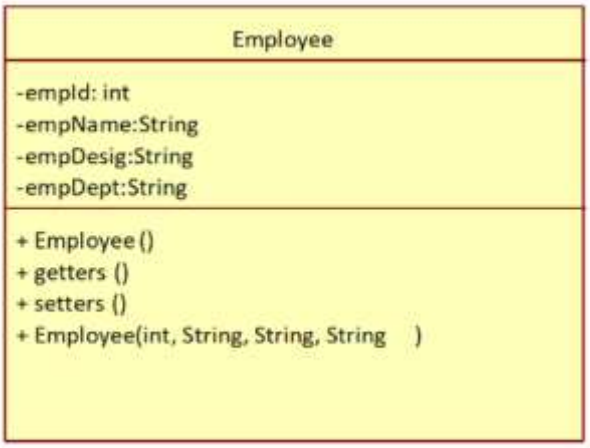
Assignment 5:

1) Create an Employee class with following attributes



Write a program, which creates an instance of employee class and sets the values for all the attributes.

- While setting value for empName, setEmpName() method should check for NullPointer and display appropriate error message.
- While setting value for empDesig, the designation must have any of the following values:developer, tester, Lead or manager. If none of these values is matching then setter method should display **'Invalid designation'** error message.
- While setting value for empDept, the department must have any of the following values: TTH, RCM, Digital, DevOps. If none of these values is matching then setter method should display **'Invalid Dept'** error message.

Source Code:

```
import java.util.Scanner;

public class Employee {

    private String empName;
    private int empId;
    private String empDept;
    private String empDesig;

    public Employee(int empId, String empName, String empDept, String empDesig) {
        this.empName = empName;
        this.empId = empId;
        this.empDept = empDept;
        this.empDesig = empDesig;
    }

    public String getempName() {
        return empName;
    }
```

```java
    public void setempName(String empName) {
        if(empName == null)
            System.out.println("Name cannot be NULL. Invalid Input!");
        else
        this.empName = empName;
    }

    public int getempId() {
        return empId;
    }

    public void setempId(int empId) {
        this.empId = empId;
    }

    public String getempDept() {
        return empDept;
    }

    public void setempDept(String empDept) {
        if(empDept !=null && ( empDept.equals("TTH") || empDept.equals("RCM") ||
empDept.equals("DevOps") || empDept.equals("Digital") ))
            this.empDept = empDept;
        else
            System.out.println("Invalid Department!");
    }

    public String getempDesig() {
        return empDesig;
    }

    public void setempDesig(String empDesig) {
        if(empDesig!=null && (empDesig.equals("Lead") || empDesig.equals("Manager") ||
empDesig.equals("Developer") || empDesig.equals("Tester") ))
            this.empDesig = empDesig;
        else
            System.out.println("Invalid Designation!");

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        // Checking for NullPointerException if the object calling setempName() is null.
If null, it will throw NullPointerException.
        // Note: Checking for e1 == null will bring dead code while printing the message.
Thus try-catch should be used.
        try {
            Employee e1 = new Employee(6828, "Akshit", "Digital" , "Lead");
            System.out.println(e1.getempName());
            System.out.println(e1.getempId());
            System.out.println(e1.getempDept());
            System.out.println(e1.getempDesig());
```

```
        } catch (Exception e) {
            // Handling excpetion with user friendly message.
            System.out.println("Employee object is null. Cannot set name for it");
        }
        sc.close();
    }
}
```

Output:

```
Akshit
6828
Digital
Lead
```

2) **Design and implement applications using basic OOP paradigms.**

- Develop a program that assists bookstore employees. For each book, the program should track the book's title, its price, its year of publication, and the author's name.
- Develop an appropriate Java Class. Create instances of the class to represent these three books:

INPUT TO BE EXECUTED:

- Daniel Defoe, Robinson Crusoe, $15.50, 1719;
- Joseph Conrad, Heart of Darkness, $12.80, 1902;
- Pat Conroy, Beach Music, $9.50, 1996.

Source Code:

```
public class BookStore {
    public static void main(String[] args) {
        Book book1 = new Book("Daniel Defoe", "Robinson Crusoe", 15.50, 1719);
        Book book2 = new Book("Joseph Conrad", "Heart of Darkness", 12.80, 1902);
        Book book3 = new Book("Pat Conroy", "Beach Music", 9.50, 1996);

        System.out.println("Book details are");
        System.out.println("Book 1: " + book1.getTitle() + " by " + book1.getAuthor() + "
(" + book1.getYearOfPublication() + ")");
        System.out.println("Book 2: " + book2.getTitle() + " by " + book2.getAuthor() + "
(" + book2.getYearOfPublication() + ")");
        System.out.println("Book 3: " + book3.getTitle() + " by " + book3.getAuthor() + "
(" + book3.getYearOfPublication() + ")");

    }
}
```

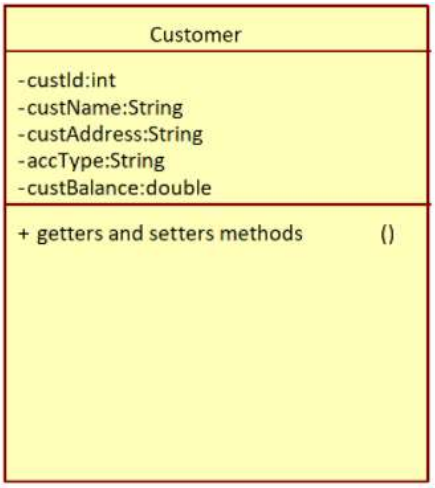
```java
class Book{
    private String title;
    private String author;
    private double price;
    private int yearOfPublication;

    public Book(String title, String author, double price, int yearOfPublication) {
        this.title = title;
        this.author = author;
        this.price = price;
        this.yearOfPublication = yearOfPublication;
    }
    public String getTitle() {
        return this.title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return this.author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public double getPrice() {
        return this.price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    public int getYearOfPublication() {
        return this.yearOfPublication;
    }

    public void setYearOfPublication(int yearOfPublication) {
        this.yearOfPublication = yearOfPublication;
    }

}
```

Output:

```
Book details are
Book 1: Daniel Defoe by Robinson Crusoe (1719)
Book 2: Joseph Conrad by Heart of Darkness (1902)
Book 3: Pat Conroy by Beach Music (1996)
```

3) Design and implement applications using basic OOP paradigms.

XYZ bank wants to maintain customer details. It will register the customer details whenever a person opens an account with the bank. Below is the customer class diagram:



At times, the customer registration process changes, here are the guidelines:

1) Admin may register customer by filling **only ID, name** and **address** details
2) Admin may register customer by filling **only ID** and **name**
3) Admin may register customer by filling **all the details**.

Write an application which implements above scenario.

Note: When other data members which are not initialized through constructors should have appropriate default values.

INPUT SOURCE CODE:

```
Customer customer1 = new Customer(1001, "Kumar", "Rajajinagar, Banglore - 10");
Customer customer2 = new Customer(1002, "Raja");
Customer customer3 = new Customer(1003, "Sthis.hanthi", "Jayanagar, Banglore - 20", "SB",
1500);
```

APPLICATION SOURCE CODE:

```
public class XYZBank {
    public static void main(String[] args) {
        Customer customer1 = new Customer(1001, "Kumar", "Rajajinagar, Banglore - 10");
        Customer customer2 = new Customer(1002, "Raja");
        Customer customer3 = new Customer(1003, "Sthis.hanthi", "Jayanagar, Banglore -
20", "SB", 1500);
        System.out.println("Customer Details are:");
        System.out.println("Customer 1:\n"+ customer1.getDetails());
        System.out.println("Customer 2:\n"+ customer2.getDetails());
        System.out.println("Customer 3:\n"+ customer3.getDetails());
    }
}
```

```
class Customer {

    private int custId;
    private String custName, custAddress, accType;
    private double custBalance;

    public String getDetails() {
        return "Customer Id: " + this.custId + ", Customer Name:" + this.custName + ",
Customer Address: "
                + this.custAddress + ", Customer Account Type: " + this.accType + ",
Customer Balance: "
                + this.custBalance;
    }

    // Constructor overloading
    Customer(int custId, String custName, String custAddress, String accType, double
custBalance) {
        this.custId = custId;
        this.custName = custName;
        this.custAddress = custAddress;
        this.accType = accType;
        this.custBalance = custBalance;
    }

    Customer(int custId, String custName, String custAddress) {
        this.custId = custId;
        this.custName = custName;
        this.custAddress = custAddress;
    }

    Customer(int custId, String custName) {
        this.custId = custId;
        this.custName = custName;
    }
}
```

Output:

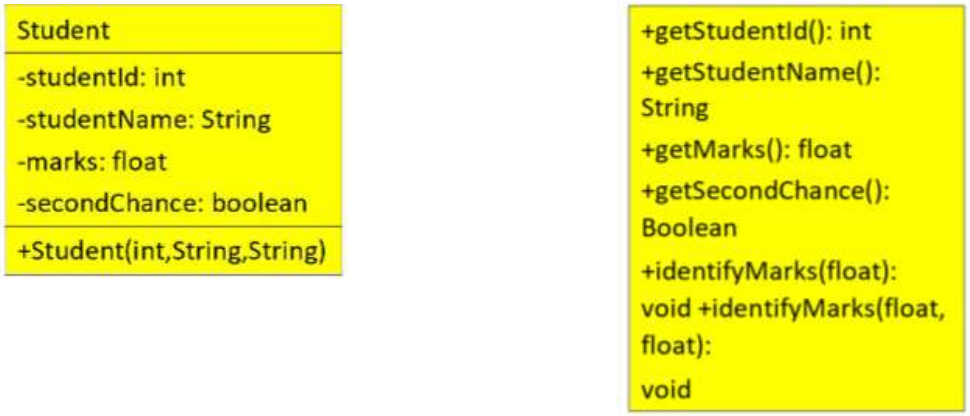
```
Customer Details are:
Customer 1:
Customer Id: 1001, Customer Name:Kumar, Customer Address: Rajajinagar, Banglore - 10,
Customer Account Type: null, Customer Balance: 0.0
Customer 2:
Customer Id: 1002, Customer Name:Raja, Customer Address: null, Customer Account Type:
null, Customer Balance: 0.0
Customer 3:
Customer Id: 1003, Customer Name:Sthis.hanthi, Customer Address: Jayanagar, Banglore - 20,
Customer Account Type: SB, Customer Balance: 1500.0
```

4) Design and implement applications using basic OOP paradigms.

Objective:

Given a class diagram for a problem, use the method and constructor overloading concepts to solve the problem and test using a set of values in an IDE.

Problem Description: The admin of a pre-university college wants to calculate the marks of students in a particular course based on some criteria. Write a Java program to implement the below given class diagram.



Student Class:

Constructor: Initializes studentId, studentName and secondChance.

identifyMarks(float) method:

This method is used to set the marks of the student if the student has cleared in the first chance itself, i.e. second chance is false. This method accepts the marks scored by the student which must be set in the marks instance variable.

identifyMarks (float, float) method:

This method is used to set the marks of the student if the student has taken the second chance i.e. second chance is true. This method accepts the marks scored by the student in the first chance and second chance. The maximum of both these marks must be identified and set in the marks instance variable.

Starter Class:

Write a starter class named **Demo**,

Step1: Create an object of Student class by passing appropriate values to the constructor.
Step2: Based on the value used for second chance instance variable, invoke the appropriate identifyMarks() method.

Step3: Invoke the getter methods and display all the instance variable values of the Student object created. Create one more object (use different value for second chance) by repeating steps 1 to 3 and test your program.

Source Code:

```
public class Demo {
    public static void main(String[] args) {
        Student student1 = new Student(6828, "Akshit Mangotra", false);
        student1.indentifyMarks(95);
        System.out.println("Student 1 DETAILS: \n - Name = " + student1.getStudentName() +
"\n - Id = " + student1.getStudentId() + "\n - Marks = " + student1.getMarks() + "\n - Is
Second Chance = " + student1.isSecondChance());

        Student student2 = new Student(2093, "Amisha Sahu", true);
        student2.indentifyMarks(88, 90);
        System.out.println("Student 2 DETAILS: \n - Name = " + student2.getStudentName() +
"\n - Id = " + student2.getStudentId() + "\n - Marks = " + student2.getMarks() + "\n - Is
Second Chance = " + student2.isSecondChance());
    }
}


class Student{
    private int studentId;
    private String studentName;
    private float marks;
    private boolean secondChance;

    // Method overloading
    public void indentifyMarks(float marks){
        this.marks = marks;
        this.secondChance = false;
    }

    public void indentifyMarks(float marks1, float marks2){
        this.marks = Math.max(marks1, marks2);
        this.secondChance = true;
    }

    //Constructor
    Student(int studentId, String studentName, boolean secondChance) {
        this.studentId = studentId;
        this.studentName = studentName;
        this.secondChance = secondChance;
    }
    // Getters and Setters
    public int getStudentId() {
        return this.studentId;
    }

    public String getStudentName() {
        return this.studentName;
    }

    public float getMarks() {
        return this.marks;
    }
```

```
    public boolean isSecondChance() {
        return this.secondChance;
    }

}
```

Output:

```
Student 1 DETAILS:
 - Name = Akshit Mangotra
 - Id = 6828
 - Marks = 95.0
 - Is Second Chance = false
Student 2 DETAILS:
 - Name = Amisha Sahu
 - Id = 2093
 - Marks = 90.0
 - Is Second Chance = true
```

5) Design and implement applications using basic OOP paradigms.

Create a method which accepts array of 'Student' objects and returns Student object who has scored highest marks.

Note: Each Student object should have following values:

- ID
- Name
- Branch
- Score

Source Code:

```
import java.util.*;
public class MaximumMarks {
    public static void main(String[] args) {
        ArrayList<Student> students = new ArrayList<Student>();
        Student student1 = new Student(1, "Akshit", "CSE", 90);
        students.add(student1);
        Student student2 = new Student(2, "Montek", "EC", 85);
        students.add(student2);
        Student student3 = new Student(3, "Amisha", "IT", 89);
        students.add(student3);
        Student student4 = new Student(4, "Kamal", "ME", 78);
        students.add(student4);

        Student topperStudent = getMaximumMarks(students);
        System.out.println("Topper Student Marks = " + topperStudent.displayMarks());
    }
```

```
    public static Student getMaximumMarks(ArrayList<Student> students) {
        // Sorts students in descending order based on marks.
        Collections.sort(students, new Comparator<Student>() {
            @Override
            public int compare(Student o1, Student o2) {
                return o2.marks - o1.marks;
            }
      });
      return students.get(0);
    }
}

class Student{
    int id, marks;
    String name, branch;

    Student(int id, String name, String branch, int marks){
        this.id = id;
        this.name = name;
        this.marks = marks;
        this.branch = branch;
    }

    public int displayMarks(){
        return marks;
    }


}
```

Output:

```
Topper Student Marks = 90
```