VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
Java Lab Exercise

1) A user defined exception class by name satishexception is given below

```
class satishexception implements Runnable
{
  public satishexception() {
        super(S);
  }
}
```

If the userdefined exception is thrown as follows

throw new satishexception(“I am userdefined exception”);

a) Identify the issue with the userdefined exception class in throwing such an exception. State the reason behind the issue and how we should correct it. Rewrite the exception class

b) Detail the reason on why super(S) is used in this exception? Demonstrate with code how S can be used in a catch block.

2) Rewrite the given code such that it handles exceptions and the message “Thanks for using Our Software” should be displayed whether an exception occurs or not.

```
public class javalabclass {

        public static void main(String[] args) {

                        calculator c = new calculator();
                        c.add();
                        c.divide();
                        c.display_namelength();

        }

}
class calculator
{
        String name;
        int num1;
        int num2;
        public calculator() {
                System.out.println("Enter the name");
                Scanner input = new Scanner(System.in);
                this.name=input.nextLine();
                System.out.println("Enter the first number");
                this.num1=input.nextInt();
                System.out.println("Enter the second number");
                this.num2=input.nextInt();
        }

        public void add() {
                System.out.println(num1+num2);
        }

        public void divide() {
                System.out.println(num1/num2);
        }

        public void display_namelength() {
                System.out.println(name.length());
        }
}
```

3) Consider a CourseRegistration class given below that has the following data members

- CourseName - String
- FacultyName - String
- NumberOfSeats – Integer

The class has a parameterized constructor to initialize an object and the following methods (The object is initialized with the given values [Course Name – Java Programming , Faculty Name – Satish , NumberOfSeats – 0])

Register_seat – This method will deduct 1 seat from the NumberOfSeats
Allot_Seats(int Seats) – This method will get the number of Seats to be allotted and add those seats to NumberOfSeats

Use Lambda expression to create the following Threads and demonstrate registration and allocation of seats to the course.

Thread 1 – Will try and register a seat for the Java Programming Course
Thread 2 – Will try and register a seat for the Java Programming Course
Thread 3- Will allot 30 seats to the course
Thread 4 – Should print the total remaining Seats after registration is complete

Thread1 and Thread2 will be in Wait if the value for NumberOfSeats is 0
Thread3 will notify other threads after allotting the seats.

4) Use Lambda Expression and create Threads that will perform the following

- Thread1 – Writes all prime numbers between 1 and 100 to the file1.txt
- Thread2 – Writes all prime numbers between 101 and 200 to the file2.txt
  (Write operations to the files should be paused for 2 seconds after each write.)

- Thread 3 – Reads and displays prime numbers from file1.txt
- Thread4 – Reads and displays prime numbers from file2.txt

Thread3 and Thread4 should start the read operations after the write operations to the file is complete. (USE DataOutputStream and DataInputStream for the file write and read operations)

5) Create a student class containing the following fields
- name
- regno
- emailId
- address

The class should have a parameterized constructor for initializing the object. Create an array of three student objects by getting the input data from the user. Serialize the array of objects to a file by name student.txt. Deserialize the array of objects from student.txt and display only those students who are from BCE batch and whose address is from vellore to the output. If there are no such students then display "No Such Students Match the Criteria" Message to the user.

6) Consider courseID, courseName, courseOfferedBy,Course_slot as the datamembers of the course class. Initialize the course class using a parameterized constructor. Read the data for four course objects from the user and serialize the objects to a file by name course.txt. Deserialize the the objects from the course.txt file and display the details of courses by name "Java Programming" that are being offered by SCOPE in slot by name C1. If there are no such courses offered then display "No Such Courses Match the Criteria" Message to the user.

7) Write a simple Java class in which all the employee details like emp_id, name, age, designation,yearsOfExperience, department, salary is maintained. Create an array of employee objects by reading in data of the employees from the user. Throw the following userdefined employee exceptions for the deviations in the input given below
- If yearsOfExperience is greater than 20 or less than 5 then throw userdefined experience exception object with the message "Experience does not Match".
- If salary is greater than 500000 or less than 5000 then throw userdefined salary exception object with the message "Salary does not fall within the range".
- If age is greater than 60 and less than 25 then throw the userdefined age exception with the message "Age not in the range"

Handle the above user defined exceptions and any other exceptions as required. Display "Thanks for using our Software" Message to the user whether the exception occurs or not.

8) Check if the given code will generate exceptions. List the parts of the code the exceptions that are bound to happen. Rewrite the code and demonstrate how such exceptions can be handled.

```
    public class javalabclass {

    public static void main(String[] args) {

                shape s[]=new shape[4];
                s[0]=new shape();
                s[0].setShapeDetails();
                s[0].display_details();
                s[0].display_shapename();
                s[0].write_datatoFile();
    }

}

class shape
{
    int numsides;
    double area;
    String shapename;

    public void setShapeDetails()
    {
                System.out.println("Enter the number of sides");
                Scanner input = new Scanner(System.in);
                this.numsides=input.nextInt()
                System.out.println("Enter the area");
                this.area=input.nextDouble();
    }

    public void display_details()
    {
                System.out.println(numsides+area);
    }

    public void display_shapename()
    {
                if(this.shapename.equals("circle"))
                {
                            System.out.println("its a circle");
                }
    }

    public void write_datatoFile()
    {
                File f = new File("satish.txt");
                FileOutputStream fout = new FileOutputStream(f);
                fout.write(numsides);
                fout.close();
    }
}
```

9) Elaborate Thread Interference and Memory Consistency issues with Threads. Consider the given code and analyse whether Thread Interference and Memory Consistency issues are generated by the code. Justify your answer.

```
public class democlass{
	public static void main(String[] args) throws InterruptedException {

		account satish = new account();
		account ramesh = new account();
		Thread trans1 = new Thread(new Runnable() {

			@Override
			public void run() {
				for(int i=0;i<500;i++)
				{
					satish.withdraw(10);
				}

			}
		});
		Thread trans2 = new Thread(new Runnable() {

			@Override
			public void run() {
				for(int i=0;i<500;i++)
				{
					ramesh.withdraw(10);
				}

			}
		});
		trans1.start();
		trans2.start();
		trans1.join();
		trans2.join();
		satish.getbalance();
	}

  }

  class account
  {
	private int balance;
	public account() {
		// TODO Auto-generated constructor stub
	this.balance=20000;
	}
```

```
        public synchronized void withdraw(int withdraw_amount) {
                this.balance=this.balance-withdraw_amount;
        }
        public void getbalance() {
                System.out.println(this.balance);
        }
}
```

10) Use Lambda Expression and create Threads that will perform the following

- Thread1 – Reads all stop words ("a", "and"," the" are considered stop words here) from the file sample1.txt and displays number of times each stop word occurs in the file.
- Thread2 – Reads all words other than stop words ("a", "and"," the" are considered stop words here) from the file sample2.txt and displays number of times the word that start with Character S is part of the file.
- Thread 3 – Displays Thanks for using our software to the user after the file read operations in Thread 1 and Thread2 is complete

Use Scanner object to perform the file reads and set the priority for Thread 1 as 2 , Thread 2 as 2 and Thread 3 as 1 in your code.

11) Write a Java application that performs the following for a user
The application should maintain the data of various projects in a file. Consider the Project class with the data members projectName, projectID,budget,location. The application should provide the option of storing the data pertaining to every Project object in a file by name project.txt.
The application should throw the following menu options to the user

a) Submit project data – Gets the project information for a specific project and writes the data to the file project.txt
b) View Project data – The user should be able to view the data of all projects from the file.

The user should be able to perform the above mentioned operations any number of times without exiting the system.

12) Write a Java application that performs the following for an user
The user is provided with the following menu option

a) Write UTF-16 Character to a file (File name is sample.txt)
b) Read all UTF-16 Characters from the file sample.txt

The user should be able to perform the above mentioned operation any number of times without exiting the system.