OBJECT ORIENTED PROGRAMMING USING JAVA LAB

LAB ASSIGNMENTS

Week 1

Objective: To understand the basic concepts of Object Oriented Programming System and to get familiar with object and class.

Assignments:

1. Write a Java program to print your name.
2. Write a Java program to add two numbers.
3. Write a Java program to change temperature from Celsius to Fahrenheit.
4. Write a Java program to change temperature from Fahrenheit to Celsius.
5. Write a Java program to find area and perimeter of a rectangle.
6. Write a Java program to find area and perimeter of a circle.
7. Write a Java Program to display whether a number is odd or even.
8. Write a Java Program to check if a number is Positive or Negative.
9. Write a Java program to find maximum of three numbers.
10. Write a Java program to swap two numbers.
11. Write a Java program to convert miles to kilometers.
12. Write a Java program to check whether a year is leap year or not.
13. Write a Java program for following grading system.

    Note: Percentage>=90% : Grade A
    Percentage>=80% : Grade B
    Percentage>=70% : Grade C
    Percentage>=60% : Grade D
    Percentage>=40% : Grade E
    Percentage<40% : Grade F
14. Write a Java program to check whether a number is divisible by 5 or not.

Week 2

Objective: To understand the basic concepts of variable, decision and loop control statements.

Assignments:

1. Write a Java program to check whether a number is Buzz or not.
2. Write a Java program to calculate factorial of 12.
3. Write a Java program for Fibonacci series.
4. Write a Java program to reverse a number.
5. Admission to a professional course is subject to the following conditions:
   (a) marks in Mathematics >= 60 (b) marks in Physics >=50
   (c) marks in Chemistry >=40 (d) Total in all 3 subjects >=200
   (Or)
   Total in Maths & Physics>=150
   Given the marks in the 3 subjects of n (user input) students, write a program to process the applications to list the eligible candidates.
6. Write a Java program to find all roots of a quadratic equation.
7. Write a Java program to calculate the sum of natural numbers up to a certain range.
8. Write a Java program to print all multiple of 10 between a given interval.
9. Write a Java program to generate multiplication table.
10. Write a Java program to find HCF of two Numbers.
11. Write a Java program to find LCM of two Numbers.
12. Write a Java program to count the number of digits of an integer.
13. Write a Java program to calculate the exponential of a number.
14. Write a Java program to check whether a number is palindrome or not.
15. Write a Java program to check whether a number is prime or not.
16. Write a Java program to convert a Binary Number to Decimal and Decimal to Binary.
17. Write a Java program to find median of a set of numbers.
18. Write a program to compute the value of Euler's number that is used as the base of natural logarithms. Use the following formula.
    e= 1+ 1/1! +1 /2! + 1/3+..... 1/n!
19. Write a Java program to generate all combination of 1, 2, or 3 using loop.
20. Write a Java program to read two integer values m and n and to decide and print whether m is multiple of n.
21. Write a Java program to display prime numbers between a given interval.
22. Write a Java program to check whether a given number is Armstrong Number or not.

Write Java programs for the patterns given bellow: **(23-25)**

23. 1
2 3 4
5 6 7 8 9

24.
```
    1
  2 1 2
3 2 1 2 3
4 3 2 1 2 3 4
```

25.
```
1     1
 2   2
  3 3
   4
```

Week 3

Objective: To understand the concepts of Array and Constructor.

Assignments:

1. Write a Java program to calculate Sum & Average of an integer array.
2. Write a Java program to implement stack using array.
3. Write a Java program to implement Queue using array.
4. Write a Java program to calculate Sum of two 2-dimensional arrays.
5. Write a Java program to find the range of a 1D array.
6. Write a Java program to search an element in an array.
7. Write a Java program to find the sum of even numbers in an integer array.
8. Write a Java program to find the sum of diagonal elements in a 2D array.
9. Reverse the elements in an array of integers without using a second array.
10. Write a Java program to enter n elements in an array and find smallest number among them.
11. Write Java program to find the sum of all odd numbers in a 2D array.
12. Write a Java program to print transpose of matrix.
13. Write a Java program to check whether a given matrix is sparse or not.
14. Write a Java program to count the prime numbers in an array.
15. Write a Java program to find second highest element of an array.
16. Write a Java program which counts the non-zero elements in an integer array.
17. Write a Java program to merge two float arrays.
18. Write a Java program where elements of two integer arrays get added index wise and get stored into a third array.
19. Write a Java program to multiply two matrices.
20. Write a Java program to subtract two matrices.
21. Write a Java program to find duplicate elements in a 1D array and find their frequency of occurrence.
22. Write a Java program to print every alternate number of a given array.

23. Given are two one-dimensional arrays A & B, which are sorted in ascending order. Write a Java program to merge them into single sorted array C that contains every item from arrays A & B, in ascending order.
24. Write a Java program to show 0-arguments constructor.
25. Write a Java program to show parameterized constructor.
26. Write a Java program to show constructor overloading.
27. Write a class, Grader, which has an instance variable, score, an appropriate constructor and appropriate methods. A method, letterGrade() that returns the letter grade as O/E/A/B/C/F.
    Now write a demo class to test the Grader class by reading a score from the user, using it to create a Grader object after validating that the value is not negative and is not greater then 100. Finally, call the letterGrade() method to get and print the grade.

28. Write a class, Commission, which has an instance variable, sales; an appropriate constructor; and a method, commission() that returns the commission.
    Now write a demo class to test the Commission class by reading a sale from the user, using it to create a Commission object after validating that the value is not negative. Finally, call the commission() method to get and print the commission. If the sales are negative, your demo should print the message "Invalid Input".

Week 4

Objective: To understand the concepts of OOPs Properties (Inheritance and Dynamic Polymorphism).

Assignments:

1. Write a Java program to implement the concept of inheritance.
2. Write a Java program to show method overloading.
3. Write a Java program to show method overriding.
4. Write a Java program to show method hiding.

5. Create a general class ThreeDObject and derive the classes Box, Cube, Cylinder and Cone from it. The class ThreeDObject has methods wholeSurfaceArea ( ) and volume ( ). Override these two methods in each of the derived classes to calculate the volume and whole surface area of each type of three-dimensional objects. The dimensions of the objects are to be taken from the users and passed through the respective constructors of each derived class. Write a main method to test these classes.

6. Write a program to create a class named Vehicle having protected instance variables regnNumber, speed, color, ownerName and a method showData ( ) to show "This is a vehicle class". Inherit the Vehicle class into subclasses named Bus and Car having

individual private instance variables routeNumber in Bus and manufacturerName in Car and both of them having showData ( ) method showing all details of Bus and Car respectively with content of the super class's showData ( ) method.

7. An educational institution maintains a database of its employees. The database is divided into a number of classes whose hierarchical relationships are shown below. Write all the classes and define the methods to create the database and retrieve individual information as and when needed.
Write a driver program to test the classes.

Staff (code, name)	*Teacher* (subject, publication) is a Staff
Officer (grade) is a Staff	*Typist* (speed) is a Staff
RegularTypist (remuneration) is a Typist	*CasualTypist* (daily wages) is a Typist.

8. Create a base class Building that stores the number of floors of a building, number of rooms and it's total footage. Create a derived class House that inherits Building and also stores the number of bedrooms and bathrooms. Demonstrate the working of the classes.

9. In the earlier program, create a second derived class Office that inherits Building and stores the number of telephones and tables. Now demonstrate the working of all three classes.

10. Write a Java program which creates a base class Num and contains an integer number along with a method shownum() which displays the number. Now create a derived class HexNum which inherits Num and overrides shownum() which displays the hexadecimal value of the number. Demonstrate the working of the classes.

11. Write a Java program which creates a base class Num and contains an integer number along with a method shownum() which displays the number. Now create a derived class OctNum which inherits Num and overrides shownum() which displays the octal value of the number. Demonstrate the working of the classes.

12. Combine Question number 10 and 11 and have all the three classes together. Now describe the working of all classes.

13. Create a base class Distance which stores the distance between two locations in miles and a method travelTime(). The method prints the time taken to cover the distance when the speed is 60 miles per hour. Now in a derived class DistanceMKS, override travelTime() so that it prints the time assuming the distance is in kilometers and the speed is 100 km per second. Demonstrate the working of the classes.

14. Create a base class called "**vehicle**" that stores number of wheels and speed.
Create the following derived classes –
"**car**" that inherits "vehicle" and also stores number of passengers.
"**truck**" that inherits "vehicle" and also stores the load limit.
Write a main function to create objects of these two derived classes and display all the information about "car" and "truck". Also compare the speed of these two vehicles - car and truck and display which one is faster.

15. Write a Java program to explain "multilevel inheritance."

Week 5

Objective: Implementing the concepts of class variable, instance variable, use of "this" keyword, use of reference variable in Java.

Assignments:

1. Create a "circle" class & a "point" class. The coordinates of the circle are given and used within the "circle" class as object of the "point" class. Display the area of circle.

2. Create a class called Time, which has three private instance variables – hour, min and sec. It contains a method called add( ) which takes one Time object as parameter and print the added value of the calling Time object and passes Time object. In the main method, declare two Time objects and assign values using constructor and call the add() method.

3. Create a class called Complex, which has three private instance variables –real and imaginary. It contains a method called add( ) which takes one Complex object as parameter and print the added value of the calling Complex object and passes Complex object. In the main method, declare two Complex objects and assign values using constructor and call the add() method.

4. Write a program to define a class having one 3-digit number, num as data member. Initialize and display reverse of that number.

5. Write a program to define a class Student with four data members such as name, roll no., sub1, and sub2. Define appropriate methods to initialize and display the values of data members. Also calculate total marks and percentage scored by student.

6. Write a program to define a class Employee to accept emp_id, emp _name, basic_salary from the user and display the gross_salary.

7. Write a program to define a class Fraction having data members numerator and denominator. Initialize three objects using different constructors and display its fractional value.

8. Write a program to define a class Item containing code and price. Accept this data for five objects using array of objects. Display code, price in tabular form and also, display total price of all items.

9. Write a program to define a class Tender containing data members cost and company name. Accept data for five objects and display company name for which cost is minimum.

10. Write a program to define a class 'employee' with data members as empid, name and salary. Accept data for 5 objects using Array of objects and print it.

11. Define a class called circle that contains:

- Two private instance variables: radius (of type double) and color (of type String),
- Initialize the variables radius and color with default value of 1.0 and "red", respectively using default constructor.
- Include a second constructor that will use the default value for color and sets the radius to the value passed as parameter.
- Two public methods: getRadius() and getArea() for returning the radius and area of the circle
- Invoke the above methods and constructors in the main.

12. Write a program which will accept an integer from the user and pass the value to a method called PrintNumberInWord that will print "ONE", "TWO",... , "NINE", "ZERO" if the integer variable "number" is 1, 2,... , 9, or 0, respectively.

13. Design a class named Account that contains:

I. A private int data field named id for the account (default 0).
II. A private double data field named balance for the account (default 0).
III. A private double data field named annualInterestRate that stores the cur-rent interest rate (default 0). Assume all accounts have the same interest rate.
IV. A private Date data field named dateCreated that stores the date when the account was created.
V. A no-arg constructor that creates a default account.
VI. A constructor that creates an account with the specified id and initial balance.
VII. The accessor and mutator methods for id,balance, and annualInterestRate.
VIII. The accessor method for dateCreated.
IX. A method named getMonthlyInterestRate() that returns the monthly interest rate.
X. A method named getMonthlyInterest() that returns the monthly interest.
XI. A method named withdraw that withdraws a specified amount from the account.
XII. A method named deposit that deposits a specified amount to the account.

14. Write a test program that prompts the user to enter the investment amount (e.g., 1000) and the interest rate (e.g., 9%), and print a table that displays future value for the years from 1 to 30, as shown below:

The amount invested: 1000
Annual interest rate: 9%

Years	Future Value
1	1093.8
2	1196.41
...	
29	13467.25
30	14730.57

15. Write method headers for the following methods:

a. Computing a sales commission, given the sales amount and the commission rate.

b. Printing the calendar for a month, given the month and year.

c. Computing a square root.

d. Testing whether a number is even, and returning true if it is.

e. Printing a message a specified number of times.

f. Computing the monthly payment, given the loan amount, number of years, and annual interest rate.

16. Write a program that reads ten numbers, computes their average, and finds out how many numbers are above the average. [Use this keyword]

17. Write a program that reads ten integers and displays them in the reverse of the order in which they were read.

18. Write a program to demonstrate use of 'this' keyword.

19. Write a program to demonstrate use of 'static' keyword.

20. Write a program to accept value of apple sales for each day of the week (using array of type float) and then, calculate the average sale of the week.

21. Write program, which finds the sum of numbers formed by consecutive digits.
Input : 2415
output : 24+41+15=80.

Week 6

Objective: To revise inheritance and to understand the concepts of Abstract class & Interface in Java.

Assignments:

1. Design an abstract class having two methods. Create Rectangle and Triangle classes by inheriting the shape class and override the above methods to suitably implement for Rectangle and Triangle class.

2. Write a program in Java to illustrate the use of interface in Java.

3. Create a general class ThreeDObject and derive the classes Box, Cube, Cylinder and Cone from it. The class ThreeDObject has methods wholeSurfaceArea ( ) and volume( ). Override these two methods in each of the derived classes to calculate the volume and whole surface area of each type of three-dimensional objects. The dimensions of

the objects are to be taken from the users and passed through the respective constructors of each derived class. Write a main method to test these classes.

4. Write a program to create a class named Vehicle having protected instance variables regnNumber, speed, color, ownerName and a method showData ( ) to show "This is a vehicle class". Inherit the Vehicle class into subclasses named Bus and Car having individual private instance variables routeNumber in Bus and manufacturerName in Car and both of them having showData ( ) method showing all details of Bus and Car respectively with content of the super class's showData ( ) method.

5. Create three interfaces, each with two methods. Inherit a new interface from the three, adding a new method. Create a class by implementing the new interface and also inheriting from a concrete class. Now write four methods, each of which takes one of the four interfaces as an argument. In main ( ), create an object of your class and pass it to each of the methods.

6. Create an interface Department containing attributes deptName and deptHead. It also has abstract methods for printing the attributes. Create a class hostel containing hostelName, hostelLocation and numberofRooms. The class contains methods for getting and printing the attributes. Then write Student class extending the Hostel class and implementing the Department interface. This class contains attributes studentName, regdNo, electiveSubject and avgMarks. Write suitable getData and printData methods for this class. Also implement the abstract methods of the Department interface. Write a driver class to test the Student class. The program should be menu driven containing the options:

i) Admit new student
ii) Migrate a student
iii) Display details of a student

For the third option a search is to be made on the basis of the entered registration number.

7. Create an interface called Player. The interface has an abstract method called play() that displays a message describing the meaning of "play" to the class. Create classes called Child, Musician, and Actor that all implement Player. Create an application that demonstrates the use of the classes(UsePlayer.java

8. Create an abstract class Accounts with the following details:
Data Members:
(a) Balance (b) accountNumber (c) accountHoldersName (d) address
Methods:
(a) withdrawl()- abstract
(b) deposit()- abstract
(c) display() to show the balance of the account number

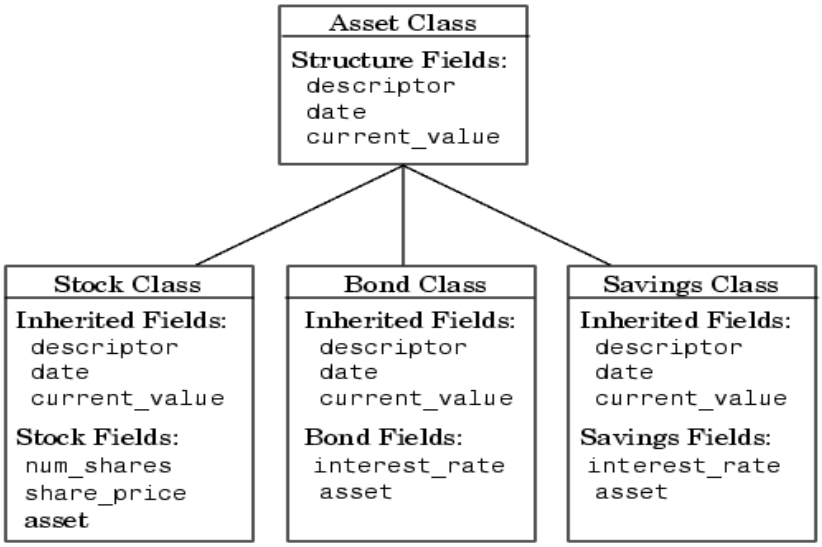
Create a subclass of this class SavingsAccount and add the following details:

Data Members:
(a) rateOfInterest
Methods:
(a) calculateAount()

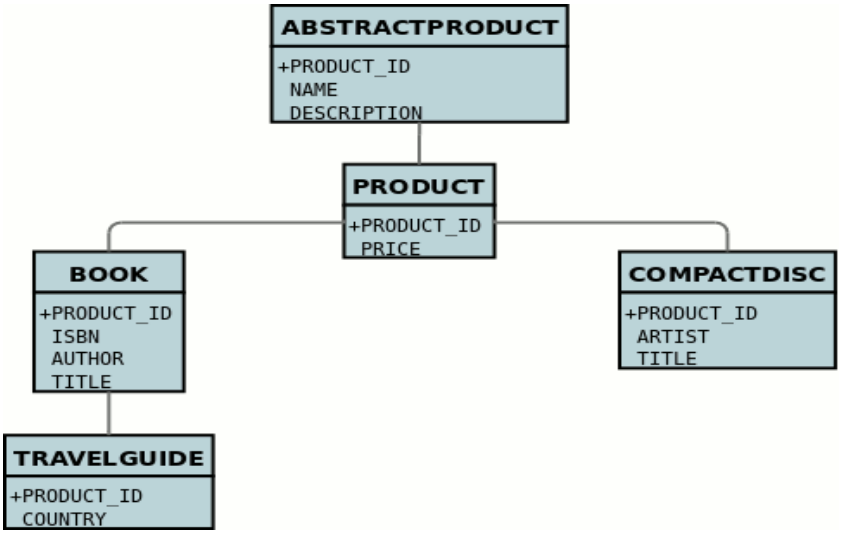
9. Create an abstract class MotorVehicle with the following details:
Data Members:
(a) modelName (b)modelNumber (c) modelPrice
Methods:
(a) display() to show all the details

Create a subclass of this class Carthat inherits the class MotorVehicle and add the following details:
Data Members:
(b) discountRate
Methods:
(a) display() method to display the Car name, model number, price and the discount rate.
(b) discount() method to compute the discount.

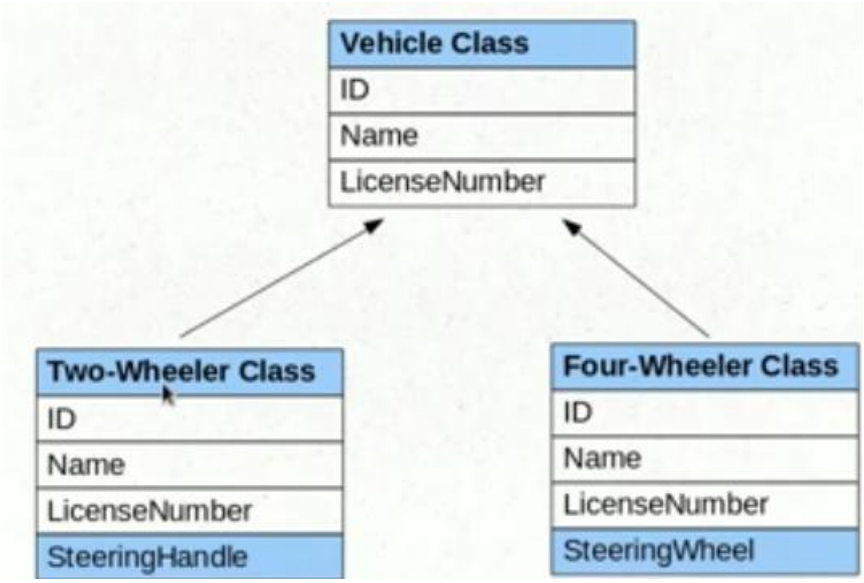
10. Implement the below Diagram.
Here, Asset class is an abstract class containing an abstract method displayDetails() method. Stock, bond and Savings class inherit the Asset class and displayDetails() method is defined in every class.



11. Implement the below Diagram. Here AbstractProduct is only abstract class.



12. Implement the below Diagram



13. Write a program to implement the Multiple Inheritance (Bank Interface, Customer & Account classes).

14. Write a program to implement the Multiple Inheritance (Gross Interface, Employee & Salary classes).

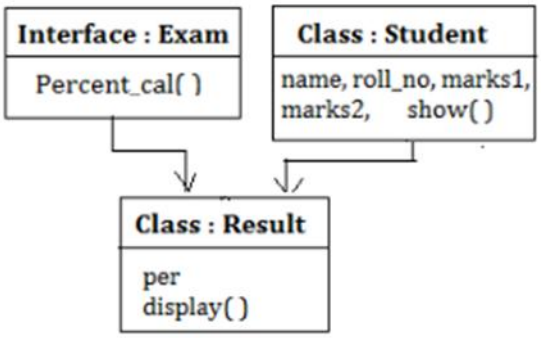
15. Program to create a interface 'Mango' and implement it in classes 'Winter' and 'Summer'.

16. Program to implement the Multiple Inheritance (Exam Interface, Student & Result classes).

17. Program to demonstrate use of hierarchical inheritance using interface.

18. Java program to Perform Payroll Using Interface (Multiple Inheritance).

19. Implement the following diagram.



Week 7

Objective: Implement the concepts of Exception Handling in Java.

Assignments:

1. Write a Java program to show the use of all keywords for exception handling.

2. Write a Java program using try and catch to generate NegativeArrayIndex Exception and Arithmetic Exception.

3. Define an exception called “NoMatchFoundException” that is thrown when a string is not equal to “University”. Write a program that uses this exception.

4. Write a class that keeps a running total of all characters passed to it (one at a time) and throws anexception if it is passed a non-alphabetic character.

5. Write a program called Factorial.java that computes factorials and catches the result in an array of type long for reuse. The long type of variable has its own range. For example 20! Is as high as the range of long type. So check the argument passes and “throw an exception”, if it is too big or too small.
    - If x is less than 0 throw an IllegalArgumentException with a message “Value of x must be positive”.

- If x is above the length of the array throw an IllegalArgumentException with a message “Result will overflow”.

Here x is the value for which we want to find the factorial.

6. Write a class that keeps a running total of all characters passed to it (one at a time) and throws an exception if it is passed a non-alphabetic character.

7. Write a program that outputs the name of the capital of the country entered at the command line. The program should throw a “NoMatchFoundException” when it fails to print the capital of the country entered at the command line.

8. Write a program that takes a value at the command line for which factorial is to be computed. The program must convert the string to its integer equivalent. There are three possible user input errors that can prevent the program from executing normally.

- The first error is when the user provides no argument while executing the program and an ArrayIndexOutOfBoundsException is raised. You must write a catch block for this.
- The second error is NumberFormatException that is raised in case the user provides a non-integer (float double) value at the command line.
- The third error is IllegalArgumentException. This needs to be thrown manually if the value at the command line is 0.

9. Create a user-defined exception named CheckArgument to check the number of arguments passed through the command line. If the number of argument is less than 5, throw the CheckArgumentexception,else print the addition of all the five numbers.

10. Consider a Student examination database system that prints the mark sheet of students. Input the following from the command line.

(a) Student’s Name
(b) Marks in six subjects

These marks should be between 0 to 50. If the marks are not in the specified range, raise a RangeException, else find the total marks and prints the percentage of the students.

11. Write a java program to create an custom Exception that would handle at least 2 kind of Arithmetic Exceptions while calculating a given equation (e.g. X+Y*(P/Q)Z-I)

12. Create two user-defined exceptions named “TooHot” and “TooCold” to check the temperature (in Celsius) given by the user passed through the command line is too hot or too cold.

- If temperature > 35, throw exception “TooHot”.
- If temperature <5, throw exception “TooCold”.

- Otherwise, print “Normal” and convert it to Farenheit.

13. Consider an Employee recruitment system that prints the candidate name based on the age criteria. The name and age of the candidate are taken as Input.Create two user-defined exceptions named “TooOlder” and “TooYounger”
- If age>45, throw exception “TooOlder”.
- If age<20, throw exception “TooYounger”.
- Otherwise, print “Eligible” and print the name of the candidate.

14. Consider a “Binary to Decimal” Number conversion system which only accepts binary number as Input. If user provides a decimal number a custom Exception “WrongNumberFormat” exception will be thrown. Otherwise, it will convert into decimal and print into the screen.

15. Write a Java Program that Implement the Nested Try Statements.

16. Write a Java Program to Create Account with 500 Rs Minimum Balance, Deposit Amount, Withdraw Amount and Also Throws LessBalanceException.
- Java Program Which has a Class Called LessBalanceException Which returns the Statement that Says WithDraw Amount(_Rs) is Not Valid
- Java Program that has a Class Which Creates 2 Accounts, Both Account Deposit Money and One Account Tries to WithDraw more Money Which Generates a LessBalanceException Take Appropriate Action for the Same.

17. Consider a Library Management System, where a user wants to find a book. If the book is present in Library (Hint: Use predefined array), then it will print the book. Otherwise it will throw an exception “BookNotFoundException”.

18. Consider a Quiz Management System, where a user needs to answer 5 questions. If any of the answer is wrong, throw an exception “NotCorrectException”. If the answer is correct give a message “good! The answer is correct”.

19. Write a program to raise a user defined exception if username is less than 6 characters and password does not match.

20. Write a program to accept a password from the user and throw 'Authentication Failure' exception if the password is incorrect.

21. Write a program to input name and age of a person and throw a user-defined exception, if the entered age is negative.

22. Write a program to throw user defined exception if the given number is not positive.

23. Write a program to throw a user-defined exception "String Mismatch Exception", if two strings are not equal (ignore the case).

24. Design a stack class. Provide your own stack exceptions namely push exception and pop exception, which throw exceptions when the stack is full and when the stack is

empty respectively. Show the usage of these exceptions in handling a stack object in the main.

25. Write an application that displays a series of at least five student ID numbers (that you have stored in an array) and asks the user to enter a numeric test score for the student. Create a ScoreException class, and throw a ScoreException for the class if the user does not enter a valid score (zero to 100). Catch the ScoreException and then display an appropriate message. In addition, store a 0 for the student's score. At the end of the application, display all the student IDs and scores.

Week 8

Objective: Implement the concepts of Keyboard input and string handling in Java.

Assignments:

1. Write a Java program for calculating Factorial. Number should be taken through user input (Using Scanner, BufferedReader both).

2. Design a palindrome class that will input a string from console and check whether the string is palindrome or not.

3. Write a Java program to merge two strings.

4. Write a Java program for reverse a string. (String will be taken as user input through console).

5. Write a Java Program to Concatenate Two Strings.

6. Write a Java Program to check if a Given String is getChar from Specific Index.

7. Write a Java Program to Find the Length of the String.

8. Write a Java Program to Find All Possible Subsets of given Length in String.

9. Write a Java Program to Remove the White Spaces from a String.

10. Write a Java Program to Compare two Strings.

11. Write a Java Program to Compare Performance of Two Strings.

12. Write a Java Program to Use Equals Method In a String Class.

13. Write a Java Program to Use EqualsIgnoreCase Method In a String Class.

14. Write a Java Program to Use compareTo Method In a String Class.

15. With a Java Program to Use compareToIgnoreCase Method In a String Class.

16. Write a Java Program to Replace Character or String.

17. Write a Java Program to Search Last Occurance of a Substring Inside a Substring.

18. Write a Java Program to Remove a Particular Character from a String.

19. Write a Java Program to Replace a Substring Inside a String by Another One.

20. Write a Java Program to Reverse a String.

21. Write a Java Program to Search a Word Inside a String.

22. Write a Java Program to Split a String into a Number of Substrings.

23. Write a Java Program to Search a Particular Word in a String.

24. Write a Java Program to Replace All Occurings of a String.

25. Write a Java Program to Make First Character of Each Word in Uppercase.

26. Write a Java Program to Delete All Repeated Words in String.

27. Write a Java Program to Reverse the String Using Both Recursion and Iteration.

28. Write a Java Program to Convert a String Totally into Upper Case.

29. Write a Java Program to Remove all Characters in Second String which are Present in First String.

30. Write a Java Program to Find the Consecutive Occurrence of any Vowel in a String.

31. Write a Java Program to Find the Largest & Smallest Word in a String.

32. Write a Java Program to Find First and Last Occurrence of Given Character in a String.

33. Write a Java Program to Display the Characters in Prime Position a Given String.

34. Write a Java Program to Sort String Ignoring Whitespaces and Repeating Characters Only Once.

35. Write a Java Program to Count Replace First Occurrence of a String.

36. Write a Java Program to Know the Last Index of a Particular Word in a String.

37. Write a Java Program to Access the Index of the Character or String.

38. Write a Java Program to Access the Characters or the ASCII of the Character Available in the String.

39. Write a Java Program to Display the Character and the Corresponding Ascii Present in the String.

40. Write a Java Program to Accept 2 String & Check Whether all Characters in First String is Present in Second String & Print.

41. Write a Java Program to Check whether a Given Character is Present in a String, Find Frequency & Position of Occurrence.

42. Write a Java Program to Count the Number of Occurrence of Each Character Ignoring the Case of Alphabets & Display them.

43. Write a Java Program to Give Shortest Sequence of Character Insertions and Deletions that Turn One String Into the Other.

44. Write a Java Program to Check Whether Date is in Proper Format or Not.

45. Write a Java Program to Validate an Email Address Format.

46. Write a Java Program to Store String Literals Using String Buffer.

47. Write a Java Program to Verify a Class is StringBuffer Class Method.

48. Write a Java Program to Ask the User His Name and Greets Him With His Name.

49. Write a Java Program to Count a Group of Words in a String.

50. Write a Java Program to Count Number of Words in a given Text or Sentence.

Week 9

Objective: Implement the concepts of Threads, multithreading & thread synchronization in Java.

Assignments:

1. Write a Java program in which total 4 threads should run. Set different priorities to the thread.
2. Create 4 threads with priority 1,3,5,7 respectively. Update a counter in each of the threads for 10 ms. Print the final value of count for each thread.
3. Write a Java Program to Use Method Level Synchronization.
4. Write a Java Program to Use Block Level Synchronization.
5. Write a Java Program to Check Whether Define run() Method as Synchronized.

6. Write a Java Program to Solve Producer Consumer Problem Using Synchronization.
7. Write a Java Program to Show that Method Will be Verified Whether it is Synchronized or Not.
8. Write a Java Program to Show How Can Class Object be Locked Using Method Level Synchronization.
9. Write a Java Program to Synchronize the Threads Acting on the Same Object. The Synchronized Block in the Program can be Executed by Only One Thread at a Time.
10. Write a Java Program to Avoid Dead Locks.
11. Write a Java Program to Solve Deadlock Using Thread.
12. Write a Java Program to Create a Thread that Implement the Runnable Interface.
13. Write a Java Program to Show the Priority in Threads.
14. Write a Java Program to Check Priority Level of a Thread.
15. Write a Java Program to Set the Priority of a Thread.
16. Write a Java Program to Get the Priorities of Running Threads.
17. Write a Java Program to Access the Priority You Can Use Method With Thread Object.
18. Write a Java Program to Use Join Thread.
19. Write a Java Program Defining Thread By Extending Thread.
20. Write a Java Program to Handle IllegalThreadStateException.
21. Write a Java Program to Check Whether Static Block will be Used.
22. Write a Java Program to Show Why Exit Method is Used in Static Method.
23. Write a Java Program to Illustrate Thread Example for setName(string name).
24. Write a Java Program to Illustrate Thread Example for Destroy().
25. Write a Java Program to Illustrate Thread Example for suspend().
26. Write a Java Program to Illustrate Thread Example for currentThread().
27. Write a Java Program to Illustrate Thread Example for run().
28. Write a Java Program to Illustrate Thread Example for getThreadGroup().
29. Write a Java Program to Illustrate Thread Example for getPriority().

30. Write a Java Program to Illustrate Thread Example for Alive().

31. Write a Java Program to Illustrate Thread Example for getName().

32. Write a Java Program to Show Interfaces Can be Extended.

33. Write a Java Program to Check a Thread is Alive or Not.

34. Write a Java Program to Get the Name of a Running Thread.

35. Write a Java Program to Get the Name of the Thread.

36. Write a Java Program to Check if a Given run() Method is Overloaded in the Thread Class.

37. Write a Java Program to Check Whether Define a Thread Class Without Defining run() Method in the Class.

38. Write a Java Program to Stop a Thread.

39. Write a Java Program to Suspend a Thread for a While.

40. Write a Java Program to Check a Thread has Stopped or Not.

Week 10

Objective: Implement the concepts of Applets in Java.

Assignments:

1. Design a Java applet that will blink “Hello Applet” message in the client area and play a musical sound in the background with a background picture in client area.

2. Design an applet that will display a text as scrolling marquee. The text can be changed by setting different “PARAMS” value.

3. Design a Java applet that displays various shapes like circle, rectangle etc.

4. Design an applet to create digital clock using thread. The clock shows system hh:mm:ss and date.

5. Write a applet to draw the following shapes:

- Rectangle with rounded corner
- Square inside a circle.

6. Write a Java Program to Create Two Lables and Two Text Fields for Entering Name and Passwords. The Password Typed by the User in the Text Field is Hidden.

7. Write a Java Program to Display Text in the Frame by Overriding PaintComponent() Method of JPanel Class.

8. Write a Java Program to Display Some Text in the Frame with the Help of a Label.

9. Write a Java Program to Create a Text Area and Display the Mouse Event When the Button on the Mouse is Clicked, When the Mouse is Moved etc is Done by the User.

10. Write a Java Program to Create a Banner Using Applet.

11. Write a Java Program to Display Clock Using Applet.

12. Write a Java Program to Create Different Shapes Using Applet.

13. Write a Java Program to Fill Colors in Shapes Using Applet.

14. Write a Java Program to go to a Link using Applet.

15. Write a Java Program to Create an Event Listener in Applet.

16. Write a Java Program to Display Image Using Applet.

****