|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Academic Year** | Aug-Dec 2019 | **Sem** | III | |
| **Course Title:** | Object Oriented Java Programming | | | |
| **Course Code:** | 19CS3PCOOJ | | | |
| **L-T-P:** | 3-0-1 | **Total Credits:** | | 4 |

**PACKAGES:**

1. Create a package CIE which has two classes- Student and Internals. The class Student has members like usn, name, sem. The class Internals has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of Internals. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.
2. Create an abstract class **Accounts** in a Package called **DataBank** with the following details:

Date members: a)balance b) accountNumber c) accountHolderName 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 **SavingAccount** in another package **SAVINGS** and add the following details:

Data members: a) rateOfInterest

Methods: a)calculateAmount( ) b) display() to display rate of interest with new balance and full account holder details.

Create a **Employee** class in another package, which will make a object reference of **SavingAccount** class and use all functionality of that class

**INTERFACES:**

1. Design an interface named **Iqueue** with the following methods:

a) insert and delete elements from the queue

b) check whether the queue is empty or not.

Implement the Iqueue with the help of arrays and if the size of the array becomes too small to hold the elements, create a new one. Test this interface by inheriting it in its subclass QueueTest.java.

1. Create an interface SAFE\_DRIVING with an interface variable Safe\_Speed and a member function speed\_Of \_the\_vehicle( ).

Create a class called drive that implements this interface. Display the following messages

1. If vehicle speed greater than Safe\_Speed then display “Drive slow”

2. If vehicle speed is less than Safe\_Speed then display “Safe Driving”

**THREADS:**

1. Develop a Java program so that it creates a thread (let's call it Thread 1). Thread 1 creates another thread (Thread 2); Thread 2 creates Thread 3; and so on, up to Thread 5. Each thread should print "Hello from Thread <num>!", but you should structure your program such that the threads print their greetings in *reverse order*.
2. Develop a Java application that creates a list of numbers and then sort in ascending order as well as in descending order simultaneously (USE THREADS).
3. Write a program which creates two threads, one thread displaying “BMS College of Engineering” once every ten seconds and another displaying “CSE” once every two seconds.

**EXCEPTION HANDLING:**

1. Create a class Student which includes instance variables usn, name and marks of three subjects. Accept the input through command line arguments. Raise an user defined exception ‘noargs’ if no arguments are given in command line . Also raise an exception when negative marks are given.
2. Implement user defined exception in a java program for Bank application which has a class with instance variables acno, name, balance and methods like deposit(amt) and withdraw(amt). If the balance falls below 2000/- then it should throw user defined exception “insufficient balance”.
3. Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called “Father” and derived class called “Son” which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge( ) when the input age<0. In Son class, implement a constructor that cases both father and son’s age and throws an exception if son’s age is >=father’s age.