Name: Vaishnavi Bhende
Date: 16/02/2024

OOPs Assignment

Q 1. Consider “Bank System” and decide classes, variables, methods about this and try to encapsulate it in the concept of encapsulation.

Ans:

```
public class Bank {
    private String accountNumber;
    private double balance;
    private String customerName;
    private String email;
    private String phoneNumber;

    // Constructor
      public Bank(String accountNumber, double balance, String customerName, String email, String phoneNumber) {
        this.accountNumber = accountNumber;
        this.balance = balance;
        this.customerName = customerName;
        this.email = email;
        this.phoneNumber = phoneNumber;
    }

    // Methods to access and modify account information
    public String getAccountNumber() {
        return accountNumber;
    }

    public double getBalance() {
        return balance;
    }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposit of $" + amount + " processed. New balance: $" + balance);
        } else {
            System.out.println("Invalid amount for deposit.");
        }
```

```
    }

    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
                System.out.println("Withdrawal of $" + amount + " processed. New balance: $" +
balance);
        } else {
            System.out.println("Invalid amount for withdrawal.");
        }
    }

    // Additional methods to manage customer information
    public String getCustomerName() {
        return customerName;
    }

    public String getEmail() {
        return email;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public static void main(String[] args) {
        // Creating a new bank account
        Bank myAccount = new Bank("123456789", 1000.0,
                "Vaishnavi", "vaishnavi@gmail.com", "857938768939");

        // Displaying initial account information
        System.out.println("Account Information:");
        System.out.println("Account Number: " + myAccount.getAccountNumber());
        System.out.println("Customer Name: " + myAccount.getCustomerName());
        System.out.println("Email: " + myAccount.getEmail());
        System.out.println("Phone Number: " + myAccount.getPhoneNumber());
        System.out.println("Balance: $" + myAccount.getBalance());

        // Performing transactions
        myAccount.deposit(500.0);
        myAccount.withdraw(200.0);

        // Displaying updated account information
        System.out.println("\nUpdated Account Information:");
```

```
        System.out.println("Balance: $" + myAccount.getBalance());
    }
}
```

Q 2. Create an abstract class Instrument which has the abstract function play.

Create three more subclasses from Instrument which are Piano, Flute, Guitar.

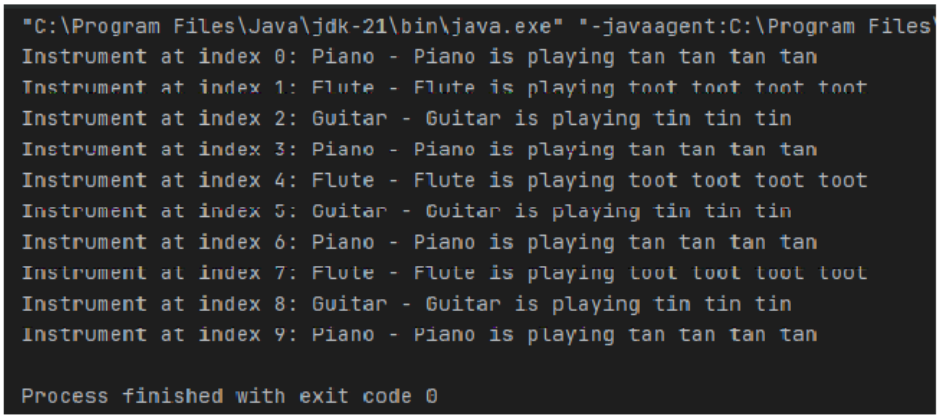
Override the play button inside all three classes printing a message. “Piano is playing tan tan tan tan” for Piano class “Flute is playing toot toot toot toot” for Flute class “Guitar is playing tin tin tin” for Guitar class

You must not allow the user to declare an object of instrument class. Create an array of 10 Instruments. Assign different types of instrument to Instrument reference. Check for the polymorphic behavior of the play method. Use the instance of operator to print which object stored at which index of instrument array

Ans:

```
abstract class Instrument {
    abstract void play();
}

class Piano extends Instrument {
    @Override
    void play() {
        System.out.println("Piano is playing tan tan tan tan");
    }
}

class Flute extends Instrument {
    @Override
    void play() {
        System.out.println("Flute is playing toot toot toot toot");
    }
}
```

```
class Guitar extends Instrument {
    @Override
    void play() {
        System.out.println("Guitar is playing tin tin tin");
    }
}

public class Main {
    public static void main(String[] args) {
        Instrument[] instruments = new Instrument[10];

        // Assigning different types of instruments
        for (int i = 0; i < instruments.length; i++) {
            if (i % 3 == 0) {
                instruments[i] = new Piano();
            } else if (i % 3 == 1) {
                instruments[i] = new Flute();
            } else {
                instruments[i] = new Guitar();
            }
        }

        // Polymorphic behavior demonstration
        for (int i = 0; i < instruments.length; i++) {
            System.out.print("Instrument at index " + i + ": ");
            if (instruments[i] instanceof Piano) {
                System.out.print("Piano - ");
            } else if (instruments[i] instanceof Flute) {
                System.out.print("Flute - ");
            } else if (instruments[i] instanceof Guitar) {
                System.out.print("Guitar - ");
            }
            instruments[i].play();
        }
    }
}
```



Or

Q. 4. Suppose you are designing a system to manage employees in a company. You have a superclass Employee and two subclasses FullTimeEmployee and PartTimeEmpoyee. The Employee class contains attributes such as name, id and method like calculateSalary(), displayDetails(). Both FullTimeEmployee and PartTimeEmployee inherit from the Employee class.

- **Implement the Employee class with appropriate attributes and methods**
- **Implement the FullTimeEmployee class that inherits from Employee and adds specific attributes/methods related to full-time employees, such as salary, benefits.**
- **Implement the PartTimeEmployee class that inherits from Employee and adds specific attributes/methods related to part-time employees, such as hourlyLate, hoursWorked.**

Ans:

```
class Employee {
    private String name;
    private int id;

    public Employee(String name, int id) {
        this.name = name;
        this.id = id;
    }

    public void displayDetails() {
        System.out.println("Name: " + name);
        System.out.println("ID: " + id);
    }
```

```
    // Placeholder method, to be overridden by subclasses
    public double calculateSalary() {
        return 0.0;
    }
}

class FullTimeEmployee extends Employee {
    private double salary;
    private String benefits;

    public FullTimeEmployee(String name, int id, double salary, String benefits) {
        super(name, id);
        this.salary = salary;
        this.benefits = benefits;
    }

    @Override
    public double calculateSalary() {
        return salary;
    }

    public String getBenefits() {
        return benefits;
    }
}

class PartTimeEmployee extends Employee {
    private double hourlyRate;
    private int hoursWorked;

    public PartTimeEmployee(String name, int id, double hourlyRate, int hoursWorked) {
        super(name, id);
        this.hourlyRate = hourlyRate;
        this.hoursWorked = hoursWorked;
    }

    @Override
    public double calculateSalary() {
        return hourlyRate * hoursWorked;
    }

    public double getHourlyRate() {
        return hourlyRate;
    }
```

```
    public int getHoursWorked() {
        return hoursWorked;
    }
}

public class Main {
    public static void main(String[] args) {
        // Creating instances of FullTimeEmployee and PartTimeEmployee
            FullTimeEmployee fullTimeEmployee = new FullTimeEmployee("Ramesh Kumar", 101,
50000, "Health Insurance");
           PartTimeEmployee partTimeEmployee = new PartTimeEmployee("Sunita Sharma", 102,
25.0, 20);

        // Displaying details and salary calculation for FullTimeEmployee
        System.out.println("Full-Time Employee Details:");
        fullTimeEmployee.displayDetails();
        System.out.println("Salary: ₹" + fullTimeEmployee.calculateSalary());
        System.out.println("Benefits: " + fullTimeEmployee.getBenefits());
        System.out.println();

        // Displaying details and salary calculation for PartTimeEmployee
        System.out.println("Part-Time Employee Details:");
        partTimeEmployee.displayDetails();
        System.out.println("Salary: ₹" + partTimeEmployee.calculateSalary());
        System.out.println("Hourly Rate: ₹" + partTimeEmployee.getHourlyRate());
        System.out.println("Hours Worked: " + partTimeEmployee.getHoursWorked());
    }
}
```

Bonus:

Implement a TestEmployee class with a main method to create instances of FullTimeEmployee and PartTimeEmployee, demonstrate inheritance, and all methods from both the superclass and subclass.

Ans:

```
public class TestEmployee {
    public static void main(String[] args) {
        // Creating a full-time employee object
             FullTimeEmployee fullTimeEmployee = new FullTimeEmployee("John", 101, 50000,
"Health Insurance");

        // Displaying details of full-time employee
```

```
        System.out.println("Details of Full-Time Employee:");
        fullTimeEmployee.displayDetails();

        // Calculating and displaying salary of full-time employee
        fullTimeEmployee.calculateSalary();

        System.out.println();

        // Creating a part-time employee object
        PartTimeEmployee partTimeEmployee = new PartTimeEmployee("Alice", 102, 15, 40);

        // Displaying details of part-time employee
        System.out.println("Details of Part-Time Employee:");
        partTimeEmployee.displayDetails();

        // Calculating and displaying salary of part-time employee
        partTimeEmployee.calculateSalary();
    }
}
```