Case Study III

INHERITANCE, INTERFACES and EXCEPTION HANDLING

Roll No: **21118**

Problem Statement:

Demonstrate inheritance interfaces and exception handling in Java using online banking as the example.

Theory:

A *class* is a user-defined data type from which objects can be created. It represents the set of properties or methods that are common to all objects of one type.

Inheritance is the mechanism of deriving a new class from an existing class. It is an important feature of OOP which provides reusability.

An *interface* in Java is a blueprint of a class. It has static constants and abstract methods. The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java. Java Interface also represents the IS-A relationship. It cannot be instantiated just like the abstract class.

Important terminology:

- Super Class: The class whose features are inherited is known as superclass (or a base class or a parent class).
- Sub Class: The class that inherits the other class is known as a subclass (or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.
- Reusability: Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

In Java, inheritance is achieved using the extends keyword.

Requirements:

1. For security reasons user profile and authentication is must
2. Account Details after logging in
3. Deposit money in account

4. Withdraw money from account
5. Transfer money from one account to another
6. Secure payments

Abstract data type and class structure:

For basic banking system following classes will be required:

- Interface *userInterface*:
    - Operations
        - showData()
        - getData()

- *Person Class*: This class will impement userInterface
    - Instance variable
        - Name
        - Age
        - Gender
    - Methods
        - showData()
        - getData()

- *Bank Account Class*: This class will inherit Person class
    - Instance variables
        - accountNumber
        - balance
        - count
    - Methods
        - depositAmount()
        - withdrawAmount()
        - transferAmount()
        - getAccountInfo()

Programming Language – Java

Programming Paradigm – Object Oriented Programming

Why OOP?

1) The data is encapsulated. Programmer can assure privacy of data.
2) There will be various objects for instance customer, bank account and hence OOP is used.

3) Certain features could be required in more than two classes for which inheritance can be used.

Source Code of banking system:

```
import java.util.Scanner;

public class Person {
	private String name;
	private int age;
	private String gender;

	public void showData() {
		System.out.println("Person Details are as follows: ");
		System.out.println("Name: " + name);
		System.out.println("Age: " + age);
		System.out.println("Gender: " + gender);
	}

	public void getData() {
		Scanner scanner = new Scanner(System.in);

		System.out.println("Please enter the following details: ");
		System.out.println("Name: ");
		try {
		this.name = scanner.nextLine();
		}
		catch(Exception e) {
			System.out.println("Please enter correct name.");
		}
		System.out.println("Age: ");
		this.age = scanner.nextInt();
		scanner.nextLine();
		System.out.println("Gender: ");
		this.gender = scanner.nextLine();

		System.out.println("Account details are successfully entered.");

		scanner.close();

	}
}



public class BankAccount extends Person {
	private int accn;
	private double balance;
	private static int cnt;

	public BankAccount() {
		cnt++;
	}

	public int getTotalAccounts() {
		return cnt;
	}
```

```
	public void deposit(double x) {
		balance += x;
		System.out.println("Deposited successfully.");
	}

	public void withdraw(double x) {
		if (balance < x) System.out.println("Withdrawal unsuccessful.");
		else {
			balance -= x;
			System.out.println("Withdrawal successful.");
		}
	}

	public void transfer(BankAccount another, double x) {
		if (balance < x) System.out.println("Transfer failed.");
		else {
			balance -= x;
			another.deposit(x);
			System.out.println("Transfer successful.");
		}
	}

	public void getAccountInfo() {
		super.showData();
		System.out.println("Bank detail are as follows:");
		System.out.println("Account number: " + accn);
		System.out.println("Balance: " + balance);
	}
}
```

Conclusion:

In this case study assignment, I have learned and programmed concept of inheritance, interface and exception handling in Java by implementing online banking system application.