United International University

CSE 1116 (Section: O)

Mid Assignment-1

Please answer the following question. If any examinee is found conducting Unfair means, that individual will get **-100% marks**. **All the best.**

All classes should use constructors, private member variables and getter and setter methods to access / modify variables.

1. 1. Create an Address Book application, where a user can create new record, update record, delete record. Your application should be able to store multiple entries in the address book. Hint: Use array of Address object.
2. 2. Create a Banking Application, where a user can create new account, deposit money, withdraw money and check the balance.
3. 3. Create an Employee record system for a company. The application will help the company to view record of a specific employee, update his info. The Company has 3 types of employee (Salaried, HourlySalaried, Commissioned), your application must handle all types of employee. [Hints: Use Inheritance and array of objects.]

Bonus: Use subclass polymorphism.

Note: 1. Salaried Employee: Salary given at the end of each month.
2. HourlySalaried Employee: Salary calculated as: Salary per hour * No of hours worked.
3. Commissioned Employee: Has a base salary + commission for each product sold.

Solutions:

AddressBookApp.java | BankingApp.java | EmployeeRecordApp.java

```java
import java.util.ArrayList;
import java.util.Scanner;

class Address {  5 usages
    private String Name;  3 usages
    private String phoneNumber;  4 usages
    private String Email;  4 usages

    public Address(String Name, String phoneNumber, String Email) {  1 usage
        this.Name = Name;
        this.phoneNumber = phoneNumber;
        this.Email = Email;
    }

    public String getName() {  2 usages
        return Name;
    }

    public String getPhoneNumber() {  no usages
        return phoneNumber;
    }

    public String getEmail() {  no usages
        return Email;
    }

    public void setPhoneNumber(String phoneNumber) {  1 usage
        this.phoneNumber = phoneNumber;
    }

    public void setEmail(String Email) {  1 usage
        this.Email = Email;
    }

    @Override
    public String toString() {
        return "Name: " + Name + ", Phone: " + phoneNumber + ", Email: " + Email;
    }
}

class AddressBook {  2 usages
    private ArrayList<Address> Address;  5 usages

    public AddressBook() {  1 usage
        Address = new ArrayList<>();
    }

    public void addContact(Address address) {  1 usage
        Address.add(address);
    }

    public void updateContact(String name, String newPhoneNumber, String newEmail) {
        for (Address contact : Address) {
            if (contact.getName().equals(name)) {
                contact.setPhoneNumber(newPhoneNumber);
                contact.setEmail(newEmail);
                return;
            }
        }
        System.out.println("Contact not found.");
    }

    public void deleteContact(String name) {  1 usage
        Address.removeIf(contact -> contact.getName().equals(name));
    }

    public void displayContacts() {  1 usage
        for (Address contact : Address) {
            System.out.println(contact);
        }
    }
}

public class AddressBookApp {
    public static void main(String[] args) {
        AddressBook addressBook = new AddressBook();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("1. Add Contact\n2. Update Contact\n3. Delete Contact\n4
            int choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    System.out.print("Name: ");
                    String name = scanner.nextLine();
                    System.out.print("Phone Number: ");
                    String phoneNumber = scanner.nextLine();
                    System.out.print("Email: ");
                    String email = scanner.nextLine();
                    addressBook.addContact(new Address(name, phoneNumber, email));
                    break;
                case 2:
                    System.out.print("Name of the contact to update: ");
                    String updateName = scanner.nextLine();
                    System.out.print("New Phone No.: ");
                    String newPhone = scanner.nextLine();
                    System.out.print("New Email: ");
                    String newEmailInput = scanner.nextLine();
                    addressBook.updateContact(updateName, newPhone, newEmailInput);
                    break;
                case 3:
                    System.out.print("Name of the contact to delete: ");
                    String deleteName = scanner.nextLine();
                    addressBook.deleteContact(deleteName);
                    break;
                case 4:
                    addressBook.displayContacts();
                    break;
                case 5:
                    scanner.close();
                    return;
                default:
                    System.out.println("Invalid choice.");
            }
        }
    }
}
```

AddressBookApp.java | BankingApp.java | EmployeeRecordApp.java

```java
import java.util.Scanner;

class BankAccount {  2 usages
    private String accountHolder;  2 usages
    private double balance;  5 usages

    public BankAccount(String accountHolder) {  1 usage
        this.accountHolder = accountHolder;
        this.balance = 0.0;
    }

    public void deposit(double amount) {  1 usage
        balance += amount;
        System.out.println("Deposited: " + amount);
    }

    public void withdraw(double amount) {  1 usage
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        } else {
            System.out.println("Insufficient balance.");
        }
    }

    public double getBalance() {  1 usage
        return balance;
    }

    public String getAccountHolder() {  no usages
        return accountHolder;
    }
}

public class BankingApp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter account holder name: ");
        String name = scanner.nextLine();
        BankAccount account = new BankAccount(name);

        while (true) {
            System.out.println("1. Deposit\n2. Withdraw\n3. Check Balance\n4. Exit");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter amount to deposit: ");
                    double depositAmount = scanner.nextDouble();
                    account.deposit(depositAmount);
                    break;
                case 2:
                    System.out.print("Enter amount to withdraw: ");
                    double withdrawAmount = scanner.nextDouble();
                    account.withdraw(withdrawAmount);
                    break;
                case 3:
                    System.out.println("Balance: " + account.getBalance());
                    break;
                case 4:
                    scanner.close();
                    return;
                default:
                    System.out.println("Invalid choice.");
            }
        }
    }
}
```

AddressBookApp.java | BankingApp.java | EmployeeRecordApp.java

```java
import java.util.ArrayList;
import java.util.Scanner;

abstract class Employee {  6 usages  3 inheritors
    private String Name;  2 usages
    private String ID;  2 usages

    public Employee(String Name, String ID) {  3 usages
        this.Name = Name;
        this.ID = ID;
    }

    public String getName() {  1 usage
        return Name;
    }

    public String getId() {  2 usages
        return ID;
    }

    public abstract double calculateSalary();  1 usage  3 implementations
}

class SalariedEmployee extends Employee {  1 usage
    private double MonthlySalary;  2 usages

    public SalariedEmployee(String Name, String ID, double monthlySalary) {  1 usage
        super(Name, ID);
        this.MonthlySalary = monthlySalary;
    }

    @Override  1 usage
    public double calculateSalary() {
        return MonthlySalary;
    }
}

class HourlySalariedEmployee extends Employee {  1 usage
    private double hourlyRate;  2 usages
    private int hoursWorked;  2 usages

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

    @Override  1 usage
    public double calculateSalary() {
        return hourlyRate * hoursWorked;
    }
}

class CommissionedEmployee extends Employee {  1 usage
    private double baseSalary;  2 usages
    private double commission;  2 usages

    public CommissionedEmployee(String name, String id, double baseSalary, double comm
        super(name, id);
        this.baseSalary = baseSalary;
        this.commission = commission;
    }

    @Override  1 usage
    public double calculateSalary() {
        return baseSalary + commission;
    }
}

class EmployeeRecordSystem {  2 usages
    private ArrayList<Employee> Employees;  3 usages

    public EmployeeRecordSystem() {  1 usage
        Employees = new ArrayList<>();
    }

    public void addEmployee(Employee employee) {  3 usages
        Employees.add(employee);
    }

    public void displayEmployeeInfo(String id) {  1 usage
        for (Employee employee : Employees) {
            if (employee.getId().equals(id)) {
                System.out.println("Name: " + employee.getName() + ", ID: " + employee
                return;
            }
        }
        System.out.println("Employee not found.");
    }
}

public class EmployeeRecordApp {
    public static void main(String[] args) {
        EmployeeRecordSystem employeeSystem = new EmployeeRecordSystem();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("1. Add Employee\n2. View Employee\n3. Exit");
            int choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    System.out.print("Enter employee type (1: Salaried, 2: Hourly, 3:
                    int type = scanner.nextInt();
                    scanner.nextLine(); // consume newline
                    System.out.print("Name: ");
                    String name = scanner.nextLine();
                    System.out.print("ID: ");
                    String id = scanner.nextLine();
                    if (type == 1) {
                        System.out.print("Monthly Salary: ");
                        double salary = scanner.nextDouble();
                        employeeSystem.addEmployee(new SalariedEmployee(name, id, sala
                    } else if (type == 2) {
                        System.out.print("Hourly Rate: ");
                        double rate = scanner.nextDouble();
                        System.out.print("Hours Worked: ");
                        int hours = scanner.nextInt();
                        employeeSystem.addEmployee(new HourlySalariedEmployee(name, id,
                    } else if (type == 3) {
                        System.out.print("Base Salary: ");
                        double base = scanner.nextDouble();
                        System.out.print("Commission: ");
                        double commission = scanner.nextDouble();
                        employeeSystem.addEmployee(new CommissionedEmployee(name, id,
                    } else {
                        System.out.println("Invalid type.");
                    }
                    break;
                case 2:
                    System.out.print("Enter employee ID to view: ");
                    String viewId = scanner.nextLine();
                    employeeSystem.displayEmployeeInfo(viewId);
                    break;
                case 3:
                    scanner.close();
                    return;
                default:
                    System.out.println("Invalid choice.");
            }
        }
    }
}
```

Outputs:

```
C:\Users\anikr\.jdks\openjdk-23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 20
1. Add Contact
2. Update Contact
3. Delete Contact
4. Display Contacts
5. Exit
1
Name: Anik Roy
Phone Number: 01521428525
Email: xyz@gmail.com
1. Add Contact
2. Update Contact
3. Delete Contact
4. Display Contacts
5. Exit
4
Name: Anik Roy, Phone: 01521428525, Email: xyz@gmail.com
1. Add Contact
2. Update Contact
3. Delete Contact
4. Display Contacts
5. Exit
```

```
C:\Users\anikr\.jdks\openjdk-23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 20
Enter account holder name: Anik Roy
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
2
Enter amount to withdraw: 5000
Insufficient balance.
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
1
Enter amount to deposit: 50000
Deposited: 50000.0
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
2
Enter amount to withdraw: 30000
Withdrawn: 30000.0
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
```

```
C:\Users\anikr\.jdks\openjdk-23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 20
1. Add Employee
2. View Employee
3. Exit
1
Enter employee type (1: Salaried, 2: Hourly, 3: Commissioned): 2
Name: Anik Roy
ID: 0112320074
Hourly Rate: 50
Hours Worked: 40
1. Add Employee
2. View Employee
3. Exit
1
Enter employee type (1: Salaried, 2: Hourly, 3: Commissioned): 2
Name: Anik Roy
ID: 55220
Hourly Rate: 50
Hours Worked: 20
1. Add Employee
2. View Employee
3. Exit
2
Enter employee ID to view: 5520
Employee not found.
1. Add Employee
2. View Employee
3. Exit
2
Enter employee ID to view: 5520
Employee not found.
1. Add Employee
2. View Employee
3. Exit
2
Enter employee ID to view: 55220
Name: Anik Roy, ID: 55220, Salary: 1000.0
1. Add Employee
2. View Employee
3. Exit
```