LAB EXAM OOPs with Java

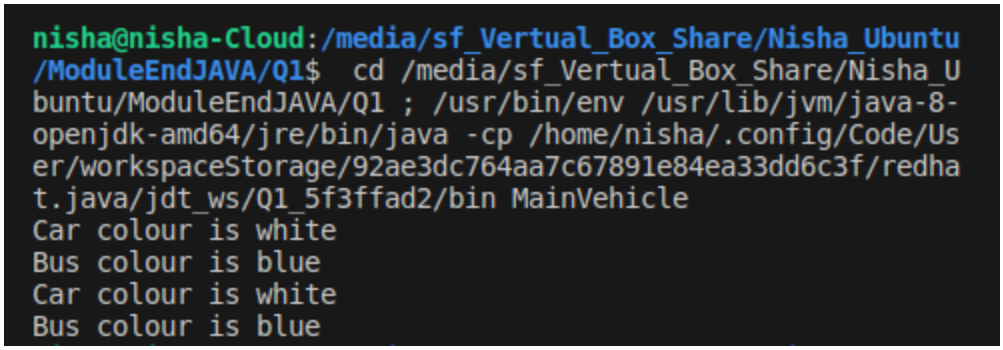
Nisha Elizabeth

1. Create an abstract class Vehicle with one abstract function color(). Create two sub classes Car and Bus from this . Invoke the function through the instance of car and bus. Also use the abstract class reference that invokes that function in main

MainVehicle.java > Vehicle

```java
/** Create an abstract class Vehicle with one abstract function color().
 * Create two sub classes Car and Bus from this .
 * Invoke the function through the instance of car and bus.
 * Also use the abstract class reference that invokes that function in main.**/


abstract class Vehicle {
    public abstract String colour();
}

class Car extends Vehicle {
    @Override
    public String colour() {
        return "Car colour is white";
    }
}

class Bus extends Vehicle {
    @Override
    public String colour() {
        return "Bus colour is blue";
    }
}

public class MainVehicle {
    Run | Debug
    public static void main(String[] args) {
        Vehicle car = new Car(); // Create instances of Car and Bus
        Vehicle bus = new Bus();

        // call the color() function through instances of Car and Bus
        System.out.println(car.colour());
        System.out.println(bus.colour());

        // call the color() function using the abstract class reference
        Vehicle vehicle1 = new Car();
        Vehicle vehicle2 = new Bus();
        System.out.println(vehicle1.colour());
        System.out.println(vehicle2.colour());
    }
}
```

Output:



2. Create a class named “Account” with following members
a. Data Members
a. accNo
b. Name
c. accType
d. accBalance
b. Function members
a. Constructor to accept all values
b. Deposit(int amt) accepting amount and adds with the accBalance
c. Withdraw(int amt) accepting amount to subtract from the accBalance
d. checkBalalce() to return the present account balance

Inherit classes such as “SavingsAccount” “PrivilegedAccount” classes from the class “Account”. The rules of these account types are as follows:-

a. SavingAccount
a. Deposit – can deposit upto a maximum of 50000 only in one transaction.
b. Withdraw – a minimum balance of 1000 should be there at any time.
b. PrivilegedAccount
a. Deposit – can deposit any amount
b. Withdraw – can take an overdraft of maximum 5000. Ie, account balance can go upto - 5000.

Write a main() with a menu driven option, which creates different types of account and do transactions on these accounts.

```java
public class Account {
    private int accNo;
    private String name;
    private String accType;
    private double accBalance;

    // Constructor to accept all values
    public Account(int accNo, String name, String accType, double accBalance) {
        this.accNo = accNo;
        this.name = name;
        this.accType = accType;
        this.accBalance = accBalance;
    }

    // Deposit method accepting amount and adding it to the accBalance
    public void deposit(double amt) {
        if (amt > 0) {
            accBalance += amt;
            System.out.println("Deposited: Rs" + amt);
        } else {
            System.out.println("Invalid deposit amount. Please enter a positive amount.");
        }
    }
```

```java
    // Withdraw method accepting amount to subtract from the accBalance
    public void withdraw(double amt) {
        if (amt > 0 && amt <= accBalance) {
            accBalance -= amt;
            System.out.println("Withdrawn: Rs" + amt);
        } else if (amt <= 0) {
            System.out.println("Invalid withdrawal amount. Please enter a positive amount.");
        } else {
            System.out.println("Insufficient funds for withdrawal.");
        }
    }

    // Check balance method to return the present account balance
    public double checkBalance() {
        return accBalance;
    }

    // Getter methods for individual data members
    public int getAccNo() {
        return accNo;
    }

    public String getName() {
        return name;
    }

    public String getAccType() {
        return accType;
    }

    public double getAccBalance() {
        return accBalance;
    }
}
```

```java
// SavingsAccount class

public class SavingsAccount extends Account {
    // Constructor for SavingsAccount
    public SavingsAccount(int accNo, String name, double accBalance) {
        super(accNo, name, accType:"Savings", accBalance);
    }

    // Override deposit method to limit deposit amount
    @Override
    public void deposit(double amt) {
        if (amt > 0 && amt <= 50000) {
            super.deposit(amt); // Call the deposit method of the parent class
        } else {
            System.out.println("Invalid deposit amount for a savings account. Maximum deposit is Rs.50,000.");
        }
    }

    // Override withdraw method to maintain minimum balance
    @Override
    public void withdraw(double amt) {
        if (amt > 0 && checkBalance() - amt >= 1000) {
            super.withdraw(amt); // Call the withdraw method of the parent class
        } else {
            System.out.println("Insufficient balance or withdrawal amount exceeds allowed limit for a savings account.");
        }
    }
}
```

```java
public class PrivilegedAccount extends Account {
    // Constructor for PrivilegedAccount
    public PrivilegedAccount(int accNo, String name, double accBalance) {
        super(accNo, name, accType:"Privileged", accBalance);
    }

    // Override deposit method to allow any amount
    @Override
    public void deposit(double amt) {
        if (amt > 0) {
            super.deposit(amt); // Call the deposit method of the parent class
        } else {
            System.out.println("Invalid deposit amount for a privileged account.");
        }
    }

    // Override withdraw method to allow an overdraft of up to 5,000
    @Override
    public void withdraw(double amt) {
        if (amt > 0 && checkBalance() - amt >= -5000) {
            super.withdraw(amt); // Call the withdraw method of the parent class
        } else {
            System.out.println("Withdrawal amount exceeds allowed limit for a privileged account.");
        }
    }
}
```

```java
import java.util.Scanner;
import java.util.HashMap;

public class MainMenu {
    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        HashMap<Integer, Account> accounts = new HashMap<>();

        while (true) {
            System.out.println("Menu:");
            System.out.println("1. Create a Savings Account");
            System.out.println("2. Create a Privileged Account");
            System.out.println("3. Deposit");
            System.out.println("4. Withdraw");
            System.out.println("5. Check Balance");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter Account Number: ");
                    int accNo = scanner.nextInt();
                    if (!accounts.containsKey(accNo)) {
                        System.out.print("Enter Name: ");
                        scanner.nextLine(); // Consume newline character
                        String name = scanner.nextLine();
                        System.out.print("Enter Initial Balance: ");
                        double initialBalance = scanner.nextDouble();
                        accounts.put(accNo, new SavingsAccount(accNo, name, initialBalance));
                        System.out.println("Savings Account created.");
                    } else {
                        System.out.println("Account with this account number already exists.");
                    }
                    break;
```

```
    case 2:
        System.out.print("Enter Account Number: ");
        int accNoPrivileged = scanner.nextInt();
        if (!accounts.containsKey(accNoPrivileged)) {
            System.out.print("Enter Name: ");
            scanner.nextLine(); // Consume newline character
            String namePrivileged = scanner.nextLine();
            System.out.print("Enter Initial Balance: ");
            double initialBalancePrivileged = scanner.nextDouble();
            accounts.put(accNoPrivileged, new PrivilegedAccount(accNoPrivileged, namePrivileged, initialBalancePrivileged));
            System.out.println("Privileged Account created.");
        } else {
            System.out.println("Account with this account number already exists.");
        }
        break;

    case 3:
        System.out.print("Enter Account Number: ");
        int depositAccNo = scanner.nextInt();
        if (accounts.containsKey(depositAccNo)) {
            Account depositAccount = accounts.get(depositAccNo);
            System.out.print("Enter deposit amount: ");
            double depositAmount = scanner.nextDouble();
            depositAccount.deposit(depositAmount);
            System.out.println("Deposit successful.");
        } else {
            System.out.println("Account with this account number does not exist. Please create an account first.");
        }
        break;
```

```
            case 4:
                System.out.print("Enter Account Number: ");
                int withdrawalAccNo = scanner.nextInt();
                if (accounts.containsKey(withdrawalAccNo)) {
                    Account withdrawalAccount = accounts.get(withdrawalAccNo);
                    System.out.print("Enter withdrawal amount: ");
                    double withdrawalAmount = scanner.nextDouble();
                    withdrawalAccount.withdraw(withdrawalAmount);
                    System.out.println("Withdrawal successful.");
                } else {
                    System.out.println("Account with this account number does not exist. Please create an account first.");
                }
                break;

            case 5:
                System.out.print("Enter Account Number: ");
                int checkBalanceAccNo = scanner.nextInt();
                if (accounts.containsKey(checkBalanceAccNo)) {
                    Account balanceAccount = accounts.get(checkBalanceAccNo);
                    double balance = balanceAccount.checkBalance();
                    System.out.println("Current balance: $" + balance);
                } else {
                    System.out.println("Account with this account number does not exist. Please create an account first.");
                }
                break;

            case 6:
                System.out.println("Exiting the program.");
                scanner.close();

            default:
                System.out.println("Invalid choice. Please select a valid option.");
        }
    }
}
```

Output:

```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/ModuleEndJAVA_LAB/Q2$  /usr/
bin/env /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -cp /home/nisha/.config/Code/Use
r/workspaceStorage/9065ef32282fd40636dadc53da3e3e15/redhat.java/jdt_ws/Q2_59e555e5/bin
MainMenu
Menu:
1. Create a Savings Account
2. Create a Privileged Account
3. Deposit
4. Withdraw
5. Check Balance
6. Exit
Enter your choice: 1
Enter Account Number: 1232456
Enter Name: Nisha
Enter Initial Balance: 50000
Savings Account created.
Menu:
1. Create a Savings Account
2. Create a Privileged Account
3. Deposit
4. Withdraw
5. Check Balance
6. Exit
Enter your choice: 2
Enter Account Number: 123457
Enter Name: Elizabeth
Enter Initial Balance: 85000
Privileged Account created.
Menu:
1. Create a Savings Account
2. Create a Privileged Account
3. Deposit
4. Withdraw
5. Check Balance
6. Exit
Enter your choice: 3
Enter Account Number: 123456
```

```
Enter your choice: 3
Enter Account Number: 123456
Account with this account number does not exist. Please create an account first.
Menu:
1. Create a Savings Account
2. Create a Privileged Account
3. Deposit
4. Withdraw
5. Check Balance
6. Exit
Enter your choice: 1
Enter Account Number: 123456
Enter Name: Jeni
Enter Initial Balance: 15000
Savings Account created.
Menu:
1. Create a Savings Account
2. Create a Privileged Account
3. Deposit
4. Withdraw
5. Check Balance
6. Exit
Enter your choice: 3
Enter Account Number: 15000
Account with this account number does not exist. Please create an account first.
Menu:
1. Create a Savings Account
2. Create a Privileged Account
3. Deposit
4. Withdraw
5. Check Balance
6. Exit
Enter your choice: 3
Enter Account Number: 123456
Enter deposit amount: 2500
Deposited: Rs2500.0
Deposit successful.
```

```
Menu:
1. Create a Savings Account
2. Create a Privileged Account
3. Deposit
4. Withdraw
5. Check Balance
6. Exit
Enter your choice: 5
Enter Account Number: 123456
Current balance: $17500.0
Menu:
1. Create a Savings Account
2. Create a Privileged Account
3. Deposit
4. Withdraw
5. Check Balance
6. Exit
Enter your choice: 4
Enter Account Number: 123457
Enter withdrawal amount: 5500
Withdrawn: Rs5500.0
Withdrawal successful.
Menu:
1. Create a Savings Account
2. Create a Privileged Account
3. Deposit
4. Withdraw
5. Check Balance
6. Exit
Enter your choice: 5
Enter Account Number: 123457
Current balance: $79500.0
Menu:
1. Create a Savings Account
2. Create a Privileged Account
3. Deposit
4. Withdraw
5. Check Balance
6. Exit
Enter your choice: 6
Exiting the program.
```