use them in real world problem.
- Learned how to use various functions on list.



Experiment-2.4

Student Name: Aafreen Khan

UID: 21BCS1397

Branch: BE-CSE

Section/Group: 606-A

Semester: 6th

Date of Performance: 14-03-2024

Subject Name: Project Based Learning in Java with Lab

Subject Code: 21CSP-319

1. Aim: Write a Program to create a menu- based Java 2.4 application with the following options.
1.Add an Employee 2. Display All 3.Exit If option1 is selected, the application should gather details of the employee like employee name, employee id, designation and salary and store it in a file. If option 2 is selected, the application should display all the employee details.

2. Objective: Write a Program to create a menu- based Java 2.4 application with the following options. 1.Add an Employee 2. Display All 3.Exit If option1 is selected, the application should gather details of the employee like employee name, employee id, designation and salary and store it in a file. If option 2 is selected, the application should display all the employee details.

3. Algo./Approach and output:

- 1) Import necessary libraries: java.util.ArrayList: For using an ArrayList to store Employee objects.
- 2) Define the Employee class with the following attributes: name, id, designation, salary.
- 3) Create a public class Main that contains the main method.
- 4) Define a constant FILE_PATH to store the path of the file where employee data will be saved.
- 5) Create an ArrayList employees to store Employee objects
- 6) Implement the addEmployee method to add a new employee to the ArrayList
- 7) Implement the saveEmployeesToFile method to save employee data to a file: Use ObjectOutputStream to write the ArrayList of Employee objects to a file specified by FILE_PATH.

4. Code:

```
import java.io.*;  
import java.util.ArrayList;  
import java.util.Scanner;
```

```
class Employee {  
    String name;  
    int id;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
String designation;
double salary;

public Employee(String name, int id, String designation, double salary) {
    this.name = name;
    this.id = id;
    this.designation = designation;
    this.salary = salary;
}

}

public class Main {
    private static final String FILE_PATH = "employee_data.txt";
    private static ArrayList<Employee> employees = new ArrayList<>();

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

        while (true) {
            System.out.println("Menu:");
            System.out.println("1. Add an Employee");
            System.out.println("2. Display All");
            System.out.println("3. Exit");
            System.out.print("Select an option: ");

            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    addEmployee(scanner);
                    break;
                case 2:
                    displayAllEmployees();
                    break;
                case 3:
                    saveEmployeesToFile();
                    System.out.println("Exiting the application. Goodbye!");
                    System.exit(0);
                    break;
                default:
                    System.out.println("Invalid option. Please try again.");
            }
        }
    }

    private static void addEmployee(Scanner scanner) {
        System.out.print("Enter Employee Name: ");
        String name = scanner.next();
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
System.out.print("Enter Employee ID: ");
int id = scanner.nextInt();
System.out.print("Enter Employee Designation: ");
String designation = scanner.next();
System.out.print("Enter Employee Salary: ");
double salary = scanner.nextDouble();

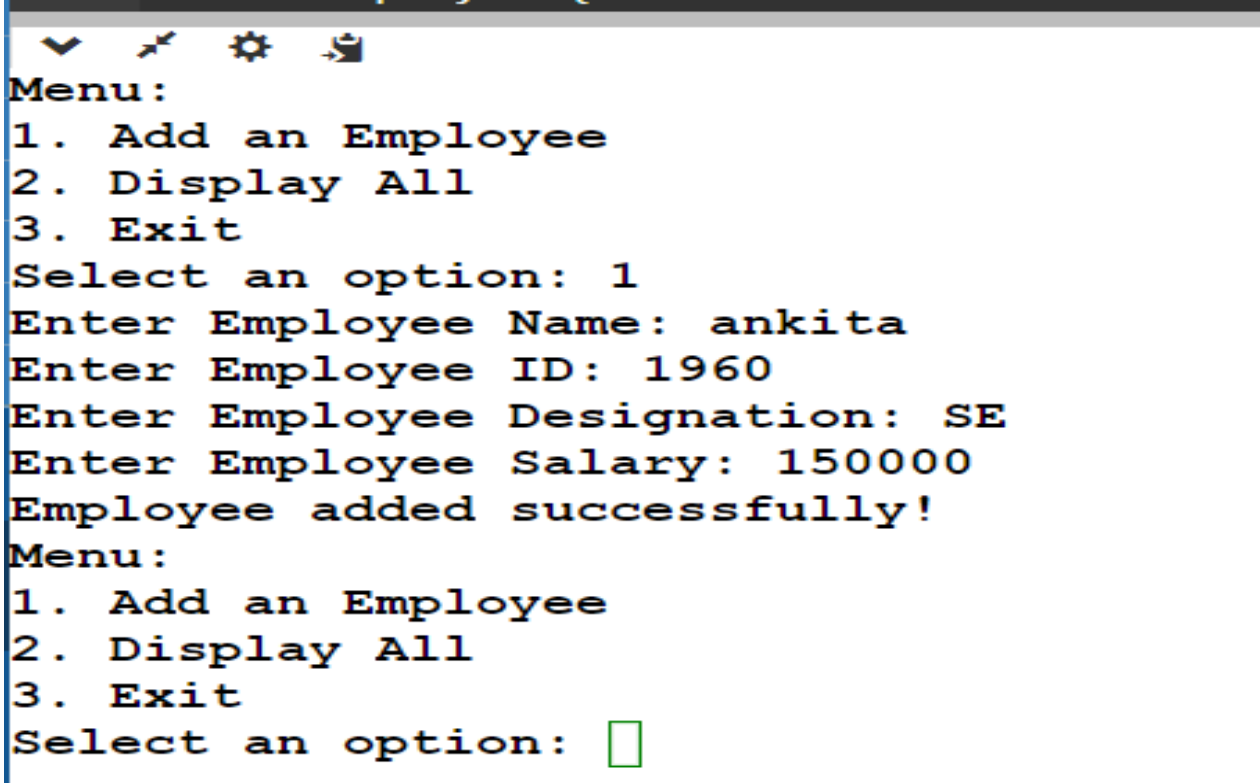
Employee newEmployee = new Employee(name, id, designation, salary);
employees.add(newEmployee);

System.out.println("Employee added successfully!");
}

private static void displayAllEmployees() {
    System.out.println("All Employees:");
    for (Employee employee : employees) {
        System.out.println("Name: " + employee.name);
        System.out.println("ID: " + employee.id);
        System.out.println("Designation: " + employee.designation);
        System.out.println("Salary: " + employee.salary);
        System.out.println(".....");
    }
}

private static void saveEmployeesToFile() {
    try (ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(FILE_PATH))) {
        outputStream.writeObject(employees);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

5. Output:



```
Menu:
1. Add an Employee
2. Display All
3. Exit
Select an option: 1
Enter Employee Name: ankita
Enter Employee ID: 1960
Enter Employee Designation: SE
Enter Employee Salary: 150000
Employee added successfully!
Menu:
1. Add an Employee
2. Display All
3. Exit
Select an option: 
```

6. Learning outcomes:

- 1) Understand the importance of handling `IOException` and proper error handling techniques when working with file operations.
- 2) Learn how to read and write objects to files using `ObjectOutputStream` and `ObjectInputStream`.
- 3) Understand constructors and how they are used to create objects.
- 4) Understand the use of **switch** statements for menu-driven programs to execute different functionalities based on user choices.
- 5) Learn how to manage an infinite loop for continuous user interaction until the user chooses to exit the program.



Experiment 3.3

Student Name: Aafreen Khan

UID: 21BCS1397

Branch: CSE

Section/Group: 606-A

Semester: 6

Date of Performance: 10/04/2024

Subject Name: PBLJ

Subject Code: 21CSH-319

1. AIM: Write a JSP application to demonstrate the expression tag by using mathematical operation. There are multiple checkboxes in the HTML page to perform different mathematical operations. By default, Addition is checked and it adds the given number in the text box. At the time of division operation when we enter any invalid data in the text box, it generates an error message from "error.jsp" page.

2. OBJECTIVE:

- To learn about the concept of Servlet in Java.
- To learn about Apache Server (Tomcat).

3. Algorithm:

- i Initiate the program and prompt the user to enter the first numerical value.
- ii Request the user to input a second numerical value.
- iii Ask the user to choose an arithmetic process: add, subtract, multiply, or divide.
- iv Calculate the outcome using the two provided numbers and the chosen arithmetic process.
- v Exhibit the outcome of the calculation to the user.
- vi In cases where a division by zero is attempted, display an error alert.
- vii Present the final result in the designated output area.

4. CODE:

```
<!DOCTYPE html>
<html>
<head>
  <title>Perform Calculations</title>
  <style>
    body {
      text-align: center;
```



```
background: linear-gradient(90deg, rgba(2,0,36,1) 0%, rgba(9,9,121,1) 35%, rgba(255,0,0,0.5) 100%);
color: black;
}
.container { border: 4px solid
#007bff; padding: 40px;
display: inline-block; text-
align: left; background-
color: #f4f4f4; font-
weight: bold;
}
.result-container { border:
2px solid #007bff;
padding: 10px; display:
none; margin-top: 10px;
text-align: center;
background-color: #f4f4f4;
}
</style>
<script> function
calculate() {
var firstNumber = parseFloat(document.getElementById("firstNumber").value); var
secondNumber = parseFloat(document.getElementById("secondNumber").value);
var operation = document.querySelector('input[name="operation"]:checked').value;
var result; switch (operation) {
case "add": result = firstNumber +
secondNumber; break;
case "subtract":
result = firstNumber - secondNumber;
break;
case "multiply":
result = firstNumber * secondNumber;
break;
case "divide":
result = secondNumber !== 0 ? firstNumber / secondNumber : "Error: Division by zero not allowed";
break; }
document.getElementById("resultDisplay").textContent = "Result: " + result;
document.getElementById("resultContainer").style.display = "block"; }
</script>
</head>
<body>
<h2>Calculator</h2>
<div class="container">
<div>
<input type="radio" name="operation" value="add" checked> Add
<input type="radio" name="operation" value="subtract"> Subtract
<input type="radio" name="operation" value="multiply"> Multiply
<input type="radio" name="operation" value="divide"> Divide
</div>
<!-- The rest of your form goes here -->
</div>
</body>
</html>
```

5. OUTPUT:

Mathematical Operations

☐ Addition ☐ Subtraction ☒ Multiplication ☐ Division

Enter First Number:

Enter Second Number:

Result: 1000

Mathematical Operations

☐ Addition ☐ Subtraction ☐ Multiplication ☒ Division

Enter First Number:

Enter Second Number:

Result: Cannot divide by zero



Experiment No. 3.1

Student Name: Aafreen Khan

UID: 21BCS1397

Branch: BE-CSE

Section/Group: 606-A

Semester: 6th

Date of Performance: 04/04/24

Subject Name: Project based learning in Java with Lab

Subject Code: 21CSH-319

1. **Aim:** Write a Program for palindrome creator application for making a longest possible palindrome out of given input string.
2. **Objective:** To develop a program which finds the largest palindromic substring out of a given string input by user.

3. Script and Output:

Code:

```
import java.util.Scanner;
```

```
public class Main {  
    public static String longestPalindrome(String s) {  
        if (s == null || s.length() < 1) return "";  
  
        int start = 0;  
        int end = 0;  
  
        for (int i = 0; i < s.length(); i++) {  
            int len1 = expandAroundCenter(s, i, i);  
            int len2 = expandAroundCenter(s, i, i + 1);  
            int len = Math.max(len1, len2);  
            if (len > end - start) {  
                start = i - (len - 1) / 2;  
                end = i + len / 2;  
            }  
        }  
        return s.substring(start, end + 1);  
    }  
  
    private static int expandAroundCenter(String s, int left, int right) {  
        while (left >= 0 && right < s.length() && s.charAt(left) == s.charAt(right)) {  
            left--;  
            right++;  
        }  
    }  
}
```

```

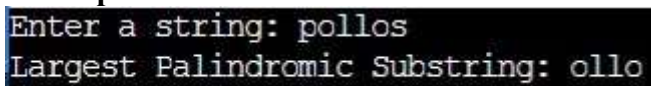
        return right - left - 1;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String s = scanner.nextLine();
        scanner.close();

        System.out.println("Largest Palindromic Substring: " + longestPalindrome(s));
    }

```

4. Output:



```

Enter a string: pollos
Largest Palindromic Substring: ollo

```

Fig 1. Test Case

5. Learning Outcomes

- 1) Develop algorithms to find the largest palindromic substring, focusing on efficiency.
- 2) Practice Java string manipulation and explore the String class.
- 3) Improve Java programming by using standard libraries and coding practices.



Experiment 3.2

Student Name: Aafreen Khan

UID: 21BCS1397

Branch: CSE

Section/Group: 606-A

Semester: 6

Date of Performance: 04/04/2024

Subject Name: Project based learning in Java with Lab

Subject Code: 21CSH-319

1. Aim: Create JSP application for addition, multiplication and division.

2. Objective: The objective of this program is to create an application that uses Java Servlet and takes input from user to add, multiply or divide numbers. The application then returns the output on the interface.

3. Code:

index.jsp:

```
<!DOCTYPE html>
<html>
<head>
  <title>Math Operations</title>
</head>
<body>
  <h2>Math Operations</h2>
  <form action="calculate.jsp" method="post">
    <input type="checkbox" name="operation" value="add" checked>Addition
    <input type="checkbox" name="operation" value="subtract">Subtraction
    <input type="checkbox" name="operation" value="multiply">Multiplication
    <input type="checkbox" name="operation" value="divide">Division<br><br>

    Enter first number: <input type="text" name="num1"><br><br>
    Enter second number: <input type="text" name="num2"><br><br>

    <input type="submit" value="Calculate">
  </form>
</body>
</html>
```

calculate.jsp:

```
<% @ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Result</title>
</head>
<body>
    <h2>Result</h2>
    <%
        String operation = request.getParameter("operation");
        double num1 = Double.parseDouble(request.getParameter("num1"));
        double num2 = Double.parseDouble(request.getParameter("num2"));
        double result = 0;
        String errorMessage = null;

        if ("add".equals(operation)) {
            result = num1 + num2;
        } else if ("subtract".equals(operation)) {
            result = num1 - num2;
        } else if ("multiply".equals(operation)) {
            result = num1 * num2;
        } else if ("divide".equals(operation)) {
            if (num2 != 0) {
                result = num1 / num2;
            } else {
                errorMessage = "Error: Cannot divide by zero.";
                request.setAttribute("error", errorMessage);
                request.getRequestDispatcher("error.jsp").forward(request, response);
            }
        }
    %>

    <% if (errorMessage == null) { %>
        <p>Result of <%= operation %>: <%= result %></p>
    <% } %>
</body>
</html>
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

error.jsp:

```
<% @ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"% >
<!DOCTYPE html>
<html>
<head>
    <title>Error</title>
</head>
<body>
    <h2>Error</h2>
    <p><%= request.getAttribute("error") %></p>
</body>
</html>
```

4. Output:

Math Operations

☒ Addition ☐ Subtraction ☐ Multiplication ☐ Division

Enter first number:

Enter second number:

Result

Result of add: 41.0

5. Learning Outcomes:

- Learned how to use Apache Tomcat to run JSP applications.
- Design and implement a dynamic JSP calculator application.
- Deploy and test the JSP calculator on a web server optimizing performance.