University of Information Technology & Sciences (UITS)

Faculty of Science and Engineering

Department of Computer Science and Engineering

Program: B.Sc. in CSE

Term Final Examination, Autumn 2024

Course Title: Object Oriented Programming Language

Course Code: CSE 0613121

Marks: 50 **Time: 3 Hours**

(Answer all of questions and it should be sequential)

			Marks
1.	a)	Illustrate with an example how static variables are shared among all instances of a class.	[03]
	b)	In Java, private variables and methods in a class are not directly accessible from outside the class, but there are techniques to access them. Create a class Person with the following private fields: • `String name` • `int age` A private method String getDetails() that returns the person's name and age in the format: `"Name: [name], Age: [age ]"`. Now write a java program to access and modify the private variables and methods using the public getter and setter methods to access.	[05]
	c)	Compare between Final keyword and Finally block in object-oriented programming.	[02]
2.	a)	**Figure:01** that illustrates the inheritance hierarchy for a system that manages different types of vehicles. You are tasked to implementing this system using Java.	[06]

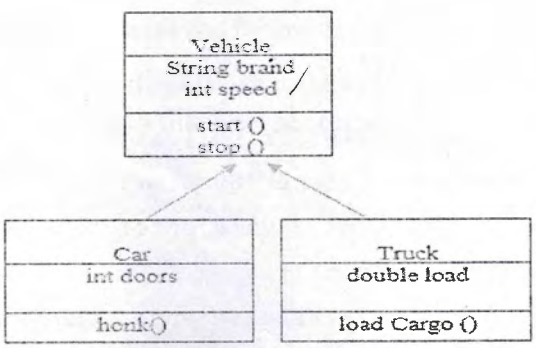

Figure:01

Base Class: `Vehicle`

Define the following methods:

- `void start()`: to print `"Vehicle started."`.
- `void stop()`: to print `"Vehicle stopped."`.

- Subclass: `Car`
  - Define a subclass `Car` that extends `Vehicle`.
  - Add the following additional attribute:
    - `int doors`: the number of doors in the car.
  - Define a method:
    - `void honk()`: to print `"Car horn: Beep beep!"`.
  - Override the `start()` method to print `"Car started."`.

- Subclass: `Truck`
  - Define a subclass `Truck` that extends `Vehicle`.
  - Add the following additional attribute:
    - `double load`: the load capacity of the truck.
  - Define a method:
    - `void loadCargo()`: to print `"Cargo loaded."`.
  - Override the `start()` method to print `"Truck started."`.

- Demonstration in the `Main` Class
  - Create objects for both `Car` and `Truck` classes.
  - Call the `start()` and `stop()` methods for both objects to demonstrate polymorphism.
  - Call their respective methods (`honk()` for `Car` and `loadCargo()` for `Truck`).

b) Explain different types of polymorphism in Java, and provide relevant examples for each type. [04]

3. a) The following code that accepts two numbers from the user and performs division. [05]

```
import java.util.Scanner;
class DivisionTest {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Please enter the numerator");
        int num1 = s.nextInt();
        System.out.println("Please enter the denominator");
        int num2 = s.nextInt();
        int[] results = new int[5];
```

```
            results[0] = num1 / num2;
            System.out.println("The result of the division is:
" + results[0]);
        }
}
```

Modify the code above with exception handling to:

a. Handle division by zero problem.
b. Handle invalid array index problem.

b) Create an abstract class `Bank` that contains an abstract method `getBalance()`. Three banks, `BankA`, `BankB`, and `BankC`, inherit from the `Bank` class. Each of these subclasses has a concrete implementation of the `getBalance()` method that returns the amount deposited in the respective bank: $60000 in `BankA`, $12500 in `BankB`, and $20000 in `BankC`. Write a Java program by calling the `getBalance()` method for each bank using objects of the respective subclasses. [05]

4. a) Design a Java program to demonstrate multiple inheritance using interfaces. Create two interfaces, `Printable` and `Showable`. [05]

Each containing a method `print()` and `show()` respectively. Then, create a class `Document` that implements both interfaces and provides definitions for the `print()` and `show()` methods which will show the following output:

Expected OutPut:

```
Printing the document...
Displaying the document...
```

b) Explain difference between errors and exceptions. Provide examples of common errors and exceptions. [03]

c) "Multiple Inheritance is not supported in java, but this problem can be solved using Interface"- **Justify** this statement. [02]

5. a) Create an interface `Animal` with the following methods: [04]

1. `eat()`: A method to display what the animal eats.
2. `sound()`: A method to display the sound the animal makes.

Create two classes, `Dog` and `Cat`, that implement the `Animal` interface.

1. In the `Dog` class, the `eat()` method should print "Dog eats bones," and the `sound()` method should print "Dog barks."
2. In the `Cat` class, the `eat()` method should print "Cat eats fish," and the `sound()` method should print "Cat meows."

In the `main()` method:

1. Create objects of the `Dog` and `Cat` classes.
2. Call the `eat()` and `sound()` methods using the respective objects.

Ensure the program demonstrates polymorphism by using the `Animal` interface as the reference type to call the methods.

b) Define the thread in OOP. Explain how threads enable multitasking in a program with the lifecycle of a thread. [03]

c) Create a class `MyThread` that extends the `Thread` class and overrides the `run()` method to print the thread name and a message five times. In the `main()` method, create and start two threads of the `MyThread` class. [03]