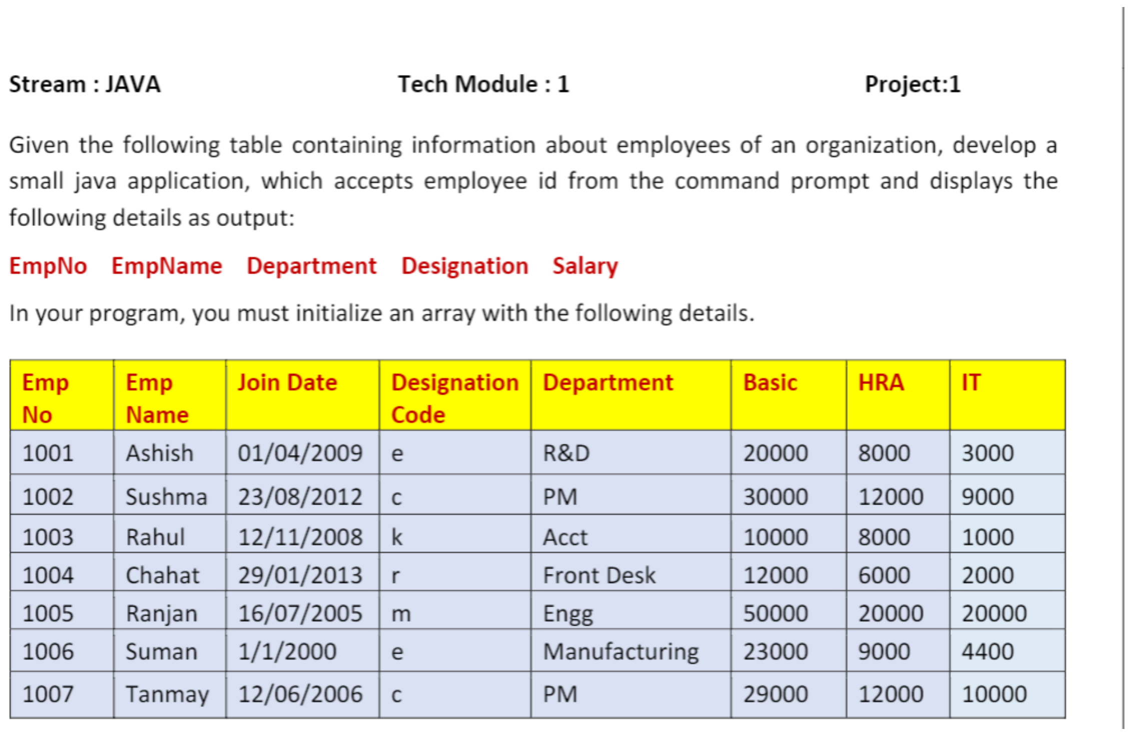
Wipro PBL

TM1 Java fundamentals

Mini project 1



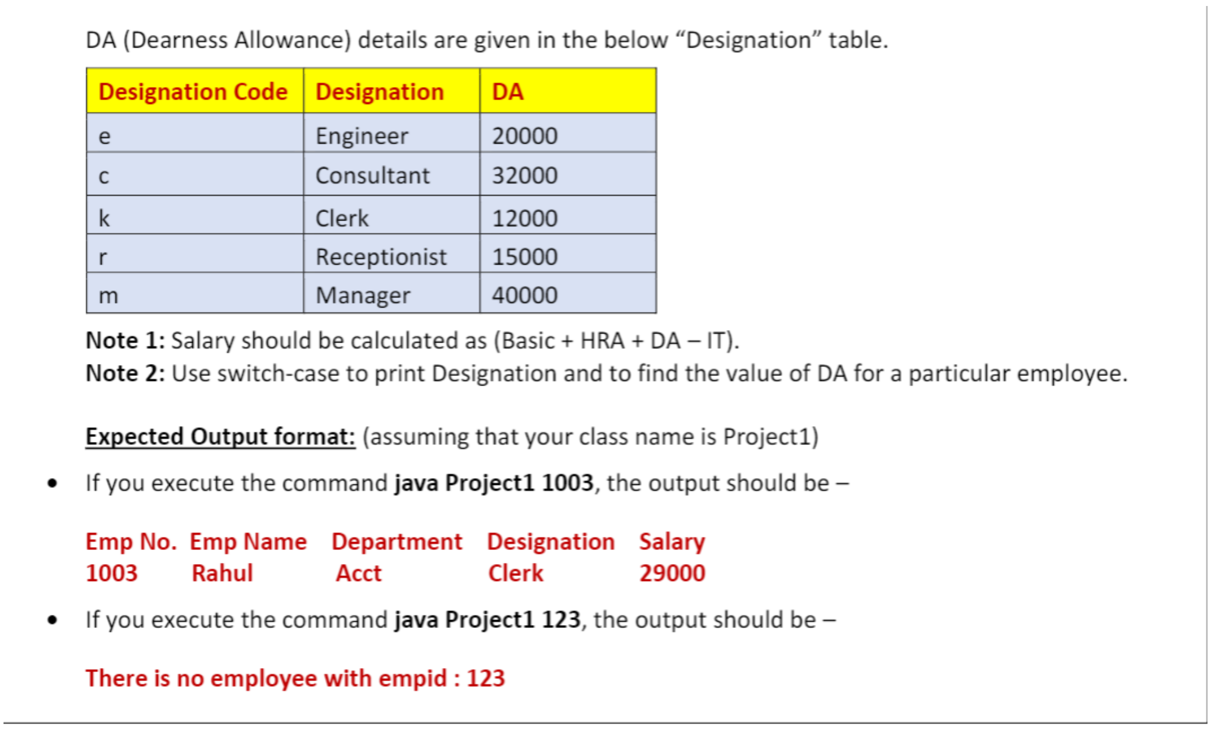
Stream : JAVA **Tech Module : 1** **Project:1**

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 following details as output:

EmpNo EmpName Department Designation Salary

In your program, you must initialize an array with the following details.

Emp No	Emp Name	Join Date	Designation Code	Department	Basic	HRA	IT
1001	Ashish	01/04/2009	e	R&D	20000	8000	3000
1002	Sushma	23/08/2012	c	PM	30000	12000	9000
1003	Rahul	12/11/2008	k	Acct	10000	8000	1000
1004	Chahat	29/01/2013	r	Front Desk	12000	6000	2000
1005	Ranjan	16/07/2005	m	Engg	50000	20000	20000
1006	Suman	1/1/2000	e	Manufacturing	23000	9000	4400
1007	Tanmay	12/06/2006	c	PM	29000	12000	10000



DA (Dearness Allowance) details are given in the below "Designation" table.

Designation Code	Designation	DA
e	Engineer	20000
c	Consultant	32000
k	Clerk	12000
r	Receptionist	15000
m	Manager	40000

Note 1: Salary should be calculated as (Basic + HRA + DA – IT).
Note 2: Use switch-case to print Designation and to find the value of DA for a particular employee.

Expected Output format: (assuming that your class name is Project1)

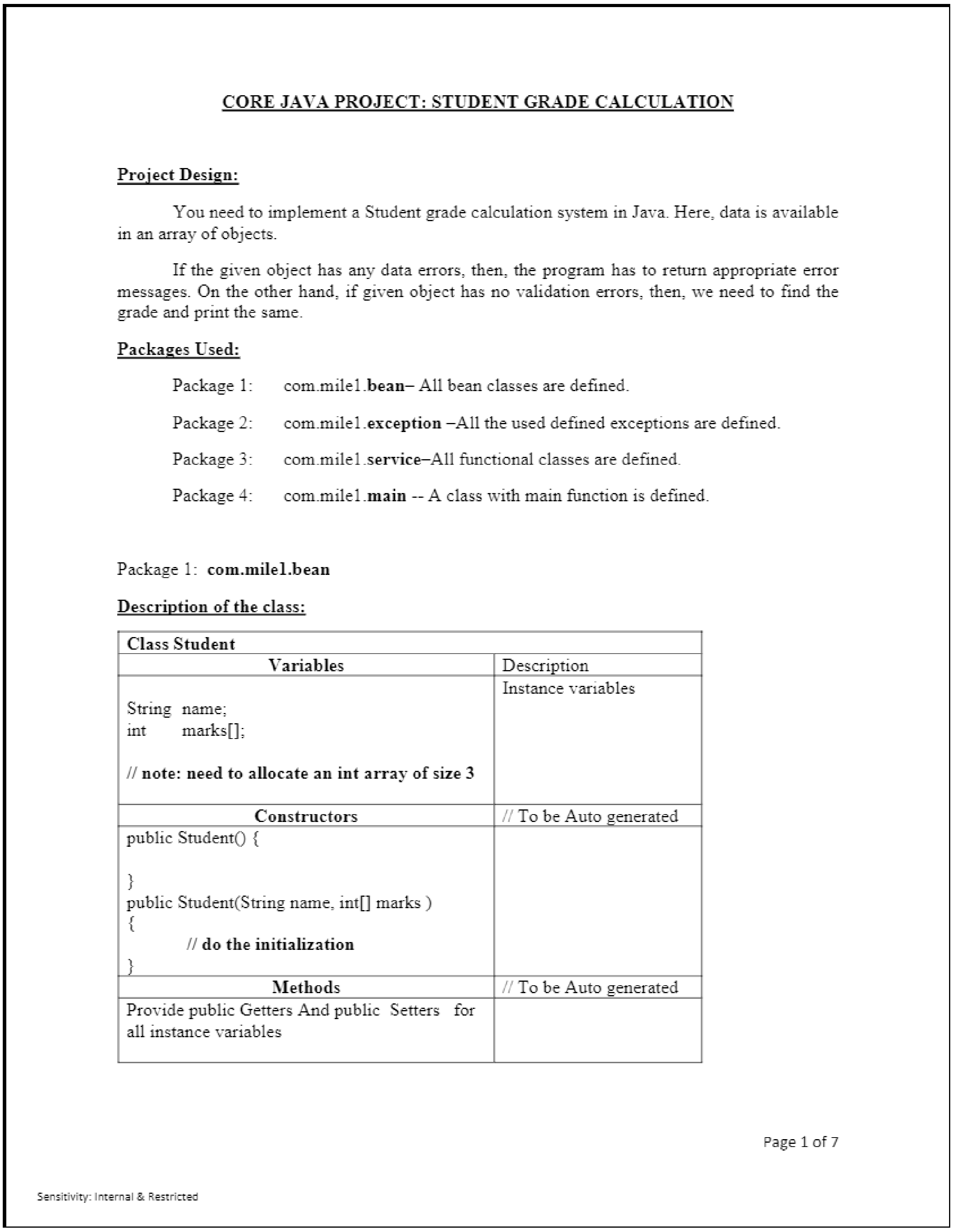
- If you execute the command **java Project1 1003**, the output should be –

```
Emp No.  Emp Name   Department   Designation   Salary
1003     Rahul      Acct         Clerk         29000
```

- If you execute the command **java Project1 123**, the output should be –

```
There is no employee with empid : 123
```

Mini project 2



CORE JAVA PROJECT: STUDENT GRADE CALCULATION

Project Design:

You need to implement a Student grade calculation system in Java. Here, data is available in an array of objects.

If the given object has any data errors, then, the program has to return appropriate error messages. On the other hand, if given object has no validation errors, then, we need to find the grade and print the same.

Packages Used:

Package 1: com.mile1.**bean**– All bean classes are defined.

Package 2: com.mile1.**exception** –All the used defined exceptions are defined.

Package 3: com.mile1.**service**–All functional classes are defined.

Package 4: com.mile1.**main** -- A class with main function is defined.

Package 1: **com.mile1.bean**

Description of the class:

Class Student	
Variables	Description

String name;
int marks[];
 	
// **note: need to allocate an int array of size 3**
	Instance variables

Constructors	// To be Auto generated

public Student() {
 	
}
public Student(String name, int[] marks)
{
// **do the initialization**
}	

Methods	// To be Auto generated
Provide public Getters And public Setters for all instance variables	

Package 2: **com.mile1.exception**

Description of the class:

All the classes in this package should ***extend*** the Exception class.

Class	**Method**	Description

NullMarksArrayException	public String toString()
	{
	}
		Return "NullMarksArrayException occurred" inside the toString() function.

NullNameException	public String toString()
	{
	}
		Return "NullNameException occurred" inside the toString() function.

NullStudentException	public String toString()
	{
	
	}
		Return "NullStudentException occurred" inside the toString() function.

Package 3: **com.mile1.service**

Description of the class:

Class **StudentReport**	
Method1	Description

public String **findGrade** (Student studentObject)
{
// write code here
 	
}
	Only valid objects are passed to this function. So, just concentrate on the logic part.
	
	Get the marks from *studentObject*.
	
	if (any one of the marks is less than 35)
	then return the "F" grade;
	else
	{
	Find the **sum** of all the marks.
	if sum is less than or equal to 150)
	then return "D" grade
	else if sum is greater than 150 and less than or equal to 200, Then return "C" grade.
	else if sum is greater than 200 and less than or equal to 250, then return "B" grade
	else if sum is greater than 250 and less than or equal to 300, then return "A" grade
	
	}

Method2	Description

public String validate
(Student studentObject)
throws NullStudentException,
NullNameException,
NullMarksArrayException
{
 	
// write code here
 	
}
	Check whether there is any null data in the given object.
	
	If given Object itself is null, then,
	{
	Throw the NullStudentException.
	
	}
	Else we do the following:
	{
	
	1) Check whether name is null. If so, throw the NullNameException.
	2) Check whether marks array is null. If so, throw the NullMarksArrayException
	
	If NullNameException and NullMarksArrayException not thrown, all data is valid. We need to call the **findGrade** function that is in the same class.
	Return the message returned by this function.
	
	}

Package3 com.mile1.service

Description of the class:

Class StudentService	
Methods	Description

public int findNumberOfNullMarks
(Student data[])
{
// write code here
}
	This function is used to count the number of objects where the marks array is null.
	
	Note: If you are not careful, you will get NullPointerException in this function.

public int findNumberOfNullNames
(Student data [])
{
 	
// write code here
}
	This function is used to count the number of objects where the name is null.
	
	Note: If you are not careful, you will get NullPointerException in this function.

public int findNumberOfNullObjects (Student data [])
{
    // write code here
}
	This function is used to count the number of objects where the given object itself is null.
	
	Hint:
	To Check whether an object is null, use (obj== null);
	
	Note: If you are not careful, you will get NullPointerException in this function.

Package 4   com.mile1.main

Description of the class:

Class StudentMain

Variables

static Student **data**[] = new Student[4];

```
// use a static block to initialize the objects
static{
        for (int i = 0; i < data.length; i++)       data [i]= new Student();

        data [0] = new Student("Sekar", new int[]{35,35,35});
        data [1] = new Student(null, new int[]{11,22,33});
        data [2] = null;
        data [3] = new Student("Manoj", null);
}
```

MAIN METHOD:
This main function used to call the various functions defined in StudentReport class and StudentService class.

Create an Object for StudentReport and do the following.

1) Call the validate function for all the objects available data [] array.
2) If any exception occurs display, the details of the exception occurred.
3) If no exception raised, then, print the result returned by the ***validate*** *function*.

Create an Object for StudentService. Using this object, do the following:

Call the findNumberOfNullMarks (data) function and print the result.

Call the findNumberOfNullNames (data) function and print the result.

Call the findNumberOfNullObjects (data) function and print the result.

<u>Sample main method looks like this:</u>

```
static   Student data [] = new Student [4];

static {

        for (int i = 0; i < s.length; i++)          data [i] = new Student ();

        data [0] = new Student ("Sekar", new int [] {35, 35, 35});

        data [1] = new Student(null,new int[]{11,22,33});

        data [2] = null;

        data [3] = new Student ("Manoj", null);

}


public static void main (String a []) {

        StudentService studentService = new StudentService ();

        StudentReport studentReport = new StudentReport ();

        System.out.println (" Grades Calculation: ");

        String x = null;

        for (int i = 0; i < data.length; i++) {

                try {x = studentReport.validate (data [i]) ;}

                catch (NullNameException e) {x="NullNameException occurred"; }

                catch (NullMarksArrayException e) {x="NullMarksArrayException occurred";}

                catch (NullStudentException e) { x="NullStudentException occurred "; }

                System.out.println ("GRADE="+x);

        }

        System.out.println ("Number of Objects with Marks array as null ="
                + studentService.findNumberOfNullMarks (data));

        System.out.println ("Number of Objects with Name as null="

                + studentService.findNumberOfNullNames(data));

        System.out.println ("Number of Objects that are entirely null="

                + studentService.findNumberOfNullObjects(data));

 }
```

SAMPLE OUTPUT:

```
Grades Calculation:
GRADE= D
GRADE= NullNameException occurred
GRADE= NullStudentException occurred
GRADE= NullMarksException occurred
Number of Objects with Marks array as null =1
Number of Objects with Name as null=1
Number of Objects that are entirely null=1
```

A NOTE ON TEST CASES:

Your solution is tested with the following set of test cases.

GRADE CALCULATION FOR VALID OBJECT:

TC1 -- Calculate the grade for **valid** objects – Check for A grade computation.

TC2 -- Calculate the grade for **valid** objects – Check for D grade computation.

TC3 -- Calculate the grade for **valid** objects – Check for F grade computation.

THROW ERROR MESSAGE FOR INVALID OBJECT:

Check whether the validate function handles the following situations.

TC4 -- If the Object is null, throw NullStudentException ().

TC5-- If the Name is null, throw NullNameException ().

TC6 -- If the Marks array is null, throw NullMarksArrayException ().

COUNTING THE NULL:

TC7 – Test findNumberOfNullName function.

TC8 – Test findNumberOfNullObjects function.

TC9 -- Test findNumberOfNullMarks function.

SAMPLE INPUT2:

```
data [0] = new Student ("A1", new int [ ] {72, 73, 74});
data [1] = new Student ("B1", new int [ ] {75, 76, 77});
data [2] = new Student ("C1", new int[ ] {99, 99, 99});
data [3] = new Student ("C3", new int[ ] {100, 100, 99});
data [4] = new Student ("B2", new int[ ] {13, 88, 13});
data [5] = new Student ("C3", new int[ ] {14, 14, 99});
data [6] = new Student ("A2", new int[ ] {77, 55, 12});
data [7] = new Student ("A5", new int[ ] {13, 88, 13});
```

SAMPLE OUTPUT2:

```
Grades Calculation:
GRADE= B
GRADE= B
GRADE= A
GRADE= A
GRADE= F
GRADE= F
GRADE= F
GRADE= F
Number of Objects with Marks array as null =0
Number of Objects with Name as null=0
Number of Objects that are entirely null=0
```

//------------------------***************************----------------------------------//

TM2 OOPS/Inheritance

2.1 Classes and objects

No.	Hands-on Assignment	Topics Covered	Status
1	Create a class Box that uses a parameterized constructor to initialize the dimensions of a box.The dimensions of the Box are width, height, depth. The class should have a method that can return the volume of the box. Create an object of the Box class and test the functionalities.	Classes and Objects, Constructor	☐
2	Create a new class called Calculator with the following methods: 1. A static method called powerInt(int num1,int num2) This method should return num1 to the power num2. 2. A static method called powerDouble(double num1,int num2). This method should return num1 to the power num2. 3. Invoke both the methods and test the functionalities. Hint: Use Math.pow(double,double) to calculate the power.	Classes and Objects, Constructor, static	☐
3	Design a class that can be used by a health care professional to keep track of a patient's vital statistics. The following are the details. Name of the class - Patient Member Variables - patientName(String),height(double),width(double) Member Function - double computeBMI() The above method should compute the BMI and return the result. The formula for computation of BMI is weight (in kg) ÷ height*height(in metres). Create an object of the Patient class and check the results.	Classes and Objects, Constructor, static	☐

2.2 Encapsulation/Abstraction

No.	Hands-on Assignment	Topics Covered	Status
1	Create a class Author with the following information. Member variables : name (String), email (String), and gender (char) Parameterized Constructor: To initialize the variables Create a class Book with the following information. Member variables : name (String), author (of the class Author you have just created), price (double), and qtyInStock (int) [Assumption: Each book will be written by exactly one Author] Parameterized Constructor: To initialize the variables Getters and Setters for all the member variables In the main method, create a book object and print all details of the book (including the author details)	Encapsulation / Abstraction	☐

2.3 Inheritance

1	Create a class named 'Animal' which includes methods like eat() and sleep(). Create a child class of Animal named 'Bird' and override the parent class methods. Add a new method named fly(). Create an instance of Animal class and invoke the eat and sleep methods using this object. Create an instance of Bird class and invoke the eat, sleep and fly methods using this object.	Inheritance	☐
2	Create a class called Person with a member variable name. Save it in a file called Person.java Create a class called Employee that will inherit the Person class.The other data members of the Employee class are annual salary (double), the year the employee started to work, and the national insurance number which is a String.Save this in a file called Employee.java Your class should have the necessary constructors and getter/setter methods. Write another class called TestEmployee, containing a main method to fully test your class definition.	Inheritance	☐
3	Create a school application with a class called Person. Create name and dateOfBirth as member variables. Create a class called Teacher that inherits from the Person class. The teacher will have additional properties like salary, and the subject that the teacher teaches. Create a class called Student that inherits from Person class. This class will have a member variable called studentId. Create a class called College Student that inherits from Student class. This class will have collegeName, the year in which the student is studying (first/second/third/fourth) etc. Create objects of each of this classes, invoke and test the methods that are available in these classes.	Inheritance	☐

2.4 Overriding/Polymorphism

No.	Hands-on Assignment	Topics Covered	Status
1	Create a base class Fruit with name ,taste and size as its attributes. Create a method called eat() which describes the name of the fruit and its taste. Inherit the same in 2 other classes Apple and Orange and override the eat() method to represent each fruit taste.	Inheritance / Overriding	☐

2	Write a program to create a class named shape. It should contain 2 methods, draw() and erase() that prints "Drawing Shape" and "Erasing Shape" respectively.
	
	For this class, create three sub classes, Circle, Triangle and Square and each class should override the parent class functions - draw () and erase ().
	
	The draw() method should print "Drawing Circle", "Drawing Triangle" and "Drawing Square" respectively.
	The erase() method should print "Erasing Circle", "Erasing Triangle" and "Erasing Square" respectively.
	
	Create objects of Circle, Triangle and Square in the following way and observe the polymorphic nature of the class by calling draw() and erase() method using each object.
	
	`Shape c=new Circle();`
	`Shape t=new Triangle();`
	`Shape s=new Square();`
		Polymorphism	☐

2.5 Garbage Collection

No Hands On Exercise

2.6 String,StringBuffer

1	Write a Program to check whether a given String is Palindrome or not.	String/StringBuffer	☐
2	Write a java program that will concatenate 2 strings and return the result. The result should be in lowercase. Note:If the concatenation creates a double-char, then one of the characters need to be omitted. Example1) i/p:Sachin,Tendulkar o/p:sachin tendulkar Example2) i/p:Mark,kate o/p:markate	String/StringBuffer	☐
3	Given a string, return a new string made of 'n' copies of the first 2 chars of the original string where 'n' is the length of the string. Example1) i/p:Wipro o/p:WiWiWiWiWi	String/StringBuffer	☐
4	Write a java program that will return the first half of the string, if the length of the string is even. It should return null for odd length string. Example1) i/p:TomCat o/p:Tom Example2) i/p:Apron o/p:null	String/StringBuffer	☐

5	`Write a java program that accepts a string and returns a new string without the first and last character of the input string.`
	
	`Example1)`
	`i/p:Suman`
	`o/p:uma`
		String/StringBuffer	☐

6	`Given 2 strings, a and b, return a new string of the form short+long+short, with the shorter string on the outside and the longer string on the inside.`
	
	`The strings will not be the same length, but they may be empty (length 0).`
	
	`If input is "hi" and "hello", then output will be "hihellohi".`
		String/StringBuffer	☐

7	`Given a string, if the first or last chars are 'x', return the string without those 'x' chars, otherwise return the string unchanged.`
	
	`If the input is "xHix", then output is "Hi".`
	`If the input is "America", then the output is "America".`
		String/StringBuffer	☐

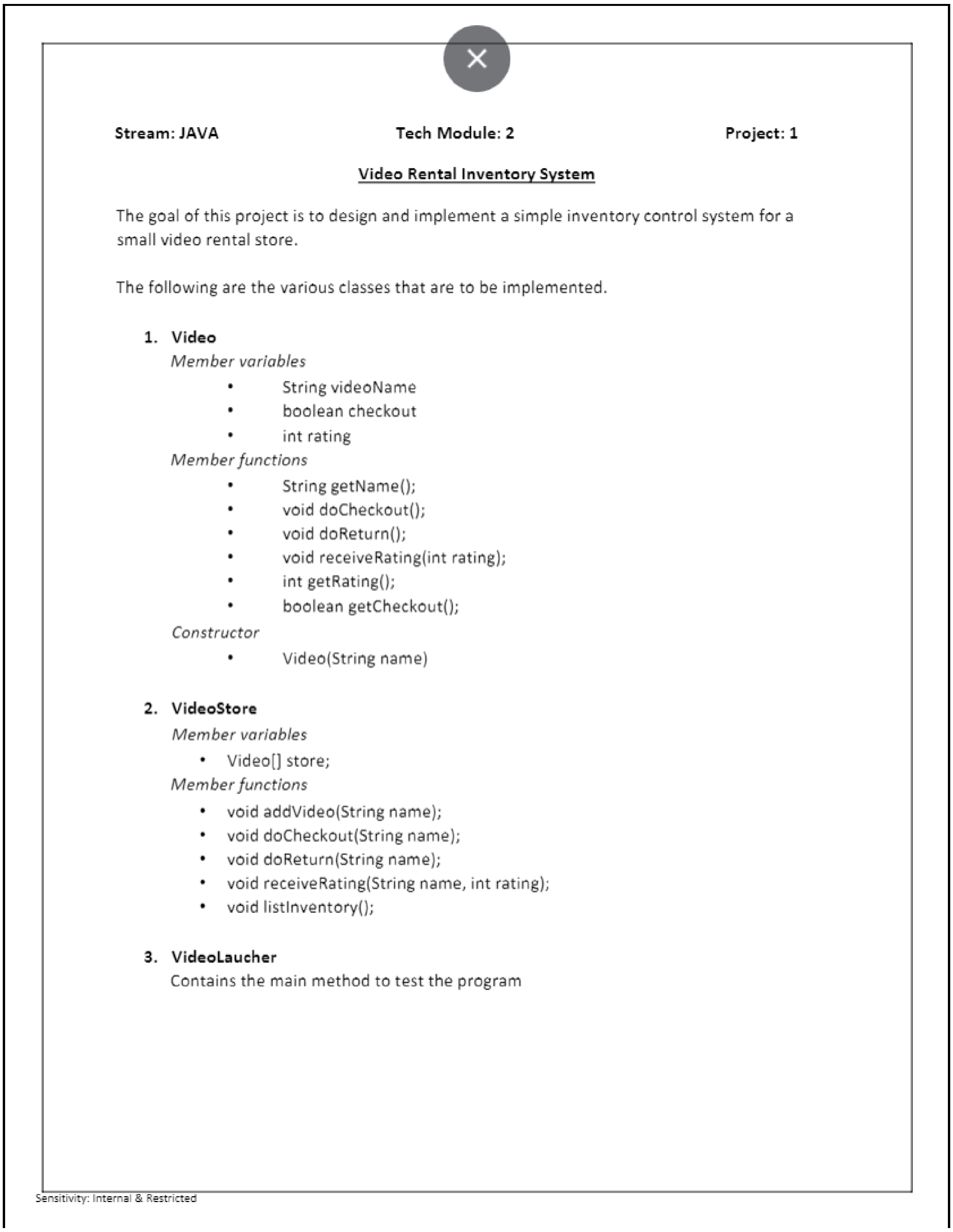
8	`Write a Java program that accepts a string (with * in it). The program should return a new string in which the following characters are removed-*,the characters that are to the left and right of *`
	
	`Example1)`
	`i/p:ab*cd`
	`o/p:ad`
		String/StringBuffer	☐

9	`Given two strings, a and b, print a new string which is made of the following combination-first character of a, the first character of b, second character of a, second character of b and so on. Any characters left, will go to the end of the result.`
	
	`Example1)`
	`i/p:Hello,World`
	`o/p:HWeolrllod`
		String/StringBuffer	☐

10	`Given a string and an integer n, print a new string made of n repetitions of the last n characters of the string.`
	`You may assume that n is between 0 and the length of the string, inclusive.`
	
	`Example1)`
	`i/p:Wipro,3`
	`o/p:propropro`
		String/StringBuffer	☐

11	`Given two strings a and b, return a new string, following the rules given below.`
	`If string b occurs in string a, then the new string should concatenate the characters that appear before and after of String b.`
	`Ignore cases where there is no character before or after the word, and a character may be included twice if it is in between two string b's.`
	
	`Example1)`
	`i/p:abcXY123XYijk,XY`
	`o/p:c13i`
	
	`Example2)`
	`i/p:XY123XY,XY`
	`o/p:13`
	
	`Example3)`
	`i/p:XY1XY,XY`
	`o/p:11`
		String/StringBuffer	☐

2. Mini projects



Stream: JAVA **Tech Module: 2** **Project: 1**

Video Rental Inventory System

The goal of this project is to design and implement a simple inventory control system for a small video rental store.

The following are the various classes that are to be implemented.

1. **Video**
   Member variables
   - String videoName
   - boolean checkout
   - int rating

   Member functions
   - String getName();
   - void doCheckout();
   - void doReturn();
   - void receiveRating(int rating);
   - int getRating();
   - boolean getCheckout();

   Constructor
   - Video(String name)

2. **VideoStore**
   Member variables
   - Video[] store;

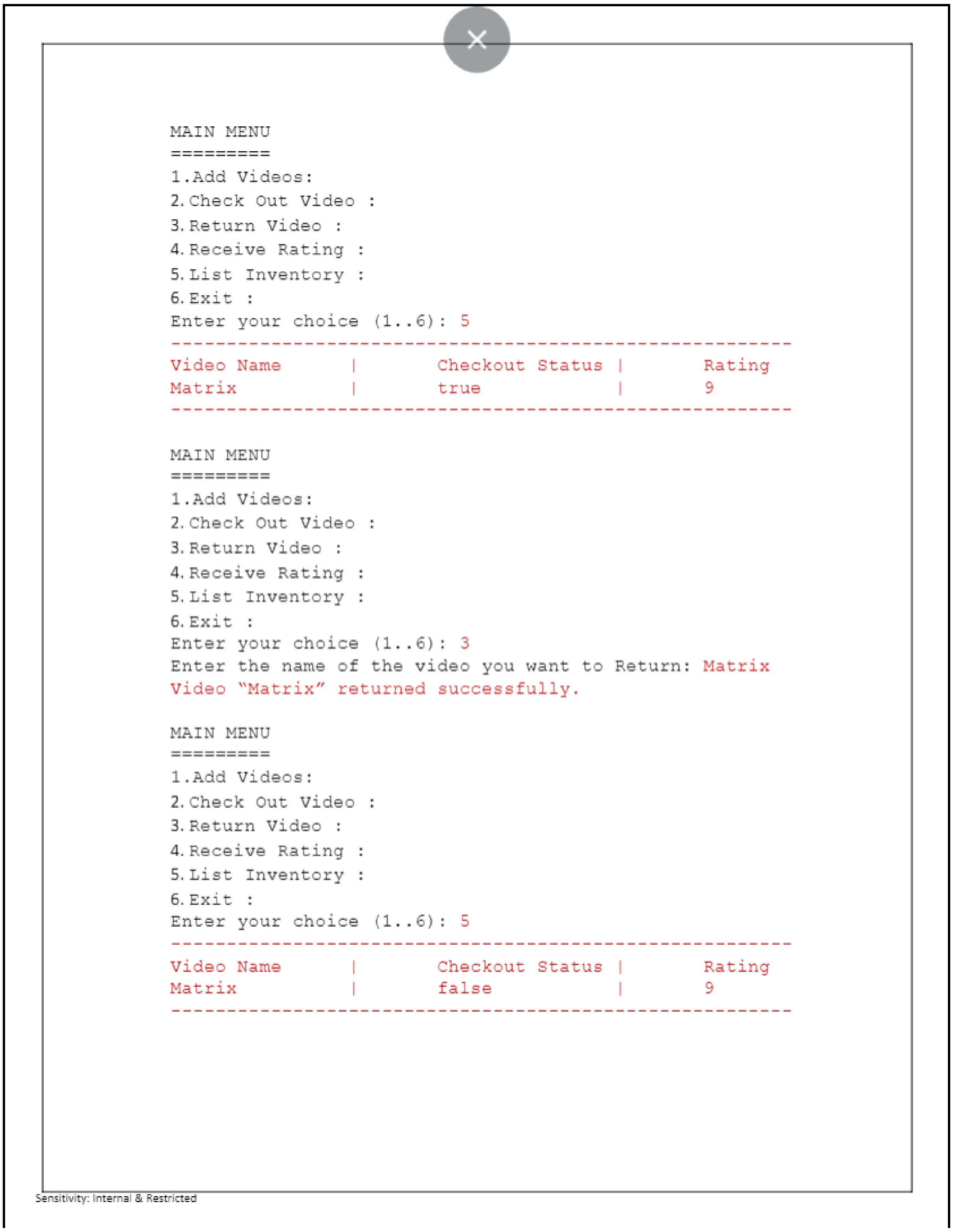
   Member functions
   - void addVideo(String name);
   - void doCheckout(String name);
   - void doReturn(String name);
   - void receiveRating(String name, int rating);
   - void listInventory();

3. **VideoLaucher**
   Contains the main method to test the program

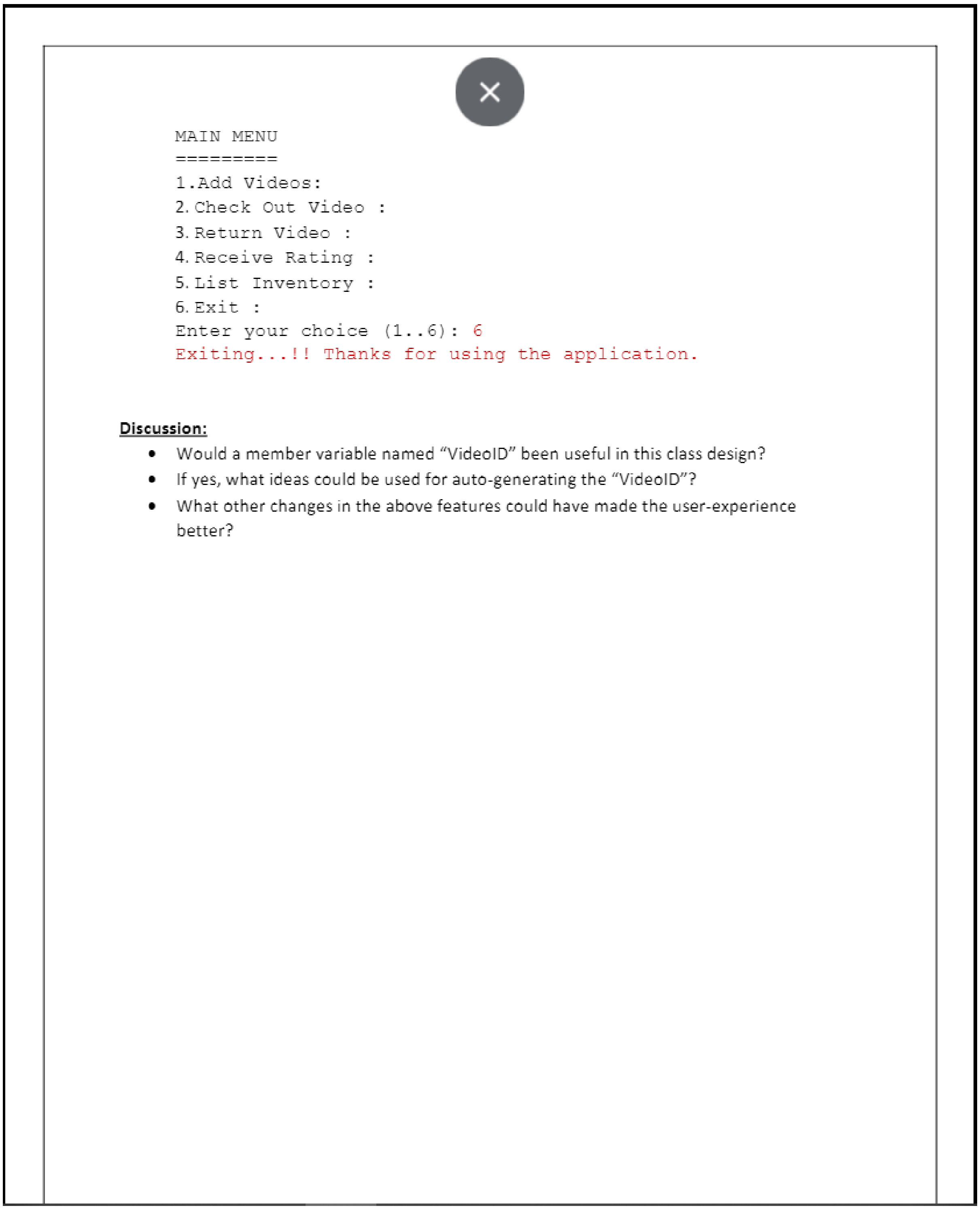
Sensitivity: Internal & Restricted

Sample Output:

```
D:\Batches\Milestone1> java VideoLauncher

MAIN MENU
=========
1.Add Videos:
2.Check Out Video :
3.Return Video :
4.Receive Rating :
5.List Inventory :
6.Exit :
Enter your choice (1..6): 1

Enter the name of the video you want to add: Matrix
Video "Matrix" added successfully.

MAIN MENU
=========
1.Add Videos:
2.Check Out Video :
3.Return Video :
4.Receive Rating :
5.List Inventory :
6.Exit :
Enter your choice (1..6): 4

Enter the name of the video you want to Rate: Matrix
Enter the rating for this video: 9
Rating "9" has been mapped to the Video "Matrix".

MAIN MENU
=========
1.Add Videos:
2.Check Out Video :
3.Return Video :
4.Receive Rating :
5.List Inventory :
6.Exit :
Enter your choice (1..6): 2
Enter the name of the video you want to check out: Matrix
Video "Matrix" checked out successfully.
```



```
MAIN MENU
=========
1.Add Videos:
2.Check Out Video :
3.Return Video :
4.Receive Rating :
5.List Inventory :
6.Exit :
Enter your choice (1..6): 5
--------------------------------------------------------
Video Name       |       Checkout Status |       Rating
Matrix           |       true            |       9
--------------------------------------------------------

MAIN MENU
=========
1.Add Videos:
2.Check Out Video :
3.Return Video :
4.Receive Rating :
5.List Inventory :
6.Exit :
Enter your choice (1..6): 3
Enter the name of the video you want to Return: Matrix
Video "Matrix" returned successfully.

MAIN MENU
=========
1.Add Videos:
2.Check Out Video :
3.Return Video :
4.Receive Rating :
5.List Inventory :
6.Exit :
Enter your choice (1..6): 5
--------------------------------------------------------
Video Name       |       Checkout Status |       Rating
Matrix           |       false           |       9
--------------------------------------------------------
```



```
MAIN MENU
=========
1.Add Videos:
2.Check Out Video :
3.Return Video :
4.Receive Rating :
5.List Inventory :
6.Exit :
Enter your choice (1..6): 6
Exiting...!! Thanks for using the application.
```

Discussion:

- Would a member variable named "VideoID" been useful in this class design?
- If yes, what ideas could be used for auto-generating the "VideoID"?
- What other changes in the above features could have made the user-experience better?

TM3 Abstraction/Packages/Exception Handling

3.1 Abstract Classes

No.	Hands-on Assignment	Topics Covered	Status

1	Create a class called GeneralBank that acts as base class for all banks. This class has getSavingsInterestRate and getFixedDepositInterestRate methods, which returns the savings account interest rate and fixed deposit account interest rate that the specific bank gives. Since GeneralBank cannot say what percentage which bank would give, make these methods abstract.
	
	Create two subclasses of GeneralBank called ICICIBank and KotMBank. Override the inherited methods from the base class. The following are the interest rates of these banks.
	ICICIBank - Savings 4% Fixed 8.5% and
	KotMBank - Savings 6% Fixed 9%.
	
	Create a main method to test the above classes and their methods. Try one by one and observe your findings
	
	a) ICICIBank i=new ICICIBank();
	
	b) KotMBank k=new KotMBank();
	
	c) GeneralBank g=new KotMBank();
	
	d) GeneralBank g=new ICICIBank();
		Abstract Classes	☐

2	Create an abstract class Compartment to represent a rail coach. Provide an abstract function notice in this class. public abstract String notice(); Derive FirstClass, Ladies, General, Luggage classes from the compartment class. Override the notice function in each of them to print notice message that is suitable to the specific type of compartment. Create a class TestCompartment.Write main function to do the following: Declare an array of Compartment of size 10. Create a compartment of a type as decided by a randomly generated integer in the range 1 to 4. Check the polymorphic behavior of the notice method. [i.e based on the random number genererated, the first compartment can be Luggage, the second one could be Ladies and so on..]	Abstract Classes	☐
3	Create an abstract class Instrument which is having the abstract function play. Create three more sub classes from Instrument which is Piano, Flute, Guitar. Override the play method inside all three classes printing a message “Piano is playing tan tan tan tan ” for Piano class “Flute is playing toot toot toot toot” for Flute class “Guitar is playing tin tin tin ” for Guitar class Create an array of 10 Instruments. Assign different type of instrument to Instrument reference. Check for the polymorphic behavior of play method. Use the instanceof operator to print which object is stored at which index of instrument array.	Abstract Classes	☐

3.2 Final Keyword

No Hands On Assignments

3.3 Packages

No.	Hands-on Assignment	Topics Covered	Status
1	Create a package called test package. Define a class called foundation inside the test package. Inside the class, you need to define 4 integer variables: var1 with private access modifier var2 with default access modifier var3 with protected access modifier var4 with public access modifier Import this class and packages in another class. Try to access all 4 variables of the foundation class and see what variables are accessible and what are not accessible.	Packages Access control Using package	☐
2	Create a class called compartment which represents the ship compartments with attributes like height, width and breadth. Take care it should not conflict with the compartment class you have created in Abstract class exercise 2. To avoid conflict create this class in a new package called com.wipro.automobile.ship	Packages User defined packages	☐
3	Create a package called com.automobile. Define an abstract class called Vehicle. Vehicle class has the following abstract methods: public String getModelName() public String getRegistrationNumber() public String getOwnerName() Create twowheeler subpackage under automobile package Hero class extends automobile.vehicle class with the following methods public int getSpeed() - returns the current speed of the vehicle. public void radio() - provides facility to control the radio device Honda class extends com.automobile.vehicle class with the following methods public int getSpeed() - Returns the current speed of the vehicle. public void cdplayer() - provides facility to control the cd player device which is available in the car. Create a test class to test the methods available in all these child class.	Packages User defined packages	☐
4	Add the following ideas to the previous hands on: Create FourWheeler subpackage under automobile package Logan class extends com.automobile.Vehicle class public int speed() - Returns the current speed of the vehicle. public int gps() - provides facility to control the gps device Ford class extends com.automobile.Vehicle class public int speed() - Returns the current speed of the vehicle. public int tempControl() - provides facility to control the air conditioning device which is available in the car Create objects of the relevant classes and test the various functionalities of the class.	Packages User defined packages	☐

3.4 Interfaces

Q1. An online library application need to be created for two types of users/roles-Adults and children. Both of these users should be able to register an account.

Any user who is less than 12 years of age will be registered as a child and they can borrow a "Kids" category book for 10 days, whereas an adult can borrow "Fiction" category books which need to be returned within 7 days.

Note: In future, more users/roles might be added to the library where similar rules will be enforced.

Develop Interfaces and classes for the categories mentioned above.

1. Create an interface LibraryUser with the following methods

void registerAccount()

void requestBook

()

2. Create 2 classes "KidUsers" and "AdultUser" which implements the LibraryUser interface.

3. Both the classes should have two instance variables - age(int),bookType(String)

4. The methods in the KidUser class should perform the following logic.

registerAccount():

if age < 12, a message displaying "You have successfully registered under a Kids Account" should be displayed in the console.

If(age>12), a message displaying, “Sorry, Age must be less than 12 to register as a kid” should be displayed in the console.

requestBook():

if bookType is “Kids”, a message displaying “Book Issued successfully, please return the book within 10 days” should be displayed in the console,else, a message displaying, “Oops, you are allowed to take only kids books” should be displayed in the console.

5. The methods in the AdultUser class should perform the following logic.

registerAccount():

if age > 12, a message displaying “You have successfully registered under an Adult Account” should be displayed in the console.

If age<12, a message displaying, “Sorry, Age must be greater than 12 to register as an adult” should be displayed in the console.

requestBook function:

if bookType is “Fiction”, a message displaying “Book Issued successfully, please return the book within 7 days” should be displayed in the console., else, a message displaying, “Oops, you are allowed to take only adult Fiction books” should be displayed in the console.

6. Create a class “LibraryInterfaceDemo.java” with a main method and test the functionalities by creating objects of KidUser and AdultUser classes.

2	Write an interface called Playable, with a method void play(); Let this interface be placed in a package called music. Write a class called Veena which implements Playable interface. Let this class be placed in a package music.string Write a class called Saxophone which implements Playable interface. Let this class be placed in a package music.wind Write another class Test in a package called live. Then, a. Create an instance of Veena and call play() method b. Create an instance of Saxophone and call play() method c. Place the above instances in a variable of type Playable and then call play()	Interfaces	☐

3.5 Exception handling

Q1

```
Get an input String from user and parse it to integer, if it is not a number it
will throw number format exception Catch it and print "Entered input is not a
valid format for an integer." or else print the square of that number. (Refer
Sample Input and Output).


Sample input and output 1:

Enter an integer: 12

The square value is 144

The work has been done successfully


Sample input and output 2:

Enter an integer: Java

Entered input is not a valid format for an integer.
```

☐ Q2

```
 Write a program that takes as input the size of the array and the elements in the
array. The program then asks the user to enter a particular index and prints the
element at that index.


This program may generate Array Index Out Of Bounds Exception. Use exception
handling mechanisms to handle this exception. In the catch block, print the class
name of the exception thrown.


Sample Input and Output 1:

Enter the number of elements in the array

3

Enter the elements in the array
```

```
20

90

4

Enter the index of the array element you want to access

2

The array element at index 2 = 4

The array element successfully accessed


Sample Input and Output 2:

Enter the number of elements in the array

3

Enter the elements in the array

20

90

4

Enter the index of the array element you want to access

6

java.lang.ArrayIndexOutOfBoundsException
```

☐ Q3

```
Write a program that takes as input the size of the array and the elements in the
array. The program then asks the user to enter a particular index and prints the
element at that index. Index  starts from zero.


This program may generate Array Index Out Of Bounds Exception  or
NumberFormatException .  Use exception handling mechanisms to handle this
exception.


Sample Input and Output 1:

Enter the number of elements in the array

2

Enter the elements in the array

50

80

Enter the index of the array element you want to access
```

```
1

The array element at index 1 = 80

The array element successfully accessed



 Sample Input and Output 2:

Enter the number of elements in the array

2

Enter the elements in the array

50

80

Enter the index of the array element you want to access

9

java.lang.ArrayIndexOutOfBoundsException



 Sample Input and Output 3:

Enter the number of elements in the array

2

Enter the elements in the array

30

j

java.lang.NumberFormatException
```

☐ Q4

```
 Write a class MathOperation which accepts 5 integers through command line. Create
an array using these parameters. Loop through the array and obtain the sum and
average of all the elements and display the result.


Various exceptions that may arise like ArithmeticException, NumberFormatException,
and so on should be handled.
```

☐ Q5

```
 Write a Program with a division method which receives two integer numbers and
performs the division operation.
```

The method should declare that it throws ArithmeticException. This exception should be handled in the main method.

☐ Q6

Write a Program to take care of Number Format Exception if user enters values other than integer for calculating average marks of 2 students. The name of the students and marks in 3 subjects are taken from the user while executing the program.

In the same Program write your own Exception classes to take care of Negative values and values out of range (i.e. other than in the range of 0-100)

☐ Q7

A student portal provides user to register their profile. During registration the system needs to validate the user should be located in India. If not the system should throw an exception.

Step 1: Create a user defined exception class named “InvalidCountryException”.

Step 2: Overload the respective constructors.

Step 3: Create a main class “UserRegistration”, add the following method,

void registerUser(String username,String userCountry) with the below implementation

• if userCountry is not equal to “India” throw a InvalidCountryException with the message “User Outside India cannot be registered”

• if userCountry is equal to “India”, print the message “User registration done successfully”

Invoke the method registerUser from the main method with the data specified and see how the program behaves.

Example1)

i/p:Mickey,US

o/p:InvalidCountryException should be thrown.

The message should be “User Outside India cannot be registered”

Example2)

i/p:Mini,India

o/p:User registration done successfully

☐ Q8

Write a program to accept name and age of a person from the command prompt(passed as arguments when you execute the class) and ensure that the age entered is >=18 and < 60.

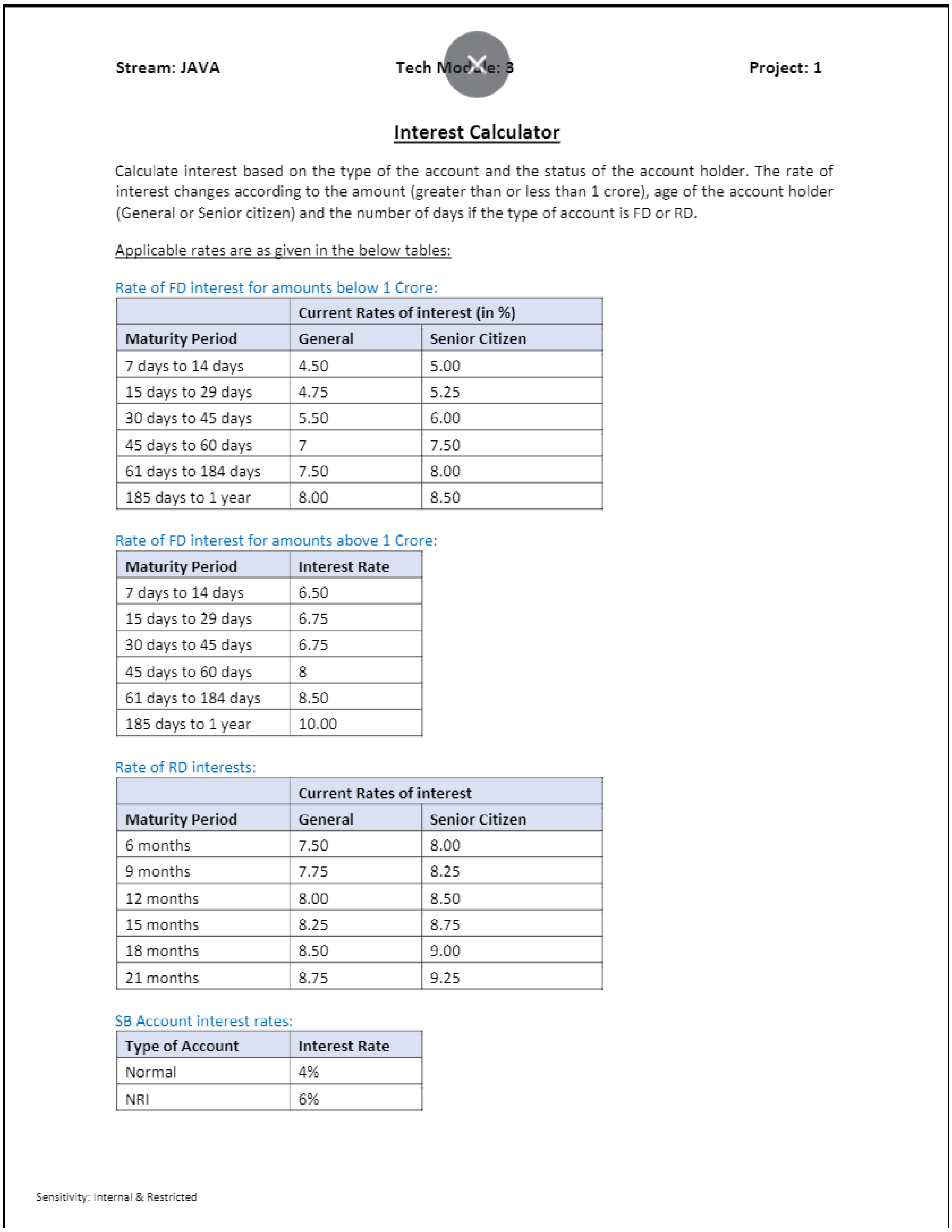
```
Display proper error messages.


The program must exit gracefully after displaying the error message in case the
arguments passed are not proper.

 (Hint : Create a user defined exception class for handling errors.)
```

☐ Q9

```
 Write a program that accepts 2 integers a and b as input and finds the quotient
of a/b.


This program may generate an Arithmetic Exception. Use exception handling
mechanisms to handle this exception.


In the catch block, print the message as shown in the sample output.

Also illustrate the use of finally block. Print the message “Inside finally
block”.


Example1)

Enter the 2 numbers

5

2

The quotient of 5/2 = 2

Inside finally block


Example2)

Enter the 2 numbers

5

DivideByZeroException caught

Inside finally block
```

☐

3.0 Mini project



Stream: JAVA **Tech Module: 3** **Project: 1**

Interest Calculator

Calculate interest based on the type of the account and the status of the account holder. The rate of interest changes according to the amount (greater than or less than 1 crore), age of the account holder (General or Senior citizen) and the number of days if the type of account is FD or RD.

Applicable rates are as given in the below tables:

Rate of FD interest for amounts below 1 Crore:

	Current Rates of interest (in %)	
Maturity Period	**General**	**Senior Citizen**
7 days to 14 days	4.50	5.00
15 days to 29 days	4.75	5.25
30 days to 45 days	5.50	6.00
45 days to 60 days	7	7.50
61 days to 184 days	7.50	8.00
185 days to 1 year	8.00	8.50

Rate of FD interest for amounts above 1 Crore:

Maturity Period	**Interest Rate**
7 days to 14 days	6.50
15 days to 29 days	6.75
30 days to 45 days	6.75
45 days to 60 days	8
61 days to 184 days	8.50
185 days to 1 year	10.00

Rate of RD interests:

	Current Rates of interest	
Maturity Period	**General**	**Senior Citizen**
6 months	7.50	8.00
9 months	7.75	8.25
12 months	8.00	8.50
15 months	8.25	8.75
18 months	8.50	9.00
21 months	8.75	9.25

SB Account interest rates:

Type of Account	**Interest Rate**
Normal	4%
NRI	6%

Stream: JAVA **Tech Module: 3** **Project: 1**

Requirements:

1. Separate classes should be created for the different types of accounts.
2. All classes should be derived from an abstract class named 'Account' which should contain a method called 'calculateInterest'.
3. Implement the 'calculateInterest' method according to the type of the account, interest rates, amount and age of the account holder.
4. If the user enters an invalid value (For e.g. negative value) in any field, raise a user defined exception.

Sample class structures are given below:

Account(Abstract)
`double interestRate`
`double amount`
`Abstract double calculateInterest()`

FDAccount
`double interestRate`
`double amount`
`int noOfDays`
`int ageOfACHolder`
`abstract double calculateInterest()`

SBAccount
`double interestRate`
`double amount`
`abstract double calculateInterest()`

RDAccount
`double interestRate`
`double amount`
`int noOfMonths;`
`double monthlyAmount;`
`abstract double calculateInterest()`

Hint: Use method overriding

Sample Output:

```
MAIN MENU
---------
   1. Interest Calculator – SB
   2. Interest Calculator – FD
   3. Interest Calculator – RD
   4. Exit
Enter your option (1..4): 1
Enter the Average amount in your account: 10000
Interest gained: Rs. 400

MAIN MENU
---------
   1. Interest Calculator –SB
   2. Interest Calculator –FD
   3. Interest Calculator –RD
   4. Exit
Enter your option (1..4): 2
Enter the FD amount: 10000
Enter the number of days: 91
Enter your age: 65
Interest gained is: Rs. 800

MAIN MENU
---------
   1. Interest Calculator –SB
   2. Interest Calculator –FD
   3. Interest Calculator –RD
   4. Exit
Enter your option (1..4): 2
Enter the FD amount: 10000
Enter the number of days:  91
Enter your age:  34
Interest gained is: Rs. 750

MAIN MENU
---------
   1. Interest Calculator –SB
   2. Interest Calculator –FD
   3. Interest Calculator –RD
   4. Exit
Enter your option (1..4): 2
Enter the FD amount: 10000
Enter the number of days: -7
Enter your age: 78
Invalid Number of days. Please enter non-negative values.
```

TM4 Junit

4.1 Introduction to Junit

No Hands on Assignments

4.2 Junit with eclipse

1	Class Name: Demo1 Method: String stringConcat(String,String) [returns concatenation of the 2 Strings received] Create the above class and method and test it using JUnit.

4.3 Assert methods and annotations

Q1

```
 Create a class Employee and implement the below method in the class.
public String findName(ArrayList employees,String name){
  String result="";
  if(employees.contains(name)){
   result="FOUND";
  }else{
   result="NOT FOUND";
  }
  return result;
 }
Write JUnit testcases to test the above method.
```

☐ Q2

```
 i) Create the following class and implement the method to check whether the given
string is a palindrome and return the result.


Class Name : Demo2
Method : palindromeCheck(String):boolean


(Hint: A String is palindrome,  If the reversed string is equal to the actual
string. Ex: madam, mom, dad, malayalam )
ii) Create a Junit test class to test the above class.
```

4.4 Test Suite

```
 Create a test suite for all the classes created in this tech module and
execute the same
```

4.0 Mini Project

S No.	JUnit – Mini Project
1	Using JUnit, test the mini project that you have created in OOPS/Inheritance Tech module
2	Using JUnit, test the mini project that you have created in Abstraction / Packages / Exception Handling Tech module

TM5 Wrapper classes

5.1 Wrapper classes

1	Write a java program that generates the minimum and maximum value for each of the Numeric Wrapper classes (Byte, Short, nteger, Long, Float, Double) Sample Output: Integer range: min: -2147483648 max: 2147483647 Double range: min: 4.9E-324 max: 1.7976931348623157E308 Long range: min: -9223372036854775808 max: 9223372036854775807 Short range: min: -32768 max: 32767 Byte range: min: -128 max: 127 Float range: min: 1.4E-45 max: 3.4028235E38	Wrapper Class	☐
2	Write a program to receive an integer number as a command line argument, and print the binary, octal and hexadecimal equivalent of the given number. Sample Output: java Test 20 Given Number :20 Binary equivalent :10100	Wrapper Class	☐

	Octal equivalent :24 Hexadecimal equivalent :14		
3	Write a Java program that reads an integer number (between 1 and 255) from the user and prints the binary representation of the number. The answer should be printed as a String. Note: The output displayed should contain 8 digits and should be padded with leading 0s(zeros), in case the returned String contains less than 8 characters. For example, if the user enters the value 16, then the output should be 00010000 and if the user enters the value 100, the output should be 01100100 You are expected to use Integer class conversion method/s described in the PDF file. Use Scanner class to accept user inputs. (Hint : You may use String.format() method for the expected output)	Wrapper Class	☐
4	Create an employee class with properties of your choice. Create an object of this class and also create a clone of the same. After making the clone, change the properties of the original employee object and print the properties of both the original and clone object and note down your observation.		

5.0 Mini projects

No mini project

TM6 I/O Streams

6.1 Introduction to I/O

NO HOA

6.2 I/O Operations

1	Write a program to count the number of times a character appears in a File. [Note : The character check is case insensitive... i.e, 'a' and 'A' are considered to be the same] Sample Input and Output: Enter the file name Input.txt Enter the character to be counted r File 'Input.txt' has 99 instances of letter 'r'.	I/O Streams	☐
2	Write a program to copy contents from one file to another and check the output. Sample Input and Output: Enter the input file name Input.txt Enter the output file name Output.txt File is copied.	I/O Streams	☐
3	Write a program to count the occurrences of each word in an input file and write the word along with its corresponding count in an output file.	I/O Streams	☐

[Note: The words should be sorted alphabetically in the output file]

(Hint : Consider using Map Collection)

For Example, let's assume the following are the contents of inputFile.txt

```
Manoj works at Wipro
Katari works at Wipro
Sureka works at Wipro
Harish works at Wipro
Anitha works at Wipro
Janani works at Wipro
```

```
D:\>Java FileWordCount inputFile.txt outputFile.txt
```

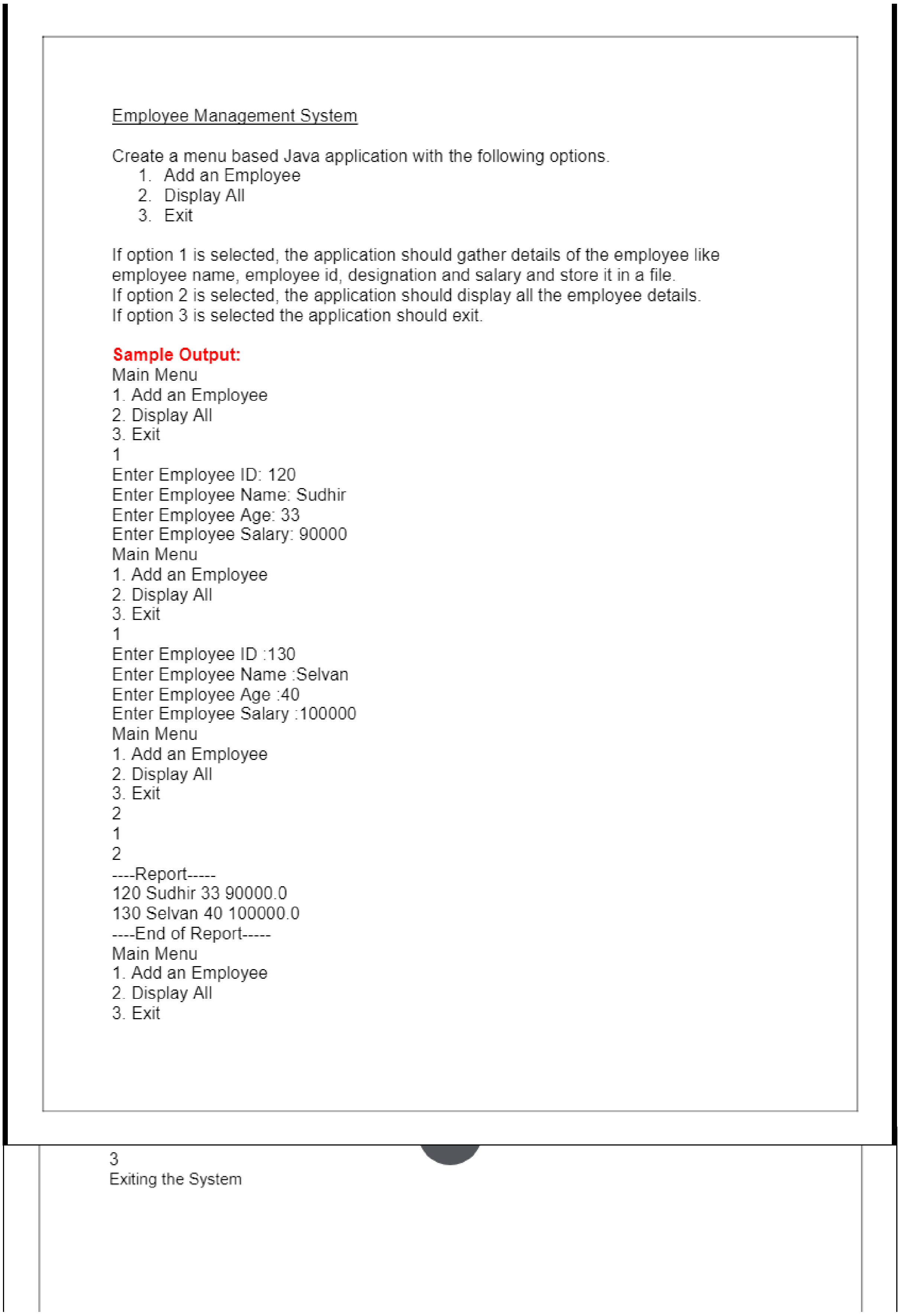
After Execution of the program the contents of outputFile.txt should be as below

```
Anitha : 1
Harish : 1
Janani : 1
Katari : 1
Manoj : 1
Sureka : 1
Wipro : 6
at : 6
works : 6
```

6.3 Object Serialization

1	`Create a class called Employee with properties name(String),dateOfBirth(java.util.Date),department(String),designation(String) and Salary(double).`
	
	`Create respective getter and setter methods and constructors (no-argument constructor and parameterized constructor) for the same.`
	
	`Create an object of the Employee class and save this object in a file called "data" using serialization.`
	
	`Later using deserialization read this object and print the properties of this object.`
		Object Serialization & Deserialization	☐

6.0 Mini project



Employee Management System

Create a menu based Java application with the following options.

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

If option 1 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.
If option 3 is selected the application should exit.

Sample Output:

```
Main Menu
1. Add an Employee
2. Display All
3. Exit
1
Enter Employee ID: 120
Enter Employee Name: Sudhir
Enter Employee Age: 33
Enter Employee Salary: 90000
Main Menu
1. Add an Employee
2. Display All
3. Exit
1
Enter Employee ID :130
Enter Employee Name :Selvan
Enter Employee Age :40
Enter Employee Salary :100000
Main Menu
1. Add an Employee
2. Display All
3. Exit
2
1
2
----Report-----
120 Sudhir 33 90000.0
130 Selvan 40 100000.0
----End of Report-----
Main Menu
1. Add an Employee
2. Display All
3. Exit
3
Exiting the System
```

TM7 Collections

7.1 Introduction to collections

NO HOA

7.2 List

1	Write a Java program to create an ArrayList, add all the months of a year and print the same.	List ArrayList	☐
2	1) Create an application for employee management with the following classes: a) Create an Employee class with following attributes and behaviors : i) int empId ii)String empName iii)String email iv)String gender v)float salary vi) void GetEmployeeDetails() -> prints employee details b) Create one more class EmployeeDB with the following attributes and behaviors. (i)ArrayList list; ii) boolean addEmployee(Employee e) -> adds the employee object to the collection iii) boolean deleteEmployee(int empId) ->delete the employee object from the collection with the given empid iv) String showPaySlip(int empId) -> returns the payslip of the employee with the given empId Provide implementation for all the methods and test your program.	List enumeration / Iterator	☐
3	Create an ArrayList that can store only Strings.	List iterator	☐

	Create a printAll method that will print all the elements of the ArrayList using an Iterator.		
4	Create an ArrayList that can store only numbers like int,float,double,etc, but not any other data type.	List ArrayList	☐
5	Implement the assignment 1 using Linked List	List Linked List	☐
6	Implement the assignment 1 using Vector	List, Vector	☐
7	Write a program that will have a Vector which is capable of storing Employee objects. Use an Iterator and enumeration to list all the elements of the Vector.	List, Vector	

7.3 Generics

NO HOA

7.4 Set

1	Develop a java class with a instance variable H1 (HashSet) add a method saveCountryNames(String CountryName) , the method should add the passed country to a HashSet (H1) and return the added HashSet(H1). Develop a method getCountry(String CountryName) which iterates through the HashSet and returns the country if exist else return null. NOTE: You can test the methods using a main method.	Set HashSet	☐
2	Write a program to store a group of employee names into a HashSet, retrieve the elements one by one using an Iterator.	Set HashSet	☐
3	Create a Collection called TreeSet which is capable of storing String objects. Then try these following operations : a) Reverse the elements of the Collection. b) Iterate the elements of the TreeSet using Iterator. c) Check if a particular element exists or not.	Set TreeSet	☐

4	Implement the assignment 1 using TreeSet	Set TreeSet	☐

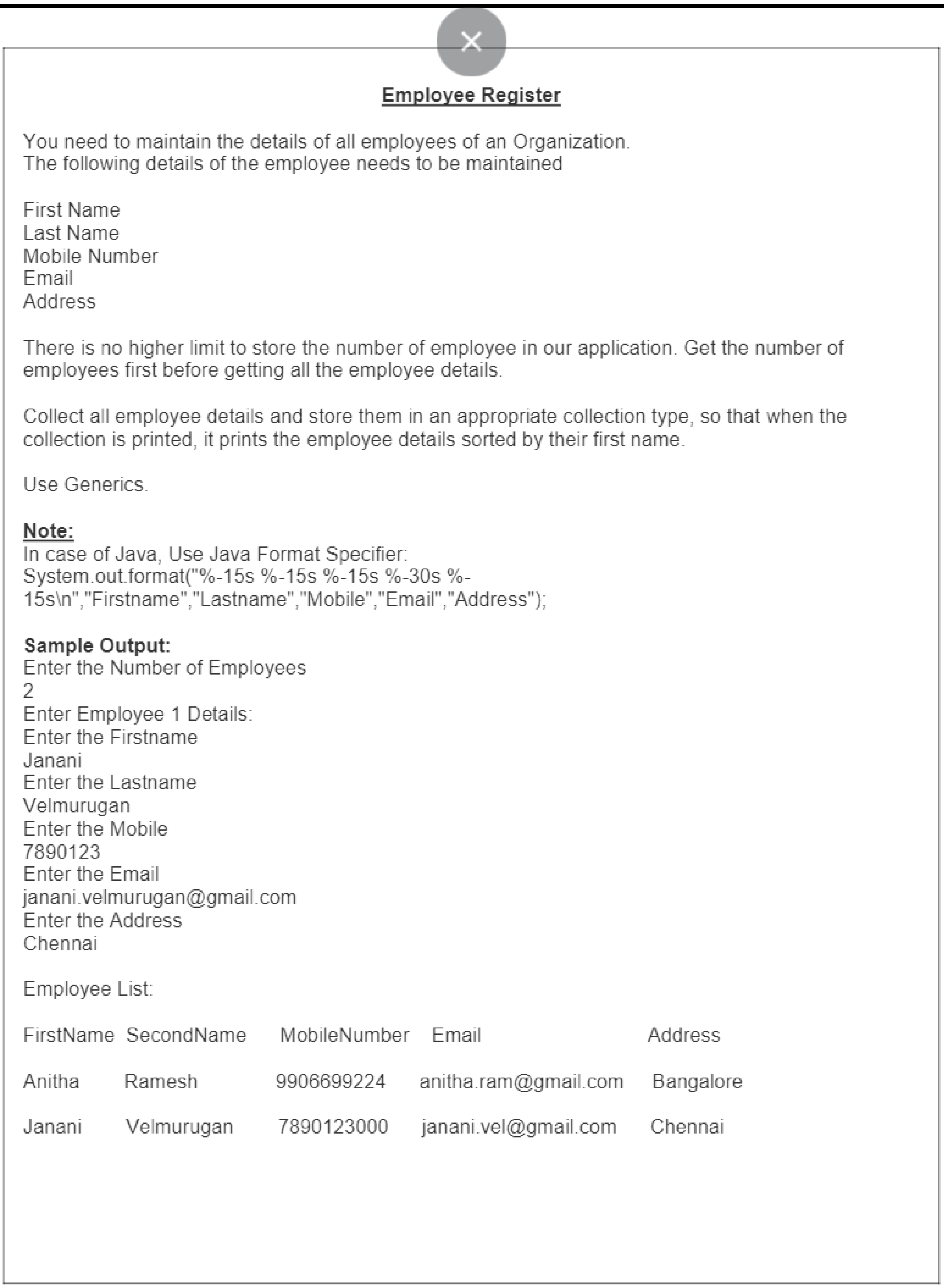
7.5 Map

1	1. Develop a java class with a instance variable M1 (HashMap) create a method saveCountryCapital(String CountryName, String capital) , the method should add the passed country and capital as key/value in the map M1 and return the Map (M1). Key- Country Value - Capital India Delhi Japan Tokyo 2. Develop a method getCapital(String CountryName) which returns the capital for the country passed, from the Map M1 created in step 1. 3. Develop a method getCountry(String capitalName) which returns the country for the capital name, passed from the Map M1 created in step 1. 4. Develop a method which iterates through the map M1 and creates another map M2 with Capital as the key and value as Country and returns the Map M2. Key – Capital Value – Country Delhi India Tokyo Japan 5. Develop a method which iterates through the map M1 and creates an ArrayList with all the Country names stored as keys. This method should return the ArrayList. NOTE: You can test the methods using a main method.	Map HashMap	☐
2	Create a Collection called HashMap which is capable of storing String objects. The program should have the following abilities a) Check if a particular key exists or not. b) Check if a particular value exists or not. c) Use Iterator to loop through the map.	Map HashMap	☐
3	Write a program that will have a Properties class object which is capable of storing some States of India and their	Map HashTable Properties	☐

	Capital. Use an Iterator to list all the elements stored in the Properties.		
4	Create a Collection "ContactList" using HashMap to store name and phone number of contacts added. The program should use appropriate generics (String, Integer) and have the following abilities: a) Check if a particular key exists or not. b) Check if a particular value exists or not. c) Use Iterator to loop through the map.	Map HashMap	☐
5	Implement the assignment 1 using TreeMap	Map TreeMap	☐
6	Implement the assignment 1 using HashTable	Map HashTable	☐

7.0 Mini projects

Mini project 1 Manage employee details



Employee Register

You need to maintain the details of all employees of an Organization.
The following details of the employee needs to be maintained

First Name
Last Name
Mobile Number
Email
Address

There is no higher limit to store the number of employee in our application. Get the number of employees first before getting all the employee details.

Collect all employee details and store them in an appropriate collection type, so that when the collection is printed, it prints the employee details sorted by their first name.

Use Generics.

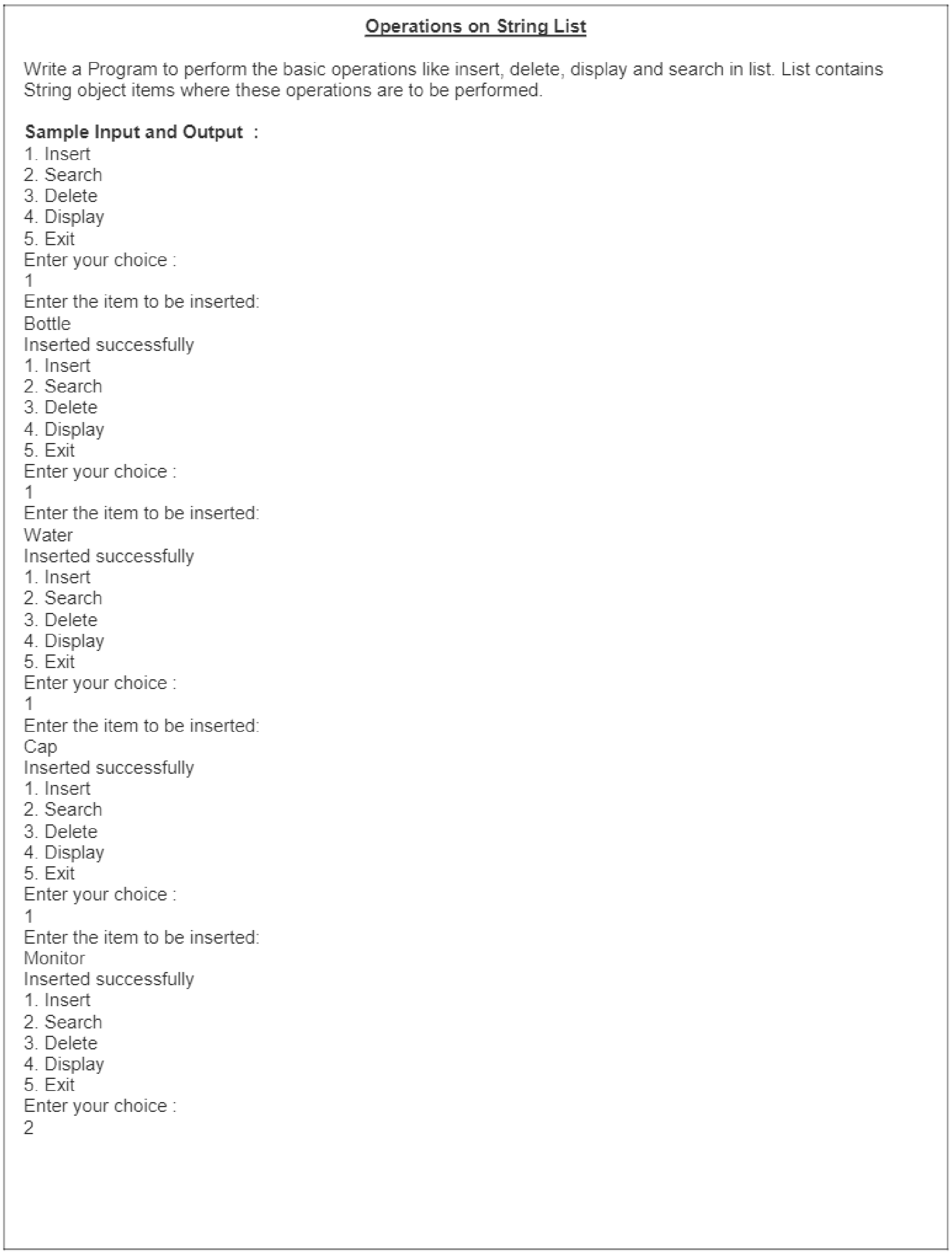
Note:
In case of Java, Use Java Format Specifier:

```
System.out.format("%-15s %-15s %-15s %-30s %-
15s\n","Firstname","Lastname","Mobile","Email","Address");
```

Sample Output:

```
Enter the Number of Employees
2
Enter Employee 1 Details:
Enter the Firstname
Janani
Enter the Lastname
Velmurugan
Enter the Mobile
7890123
Enter the Email
janani.velmurugan@gmail.com
Enter the Address
Chennai

Employee List:

FirstName  SecondName   MobileNumber   Email                   Address

Anitha     Ramesh       9906699224     anitha.ram@gmail.com    Bangalore

Janani     Velmurugan   7890123000     janani.vel@gmail.com    Chennai
```

Mini project 2 String Operations

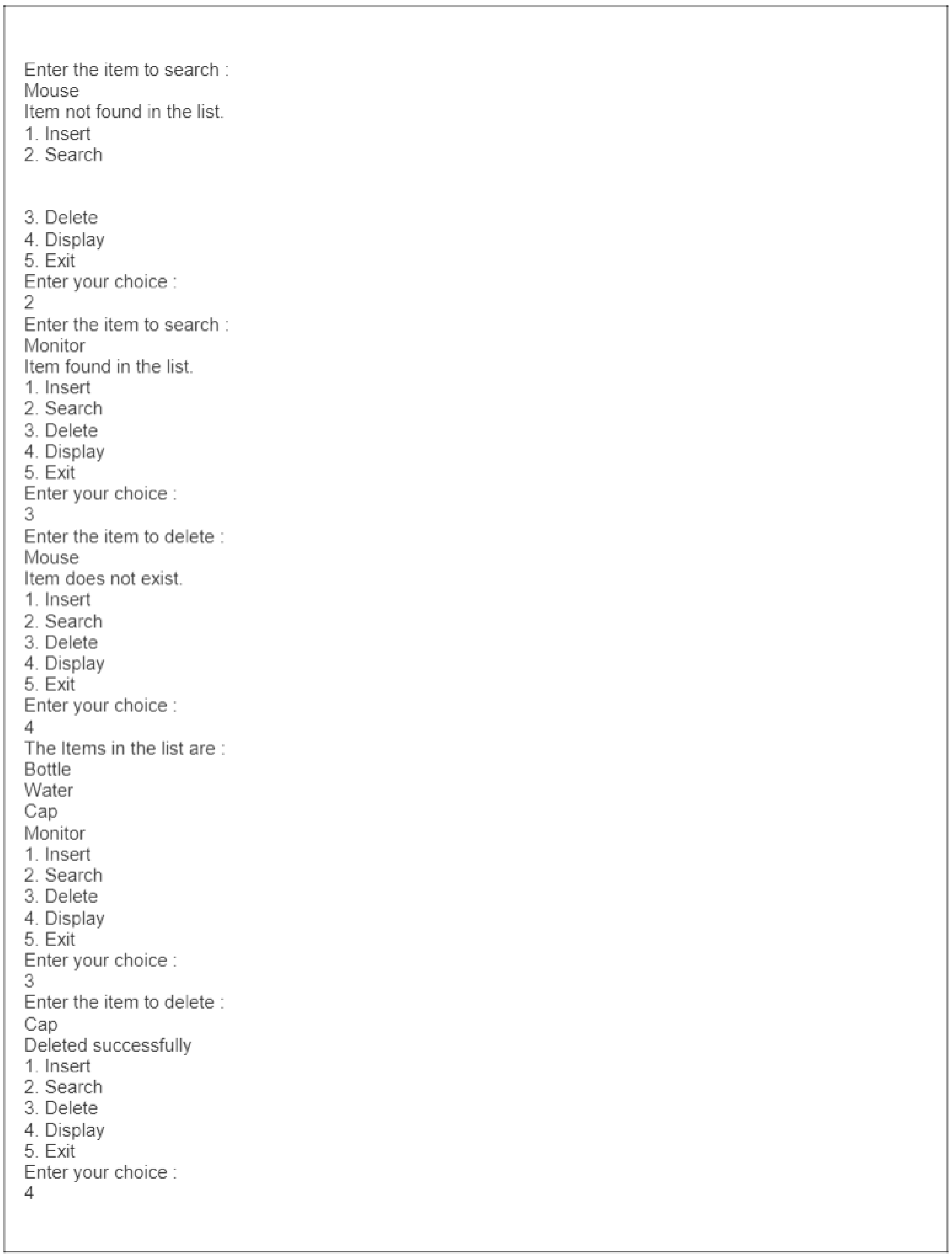


Operations on String List

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.

Sample Input and Output :

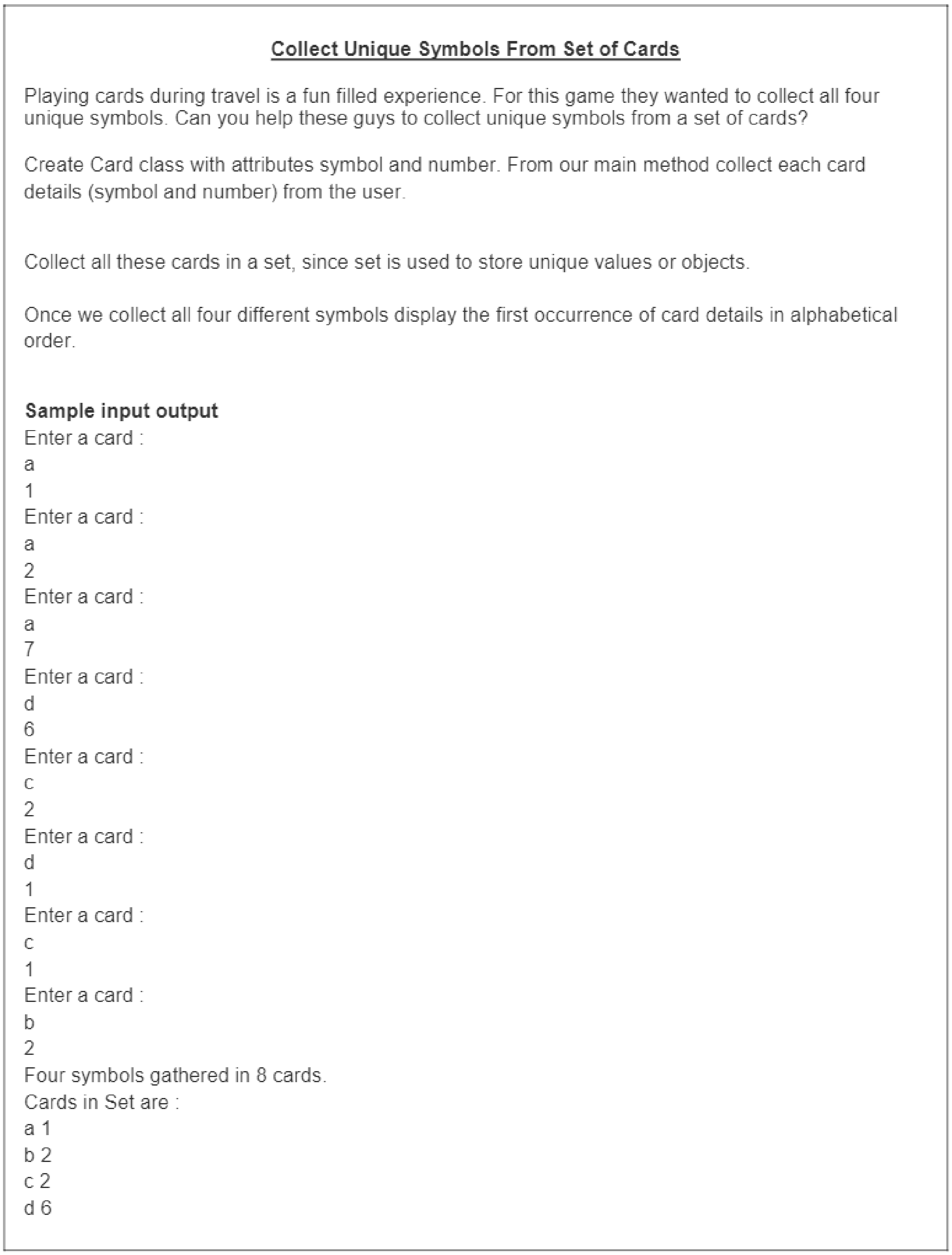
```
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice :
1
Enter the item to be inserted:
Bottle
Inserted successfully
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice :
1
Enter the item to be inserted:
Water
Inserted successfully
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice :
1
Enter the item to be inserted:
Cap
Inserted successfully
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice :
1
Enter the item to be inserted:
Monitor
Inserted successfully
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice :
2
```



```
Enter the item to search :
Mouse
Item not found in the list.
1. Insert
2. Search


3. Delete
4. Display
5. Exit
Enter your choice :
2
Enter the item to search :
Monitor
Item found in the list.
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice :
3
Enter the item to delete :
Mouse
Item does not exist.
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice :
4
The Items in the list are :
Bottle
Water
Cap
Monitor
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice :
3
Enter the item to delete :
Cap
Deleted successfully
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice :
4
```

```
The Items in the list are :
Bottle
Water
Monitor
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice :
5
```

Mini project 3 Collect unique symbols from set of cards



Collect Unique Symbols From Set of Cards

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.

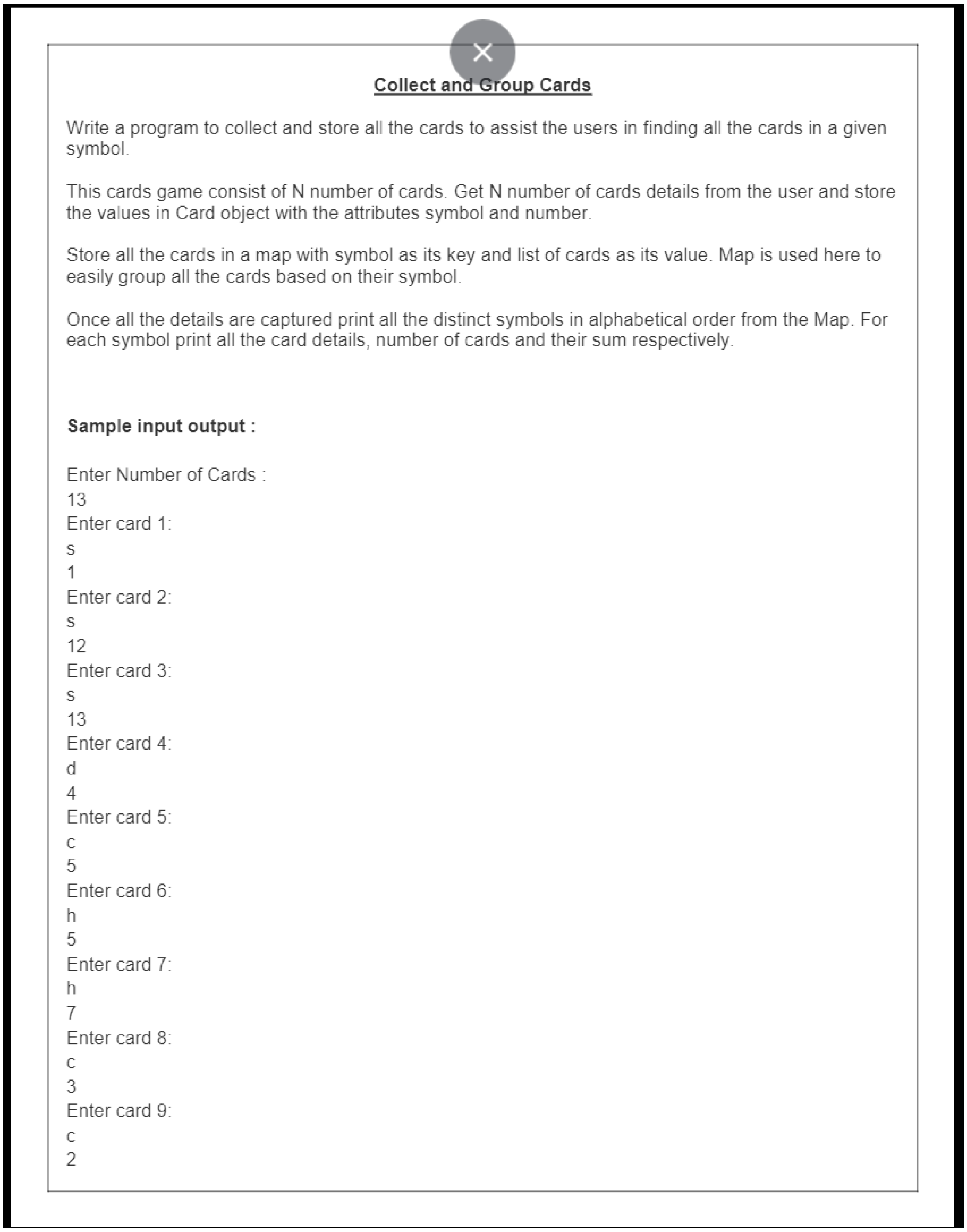
Collect all these cards in a set, since set is used to store unique values or objects.

Once we collect all four different symbols display the first occurrence of card details in alphabetical order.

Sample input output

```
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 8 cards.
Cards in Set are :
a 1
b 2
c 2
d 6
```

Mini project 4 Collect and group cards



Collect and Group Cards

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

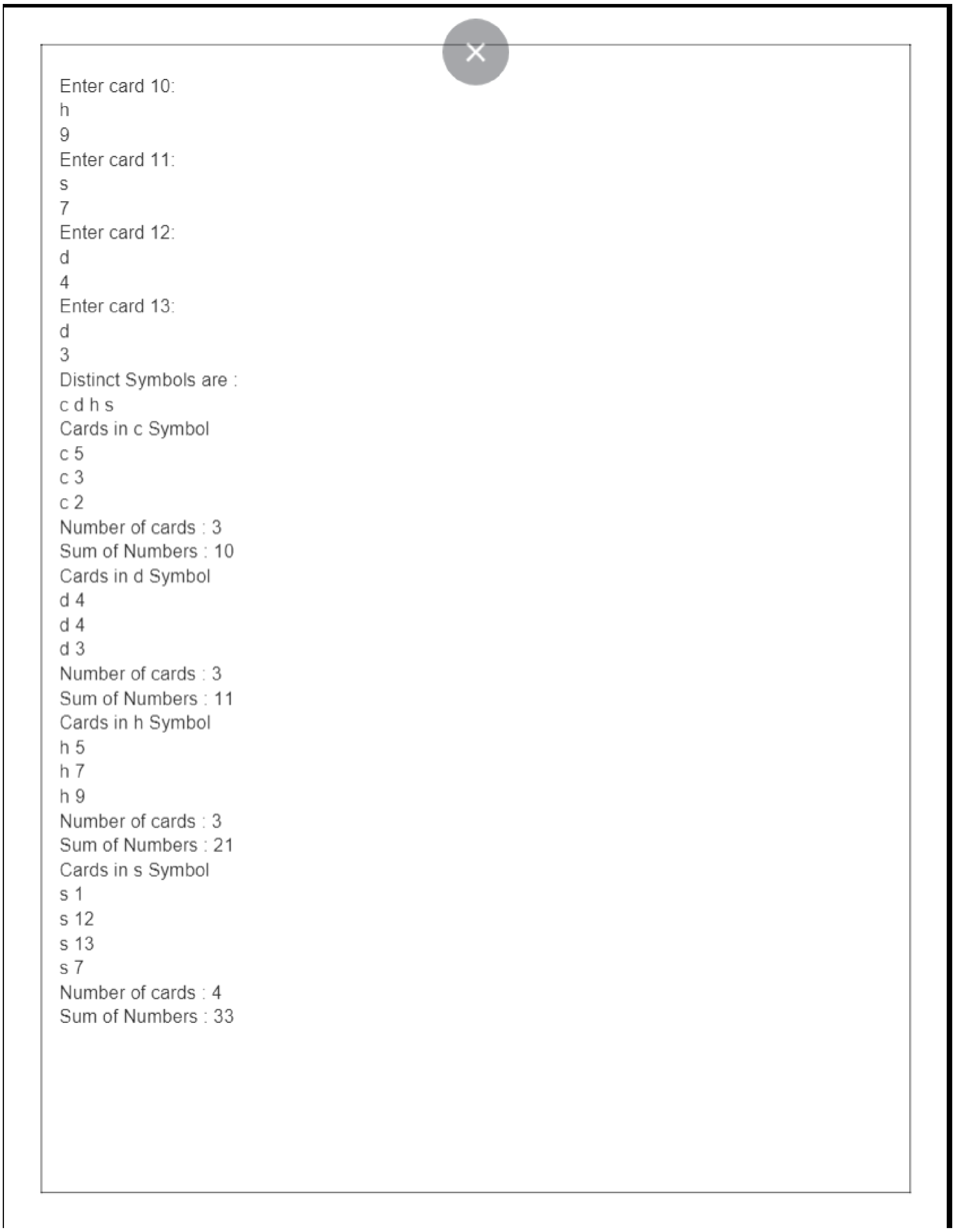
This cards game consist of N number of cards. Get N number of cards details from the user and store the values in Card object with the attributes symbol and number.

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.

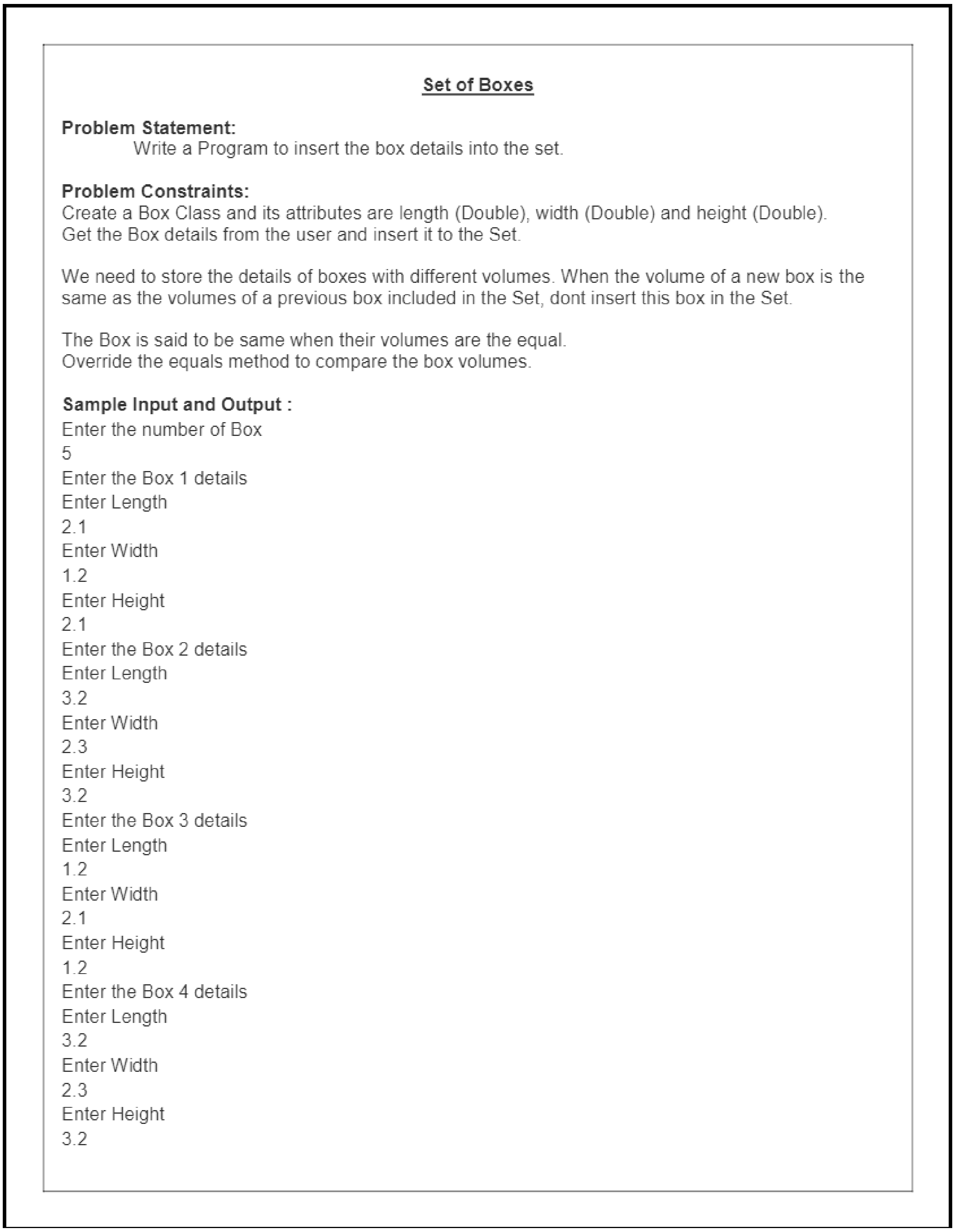
Sample input output :

```
Enter Number of Cards :
13
Enter card 1:
s
1
Enter card 2:
s
12
Enter card 3:
s
13
Enter card 4:
d
4
Enter card 5:
c
5
Enter card 6:
h
5
Enter card 7:
h
7
Enter card 8:
c
3
Enter card 9:
c
2
```



```
Enter card 10:
h
9
Enter card 11:
s
7
Enter card 12:
d
4
Enter card 13:
d
3
Distinct Symbols are :
c d h s
Cards in c Symbol
c 5
c 3
c 2
Number of cards : 3
Sum of Numbers : 10
Cards in d Symbol
d 4
d 4
d 3
Number of cards : 3
Sum of Numbers : 11
Cards in h Symbol
h 5
h 7
h 9
Number of cards : 3
Sum of Numbers : 21
Cards in s Symbol
s 1
s 12
s 13
s 7
Number of cards : 4
Sum of Numbers : 33
```

Mini project 5 Set of Boxes



Set of Boxes

Problem Statement:

Write a Program to insert the box details into the set.

Problem Constraints:

Create a Box Class and its attributes are length (Double), width (Double) and height (Double).
Get the Box details from the user and insert it to the Set.

We need to store the details of boxes with different volumes. When the volume of a new box is the same as the volumes of a previous box included in the Set, dont insert this box in the Set.

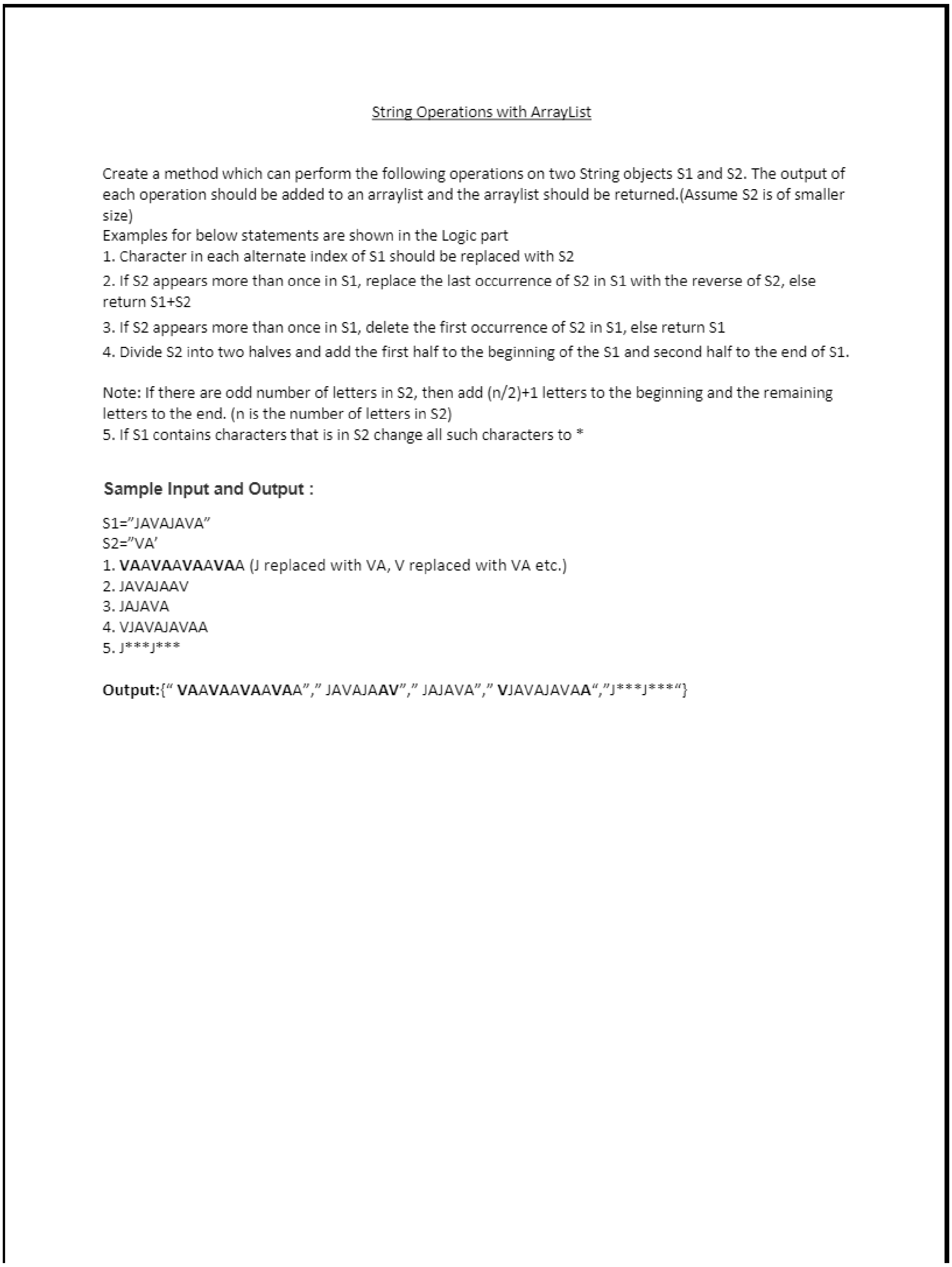
The Box is said to be same when their volumes are the equal.
Override the equals method to compare the box volumes.

Sample Input and Output :

```
Enter the number of Box
5
Enter the Box 1 details
Enter Length
2.1
Enter Width
1.2
Enter Height
2.1
Enter the Box 2 details
Enter Length
3.2
Enter Width
2.3
Enter Height
3.2
Enter the Box 3 details
Enter Length
1.2
Enter Width
2.1
Enter Height
1.2
Enter the Box 4 details
Enter Length
3.2
Enter Width
2.3
Enter Height
3.2
```

```
Enter the Box 5 details
Enter Length
3.3
Enter Width
2.2
Enter Height
1.1
Unique Boxes in the Set are
Length =1.2 Width =2.1 Height =1.2 Volume =3.02
Length =2.1 Width =1.2 Height =2.1 Volume =5.29
Length =3.3 Width =2.2 Height =1.1 Volume =7.99
Length =3.2 Width =2.3 Height =3.2 Volume =23.55
```

Mini project 6 String Operations with ArrayLists



String Operations with ArrayList

Create a method which can perform the following operations on two String objects S1 and S2. The output of each operation should be added to an arraylist and the arraylist should be returned.(Assume S2 is of smaller size)
Examples for below statements are shown in the Logic part
1. Character in each alternate index of S1 should be replaced with S2
2. If S2 appears more than once in S1, replace the last occurrence of S2 in S1 with the reverse of S2, else return S1+S2
3. If S2 appears more than once in S1, delete the first occurrence of S2 in S1, else return S1
4. Divide S2 into two halves and add the first half to the beginning of the S1 and second half to the end of S1.

Note: If there are odd number of letters in S2, then add (n/2)+1 letters to the beginning and the remaining letters to the end. (n is the number of letters in S2)
5. If S1 contains characters that is in S2 change all such characters to *

Sample Input and Output :

S1="JAVAJAVA"
S2="VA'
1. **VA**A**VA**A**VA**A**VA**A (J replaced with VA, V replaced with VA etc.)
2. JAVAJAAV
3. JAJAVA
4. VJAVAJAVAA
5. J***J***

Output:{" **VA**A**VA**A**VA**A**VA**A"," JAVAJA**AV**"," JAJAVA"," **V**JAVAJAVA**A**","J***J***"}

TM8 RDBMS/SQL/JDBC

8.1 Introduction to RDBMS

No HOA

8.2 Oracle 11g Introduction

1	Using SQL Developer: Create a database connection using the following information: Connection Name: myconnection Username: hr Password: hr Hostname: localhost Port: 1521 SID: ORCL Ensure that you select the Save Password check box. Testing and Connecting Using the Oracle SQL Developer Database Connection If the status is Success, connect to the database using this new connection.	Starting up SQL Plus and SQL Developer	☐
2	Expand MyConnection -- > Explore All the available table structure of Employee table - its columns view the data tab of the Employee tables	Starting up SQL Plus and SQL Developer	☐
3	Start SQLPLUS using UserName : hr password : hr	Starting up SQL Plus and SQL Developer	☐

8.3 Select Statement

1	Determine the structure of the DEPARTMENTS table and its contents.	Select Statement	☐
2	Create a query to display the last name, job ID, hire date, and employee ID for each employee, with the employee ID appearing first. Provide an alias STARTDATE for the HIRE_DATE column.	Select Statement	☐
3	Create a query to display all unique job IDs from the EMPLOYEES table.	Select Statement	☐
4	Create a query to display employee id, last name, job id and hiredate from employee table with more describing column names. Name the column headings Emp # , Employee , Job and Hire Date respectively.	Select Statement	☐
5	Create a report of all employees and their job IDs. Display the last name concatenated with the job ID (separated by a comma and space) and name the column as "Employee and Title"	Select Statement	☐

8.4 Restricting and Sorting Data

1	Create a report that displays the last name and salary of employees who earn more than $12,000.	Restricting and Sorting Data	☐
2	Create a report that displays the last name and department number for employee number 176.	Restricting and Sorting Data	☐
3	To find high-salary and low-salary employees. Create a query to display the last name and salary for any employee whose salary is not in the range of $5,000 to $12,000	Restricting and Sorting Data	☐
4	Create a report to display the last name, job ID, and hire date for employees with the last names of Matos and Taylor. Order the query in ascending order by the hire date.	Restricting and Sorting Data	☐

5	Display the last name and department ID of all employees in departments 20 or 50 in ascending alphabetical order by name.	Restricting and Sorting Data	☐
6	List employees who earn between $5,000 and $12,000, and are in department 20 or 50. Label the columns as Employee and Monthly Salary, respectively.	Restricting and Sorting Data	☐
7	Create a report that displays the last name and hire date for all employees who were hired in 1994.	Restricting and Sorting Data	☐
8	Create a report to display the last name and job title of all employees who do not have a manager.	Restricting and Sorting Data	☐
9	Create a report to display the last name, salary, and commission of all employees who earn commissions. Sort data in descending order based on salary and commissions. Use the column's numeric position in the ORDER BY clause.	Restricting and Sorting Data	☐
10	Create a report that displays the last name and salary of employees who earn more than an amount that the user specifies after a prompt. If you enter 12000, it should display all employees earning more than 12000. Eg: Salary_value: 12000	Restricting and Sorting Data	☐
11	Create a query that prompts the user for a manager ID and generates the employee ID, last name, salary and department for that manager's employees and prompts a column name by which result should be sorted. Eg: manager_id :103 sorted_by : last_name	Restricting and Sorting Data	☐
12	Display all employee last names in which the third letter of the name is "a".	Restricting and Sorting Data	☐

13	Display the last names of all employees who have both an “a” and an “e” in their last name.	Restricting and Sorting Data	☐
14	Display the last name, job, and salary for all employees whose jobs are either those of a sales representative or of a stock clerk, and whose salaries are not equal to $2,500, $3,500, or $7,000.	Restricting and Sorting Data	☐

8.5 DML

1	Run the below script Create table MY_EMPLOYEE as Select employee_id,first_name,last_name,department_id,salary from EMPLOYEES where 1=2;	DML	☐

2	Test the table creation by viewing the structure using describe command
	
	`Name                           Null     Type`
	`------------------------------ -------- ----------------------------`
	`--`
	`EMPLOYEE_ID                             NUMBER(6)`
	`FIRST_NAME                              VARCHAR2(20)`
	`LAST_NAME                      NOT NULL VARCHAR2(25)`
	`DEPARTMENT_ID                           NUMBER(4)`
	`SALARY                                  NUMBER(8,2)`
	`5 rows selected`
		DML	☐

3	Insert one record without listing the column names in the insert statement. Check whether data is inserted
	Eg:
	`employee_id    first_name    last_name    department_id     salary`
	`201             Michael       Hartstein      20          13000`
		DML	☐

4	Insert one record without listing the column names in the insert statement where salary value remain undetermined. Check whether data is inserted Eg: employee_id first_name last_name department_id salary 201 Michael Hartstein 20 13000 202 Pat Fay 20 (null)	DML	☐
5	Insert one record with listing the column names avoiding salary column in the insert statement where salary value remain undetermined. Check whether data is inserted employee_id first_name last_name department_id salary 201 Michael Hartstein 20 13000 202 Pat Fay 20 (null) 203 Susan Mavris 40 (null)	DML	☐
6	Use the above Script to insert the below given records employee_id first_name last_name department_id salary 205 Shelley Higgins 110 12000 100 Steven King 90 24000 101 Neena Kochhar 90 17000 102 Lex De Haan 90 17000 111 Ismael Sciarra 100 7700 112 Jose Manuel Urman 100 7800 204 Hermann Baer 70 10000	DML	☐
7	Create a query to increase salary by 10% for all employees in dept 90.	DML	☐
8	Create a query to update Last_name of emp 202 to Higgins.	DML	☐
9	Delete employees whose name either first or last name has char seq of ‘man’	DML	☐

8.6 DDL

1	Create the DEPT table based on the following table instance chart. Save the statement in a script called lab_10_01.sql , and then execute the statement in the script to create the table. Confirm that the table is created. Specification Values: Column named Dept_ID of Numeric 7 size and would be a primary key. Column named Dept_Name of varchar2 size 20.	DDL	☐
2	Populate the DEPT table with data from the DEPARTMENTS table. Include only columns that you need. Insert dept Id 10 and Name Accounts Insert dept Id as null and Name as TT Correct by giving 20 and TT Insert A1 as Id and Accounts Correct by giving 30 and Accounts	DDL	☐
3	Create the EMP table based on the following table instance chart. Save the statement in a script called lab_10_03.sql , and then execute the statement in the script to create the table. Confirm that the table is created. Specification- Values Column Name: ID, LAST_NAME, FIRST_NAME, DEPT_ID Key Type: PK, -, -, FK Nulls /Unique: -, Not null, -, -, FK Table: -, -, -, Dept FK Column: -, -, -, ID Data type: NUMBER, VARCHAR2, VARCHAR2, NUMBER Length: 7, 25, 25, 7 Insert 101,Sam,Sundar,10 Insert 101,Ram,Krishna,20 Insert 102,Gopi,null,40 Insert 103,null,ram,20	DDL	☐

8.7 Introduction to JDBC

NO HOA

8.8 Establishing connection

1	Write a java program that establishes a connection to oracle database successfully. If the connection is successful, it should display a message "Connection Established successfully". In case, it is not able to do so due to any exception, it should display the message "Connection could not be established ". If there is an exception, it should also display the description of the exception.	JDBC, Driver Manager, Connection	☐
2	In the just concluded exercise, where you have established the connection successfully, exclude the registration process(by commenting the line containing the code Class.forName("..")). Observe the result.	JDBC, Driver Manager, Connection	☐

8.9 Executing query and Processing results

1	Write a java program that connects to oracle database, queries the inbuilt table "emp" and displays the first two columns (empno using column index and ename using column name) of all the rows.	JDBC, Driver Manager, Connection,Statements	☐
2	Modify the above program to display all the rows where sal is greater than 1000 and less than 2000. Display the columns ename, job, sal and comm.	JDBC, Driver Manager, Connection,Statements	☐

8.10 Using prepared statements and MetaData Objects

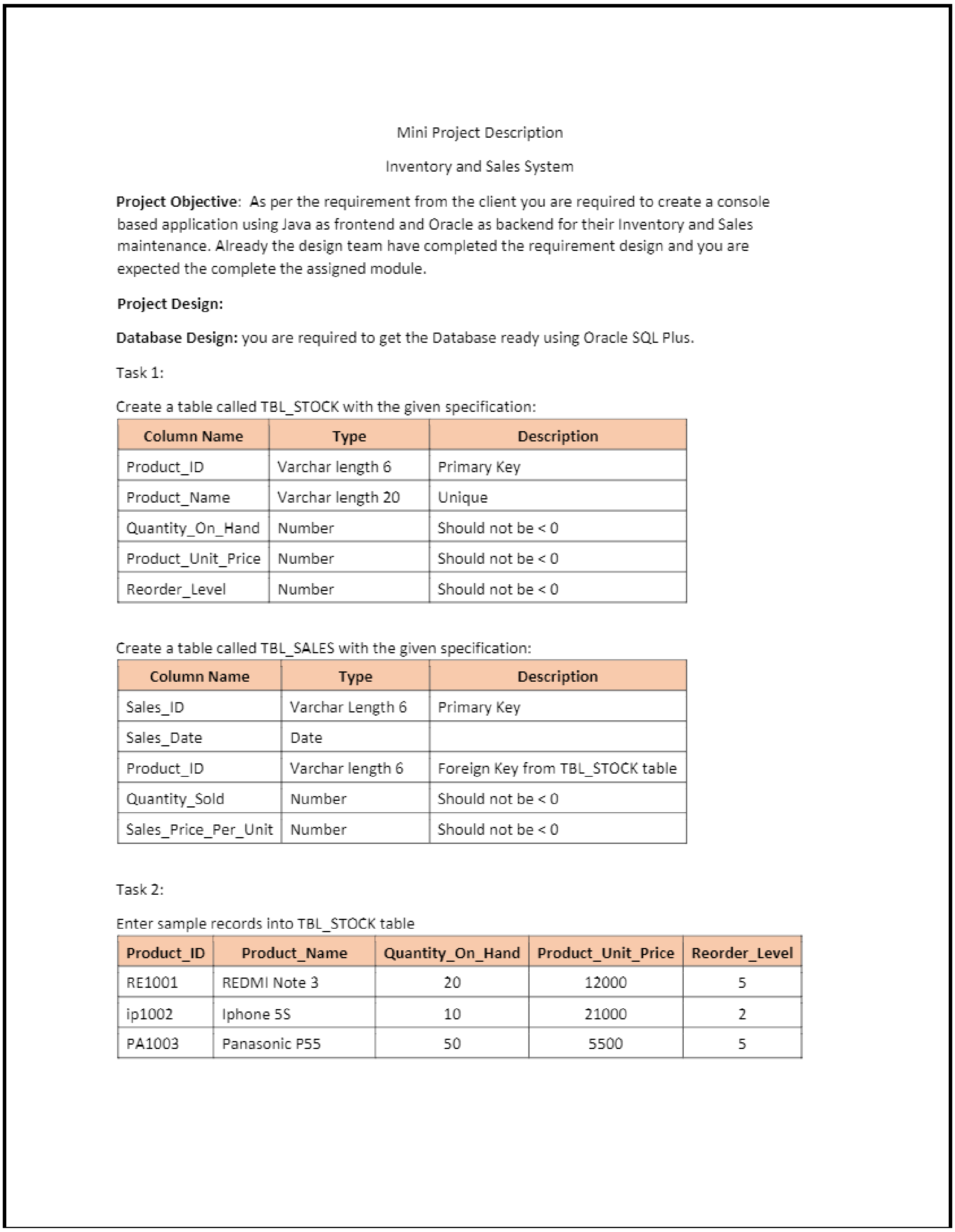
1	Develop a jdbc program containing main method, which should instantiate a class called DAOClass, which should contain methods called insert, delete, modify and display. Description of what each of these methods are expected to do is given below. Necessary details required for executing these methods, are passed from command line argument. For e.g. If the name of the class containing the main method is JDBCCalls, then if you want to insert a record, you will execute this class as java JDBCCalls 1 101 “Ajit” “IV” “20-Nov-2001” 4000 Where 1 is the option for inserting the record and all other details are the values for the columns in each row of the student table. The structure of student table is given below. Similarly, for deleting a record, you have to execute the code as java JDBCCalls 2 101 where 2 is the option for deleting a record and 101 is the rollno of the student, whose record has to be deleted. For modifying a record, you will use java JDBCCalls 3 101 4500, where 3 is the option for modifying a record and the 4500 is the new fee which needs to replace the old fee value. For Displaying records, if the main class is executed as follows java JDBCCalls 4 101 it should display only one record, that of the student with roll no. 101. 4 option is for displaying the record. If the main class is executed as java JDBCCalls 4 (without specifying the rollno.), it means that details of all the students should be displayed.	PreparedStatement	☐
2	Inserting a record ABC International School wants to computerize students details. The school maintains a database of students in Oracle. The student table contains information related to students and is shown in the following student table structure. Column Name Type Constraint Rollno Number (4) Primary Key StudentName Varchar (20) Not Null	PreparedStatement	☐

	Standard Varchar (2) Not Null Date_Of_Birth Date Fees Number (9,2) When a new student joins the school, the student record is inserted in the student table. The valid student details are as follows: • Rollno: Valid value is a 4-digit number • StudentName: Valid value can contain maximum 20 letters in uppercase • Standard : Valid values are Roman Letters representing I to X(I, II, III, IV….IX, X) Write a Java program to insert some records to the table		
3	Deleting a Student’s record When a student leaves the school, the record related to that student needs to be deleted from the Student table. The student’s roll no, whose record has to be deleted, should be passed as a command line argument. Upon deletion, the Student details must be stored in another table named StudentLog which will maintain the details such as Rollno, StudentName, Standard and Leaving_date.	PreparedStatement	☐
4	Modification of Student record When there is a change in the fee to be paid by a student, the respective row should be appropriately updated. Pass the rollno from the command prompt along with the new fee amount and this amount should be reflected in the table for that particular student.	PreparedStatement	☐
5	Display Student details Write the code to display details of all the students, if no roll no. is passed, while executing the main program. If while executing the main program, the roll no. is passed, then it should display the record of that particular student.	PreparedStatement	☐

8.11 Using Callable Statements and Transactions

1	Create a stored procedure that calculates net salary of all the employees whose records are stored in table "emp". The criteria for calculating net salary is as follows : Gross salary = sal + comm. Net Salary = gross salary - IT If the employee's commission is null then IT is calculated as IT = 10% of gross salary else if the employees commission is less than 500, then IT is calculated as IT = 15% of gross salary else IT = 20% of gross salary. Develop a jdbc program that invokes this stored procedure by passing the empno. and in return gets the net salary of each employee. Display on screen the empno., ename and net salary of all the employees.	CallableStatement	☐

8.0 Mini project Inventory and sales System



Mini Project Description

Inventory and Sales System

Project Objective: As per the requirement from the client you are required to create a console based application using Java as frontend and Oracle as backend for their Inventory and Sales maintenance. Already the design team have completed the requirement design and you are expected the complete the assigned module.

Project Design:

Database Design: you are required to get the Database ready using Oracle SQL Plus.

Task 1:

Create a table called TBL_STOCK with the given specification:

Column Name	Type	Description
Product_ID	Varchar length 6	Primary Key
Product_Name	Varchar length 20	Unique
Quantity_On_Hand	Number	Should not be < 0
Product_Unit_Price	Number	Should not be < 0
Reorder_Level	Number	Should not be < 0

Create a table called TBL_SALES with the given specification:

Column Name	Type	Description
Sales_ID	Varchar Length 6	Primary Key
Sales_Date	Date	
Product_ID	Varchar length 6	Foreign Key from TBL_STOCK table
Quantity_Sold	Number	Should not be < 0
Sales_Price_Per_Unit	Number	Should not be < 0

Task 2:

Enter sample records into TBL_STOCK table

Product_ID	Product_Name	Quantity_On_Hand	Product_Unit_Price	Reorder_Level
RE1001	REDMI Note 3	20	12000	5
ip1002	Iphone 5S	10	21000	2
PA1003	Panasonic P55	50	5500	5

Task 3:

Create the following sequences:

Sequence Name	Start value	Incremental Value
SEQ_SALES_ID	1000	1
SEQ_PRODUCT_ID	1004	1

Task 4:

Create a view named V_SALES_REPORT using TBL_SALES table joined with TBL_STOCK table based on ProductID order the result based on Profit_Amount in descending and Sales_ID in Ascending.

Column Name	Description
Sales_ID	
Sales_Date	
Product_ID	
Product_Name	
Quantity_Sold	
Product_Unit_Price	
Sales_Price_Per_Unit	
Profit_Amount	returns the difference between the Sales_Price_Per_Unit and Product_Unit_Price

Application Design: Create Java Application to manage the Sales:

Create a Java Application to manage the Sales and control the inventory. Create a new Java project under eclipse.

Task 5:

Create the following packages and the specified classes below.

Name of the package	Usage
com.wipro.sales.util	Contains the class that establishes the database connection
com.wipro.sales.bean	Contains all the bean classes
com.wipro.sales.dao	Contains the DAO classes that performs the real JDBC operations
com.wipro.sales.service	Contains the administrator class that receives input from Servlets and that invokes the respective DAO class methods
com.wipro.sales.main	Contains executable class with the main method

Under the package com.wipro.sales.util create the following classes.

Class	Method and Variables	Description
DBUtil		DB connection class
	public static Connection **getDBConnection**()	Establish a connection to the database and return the java.sql.Connection reference

Under the package com.wipro.sales.bean create the following classes.

Class	Method and Variables	Description
Product		**Bean Class**
String	productID	
String	productName	
int	quantityOnHand	
double	productUnitPrice	
int	reorderLevel	
	Setters and Getters for all properties	Using Eclipse, create getters and setters for all the properties

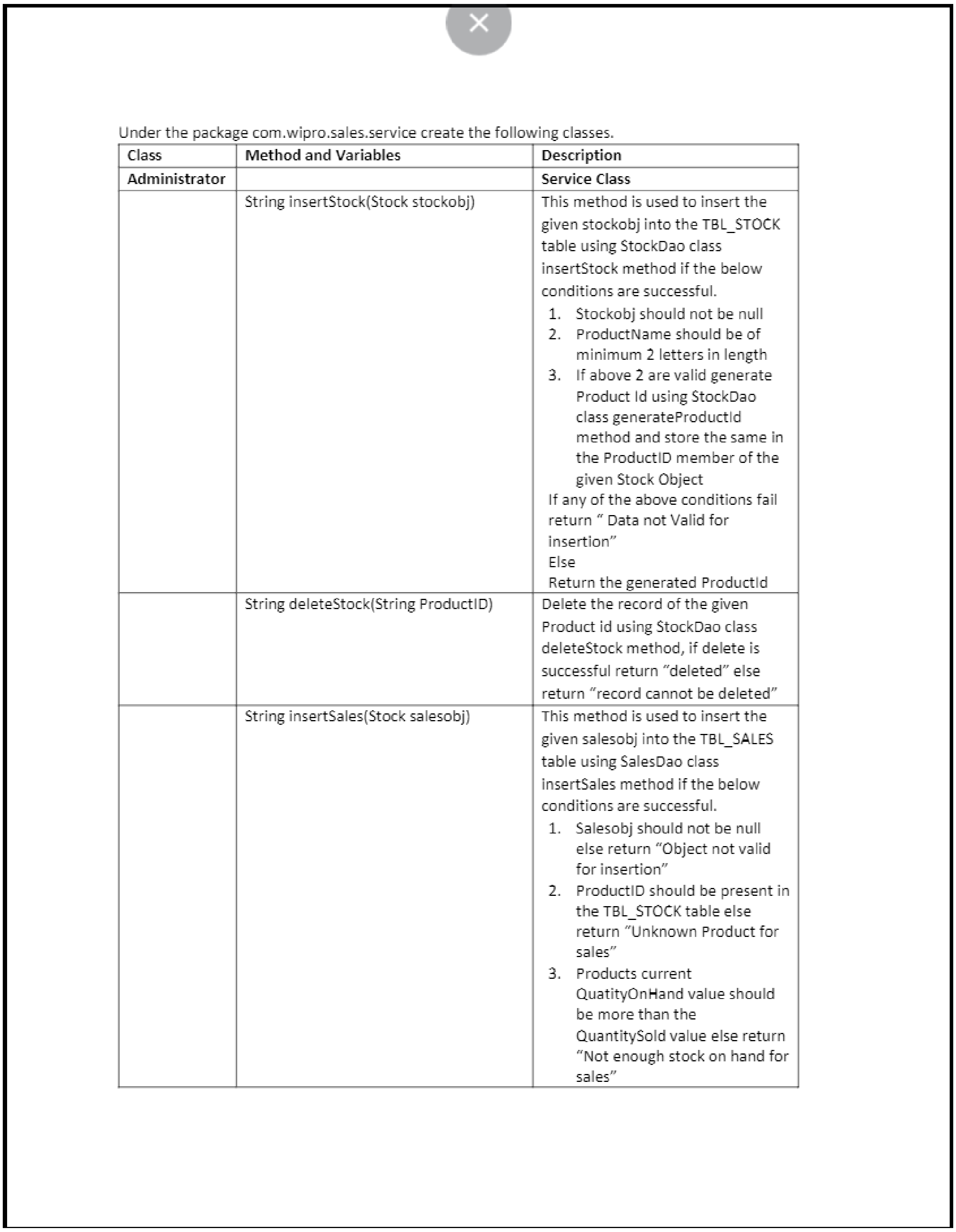
Class	Method and Variables	Description
Sales		**Bean Class**
String	salesID	
Java.util.Date	salesDate	
String	productID	
int	quantitySold	
double	salesPricePerUnit	
	Setters and Getters for all properties	Using Eclipse, create getters and setters for all the properties

Class	Method and Variables	Description
SalesReport		**Bean Class**
String	salesID	
Java.util.Date	salesDate	
String	productID	
String	productName	
int	quantitySold	
double	productUnitPrice	
double	salesPricePerUnit	
double	profitAmount	
	Setters and Getters for all properties	Using Eclipse, create getters and setters for all the properties

Under the package com.wipro.sales.dao create the following classes.

Class	Method and Variables	Description
SalesDao		Dao Class
	Int insertSales(Sales sales)	This method is used to insert the given sales obj into TBL_SALES table
	String generateSalesID(java.util.Date salesDate)	This method is used to generate Sales ID using the last2digit of the year part of the given date concatenated with the SEQ_SALES_ID sequence generated number.
	ArrayList<SalesReport> getSalesReport()	This method runs the V_SALES_REPORT view and stores every record in SalesREport Bean adding them to an arraylist. Which is return back to the user.

Class	Method and Variables	Description
StockDao		Dao Class
	insertStock(Stock sales)	This method is used to insert the given stock obj into TBL_STOCK table
	generateProductID(String productName)	This method is used to generate Stock ID using the First 2 letters of the given product name concatenated with the SEQ_PRODUCT_ID sequence generated number.
	updateStock(String productID,int soldQty)	This method is used to update the Stock table by subtracting the current Quantity_On_Hand by the given soldQty of the given productID.
	Stock getStock(String productID)	This method is used to fetch a specific record details from the Stock table for the given productID, store the information to a Stock bean object the return the same.
	deleteStock(String productID)	This method is used to delete the stock record of the given ProductID



Under the package com.wipro.sales.service create the following classes.

Class	Method and Variables	Description
Administrator		**Service Class**
	String insertStock(Stock stockobj)	This method is used to insert the given stockobj into the TBL_STOCK table using StockDao class insertStock method if the below conditions are successful. 1. Stockobj should not be null 2. ProductName should be of minimum 2 letters in length 3. If above 2 are valid generate Product Id using StockDao class generateProductId method and store the same in the ProductID member of the given Stock Object If any of the above conditions fail return " Data not Valid for insertion" Else Return the generated ProductId
	String deleteStock(String ProductID)	Delete the record of the given Product id using StockDao class deleteStock method, if delete is successful return "deleted" else return "record cannot be deleted"
	String insertSales(Stock salesobj)	This method is used to insert the given salesobj into the TBL_SALES table using SalesDao class insertSales method if the below conditions are successful. 1. Salesobj should not be null else return "Object not valid for insertion" 2. ProductID should be present in the TBL_STOCK table else return "Unknown Product for sales" 3. Products current QuatityOnHand value should be more than the QuantitySold value else return "Not enough stock on hand for sales"

		4. SalesDate should be currentdate or earlier date and not future date, else return "Invalid date" 5. If above 4 are valid generate Sales Id using SalesDao class generateSalesId method and store the same in the SalesID member of the given Sales Object Call the **insertSales** method of **SalesDao** and insert the record. If insertion is successful call the **updateStock** method of the **StockDao** and update the sold quantity to the stock. On successful completion of both the transaction return "Sales Completed" else "Error".
	ArrayList<SalesReport> getSalesReport()	This method calls the getSalesReport of the SalesDao and returns the ArrayList

Under the package com.wipro.sales.main create the following classes.

Class	Method and Variables	Description
SalesApplication		**Executable Class**
	public static void main(String args[])	This method has to display a main menu with following Options: 1. Insert Stock 2. Delete Stock 3. Insert Sales 4. View Sales Report Enter your Choice: On selecting the choice It should accept the required data from the user create appropriate object and call the valid method from the Administrator class. Eg: if the selected option is 1. Then create Stock bean object and get all Stock bean data from user and set it to the object and call insertStock method from the Administrator class.