Day 5 – Assignment

Pratik K Kamble
Employee ID : 46263548
15 – 09 – 2022

Q1: Work on the below requirement.

1. Make a parent class Person.

a. Make Employee, UnEmployed and SelfEmployed as the subclasses. Make calculateSalary method as a base method in Person class.
b. Override this calculation method in all the other classes (Employee, Unemployed, and SelfEmployed)
c. Make parameterized constructor in individual derived subclasses which takes calculation parameters (like tax %, Salary, etc) as parameters and does the salary calculations.
d. Make another class CalculateSalary with overloaded parameterized constructors. These parameterized constructors take empType and other parameters as tax%, salary etc. and calculates the salary.

2. Make use of access Specifies in the above program:
a. Make the calculateArea method private in the parent class.
b. Make the overridden methods public in the subclasses.

Solution –

```java
package com.Assignment_day5;

public class Person {
    1 usage
    float salary;
    1 usage
    float tax;

    public Person(float salary, float tax) {
        this.salary = salary;
        this.tax = tax;
    }
    4 usages
    public Person() {
    }
    3 usages   3 overrides
    float calculateSalary() { return -1; }

    private float calculateArea() {
        return -1;
    }
}
```

```java
2 usages
class Employee extends Person {
    3 usages
    float salary;
    2 usages
    float tax;
    2 usages
    float inHandSalary;

    1 usage
    public Employee(float salary, float tax){
        this.salary = salary;
        this.tax = tax;
    }
    3 usages
    @Override
     float calculateSalary() {
        inHandSalary = salary - (salary*(tax/100));
        return inHandSalary;
    }
}
```

```java
2 usages
class UnEmployed extends Person{
    2 usages
    float salary;
    2 usages
    float tax;
    float inHandSalary;
    1 usage
    UnEmployed(){
        this.salary = salary;
        this.tax = tax;
    }
    3 usages
    @Override
    float calculateSalary() { return 0.0f; }
}
```

```java
2 usages
class SelfEmployed extends Person{
    3 usages
    float salary;
    2 usages
    float tax = 5f;
    3 usages
    float assets;
    2 usages
    float inHandSalary;
    1 usage
    SelfEmployed(float salary, float tax, float assets){
        this.salary = salary;
        this.tax = 7f;
        this.assets = assets;
    }
    3 usages
    @Override
    float calculateSalary() {
        this.inHandSalary = this.salary - this.assets - ((this.salary-this.assets)*(this.tax/100));
        return this.inHandSalary;
    }
}
```

```java
6 usages
class CalculateSalary{
    4 usages
    String empType;
    float salary;
    float tax;
    4 usages
    float inHandSalary;
    3 usages
    CalculateSalary(String empType, float salary, float tax, float assets) {
        switch (empType) {
            case "employed":
                tax = 18f;
                this.empType = "employed";
                inHandSalary = salary - (salary*(tax/100));
                break;
            case "unemployed":
                tax = 10f;
                this.empType = "unemployed";
                inHandSalary = 0.0f;
                break;
            case "selfemployed":
                tax = 7f;
                this.empType = "selfemployed";
                inHandSalary = salary - assets - ((salary-assets)*(tax/100));
        }
    }
    3 usages
    void printSalary() { System.out.println("The " + empType + " has total inhand salary of " + inHandSalary); }
}
```

```java
class PersonTest {
    public static void main(String[] args) {
        Person per = new Person();
        Employee emp = new Employee( salary: 25000,  tax: 18f);
        UnEmployed unemp = new UnEmployed();
        SelfEmployed self = new SelfEmployed( salary: 30000,  tax: 5f,  assets: 8000);

        CalculateSalary cal1 = new CalculateSalary( empType: "employed",  salary: 25000,  tax: 18f,  assets: 0);
        CalculateSalary cal2 = new CalculateSalary( empType: "unemployed",  salary: 0,  tax: 0,  assets: 0);
        CalculateSalary cal3 = new CalculateSalary( empType: "selfemployed",  salary: 30000,  tax: 5f,  assets: 8000);

        System.out.println("Inherited Classes test...");
        System.out.println("Employed person has inhand salary of - " + emp.calculateSalary());
        System.out.println("Unemployed Person has inhand salary of - " + unemp.calculateSalary());
        System.out.println("Self-Employed Person has inhand salary of - " + self.calculateSalary());

        System.out.println("");
        System.out.println("Constructor Overloading test...");
        cal1.printSalary();
        cal2.printSalary();
        cal3.printSalary();

    }
}
```

```
D:\so_cket\Capgemini\java\bin\java.exe "-javaagent:D:\Join
Inherited Classes test...
Employed person has inhand salary of - 20500.0
Unemployed Person has inhand salary of - 0.0
Self-Employed Person has inhand salary of - 20460.0

Constructor Overloading test...
The employed has total inhand salary of 20500.0
The unemployed has total inhand salary of 0.0
The selfemployed has total inhand salary of 20460.0

Process finished with exit code 0
```

Q2: Create a class that imitates part of the functionality of the basic data type int. Call the class Int (note different capitalization). The only data in this class is an int variable. Include member functions to initialize an Int to 0, to initialize it to an int value, to display it (it looks just like an int), and to add two Int values.

Write a program that exercises this class by creating one uninitialized and two initialized Int values, adding the two initialized values and placing the response in the uninitialized value, and then displaying this result.

Follow below table for more understanding.

ClassName	Public class Int
Fields	private int sum
Methods	Generate all getter and setter and parameterized constructor
Must have	Default constructor, that will initialize the sum with default value as 0
	Getter for sum field
public void addNumber(int i, int j){}	Will add the value of two ints and keep it in sum
ClassName	public class CheckInt
Method	main(String[] args): to test Int class

Solution –

```java
package com.Assignment_day5;

public class Int {
    5 usages
    private int sum;
    1 usage
    Int() {
        sum = 0;
    }
    Int(int sum) {
        this.sum = sum;
    }
    1 usage
    public int getSum() {
        return sum;
    }
    public void setSum(int sum) {
        this.sum = sum;
    }
    1 usage
    public void addNumber(int i, int j) {
        sum = i + j;
    }
}

class CheckInt {
    public static void main(String[] args) {
        Int intmain = new Int();
        intmain.addNumber( i: 5,  j: 8);
        System.out.println("Testing the Int class and sum comes to be: " +intmain.getSum());
    }
}
```

```
D:\so_cket\Capgemini\java\bin\java.exe "-javaagent:D:\Joining Capgemini\
Testing the Int class and sum comes to be: 13

Process finished with exit code 0
```

Q3: Imagine a tollbooth at a bridge. Cars passing by the booth are expected to pay Rs 50 toll. Mostly they do, but sometimes a car goes by without paying. The tollbooth keeps track of the number of cars that have gone by, and of the total amount of money collected. Model this tollbooth with a class called TollBooth. The two data items are a type int to hold the total number of cars, and a type double to hold the total amount of money collected. A constructor initializes both of these to 0. A member function called payingCar() increments the car total and adds Rs 50 to the cash total. Another function called noPayCar(), increments the car total but adds nothing to the cash total. Finally, a member function called display() displayes the two totals. Make appropriate member functions.

Create a main class to test this app. This program should allow the user to push one key to count a paying car, and another to count a nonpaying car. Pushing the Esc key should cause the program to print out the total cars and total cash and then exit

Solution –

```java
package com.Assignment_day5;

import java.util.Objects;
import java.util.Scanner;

2 usages
public class TollBooth {
    5 usages
    static int noOfCars;
    5 usages
    static double money;
    1 usage
    TollBooth() {
        noOfCars = 0;
        money = 0;
    }
//...
    1 usage
    void payingCar() {
        noOfCars++;
        money += 50;
    }

    1 usage
    void noPayCar() {
        noOfCars++;
        money += 0;
    }
```

```
D:\so_cket\Capgemini\java\bin\java.exe "-javaagent:D:\Joining Capgemini\
y
Paying Car Added
y
Paying Car Added
y
Paying Car Added
y
Paying Car Added
Y
Paying Car Added
Y
Paying Car Added
n
Non Paying Car Added
n
Non Paying Car Added
n
Non Paying Car Added
N
Non Paying Car Added
N
Non Paying Car Added
q
Total Cars passed through TollBooth: 11
Total Money Collected by cars passed through TollBooth: 300.0
Total Number of Non-Paying Cars passed through TollBooth: 5

Process finished with exit code 0
```

```java
    void display() {
        System.out.println("Total Cars passed through TollBooth: " + noOfCars);
        System.out.println("Total Money Collected by cars passed through TollBooth: " + money);
        System.out.println("Total Number of Non-Paying Cars passed through TollBooth: "
                + (int)(noOfCars-(money/50)));
    }
}

class Test {
    public static void main(String[] args) {

        TollBooth booth = new TollBooth();
        Scanner scan = new Scanner(System.in);
        boolean quit = false;
        String entry;
        while(!quit) {
            entry = scan.next();
            if(Objects.equals(entry, b: "Y") || Objects.equals(entry, b: "y")) {
                booth.payingCar();
                System.out.println("Paying Car Added");
            } else if (Objects.equals(entry, b: "N") || Objects.equals(entry, b: "n")) {
                booth.noPayCar();
                System.out.println("Non Paying Car Added");
            } else if (Objects.equals(entry, b: "q")) {
                quit = true;
            }
        }
        scan.close();
        booth.display();
    }
}
```

```
D:\so_cket\Capgemini\java\bin\java.exe "-javaagent:D:\Joining Capgemini\
y
Paying Car Added
y
Paying Car Added
y
Paying Car Added
y
Paying Car Added
Y
Paying Car Added
Y
Paying Car Added
n
Non Paying Car Added
n
Non Paying Car Added
n
Non Paying Car Added
N
Non Paying Car Added
N
Non Paying Car Added
q
Total Cars passed through TollBooth: 11
Total Money Collected by cars passed through TollBooth: 300.0
Total Number of Non-Paying Cars passed through TollBooth: 5

Process finished with exit code 0
```

Q4: You need to work on Doctor Information System. Below is the problem definition and business requirement.

Problem Definition

You need to store the information of Doctors, which is going to hold information like, name, speciality, and Patient. Patient will have information like name, problem. In this application doctors will be of below specialities. Neurologist, Dermitologist, Dentst, Orthopedist. If patient inquire about Doctor's speciality which is not available in record, specific exceptin must be reported. Data should be stored in array as local repository. Use the standard practice of development, your code must be properly structured and documented.

Business Requirement

• Application must have proper table structure with proper relationships.

• There should not be a prefilled data local repository.

• Perform all the CRUD operations on arrays

• You must use the concepts of oops properly at appropriate situations

• Modular approach must be used in application development

• Create a menu driven program

You must have main menu that must have operations related to Doctor and Patients

Solution –

```java
package com.Assignment_day5;

import java.util.Objects;
import java.util.Scanner;

7 usages
class Doctor{
    6 usages
    String name;
    3 usages
    String speciality;

    1 usage
    public Doctor(String name) { this.name = name; }
    1 usage
    public Doctor(String name, String speciality) {
        this.name = name;
        this.speciality = speciality;
    }

}
```

```java
7 usages
class Patient {
    6 usages
    String name;
    3 usages
    String problem;
    2 usages
    String reqspecial;

    1 usage
    public Patient(String name) { this.name = name; }
    1 usage
    public Patient(String name, String problem, String reqspecial) {
        this.name = name;
        this.problem = problem;
        this.reqspecial = reqspecial;
    }
}
```

```java
public class HospitalManagement {
    11 usages
    Doctor[] doctors = new Doctor[10];
    12 usages
    Patient[] patients = new Patient[10];

    1 usage
    public void addDoctor(String name, String speciality){
        Doctor newd = new Doctor(name, speciality);
        for (int i = 0; i < doctors.length; i++) {
            if(this.doctors[i] == null) {
                this.doctors[i] = newd;
                break;
            }
        }
        System.out.println("Doctor datails added successfully...");
        System.out.println("");
    }

    1 usage
    public void addPatient(String name, String problem, String reqspecial) {
        Patient newp = new Patient(name, problem, reqspecial);
        for (int i = 0; i < this.patients.length; i++) {
            if(patients[i] == null) {
                patients[i] = newp;
                break;
            }
        }
        System.out.println("Patient datails added successfully...");
        System.out.println("");
    }
```

```java
public void removeDoctor(String name) {
    Doctor rd = new Doctor(name);
    for(int i = 0; i < this.doctors.length; i++) {
        if(Objects.equals(doctors[i].name, rd.name)) {
            doctors[i] = null;
            break;
        }
    }
    System.out.println("Doctor has been removed successfully...");
    System.out.println("");
}

1 usage
public void removePatient(String name) {
    Patient rp = new Patient(name);
    for(int i = 0; i < this.patients.length; i++) {
        if(Objects.equals(patients[i].name, rp.name)) {
            patients[i] = null;
            break;
        }
    }
    System.out.println("Patient removed successfully...");
    System.out.println("");
}
```

```java
public void showDoctors() {
    System.out.println("All the available doctors in the hospital - ");
    int count = 1;
    for (Doctor doctor : doctors) {
        if (doctor == null) {
        } else {
            System.out.println(count++ + " | " + doctor.name + " | " + doctor.speciality);
        }
    }
    System.out.println("");
}

1 usage
public void showPatients() {
    System.out.println("All the Patients having today's appointment - ");
    int count = 1;
    for (Patient patient : patients) {
        if (patient == null) {
            continue;
        } else {
            System.out.println(count++ + " | " + patient.name + " | " + patient.problem);
        }
    }
    System.out.println("");
}
```

```java
1 usage
public void viewAppointment() {
    int count = 1;
    for(int i = 0; i < doctors.length; i++) {
        if (doctors[i] != null){
            for (int j = 0; j < patients.length; j++) {
                if(patients[j] != null) {
                    if (doctors[i].speciality == patients[j].reqspecial) {
                        System.out.println(count++ + " | " + doctors[i].name
                                + " -|- " + this.patients[j].name + " -|- "
                                + patients[j].problem);
                    }
                }
                System.out.println("Patient is not present in the database...");
                break;
            }
        }
        else {
            System.out.println("Doctor is not present in the database...");
            break;
        }
    }
}
```

```java
public static void main(String[] args) {
    HospitalManagement hm = new HospitalManagement();

    System.out.println("Welcome to the Hospital Management System...");
    System.out.println("");
    Scanner scan = new Scanner(System.in);

    boolean quit = false;
    while(!quit) {
        System.out.println("Choose your Option - ");
        System.out.println("A - Add Doctor");
        System.out.println("B -  Remove Doctor");
        System.out.println("C - Add Patient");
        System.out.println("D - Remove Patient");
        System.out.println("E - Show Todays Appointments");
        System.out.println("Q - Exit !");
        String choice = scan.next();
```

```java
switch (choice) {
    case "A" :
        System.out.println("Please Enter the name of the Doctor - ");
        String dname = scan.next();
        System.out.println("Enter the Speciality - ");
        String speciality = scan.next();
        hm.addDoctor(dname, speciality);
        System.out.println("");
        hm.showDoctors();
        break;
    case "B" :
        System.out.println("Please Enter the name of the Doctor - ");
        String drname = scan.next();
        hm.removeDoctor(drname);
        break;
    case "C" :
        System.out.println("Please Enter the name of the Patient - ");
        String pname = scan.next();
        System.out.println("Please Enter the Problem - ");
        String problem = scan.next();
        System.out.println("Which Specialist do you need ?");
        String reqspecial = scan.next();
        hm.addPatient(pname, problem, reqspecial);
        System.out.println("");
        hm.showPatients();
        break;
```

```
            case "D" :
                System.out.println("Please Enter the name of the Patient - ");
                String prname = scan.next();
                hm.removePatient(prname);
                break;
            case "E" :
                hm.viewAppointment();
                break;
            case "Q" :
                System.out.println("Exiting from the Database...");
                quit = true;
                break;
            default :
                System.out.println("Please Enter the correct choice - ");
                break;
        }
    }
```

```
Welcome to the Hospital Management System...

Choose your Option -
A - Add Doctor
B -  Remove Doctor
C - Add Patient
D - Remove Patient
E - Show Todays Appointments
Q - Exit !
A
Please Enter the name of the Doctor -
Praveen
Enter the Speciality -
Neurologist
Doctor datails added successfully...


All the available doctors in the hospital -
1 | Praveen | Neurologist

Choose your Option -
A - Add Doctor
B -  Remove Doctor
C - Add Patient
D - Remove Patient
E - Show Todays Appointments
Q - Exit !
E
Patient is not present in the database...
Doctor is not present in the database...
```

```
Choose your Option -
A - Add Doctor
B -  Remove Doctor
C - Add Patient
D - Remove Patient
E - Show Todays Appointments
Q - Exit !
B
Please Enter the name of the Doctor -
Praveen
Doctor has been removed successfully...

Choose your Option -
A - Add Doctor
B -  Remove Doctor
C - Add Patient
D - Remove Patient
E - Show Todays Appointments
Q - Exit !
C
Please Enter the name of the Patient -
Neeta
Please Enter the Problem -
Headache
Which Specialist do you need ?
Neurologist
Patient datails added successfully...
```

```
All the Patients having today's appointment -
1 | Neeta | Headache

Choose your Option -
A - Add Doctor
B -  Remove Doctor
C - Add Patient
D - Remove Patient
E - Show Todays Appointments
Q - Exit !
Q
Exiting from the Database...
```