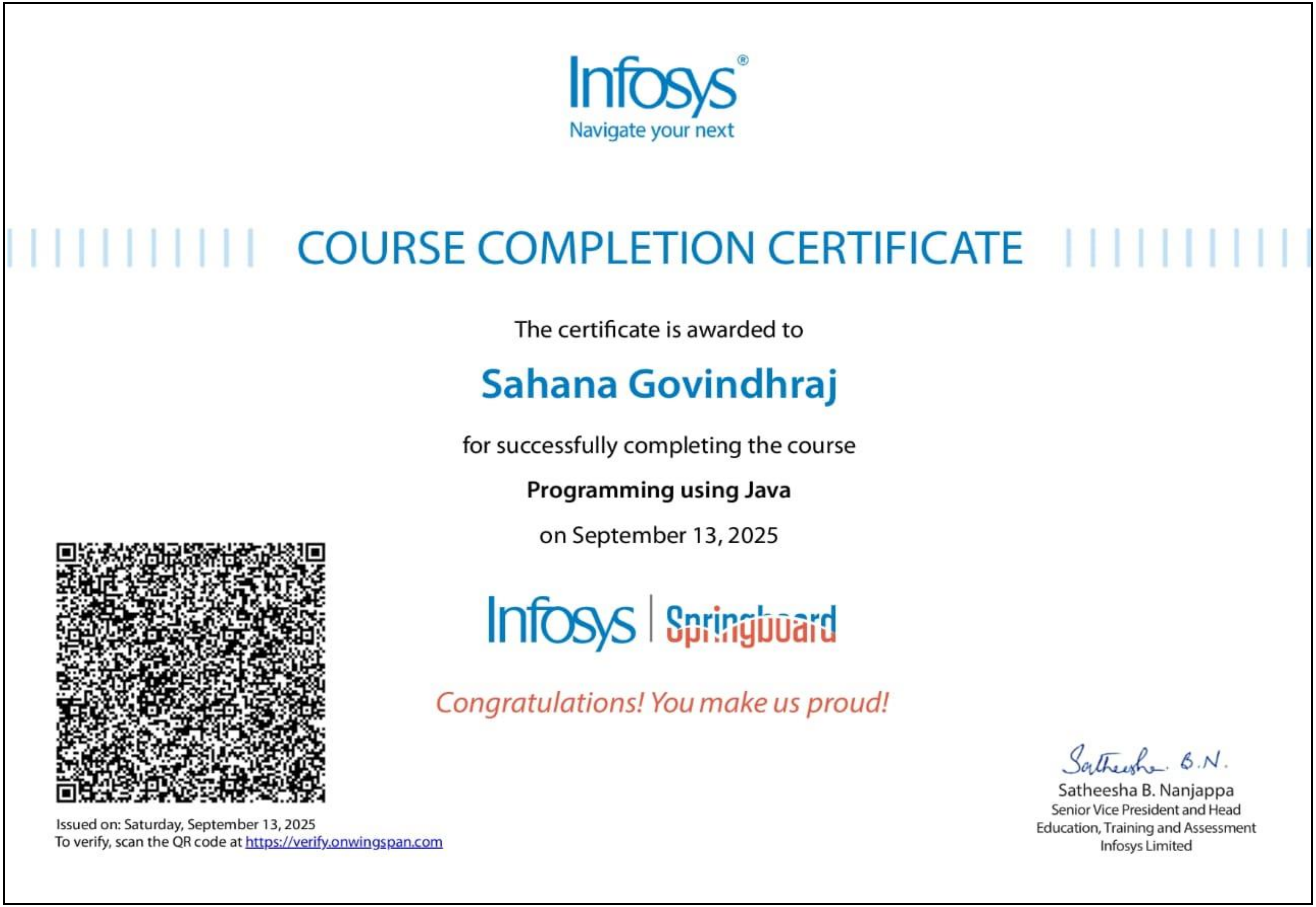


Infosys®
Navigate your next

COURSE COMPLETION CERTIFICATE

The certificate is awarded to

Sahana Govindhraj

for successfully completing the course

Programming using Java

on September 13, 2025

Infosys | Springboard

Congratulations! You make us proud!

Issued on: Saturday, September 13, 2025
To verify, scan the QR code at https://verify.onwingspan.com

Satheesha B. Nanjappa
Senior Vice President and Head
Education, Training and Assessment
Infosys Limited

Infosys springboard assignment

ASSIGNMENT1

Problem Statement

Create a new Java project with "AddressDetails.java" file and implement a Java code to display your address.

Sample Output

```
Door No: D089
Street: St. Louis Street
City: Springfield
ZIP Code: 62729
```

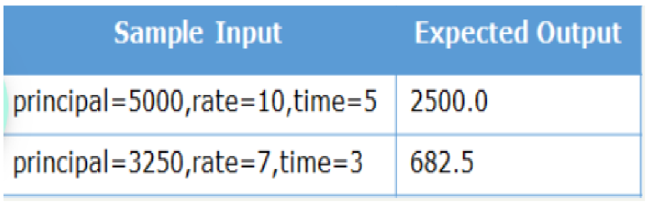
```
class TraineeDetails {

    public static void main(String[] args) {

        System.out.println("Door No: D089”);

        System.out.println("Street: St. Louis Street”);

        System.out.println("City: Springfield”);

        System.out.println("ZIP Code: 62729”);

    }

}
```

Data Types and Operators - Exercise 1

Implement a program to calculate the Simple Interest by using the formula given below and display the calculated Simple Interest.

Simple Interest = (principal*rate of interest*time)/100

Sample Input and Output



Sample Input	Expected Output
principal=5000,rate=10,time=5	2500.0
principal=3250,rate=7,time=3	682.5

```
class SimpleInterest {

           public static void main(String[] args) {

        double principal = 10000;

        double rate = 5;

        double time = 2;

        double simpleInterest = (principal * rate * time) / 100;

 System.out.println("Simple Interest is: " + simpleInterest);

    }

}
```

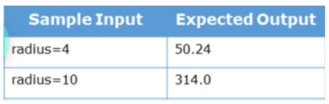
Data Types and Operators - Assignment 1

Problem Statement

Implement a program to find the area of a circle by using the formula given below and display the calculated area.

Area = pi*radius*radius

The value of pi is 3.14

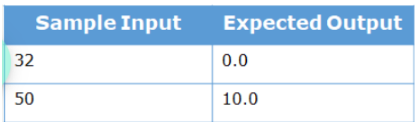


Sample Input	Expected Output
radius=4	50.24
radius=10	314.0

```
class CircleArea {
    public static void main(String[] args) {
        double pi = 3.14;
        double radius = 7; // Set radius here
       double area = pi * radius * radius;
       System.out.println("Area of the circle is: " + area);
    }
}
```

Data Types and Operators - Assignment 2

Problem Statement:Implement a program to convert temperature from Fahrenheit to Celsius degree by using the formula given below and display the converted value. C = ((F-32)/9)*5 where, C represents temperature in Celsius and F represents temperature in Fahrenheit.Sample Input and Output



Sample Input	Expected Output
32	0.0
50	10.0

```
class FahrenheitToCelsius {
    public static void main(String[] args) {
        double fahrenheit = 98.6; // Example Fahrenheit value
        double celsius = ((fahrenheit - 32) * 5) / 9;
        System.out.println(fahrenheit + " Fahrenheit is equal to " + celsius + " Celsius.");
    }
}
```

Selection Control Structure - Exercise 1

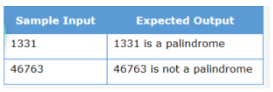
```
class MaximumOfThree {
    public static void main(String[] args) {
        int num1 = 15;
        int num2 = 42;
        int num3 = 27;
        int max = Math.max(num1, Math.max(num2, num3));
System.out.println("The maximum number is: " + max);
    }
}
```

Iteration Control Structure - Exercise 1

Implement a program to check whether a given number is a palindrome.

Palindrome is a sequence that reads the same backwards as forwards.

E.g.: 121, 1331, 2332, 78900987, 123456654321, etc



Sample Input	Expected Output
1331	1331 is a palindrome
46763	46763 is not a palindrome

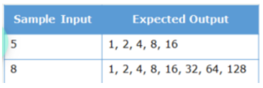
```
class PalindromeNumber {
    public static void main(String[] args) {
        int number = 12321; // Number to check
        int original = number;
        int reversed = 0;
```

```
 // Reverse the number
        while (number > 0) {
            int digit = number % 10;
            reversed = reversed * 10 + digit;
            number /= 10;
        }
// Check if original equals reversed
        if (original == reversed) {
            System.out.println(original + " is a palindrome.");
        } else {
            System.out.println(original + " is NOT a palindrome.");
        }
    }
}
```

Iteration Control Structure - Exercise 2

Implement a program to display the geometric sequence as given below for a given value n, where n is the number of elements in the sequence.

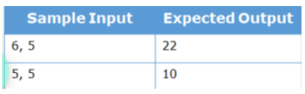
1, 2, 4, 8, 16, 32, 64, ..., 1024



Sample Input	Expected Output
5	1, 2, 4, 8, 16
8	1, 2, 4, 8, 16, 32, 64, 128

```
class GeometricSequence {
    public static void main(String[] args) {
        int n = 11; // Number of elements in the sequence
        int term = 1; // First term of the sequen
      System.out.print("Geometric sequence: ");
        for (int i = 1; i <= n; i++) {
            System.out.print(term);
            if (i != n) {
                System.out.print(", ");
            }
            term *= 2; // Each term is double the previous
        }
    }
}
```

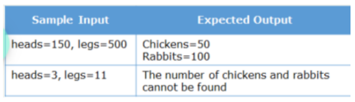
Assignment
Iteration Control Structure - Assignment 1Implement a program to display the sum of two given numbers if the numbers are same. If the numbers are not same, display the double of the sum.



Sample Input	Expected Output
6, 5	22
5, 5	10

```
class SumOrDouble {
    public static void main(String[] args) {
        int num1 = 10; // First number
        int num2 = 10; // Second number
        int sum = num1 + num2;
    if (num1 == num2) {
            System.out.println("Numbers are the same. Sum is: " + sum);
        } else {
            System.out.println("Numbers are different. Double of sum is: " + (2 * sum));
        }
    }
}
```

Iteration Control Structure - Assignment 2Implement a program to find the number of rabbits and chickens in a farm. Given the number of heads and legs of the chickens and rabbits in a farm, identify and display the number of chickens and rabbits in the farm.If the given input cannot make a valid number of rabbits and chickens, then display an appropriate message.



Sample Input	Expected Output

heads=150, legs=500	Chickens=50
	Rabbits=100

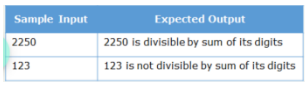
heads=3, legs=11	The number of chickens and rabbits cannot be found

```
class FarmAnimals {
    public static void main(String[] args) {
        int heads = 10; // Total number of heads
        int legs = 32;  // Total number of legs
        // Let c = number of chickens, r = number of rabbits
        // Equations:
        // c + r = heads
        // 2*c + 4*r = legs
        int rabbits = (legs - 2 * heads) / 2;
        int chickens = heads - rabbits;// Check for valid solution
        if (rabbits >= 0 && chickens >= 0 && (2 * chickens + 4 * rabbits == legs)) {
            System.out.println("Number of chickens: " + chickens);
            System.out.println("Number of rabbits: " + rabbits);
        } else {
            System.out.println("No valid solution for the given heads and legs.");
        }
    }
}
```

Iteration Control Structure - Assignment 3

Implement a program to find out whether a number is divisible by the sum of its digits.

Display appropriate messages.



Sample Input	Expected Output
2250	2250 is divisible by sum of its digits
123	123 is not divisible by sum of its digits

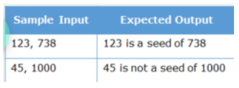
```
class DivisibleByDigitSum {
    public static void main(String[] args) {
        int number = 132; // Change this number to test
        int sumOfDigits = 0;
        int temp = number;
        // Calculate sum of digits
        while (temp > 0) {
            sumOfDigits += temp % 10;
            temp /= 10;
        }
        // Check divisibility
        if (number % sumOfDigits == 0) {
            System.out.println(number + " is divisible by the sum of its digits (" + sumOfDigits + ").");
        } else {
            System.out.println(number + " is NOT divisible by the sum of its digits (" + sumOfDigits + ").");
        }
    }
}
```

Iteration Control Structure - Assignment 4

Implement a program to find out whether a number is a seed of another number.

A number X is said to be a seed of number Y if multiplying X by its every digit equates to Y.

E.g.: 123 is a seed of 738 as 123*1*2*3 = 738



Sample Input	Expected Output
123, 738	123 is a seed of 738
45, 1000	45 is not a seed of 1000

```
class SeedNumber {
    public static void main(String[] args) {
        int X = 123; // Potential seed number
        int Y = 738; // Target number
        int product = X;
        int temp = X;
        // Multiply X by each of its digits
        while (temp > 0) {
            int digit = temp % 10;
            product *= digit;
            temp /= 10;
        }
        // Check if product equals Y
        if (product == Y) {
            System.out.println(X + " is a seed of " + Y);
        } else {
            System.out.println(X + " is NOT a seed of " + Y);
        }
    }
}
```

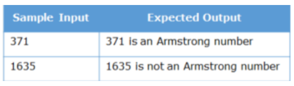
Iteration Control Structure - Assignment 5

Implement a program to check whether a given number is an Armstrong number.An Armstrong number is an n-digit number that is equal to the sum of the nth powers of its individual digits.E.g.: 371 is an Armstrong number as $3^3 + 7^3 + 1^3$=371

1634 is an Armstrong number as $1^4 + 6^4 + 3^4 + 4^4$=1634

Hint

- **Use Math.pow(double a, double b) method to calculate the power of a number**



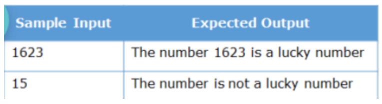
Sample Input	Expected Output
371	371 is an Armstrong number
1635	1635 is not an Armstrong number

```
class ArmstrongNumber {
    public static void main(String[] args) {
        int number = 371; // Number to check
        int sum = 0;
        int temp = number;
        // Count number of digits
        int n = String.valueOf(number).length();
        // Calculate sum of nth powers of digits
        while (temp > 0) {
            int digit = temp % 10;
            sum += Math.pow(digit, n);
            temp /= 10;
        }
        // Check if sum equals original number
        if (sum == number) {
            System.out.println(number + " is an Armstrong number.");
        } else {
            System.out.println(number + " is NOT an Armstrong number.");
        }
    }
}
```

Iteration Control Structure - Assignment 6

Implement a program to check whether a given number is a lucky number.A lucky number is a number whose sum of squares of every even-positioned digit (starting from the second position) is a multiple of 9.

E.g. - 1623 = 6^2+3^2 = 45 is a multiple of 9 and hence is a lucky number.

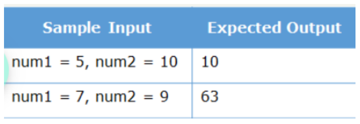


Sample Input	Expected Output
1623	The number 1623 is a lucky number
15	The number is not a lucky number

```
class LuckyNumber {
    public static void main(String[] args) {
        int number = 1623; // Number to check
        String numStr = String.valueOf(number);
        int sum = 0;
        // Iterate through digits
        for (int i = 1; i < numStr.length(); i += 2) {
            // i = 1, 3, 5 ... (even positions starting from 2)
            int digit = Character.getNumericValue(numStr.charAt(i));
            sum += digit * digit;
        }
        // Check if sum is multiple of 9
        if (sum % 9 == 0) {
            System.out.println(number + " is a lucky number.");
        } else {
            System.out.println(number + " is NOT a lucky number.");
        }
    }
}
```

Iteration Control Structure - Assignment 7

Implement a program to find and display the least common multiple (LCM) of two whole numbers.
Least Common Multiple (LCM) of two numbers, num1 and num2 is the smallest positive number that is divisible by both num1 and num2.



Sample Input	Expected Output
num1 = 5, num2 = 10	10
num1 = 7, num2 = 9	63

```
class LCM {
    public static void main(String[] args) {
        int num1 = 12; // First number
        int num2 = 18; // Second number
        // Find LCM using formula: LCM(a,b) = (a*b)/GCD(a,b)
        int lcm = (num1 * num2) / gcd(num1, num2);
        System.out.println("LCM of " + num1 + " and " + num2 + " is: " + lcm);
    }
    // Method to calculate GCD using Euclidean algorithm
    public static int gcd(int a, int b) {
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }
}
```

Iteration Control Structure - Assignment 8

Implement a program to display the below pattern.

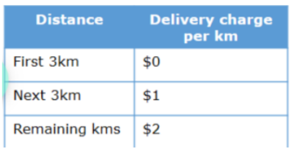
```
*****
****
***
```

```
**

*
```

```
class StarPattern {

    public static void main(String[] args) {

        int rows = 5; // Number of rows

        for (int i = rows; i >= 1; i--) {

            for (int j = 1; j <= i; j++) {

                System.out.print("*");

            }

            System.out.println(); // Move to next line

        }

    }

}
```

Selection Control Structure - Assignment 4

**Food Corner home delivers vegetarian and non-vegetarian meals to its customers based on the order.
A vegetarian combo costs $12 per plate and a non-vegetarian combo costs $15 per plate. Apart from the cost per plate of food, customers are also charged for home delivery based on the distance in kms from the restaurant to the delivery point. The delivery charges are as mentioned below:**



Distance	Delivery charge per km
First 3km	$0
Next 3km	$1
Remaining kms	$2

```
class FoodCornerBill {

    public static void main(String[] args) {

        char foodType = 'V';   // 'V' for vegetarian, 'N' for non-vegetarian

        int quantity = 3;      // Number of plates

        int distance = 5;      // Distance in kms

        int billAmount = -1; // Default invalid value

        // Validate inputs

        if ((foodType == 'V' || foodType == 'N') && distance > 0 && quantity >= 1) {

            // Price per plate

            int pricePerPlate = (foodType == 'V') ? 12 : 15;

            // Delivery charges based on distance

            int deliveryCharges = calculateDeliveryCharges(distance);

            // Final bill

            billAmount = (pricePerPlate * quantity) + deliveryCharges;

        }

        System.out.println(billAmount);

    }

    // Example delivery charge function

    // Modify the rates as needed

    private static int calculateDeliveryCharges(int distance) {

        if (distance <= 3) {
            return 5;
        } else if (distance <= 6) {
            return 10;
        } else {
            return 15;
        }
    }
}
```

Selection Control Structure - Assignment 2

Quadratic equation is an equation with degree 2 in the form of $ax^2 + bx + c = 0$ where a, b and c are the coefficients. Implement a program to solve a quadratic equation.

Find the discriminant value using the formula given below.

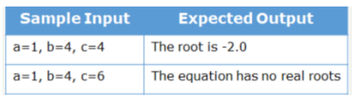
discriminant = $b^2 - 4ac$.If the discriminant is 0, the values of both the roots will be same. Display the value of the root.

If the discriminant is greater than 0, the roots will be unequal real roots. Display the values of both the roots.

If the discriminant is less than 0, there will be no real roots. Display the message "The equation has no real root"

Use the formula given below to find the roots of a quadratic equation.

$x = (-b \pm \text{discriminant})/2a$

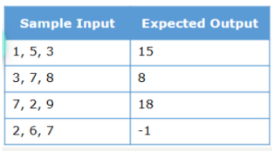


Sample Input	Expected Output
a=1, b=4, c=4	The root is -2.0
a=1, b=4, c=6	The equation has no real roots

```
class QuadraticEquation {
    public static void main(String[] args) {
        double a = 1;  // Coefficient of x^2
        double b = -5; // Coefficient of x
        double c = 6;  // Constant term
        // Check if 'a' is zero
        if (a == 0) {
            System.out.println("Not a quadratic equation.");
            return;
        }
        // Calculate discriminant
        double discriminant = b * b - 4 * a * c;
        if (discriminant > 0) {
            // Two unequal real roots
            double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
            double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
            System.out.println("Roots are real and unequal:");
            System.out.println("Root 1 = " + root1);
            System.out.println("Root 2 = " + root2);
        } else if (discriminant == 0) {
            // Both roots are equal
            double root = -b / (2 * a);
            System.out.println("Roots are real and equal:");
            System.out.println("Root = " + root);
        } else {
            // Discriminant < 0 → No real roots
            System.out.println("The equation has no real root");
        }
    }
}
```

Selection Control Structure - Assignment 3

Problem Statement:Implement a program to calculate the product of three positive integer values. However, if one of the integers is 7, consider only the values to the right of 7 for calculation. If 7 is the last integer, then display -1.Note: Only one of the three values can be 7.



Sample Input	Expected Output
1, 5, 3	15
3, 7, 8	8
7, 2, 9	18
2, 6, 7	-1

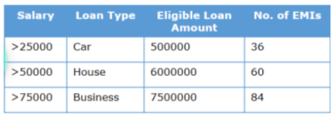
```
class ProductWithSeven {
    public static void main(String[] args) {
        int num1 = 3;
        int num2 = 7;
        int num3 = 4;
        int product = 1;
        if (num1 == 7) {
            // Ignore num1, calculate product of num2 and num3
            product = num2 * num3;
        } else if (num2 == 7) {
            // Ignore num2 and everything to its left, only num3 remains
            product = num3;
        } else if (num3 == 7) {
            // 7 is last → no numbers to multiply
            product = -1;
        } else {
            // No 7 → multiply all three
            product = num1 * num2 * num3;
        }
        System.out.println(product);
    }
}
```

Selection Control Structure - Assignment 5

The Metro Bank provides various types of loans such as car loans, business loans and house loans to its account holders, i.e., customers.Implement a program to determine the eligible loan amount and the EMI that the bank can provide to its customers based on their salary and the loan type they expect to avail.The values required for determining the eligible loan amount and the EMI are:

- **account number of the customer,account balance of the customer,salary of the customer,loan type ,expected loan amount,expected no. of EMIsThe following validations should be performed:The account number should be of 4 digits and its first digit should be 1.The customer should have a minimum balance of $1000 in the account.Display**

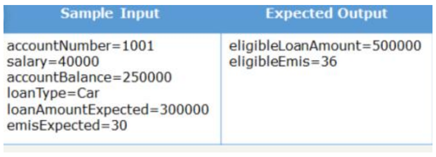
appropriate error messages if the validations fail..If the validations pass, determine whether the bank would provide the loan or not. The bank would provide the loan, only if the loan amount and the number of EMIs expected by the customer is less than or equal to the loan amount and the number of EMIs decided by the bank respectively. The bank decides the eligible loan amount and the number of EMIs based on the below table.



Salary	Loan Type	Eligible Loan Amount	No. of EMIs
>25000	Car	500000	36
>50000	House	6000000	60
>75000	Business	7500000	84

Display the account number, eligible and requested loan amount and the number of EMIs if the bank provides the loan.

Display an appropriate message if the bank does not provide the loan.



Sample Input	Expected Output

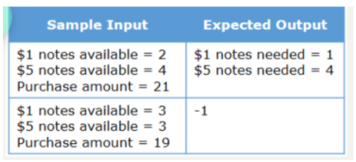
accountNumber=1001
salary=40000
accountBalance=250000
loanType=Car
loanAmountExpected=300000
emisExpected=30
	eligibleLoanAmount=500000
	eligibleEmis=36

```
class MetroBankLoan {
  public static void main(String[] args) {
    // Customer details (change values to test)
    int accountNumber = 1234;
    double accountBalance = 5000;
    double salary = 25000;
    String loanType = "Car"; // Car, House, Business
    double requestedLoanAmount = 40000;
    int requestedEMIs = 36;
   // Validate account number
    if (accountNumber < 1000 || accountNumber > 9999 || String.valueOf(accountNumber).charAt(0) != '1') {
      System.out.println("Invalid account number. Loan cannot be processed.");
      return; // Exclude invalid account number
    }
    // Validate account balance
    if (accountBalance < 1000) {
      System.out.println("Insufficient account balance. Loan cannot be processed.");
      return; // Exclude low balance
    }
    // Validate loan type
    if (!loanType.equalsIgnoreCase("Car") &&
      !loanType.equalsIgnoreCase("House") &&
      !loanType.equalsIgnoreCase("Business")) {
      System.out.println("Invalid loan type. Loan cannot be processed.");
      return; // Exclude invalid loan type
    }
    // Determine eligible loan amount and EMIs based on salary and loan type
    double eligibleLoanAmount = 0;
    int eligibleEMIs = 0;
    if (salary > 25000) {
      switch (loanType.toLowerCase()) {
```

```
                case "car": eligibleLoanAmount = 500000; eligibleEMIs = 36; break;
                case "house": eligibleLoanAmount = 6000000; eligibleEMIs = 60; break;
                case "business": eligibleLoanAmount = 7500000; eligibleEMIs = 84; break;
            }
        } else if (salary > 20000) {
            switch (loanType.toLowerCase()) {
                case "car": eligibleLoanAmount = 300000; eligibleEMIs = 36; break;
                case "house": eligibleLoanAmount = 5000000; eligibleEMIs = 60; break;
                case "business": eligibleLoanAmount = 6000000; eligibleEMIs = 60; break;
            }
        } else if (salary > 10000) {
            switch (loanType.toLowerCase()) {
                case "car": eligibleLoanAmount = 200000; eligibleEMIs = 12; break;
                case "house": eligibleLoanAmount = 3000000; eligibleEMIs = 36; break;
                case "business": eligibleLoanAmount = 3500000; eligibleEMIs = 60; break;
            }
        } else {
            System.out.println("Salary too low for any loan. Loan cannot be processed.");
            return; // Exclude low salary
        }
        // Check requested loan and EMIs against eligibility
        if (requestedLoanAmount <= eligibleLoanAmount && requestedEMIs <= eligibleEMIs) {
            System.out.println("Account Number: " + accountNumber);
            System.out.println("Requested Loan Amount: $" + requestedLoanAmount);
            System.out.println("Eligible Loan Amount: $" + eligibleLoanAmount);
            System.out.println("Requested EMIs: " + requestedEMIs);
            System.out.println("Eligible EMIs: " + eligibleEMIs);
            System.out.println("Loan Approved!");
        } else {
            System.out.println("Loan cannot be provided. Requested amount or EMIs exceed eligibility.");
        }
    }
}
```

Selection Control Structure - Assignment 6

You have x number of $5 notes and y number of $1 notes. You want to purchase an item for amount z. The shopkeeper wants you to provide exact change. You want to pay using a minimum number of notes. How many $5 notes and $1 notes will you use?Implement a program to find out how many $5 notes and $1 notes will be used. If an exact change is not possible, then display -1.



Sample Input	Expected Output
$1 notes available = 2 $5 notes available = 4 Purchase amount = 21	$1 notes needed = 1 $5 notes needed = 4
$1 notes available = 3 $5 notes available = 3 Purchase amount = 19	-1

```
class ExactChange {
    public static void main(String[] args) {
        int x = 3; // Number of $5 notes available
        int y = 5; // Number of $1 notes available
        int z = 17; // Amount to pay
        int fiveNotesUsed = 0;
        int oneNotesUsed = 0;
        boolean exactChangePossible = false;
        // Use maximum possible $5 notes first
        for (int fives = Math.min(x, z / 5); fives >= 0; fives--) {
            int remaining = z - (fives * 5);
            if (remaining <= y) {
                fiveNotesUsed = fives;
                oneNotesUsed = remaining;
                exactChangePossible = true;
                break;
            }
        }
        if (exactChangePossible) {
            System.out.println("Number of $5 notes used: " + fiveNotesUsed);
            System.out.println("Number of $1 notes used: " + oneNotesUsed);
        } else {
            System.out.println(-1);
        }
    }
}
```

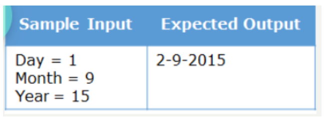
Selection Control Structure - Assignment 7

Implement a program to generate and display the next date of a given date.

The date will be provided as day, month and year as shown in the below table.

The output should be displayed in the format: day-month-year.

Assumption: The input will always be a valid date.



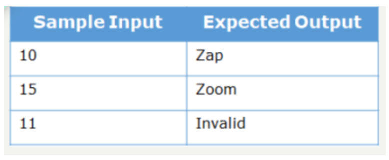
Sample Input	Expected Output

Day = 1
Month = 9
Year = 15
	2-9-2015

```
class NextDate {
    public static void main(String[] args) {
        int day = 28;   // Input day
        int month = 2;  // Input month
        int year = 2024; // Input year
        // Determine the number of days in the current month
        int daysInMonth;
        switch (month) {
            case 1: case 3: case 5: case 7: case 8: case 10: case 12:
                daysInMonth = 31;
                break;
            case 4: case 6: case 9: case 11:
                daysInMonth = 30;
                break;
            case 2:
                if (isLeapYear(year)) {
                    daysInMonth = 29;
                } else {
                    daysInMonth = 28;
                }
                break;
            default:
                daysInMonth = 30; // Default, though input assumed valid
        }
        // Calculate next date
        day++;
        if (day > daysInMonth) {
            day = 1;
            month++;
            if (month > 12) {
                month = 1;
                year++;
            }
        }
        System.out.println("Next Date: " + day + "-" + month + "-" + year);
    }
    // Method to check leap year
    private static boolean isLeapYear(int year) {
        return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
    }
}
```

Selection Control Structure - Assignment 8

Implement a program that displays a message for a given number based on the below conditions.If the number is a multiple of 3, display "Zip".If the number is a multiple of 5, display "Zap".If the number is a multiple of both 3 and 5, display "Zoom",For all other cases, display "Invalid".

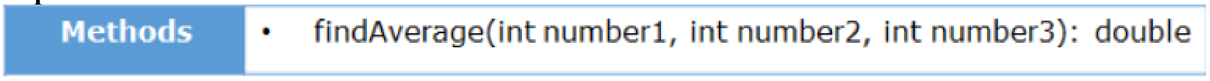


Sample Input	Expected Output
10	Zap
15	Zoom
11	Invalid

```
class ZipZapZoom {
    public static void main(String[] args) {
        int number = 15; // Input number
        if (number % 3 == 0 && number % 5 == 0) {

            System.out.println("Zoom");

        } else if (number % 3 == 0) {

            System.out.println("Zip");

        } else if (number % 5 == 0) {

            System.out.println("Zap");

        } else {

            System.out.println("Invalid");

        }

    }

}
```

Methods - Exercise 1

Implement a class Calculator with the method mentioned below.

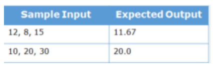


Methods	• findAverage(int number1, int number2, int number3): double

Method Description

findAverage()

Calculate the average of three numbers.Return the average rounded off to two decimal digits
Test the functionalities using the provided Tester class.

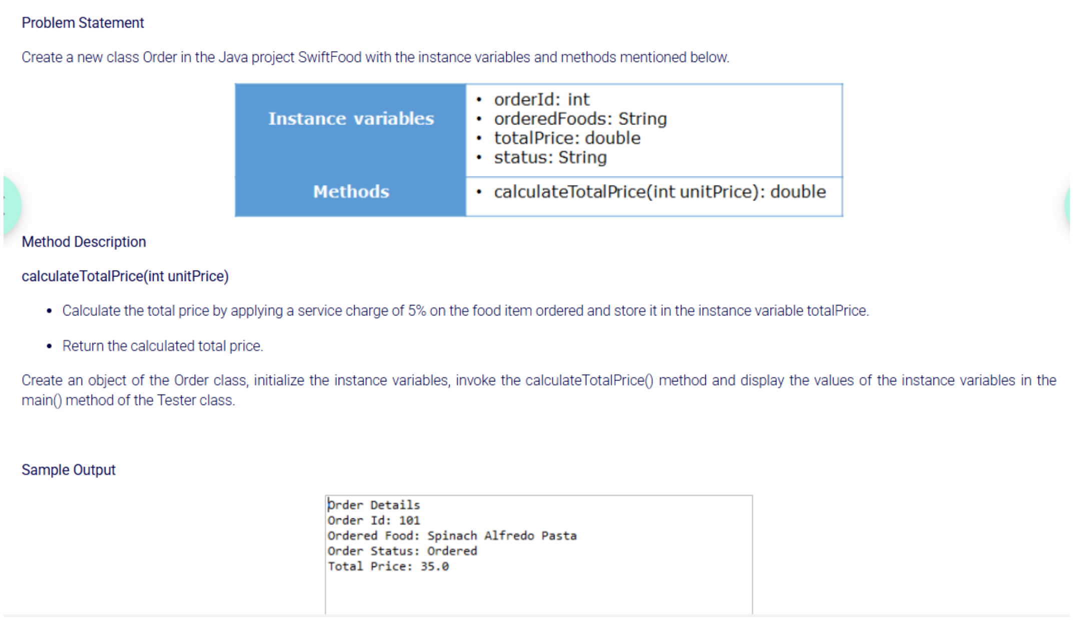


Sample Input	Expected Output
12, 8, 15	11.67
10, 20, 30	20.0

Hint: For round-off to two decimal digits:double num1 = 65, num2 = 175;
double num3 = num1/num2;System.out.println(Math.round(num3*100.0)/100.0);

```
import java.math.BigDecimal;

import java.math.RoundingMode;

public class Calculator {

    // Method to calculate average of three numbers rounded to 2 decimal places

    public double findAverage(double num1, double num2, double num3) {

        double average = (num1 + num2 + num3) / 3.0;

        // Use BigDecimal for precise rounding

        BigDecimal bd = new BigDecimal(average);

        bd = bd.setScale(2, RoundingMode.HALF_UP);

        return bd.doubleValue();

    }

}
```

Methods-Assignment 1



Problem Statement

Create a new class Order in the Java project SwiftFood with the instance variables and methods mentioned below.

Instance variables	• orderId: int • orderedFoods: String • totalPrice: double • status: String
Methods	• calculateTotalPrice(int unitPrice): double

Method Description

calculateTotalPrice(int unitPrice)

- Calculate the total price by applying a service charge of 5% on the food item ordered and store it in the instance variable totalPrice.
- Return the calculated total price.

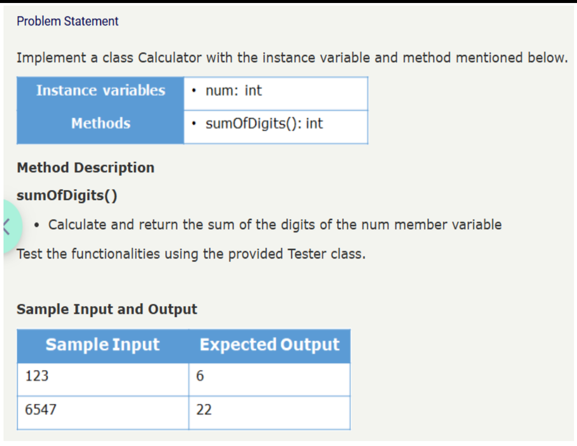
Create an object of the Order class, initialize the instance variables, invoke the calculateTotalPrice() method and display the values of the instance variables in the main() method of the Tester class.

Sample Output

```
Order Details
Order Id: 101
Ordered Food: Spinach Alfredo Pasta
Order Status: Ordered
Total Price: 35.0
```

```
class Order {
    int orderId;
    String orderedFoods;
    double totalPrice;
    String status;
    public double calculateTotalPrice(int unitPrice) {
        totalPrice = unitPrice + (unitPrice * 0.05);
        return totalPrice;
    }
}

public class Methods_Assignment_1 {
    public static void main(String[] args) {
        Order order = new Order();
        order.orderId = 101;
        order.orderedFoods = "Spinach Alfredo Pasta";
        order.status = "Ordered";
        int unitPrice = 33;
        order.calculateTotalPrice(unitPrice);
        System.out.println("Order Details");
        System.out.println("Order Id: " + order.orderId);
        System.out.println("Ordered Food: " + order.orderedFoods);
        System.out.println("Order Status: " + order.status);
        System.out.println("Total Price: " + order.totalPrice);
    }
}
```

Methods - Assignment 3


Problem Statement

Implement a class Calculator with the instance variable and method mentioned below.

Instance variables	• num: int
Methods	• sumOfDigits(): int

Method Description

sumOfDigits()

- Calculate and return the sum of the digits of the num member variable

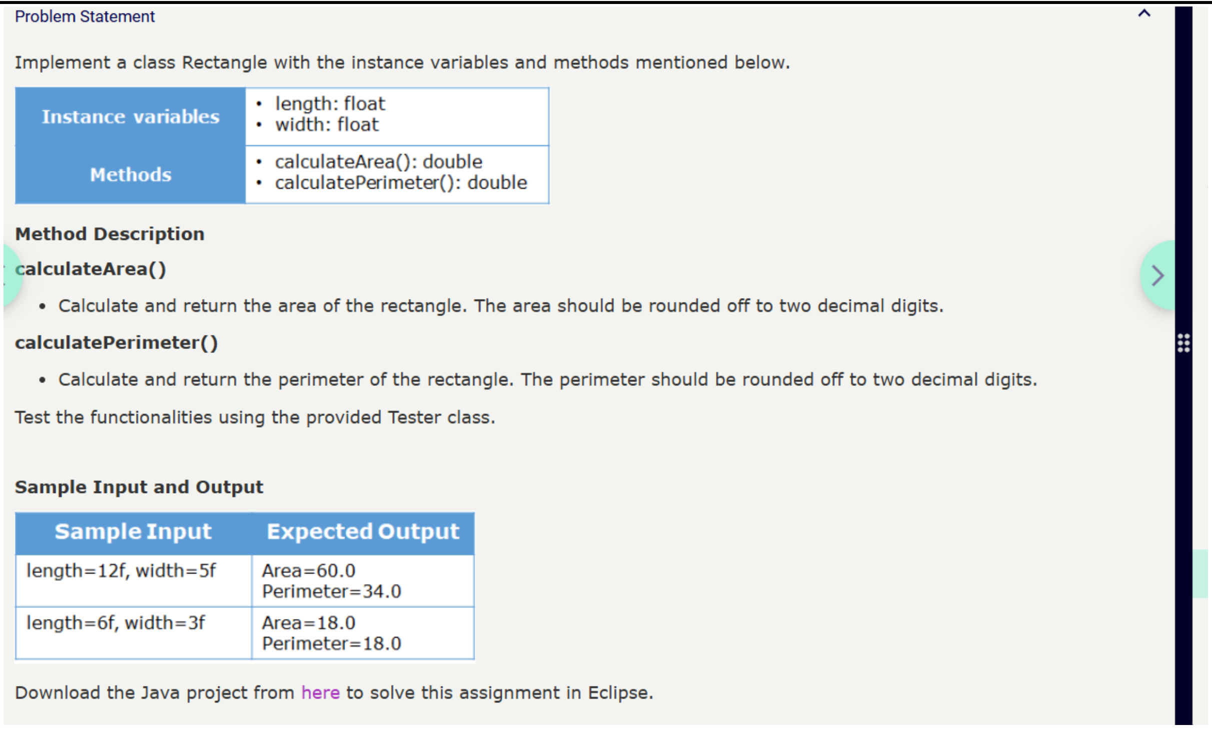
Test the functionalities using the provided Tester class.

Sample Input and Output

Sample Input	Expected Output
123	6
6547	22

```
class Calculator {
            // Implement your code here
            public int num;
            public int sumOfDigits(){
               int sum=0;
               while(this.num >0){
                  sum += this.num%10;
                  this.num /= 10;
               }
               return sum;
            }

}

public class Methods_Assignment_3 {
            public static void main(String args[]) {
                        Calculator calculator = new Calculator();
      calculator.num = 6547;
      int sum = calculator.sumOfDigits();
                        System.out.print(sum);

            }
}
```

Methods - Assignment 4


Problem Statement

Implement a class Rectangle with the instance variables and methods mentioned below.

Instance variables	• length: float • width: float
Methods	• calculateArea(): double • calculatePerimeter(): double

Method Description

calculateArea()

- Calculate and return the area of the rectangle. The area should be rounded off to two decimal digits.

calculatePerimeter()

- Calculate and return the perimeter of the rectangle. The perimeter should be rounded off to two decimal digits.

Test the functionalities using the provided Tester class.

Sample Input and Output

Sample Input	Expected Output

length=12f, width=5f	Area=60.0
	Perimeter=34.0

length=6f, width=3f	Area=18.0
	Perimeter=18.0

Download the Java project from here to solve this assignment in Eclipse.

```
import java.util.Scanner;
class Rectangle {
    //Implement your code here
    public float length;
    public float width;
    public double calculateArea(){
        double result = (this.length*this.width);
        result =  Math.round(result * 100.0) / 100.0;
        return result;
    }

    public double calculatePerimeter(){
        double result =  2*(this.length + this.width);
        result =  Math.round(result * 100.0) / 100.0;
        return result;
    }
}
public class Methods_Assignment_4
 {
            public static void main(String args[]) {
                        Rectangle rectangle=new Rectangle();
                        Scanner scan = new Scanner(System.in);
                        //Assign values to the member variables of Rectangle class
                        rectangle.length = scan.nextInt();
                        rectangle.width = scan.nextInt();
        scan.close();
                        //Invoke the methods of the Rectangle class to calculate the area and perimeter
                        double res1 = rectangle.calculateArea();

                        double res2 = rectangle.calculatePerimeter();

                        //Display the area and perimeter using the lines given below

                        System.out.println("Area of the rectangle is " + res1);

                        System.out.println("Perimeter of the rectangle is "+ res2);

            }

}
```

Encapsulation assignment 2

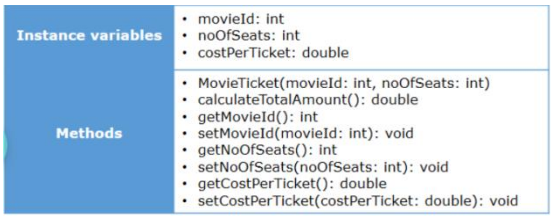
Problem Statement

You have already created the Order and Food classes in the SwiftFood project. Make necessary changes to the Order and Food classes by making all the instance variables private and adding getter and setter methods for the instance variables.

```
public class Food {
// Make instance variables private
private String name;
private double price;
private int quantity;
// Constructor
public Food(String name, double price, int quantity) {
this.name = name;
this.price = price;
this.quantity = quantity;
}
// Getter and Setter for name
public String getName() {
return name;
}
public void setName(String name) {
this.name = name;
}

// Getter and Setter for price
public double getPrice() {
return price;
}
public void setPrice(double price) {
this.price = price;
}
// Getter and Setter for quantity
public int getQuantity() {
return quantity;
}
public void setQuantity(int quantity) {
this.quantity = quantity;
}
}
```

Encapsulation - Assignment 2

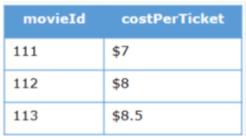
Implement the class MovieTicket with the instance variables and methods mentioned below.



Instance variables	• movieId: int • noOfSeats: int • costPerTicket: double
Methods	• MovieTicket(movieId: int, noOfSeats: int) • calculateTotalAmount(): double • getMovieId(): int • setMovieId(movieId: int): void • getNoOfSeats(): int • setNoOfSeats(noOfSeats: int): void • getCostPerTicket(): double • setCostPerTicket(costPerTicket: double): void

Method Description

MovieTicket (int movieId, int noOfSeats).Initialize the member variables movieId and noOfSeats appropriately with the values passed to the constructor.,calculateTotalAmount().Calculate the total amount to be paid based on the costPerTicket and noOfSeats by applying a 2% tax on the total amount.The costPerTicket for the movies are provided in the below table. Set the value of costPerTicket instance variable appropriately.

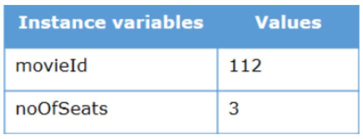
- 

movieId	costPerTicket
111	$7
112	$8
113	$8.5

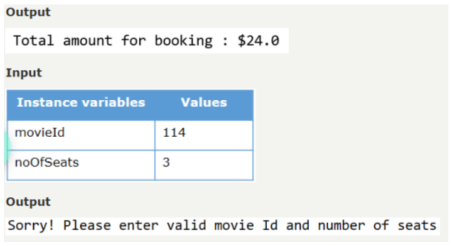
  Note: Return the total amount calculated after rounding off using the Math.round(double a) method.

Test the functionalities using the provided Tester class.

Sample Input and Output



Instance variables	Values
movieId	112
noOfSeats	3



Output

```
Total amount for booking : $24.0
```

Input

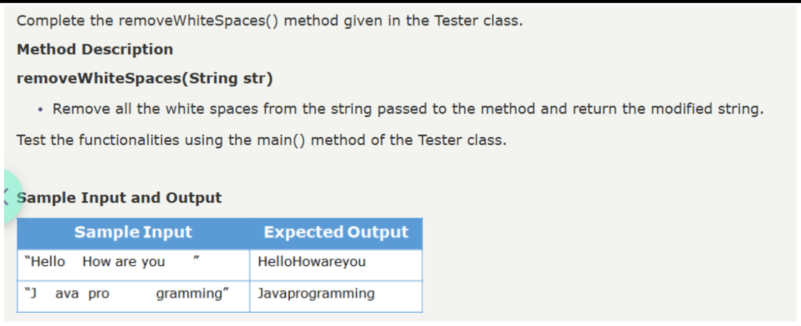
Instance variables	Values
movieId	114
noOfSeats	3

Output

```
Sorry! Please enter valid movie Id and number of seats
```

```
class MovieTicket {
  private int movieId;
  private int noOfSeats;
  private double costPerTicket;
  public MovieTicket(int movieId, int noOfSeats) {
    setMovieId(movieId);
    setNoOfSeats(noOfSeats);
    setCostPerTicket(movieId);
  }
  // Getter methods
  public int getMovieId() {
    return movieId;
  }
  public int getNoOfSeats() {
    return noOfSeats;
  }
  public double getCostPerTicket() {
    return costPerTicket;
  }
  // Setter methods
  public void setMovieId(int movieId){
    this.movieId = movieId;
  }
  public void setNoOfSeats(int noOfSeats) {
    this.noOfSeats = noOfSeats;
  }
  public void setCostPerTicket(double costPerTicket) {
        int mID = (int) this.movieId;
        switch (mID) {
            case 111:
                this.costPerTicket = 7.0;
                break;
            case 112:
                this.costPerTicket = 8.0;
                break;
            case 113:
                this.costPerTicket = 8.5;
                break;
            default:
                this.costPerTicket = 0.0;
                break;
        }
    }
    public double calculateTotalAmount() {
        double cpt = getCostPerTicket();
        int nos = getNoOfSeats();
        double price = cpt * nos;
        double totalAmount = price * 1.02;
        totalAmount = Math.round(totalAmount);
        return totalAmount;
    }
}
class Tester {
    public static void main(String[] args) {
        MovieTicket movieTicket = new MovieTicket(112, 3);
        double amount = movieTicket.calculateTotalAmount();
        if (amount == 0)
            System.out.println("Sorry! Please enter a valid movie Id and number of seats");
        else
            System.out.println("Total amount for booking: $" + amount);
    }
}
```

String - Exercise 1


Complete the removeWhiteSpaces() method given in the Tester class.

Method Description

removeWhiteSpaces(String str)

- Remove all the white spaces from the string passed to the method and return the modified string.

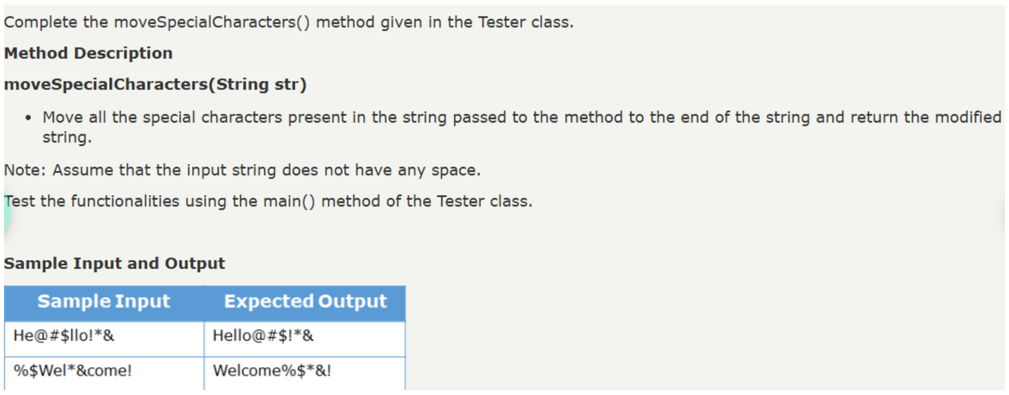
Test the functionalities using the main() method of the Tester class.

Sample Input and Output

Sample Input	Expected Output
"Hello How are you "	HelloHowareyou
"J ava pro gramming"	Javaprogramming

```
public class Tester {

    public static String removeWhiteSpaces(String str) {

        if (str == null) {

            return ""; // Handle null case safely

        }

        return str.replaceAll("\\s+", "");

    }


    public static void main(String[] args) {

        // Sample Test

        String input = "I am learning Java";

        String result = removeWhiteSpaces(input);

        System.out.println("String after removing white spaces: " + result);

    }

}
```

STRING ASSIGNMENT 1


Complete the moveSpecialCharacters() method given in the Tester class.

Method Description

moveSpecialCharacters(String str)

- Move all the special characters present in the string passed to the method to the end of the string and return the modified string.

Note: Assume that the input string does not have any space.

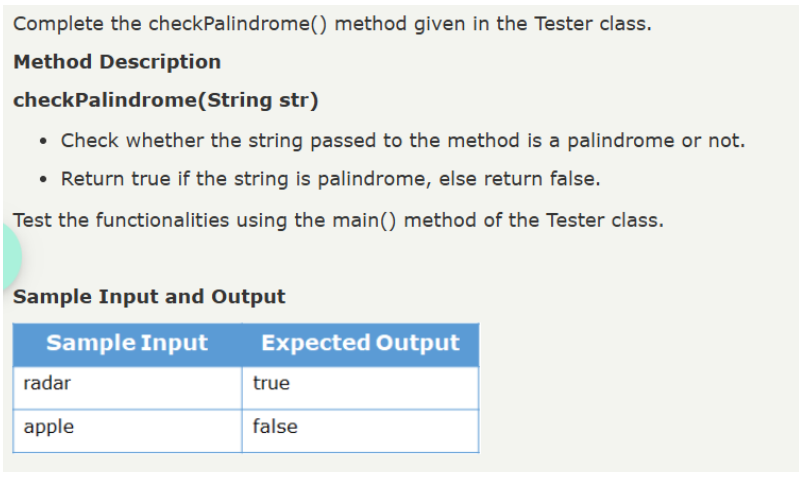
Test the functionalities using the main() method of the Tester class.

Sample Input and Output

Sample Input	Expected Output
He@#$llo!*&	Hello@#$!*&
%$Wel*&come!	Welcome%$*&!

```
class Tester {

    public static String moveSpecialCharacters(String str) {

        // Create two StringBuilder objects to separate special characters and other characters

        StringBuilder specialChars = new StringBuilder();

        StringBuilder otherChars = new StringBuilder();

        for (int i = 0; i < str.length(); i++) {

            char c = str.charAt(i);

            // special character

            if (!Character.isLetterOrDigit(c)) {

                specialChars.append(c);

            } else {
                otherChars.append(c);
            }
        }
        otherChars.append(specialChars);
        return otherChars.toString();
    }
    public static void main(String args[]) {
        String str = "He@#$llo!*&";
        System.out.println(moveSpecialCharacters(str));
    }
}
```

STRING ASSIGNMENT 2



Complete the checkPalindrome() method given in the Tester class.

Method Description

checkPalindrome(String str)

- Check whether the string passed to the method is a palindrome or not.
- Return true if the string is palindrome, else return false.

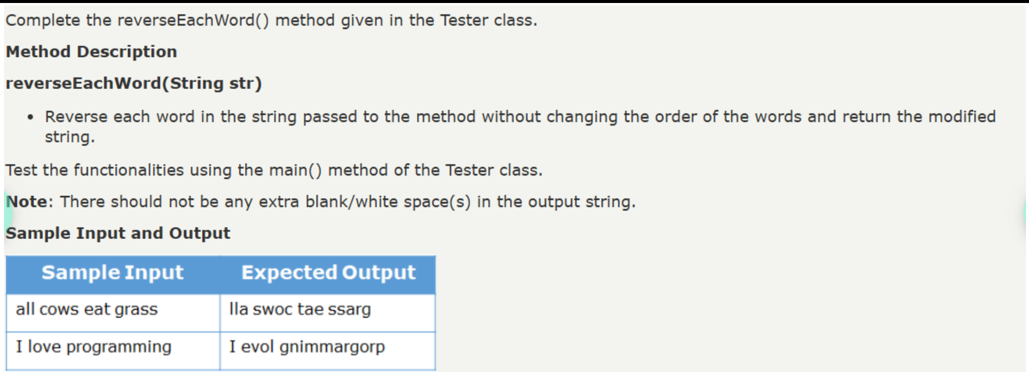
Test the functionalities using the main() method of the Tester class.

Sample Input and Output

Sample Input	Expected Output
radar	true
apple	false

```
class Tester {
    public static boolean checkPalindrome(String str) {
        StringBuilder reverse = new StringBuilder();
        for (int i = str.length() - 1; i >= 0; i--) {
            char c = str.charAt(i);
            reverse.append(c);
        }
        return str.equals(reverse.toString());
    }
    public static void main(String args[]) {
        String str = "radar";
        if (checkPalindrome(str))
            System.out.println("The string is a palindrome!");
        else
            System.out.println("The string is not a palindrome!");
    }
}
```

STRING ASSIGNMENT 3



Complete the reverseEachWord() method given in the Tester class.

Method Description

reverseEachWord(String str)

- Reverse each word in the string passed to the method without changing the order of the words and return the modified string.

Test the functionalities using the main() method of the Tester class.

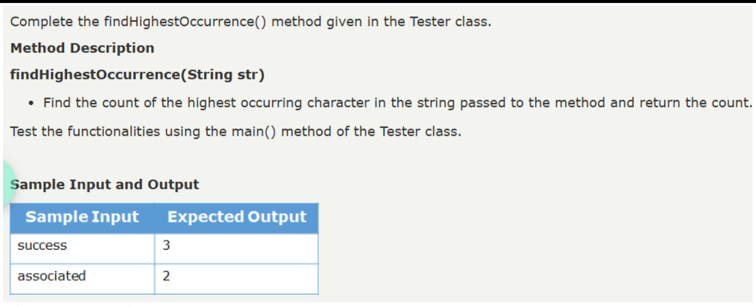
Note: There should not be any extra blank/white space(s) in the output string.

Sample Input and Output

Sample Input	Expected Output
all cows eat grass	lla swoc tae ssarg
I love programming	I evol gnimmargorp

```
class Tester {
  public static String reverseEachWord(String str) {
    String[] words = str.split(" ");
    StringBuilder reversedStr = new StringBuilder();
    for (int i = 0; i < words.length; i++) {
      String word = words[i];
      String reversedWord = reverseWord(word);
      reversedStr.append(reversedWord);
      if (i < words.length - 1) {
        reversedStr.append(" ");
      }
    }
    return reversedStr.toString();
  }
  private static String reverseWord(String word) {
    char[] chars = word.toCharArray();
    int start = 0;
    int end = chars.length - 1;
    //  reverse the word
    while (start < end) {
      char temp = chars[start];
      chars[start] = chars[end];
      chars[end] = temp;
      start++;
      end--;
    }
    return new String(chars);
  }
  public static void main(String args[]) {
    String str = "all cows eat grass";
    System.out.println(reverseEachWord(str));
  }
}
```

STRING ASSIGNMENT 4


Complete the findHighestOccurrence() method given in the Tester class.

Method Description

findHighestOccurrence(String str)

- Find the count of the highest occurring character in the string passed to the method and return the count.

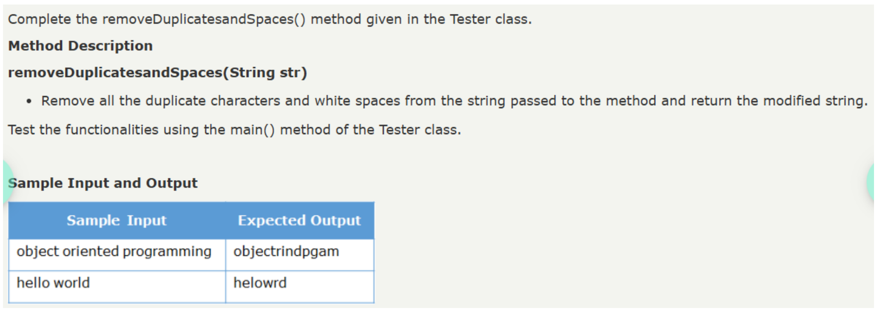
Test the functionalities using the main() method of the Tester class.

Sample Input and Output

Sample Input	Expected Output
success	3
associated	2

```
class Tester {
        public static int findHighestOccurrence(String str){
                //Implement your code here and change the return value accordingly
                Map<Character, Integer> occurenceMap = new HashMap<>();
                for(char ch : str.toCharArray()){
                  if(ch != ' '){
                    occurenceMap.put(ch, occurenceMap.getOrDefault(ch, 0) + 1);
                  }
                }
                char highestChar = '\0';
                int maxFrequency = 0;
                for(Map.Entry<Character, Integer> entry : occurenceMap.entrySet()){
                  if(entry.getValue() > maxFrequency){
                    highestChar = entry.getKey();
                    maxFrequency = entry.getValue();
                  }
                }
    return highestChar;
        }
        public static void main(String args[]){
          String str = "success";
          System.out.println(findHighestOccurrence(str));
        }
}
```

STRING ASSIGNMENT 5


Complete the removeDuplicatesandSpaces() method given in the Tester class.

Method Description

removeDuplicatesandSpaces(String str)

- Remove all the duplicate characters and white spaces from the string passed to the method and return the modified string.

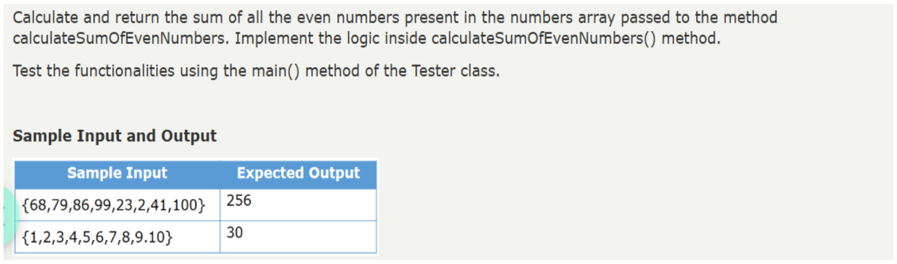
Test the functionalities using the main() method of the Tester class.

Sample Input and Output

Sample Input	Expected Output
object oriented programming	objectrindpgam
hello world	helowrd

```
class Tester{
  public static String removeDuplicatesandSpaces(String str){
    Set<Character> seen = new HashSet<>();
    StringBuilder result = new StringBuilder();
    for (char ch : str.toCharArray()) {
            if (ch != ' ' && !seen.contains(ch)) {
                seen.add(ch);
                result.append(ch);
            }
        }
        return result.toString();
            }
            public static void main(String args[]){
                String str = "object oriented programming";
                System.out.println(removeDuplicatesandSpaces(str));
            }
}
```

Array - Exercise 1


Calculate and return the sum of all the even numbers present in the numbers array passed to the method calculateSumOfEvenNumbers. Implement the logic inside calculateSumOfEvenNumbers() method.

Test the functionalities using the main() method of the Tester class.

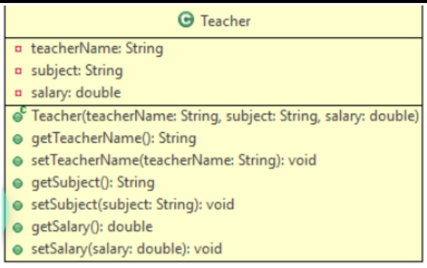
Sample Input and Output

Sample Input	Expected Output
{68,79,86,99,23,2,41,100}	256
{1,2,3,4,5,6,7,8,9.10}	30

```
public class Tester {
    public static int calculateSumOfEvenNumbers(int[] numbers) {
        int sum = 0;
        if (numbers == null) {
            return 0; // Handle null array safely
        }
        for (int num : numbers) {
            if (num % 2 == 0) { // check if number is even
                sum += num;
            }
        }
        return sum;
    }
    public static void main(String[] args) {
        // Sample test
        int[] numbers = {10, 15, 20, 25, 30};
        int result = calculateSumOfEvenNumbers(numbers);
        System.out.println("Sum of even numbers: " + result);
    }
}
```

Array - Assignment 1

Implement the class Teacher based on the class diagram and description given below.



```
Teacher
teacherName: String
subject: String
salary: double
Teacher(teacherName: String, subject: String, salary: double)
getTeacherName(): String
setTeacherName(teacherName: String): void
getSubject(): String
setSubject(subject: String): void
getSalary(): double
setSalary(salary: double): void
```

Method Description

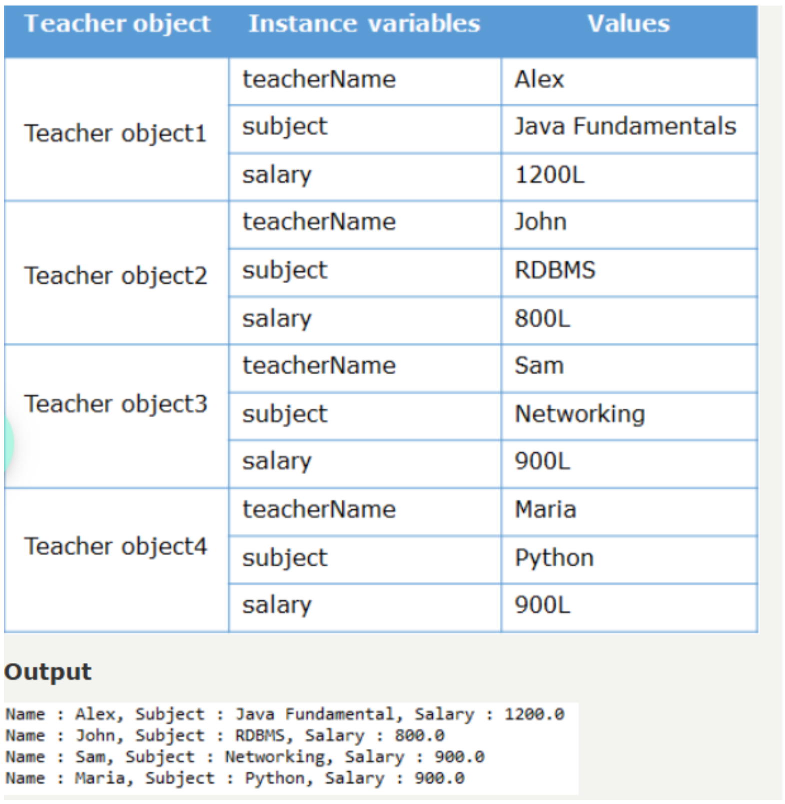
Teacher(String teacherName, String subject, double salary)

Initialize the values of all the instance variables appropriately with the values passed

Create a Tester class. Create 4 objects of Teacher class. Create an array of type Teacher store the created objects and display the details of the teachers.

Sample Input and Output

Input



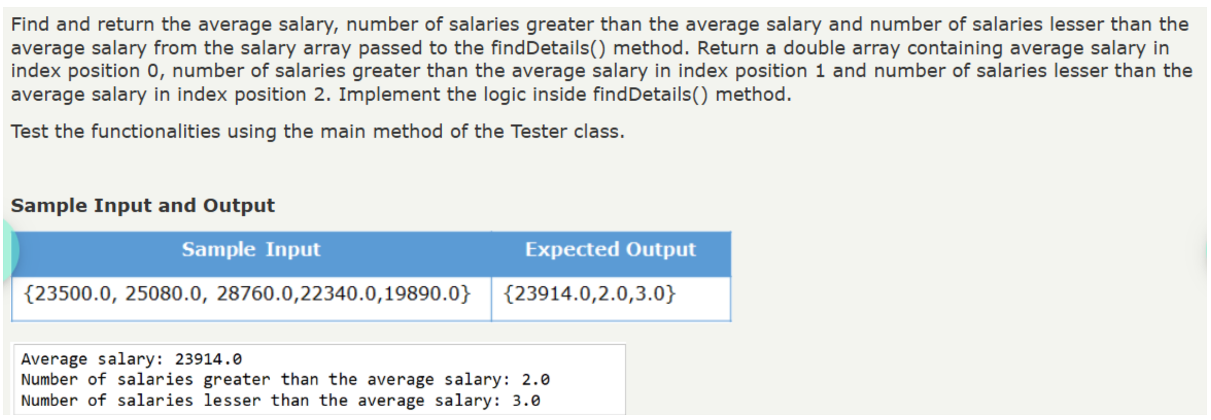
Teacher object	Instance variables	Values
Teacher object1	teacherName	Alex
	subject	Java Fundamentals
	salary	1200L
Teacher object2	teacherName	John
	subject	RDBMS
	salary	800L
Teacher object3	teacherName	Sam
	subject	Networking
	salary	900L
Teacher object4	teacherName	Maria
	subject	Python
	salary	900L

Output

```
Name : Alex, Subject : Java Fundamental, Salary : 1200.0
Name : John, Subject : RDBMS, Salary : 800.0
Name : Sam, Subject : Networking, Salary : 900.0
Name : Maria, Subject : Python, Salary : 900.0
```

```
class Teacher {
    private String teacherName;
    private String subject;
    private double salary;
    public Teacher(String teacherName, String subject, double salary) {
        this.teacherName = teacherName;
        this.subject = subject;
        this.salary = salary;
    }
    public String getTeacherName() {
        return teacherName;
    }
    public void setTeacherName(String teacherName) {
        this.teacherName = teacherName;
    }
    public String getSubject() {
        return subject;
    }
    public void setSubject(String subject) {
        this.subject = subject;
    }
    public double getSalary() {
        return salary;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }
    public void displayDetails() {
        System.out.println("Name: " + teacherName + ", Subject: " + subject + ", Salary: " + salary);
    }
}
class Tester {
    public static void main(String[] args) {
        Teacher teacher1 = new Teacher("Alex", "Java Fundamentals", 1200);
        Teacher teacher2 = new Teacher("John", "RDBMS", 800);
        Teacher teacher3 = new Teacher("Sam", "Networking", 900);
        Teacher teacher4 = new Teacher("Maria", "Python", 900);
        Teacher[] teachers = {teacher1, teacher2, teacher3, teacher4};
        for (Teacher teacher : teachers) {
            teacher.displayDetails();
        }
    }
}
```

Array - Assignment 2



Find and return the average salary, number of salaries greater than the average salary and number of salaries lesser than the average salary from the salary array passed to the findDetails() method. Return a double array containing average salary in index position 0, number of salaries greater than the average salary in index position 1 and number of salaries lesser than the average salary in index position 2. Implement the logic inside findDetails() method.

Test the functionalities using the main method of the Tester class.

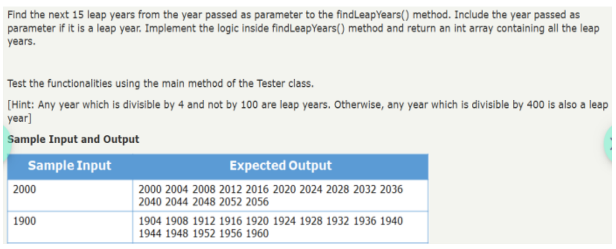
Sample Input and Output

Sample Input	Expected Output
{23500.0, 25080.0, 28760.0,22340.0,19890.0}	{23914.0,2.0,3.0}

```
Average salary: 23914.0
Number of salaries greater than the average salary: 2.0
Number of salaries lesser than the average salary: 3.0
```

```
class Tester {
    public static double[] findDetails(double[] salary) {
        //Implement your code here and change the return value accordingly
        double aboveAverage = 0;
        double belowAverage = 0;
        double salarySum = Arrays.stream(salary).sum();
        double average = salarySum / salary.length();
        for(double entry : salary){
            if(entry > average) aboveAverage++;
            if(entry < average) belowAverage++;
        }
        return new double[]{average, aboveAverage, belowAverage};
    }
    public static void main(String[] args) {
        double[] salary = { 23500.0, 25080.0, 28760.0, 22340.0, 19890.0 };
        double[] details = findDetails(salary);
        System.out.println("Average salary: "+ details[0]);
        System.out.println("Number of salaries greater than the average salary: "+ details[1]);
        System.out.println("Number of salaries lesser than the average salary: "+ details[2]); }}
```

Array - Assignment 3


Find the next 15 leap years from the year passed as parameter to the findLeapYears() method. Include the year passed as parameter if it is a leap year. Implement the logic inside findLeapYears() method and return an int array containing all the leap years.

Test the functionalities using the main method of the Tester class.

[Hint: Any year which is divisible by 4 and not by 100 are leap years. Otherwise, any year which is divisible by 400 is also a leap year]

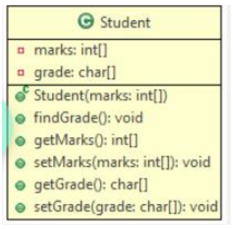
Sample Input and Output

Sample Input	Expected Output
2000	2000 2004 2008 2012 2016 2020 2024 2028 2032 2036 2040 2044 2048 2052 2056
1900	1904 1908 1912 1916 1920 1924 1928 1932 1936 1940 1944 1948 1952 1956 1960

```
class Tester {
   public static int[] findLeapYears(int year){
       int[] leapYears = new int[15];
       int index = 0;
       while (index < 15) {
           if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) {
               // If it's a leap year, add it to the array
               leapYears[index] = year;
               index++;
           }
           year++;
       }
       return leapYears;
   }
   public static void main(String[] args) {
       int year = 2001;
       int[] leapYears = findLeapYears(year);
       // Print the leap years
       for (int leapYear : leapYears) {
        System.out.println(leapYear);
      }
    }
  }
```

Array - Assignment 4

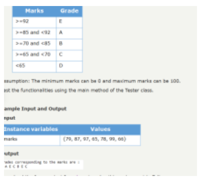
The results of a mid-term assessment have been declared. The marks scored by a student in different subjects are stored in an array.You need to find the corresponding grade for each subject by implementing the class Student based on the class diagram and description given below.



```
Student
marks: int[]
grade: char[]
Student(marks: int[])
findGrade(): void
getMarks(): int[]
setMarks(marks: int[]): void
getGrade(): char[]
setGrade(grade: char[]): void
```

Method description

Student(int[] marks).Initialize the value of the instance variable marks with the value passed to the constructor.Initialize the grade with the same size as marks array.findGrade(),Find the corresponding grade for each marks present in the marks array based on the below details and update the grade array.



Marks	Grade
>=92	E
>=85 and <92	A
>=70 and <85	B
>=65 and <70	C
<65	D

ssumption: The minimum marks can be 0 and maximum marks can be 100.

est the functionalities using the main method of the Tester class.

ample Input and Output

nput

Instance variables	Values
marks	{79, 87, 97, 65, 78, 99, 66}

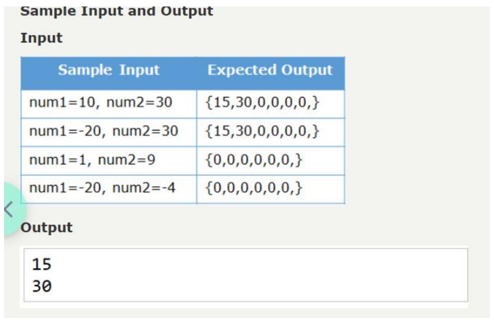
utput

```
rades corresponding to the marks are :
A E C B E C
```

```
class Student {
    private int[] marks;
    private char[] grade;
    public Student(int[] marks) {
        setMarks(marks);
        grade = new char[marks.length];
    }
    public void setMarks(int[] marks) {
        this.marks = marks;
    }
    public void findGrade() {
        char[] grade = new char[marks.length];
        for (int i = 0; i < marks.length; i++) {
            if (marks[i] >= 92) grade[i] = 'E';
            else if (marks[i] >= 85) grade[i] = 'A';
            else if (marks[i] >= 70) grade[i] = 'B';
            else if (marks[i] >= 65) grade[i] = 'C';
            else tempGrades[i] = 'D';
        }
        setGrade(grades);
    }
    public void setGrade(char[] grade) {
        this.grade = grade;
    }
    public int[] getMarks() {
        return marks;
    }
    public char[] getGrade() {
        return grade;
    }
}
class Tester {
    public static void main(String[] args) {
        int[] marks = {79, 87, 97, 65, 78, 99, 66};
        Student student = new Student(marks);
        student.findGrade();
        System.out.println("Grades corresponding to the marks are : ");
        char[] grades = student.getGrade();
        for (int index = 0; index < grades.length; index++) {
            System.out.print(grades[index] + " ");
        }
    }
}
```

Array - Assignment 5

Implement the method findNumbers accepting two numbers num1 and num2 based on the conditions given below:Validate that num1 should be less than num2.If the validations are successful.populate all the 2 digit positive numbers between num1 and num2 into the provided numbers array if.sum of the digits of the number is a multiple of 3.number is a multiple of 5,Return the numbers array.Test the functionalities using the main method of the Tester class.



Sample Input and Output

Input

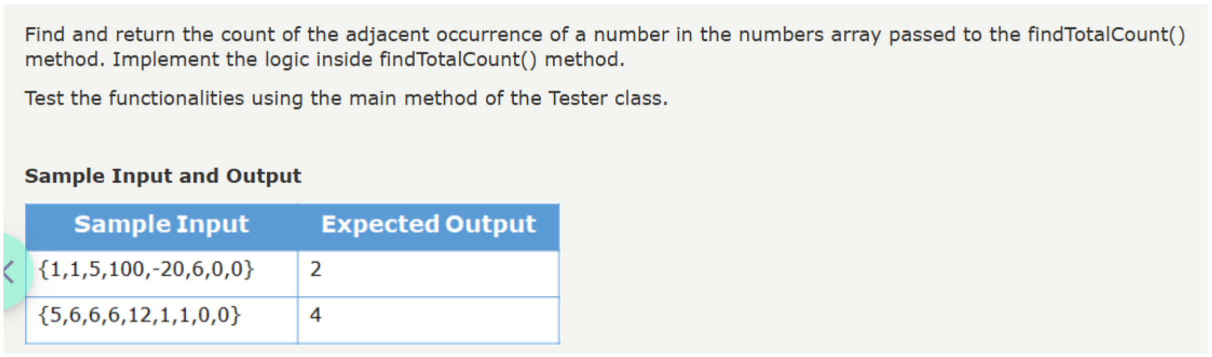
Sample Input	Expected Output
num1=10, num2=30	{15,30,0,0,0,0,}
num1=-20, num2=30	{15,30,0,0,0,0,}
num1=1, num2=9	{0,0,0,0,0,0,}
num1=-20, num2=-4	{0,0,0,0,0,0,}

Output

```
15
30
```

```
class Tester {
    public static int[] findNumbers(int num1, int num2) {
        int[] numbers = new int[6];
        if (num1 >= num2) {
            return numbers;
        }
        int index = 0;
        for (int i = num1; i <= num2; i++) {
            if (i > 9 && i < 100) {
                int sumOfDigits = (i / 10) + (i % 10);
                if (sumOfDigits % 3 == 0 && i % 5 == 0) {
                    if (index < 6) {
                        numbers[index] = i;
                        index++;
                    }
                }
            }
        }
        return numbers;
    }

    public static void main(String[] args) {
        int num1 = 10;
        int num2 = 30;
        int[] numbers = findNumbers(num1, num2);
        if (numbers[0] == 0) {
            System.out.println("There is no such number!");
        } else {
            for (int index = 0; index < numbers.length; index++) {
                if (numbers[index] == 0) {
                    break;
                }
                System.out.println(numbers[index]);
            }
        }
    }
}
```

Array - Assignment 6



Find and return the count of the adjacent occurrence of a number in the numbers array passed to the findTotalCount() method. Implement the logic inside findTotalCount() method.

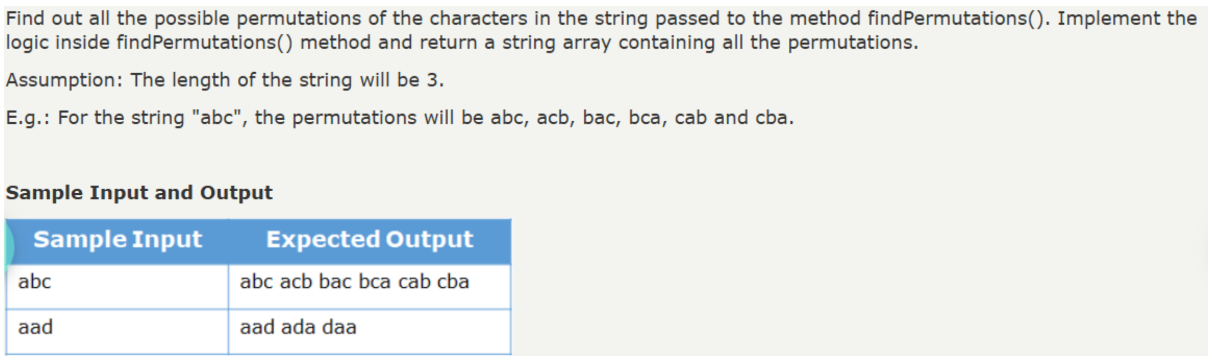
Test the functionalities using the main method of the Tester class.

Sample Input and Output

Sample Input	Expected Output
{1,1,5,100,-20,6,0,0}	2
{5,6,6,6,12,1,1,0,0}	4

```
class Tester {
    public static int findTotalCount(int[] numbers) {
        int count = 0;
        for (int i = 0; i < numbers.length - 1; i++) {
            if (numbers[i] == numbers[i + 1]) {
                count++;
            }
        }
        return count;
    }
    public static void main(String[] args) {
        int[] numbers1 = {1, 1, 5, 100, -20, 6, 0, 0};
        int[] numbers2 = {5, 6, 6, 6, 12, 1, 1, 0, 0};

        System.out.println("Count of adjacent occurrence in first array: " + findTotalCount(numbers1));
        System.out.println("Count of adjacent occurrence in second array: " + findTotalCount(numbers2));
    }
}
```

Array - Assignment 7



Find out all the possible permutations of the characters in the string passed to the method findPermutations(). Implement the logic inside findPermutations() method and return a string array containing all the permutations.

Assumption: The length of the string will be 3.

E.g.: For the string "abc", the permutations will be abc, acb, bac, bca, cab and cba.

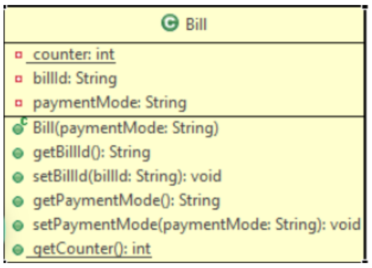
Sample Input and Output

Sample Input	Expected Output
abc	abc acb bac bca cab cba
aad	aad ada daa

```
class Tester{
    public static String[] findPermutations(String str){
        //Implement your code here and change the return value accordingly
        String[] permutations = new String[6];
        int index=0;
        for(int i=0; i<3; i++){
            for(int j=0; j<3; j++){
                if(i==j) continue;
                for(int k=0; k<3; k++){
                    if(i==k || j==k) continue;
                    String prem = "" + str.charAt(i) + str.charAt(j) + str.charAt(k);
                    if(!contains(permutations, prem)){
                        permutations[index++] = prem;
                    }
                }
            }
        }
        return permutations;
    }
    private static boolean contains(String arr[], String value){
        for(String s:arr){
            if(s != null && s.equals(value)){
                return true;
            }
        }
        return false;
    }
    public static void main(String args[]){
        String str = "abc";
        String permutations[] = findPermutations(str);
        for(String permutation: permutations){
            if (permutation!=null)
                System.out.println(permutation);
        }
    }
}
```

Static Exercise:

Implement the class Bill based on the class diagram and description given below.



Method Description

Bill(String paymentMode)

- **Initialize the paymentMode instance variable with the value passed to the parameter.**
- **Generate the billId using counter. The value of billId should start from 'B9001' and the numerical part should be incremented by 1 for the subsequent values. Initialize the counter in static block.**

Implement the appropriate getter and setter methods.

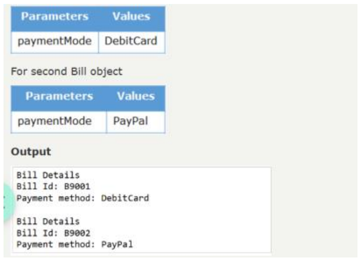
Test the functionalities using the provided Tester class. Create two or more Bill objects and validate that the billId is being generated properly.

Sample Input and Output

For constructor

Input

For first Bill object



Parameters	Values
paymentMode	DebitCard

For second Bill object

Parameters	Values
paymentMode	PayPal

Output

```
Bill Details
Bill Id: B9001
Payment method: DebitCard

Bill Details
Bill Id: B9002
Payment method: PayPal
```

```
class Bill{
    //Implement your code here
    private  static int counter;
    private String billId;
    private String paymentMode;
    public Bill(String paymentMode){
        this.paymentMode = paymentMode;
        String bId = 'B' + Integer.toString(Bill.counter);
        counter++;
        setBillId(bId);
    }
    static{
        counter = 9001;
    }
    public static int getCounter(){
        return Bill.counter;
    }
    public String getBillId(){
        return this.billId;
    }
    public void setBillId(String billId){
        this.billId = billId;
    }
    public String getPaymentMode(){
        return this.paymentMode;
    }
    public void setPaymentMode(String paymentMode){
        this.paymentMode = paymentMode;
    }
}
class Tester {
    public static void main(String[] args) {
        Bill bill1 = new Bill("DebitCard");
        Bill bill2 = new Bill("PayPal");
        //Create more objects and add them to the bills array for testing your code
        Bill[] bills = { bill1, bill2 }
        for (Bill bill : bills) {
            System.out.println("Bill Details");
            System.out.println("Bill Id: " + bill.getBillId());
            System.out.println("Payment method: " + bill.getPaymentMode());
            System.out.println();
        }
    }
```

}

Static - Assignment 2

A dance club is conducting auditions to admit interested candidates. You need to implement a Participant class for the dance club based on the class diagram and description given below.Method Description:Partcipant(String name, long contactNumber, String city).Initialize the name, contactNumber and city instance variables appropriately with the values passed to the constructor.Generate the registrationId using the static variable counter. The value of registrationId should start from 'D10001' and the numerical part should be incremented by 1 for the subsequent values. Initialize the counter in static block.Implement the appropriate getter and setter methods.Test the functionalities using the provided Tester class. Create two or more Participant objects and validate that the values of the member variables are proper.

```
class Participant {

    // Implement your code here

    private static int counter;

    private String registrationId;

    private String name;

    private long contactNumber;

    private String city;

    static {

        Participant.counter = 10000;

    }

    // Constructor

    public Participant(String name, long contactNumber, String city) {

        this.name = name;

        this.contactNumber = contactNumber;

        this.city = city;

        this.registrationId = "D" + (counter + 1);

        counter++;

    }

    // Registration ID

    public String getRegistrationId() {

        return this.registrationId;

    }

    // Counter

    public static void setCounter(int counter) {

        Participant.counter = counter;

    }

    public static int getCounter() {

        return Participant.counter;

    }

    // Name

    public void setName(String name) {

        this.name = name;

    }

    public String getName() {

        return this.name;

    }
    // Contact Number
    public void setContactNumber(long contactNumber) {
        this.contactNumber = contactNumber;
    }
    public long getContactNumber() {
        return this.contactNumber;
    }
    // City
    public void setCity(String city) {
        this.city = city;
    }
    public String getCity() {
        return this.city;
    }
}
class Tester {
    public static void main(String[] args) {
        Participant participant1 = new Participant("Franklin", 7656784323L, "Texas");
        Participant participant2 = new Participant("Merina", 7890423112L, "New York");
        // Create more objects and add them to the participants array for testing your code
        Participant participant3 = new Participant("John", 9876543210L, "Los Angeles");
        Participant[] participants = { participant1, participant2, participant3 };
        for (Participant participant : participants) {
            System.out.println("Hi " + participant.getName() + "! Your registration id is " + participant.getRegistrationId());
        }
    }
}


    }
    // constructor
    public Participant(String name,long contactNumber,String city){
        setName(name);
        setContactNumber(contactNumber);
        setCity(city);
        this.registrationId = 'D' + Integer.toStringcounter;
        counter++;
    }
    // Registration
    public String getRegistrationId(){
        return this.registrationId;
    }
    // Counter
    public static void setCounter(int counter){
        this.counter = counter;
    }
    public static int getCounter(){
        return this.counter;
    }
    // Name
    public void setName(String name){
        this.name = name;
    }
    public String getName(){
        return this.name;
    }
    // Contact Number
    public void setContactNumber(long contactNumber){
        this.contactNumber = contactNumber;
    }
    public long getContactNumber(){
        return this.contactNumber;
    }
    // city
    public void setCity(String city){
        return this.city;
    }
    public String getCity(){
        return this.city;
    }
}
class Tester {
        public static void main(String[] args) {
                Participant participant1 = new Participant("Franklin", 7656784323L, "Texas");
                Participant participant2 = new Participant("Merina", 7890423112L, "New York");
                //Create more objects and add them to the participants array for testing your code
                Participant[] participants = { participant1, participant2 };
                for (Participant participant : participants) {
                        System.out.println("Hi "+participant.getName()+"! Your registration id is "+participant.getRegistrationId());
                }
        }
}
```

STATIC ASSIGNMENT 3

Rainbow Cinemas is an upcoming multiplex in the city with a seating capacity of 400 people. They need an application to be developed for booking of tickets.You need to implement a Booking class based on the class diagram and description given below.Method Description:Booking(String customerEmail, int seatsRequired).Initialize the customerEmail and seatsRequired instance

variable appropriately with the values passed to the constructor..If the required number of seats are available, set the value of isBooked to true and update the value of seatsAvailable accordingly. The total number of seats available is 400 and should be initialized in static block..If the required number of seats are not available, set the value of isBooked to false.Implement the appropriate getter and setter methods.Test the functionalities using the provided Tester class. Create two or more Booking objects and validate that the values of the member variables are proper.

```
class Booking {

    // Implement your code here

    private String customerEmail;

    private int seatsRequired;

    private boolean isBooked;

    private static int seatsAvailable;

    static {

        seatsAvailable = 400;

    }

    public Booking(String customerEmail, int seatsRequired) {

        this.customerEmail = customerEmail;

        this.seatsRequired = seatsRequired;

        if (seatsRequired <= seatsAvailable) {

            this.isBooked = true;

            seatsAvailable -= seatsRequired;

        } else {

            this.isBooked = false;

        }

    }

    // Customer Email

    public void setCustomerEmail(String customerEmail) {

        this.customerEmail = customerEmail;

    }

    public String getCustomerEmail() {

        return customerEmail;

    }


    // Seats Required

    public void setSeatsRequired(int seatsRequired) {

        this.seatsRequired = seatsRequired;

    }

    public int getSeatsRequired() {

        return seatsRequired;

    }

    // Is Booked

    public void setBooked(boolean isBooked) {

        this.isBooked = isBooked;

    }

    public boolean isBooked() {
        return isBooked;
    }
    // Seats Available
    public static void setSeatsAvailable(int seatsAvailable) {
        Booking.seatsAvailable = seatsAvailable;
    }
    public static int getSeatsAvailable() {
        return seatsAvailable;
    }
}
class Tester {
    public static void main(String[] args) {
        Booking booking1 = new Booking("jack@email.com", 100);
        Booking booking2 = new Booking("jill@email.com", 350);

        // Create more objects and add them to the bookings array for testing your code
        Booking booking3 = new Booking("john@email.com", 50);
        Booking[] bookings = { booking1, booking2, booking3 };
        for (Booking booking : bookings) {
            if (booking.isBooked()) {
                System.out.println(booking.getSeatsRequired() + " seats successfully booked for " + booking.getCustomerEmail());
            } else {
                System.out.println("Sorry " + booking.getCustomerEmail() + ", required number of seats are not available!");
                System.out.println("Seats available: " + Booking.getSeatsAvailable());
            }
        }
    }
}
```

AGGREGATION EXERCISE 1

```
class Address {
private String doorNo;
private String street;
private String city;
private int zipcode;
public Address(String doorNo, String street, String city, int zipcode) {
this.doorNo = doorNo;
this.street = street;
this.city = city;
this.zipcode = zipcode;
}
public String getDoorNo() {
return doorNo;
}
public void setDoorNo(String doorNo) {
this.doorNo = doorNo;
}
public String getStreet() {
return street;
}
public void setStreet(String street) {
```

```
this.street = street;
}
public String getCity() {
return city;
}
public void setCity(String city) {
this.city = city;
}
public int getZipcode() {
return zipcode;
}
public void setZipcode(int zipcode) {
this.zipcode = zipcode;
}
}
class Customer {
private String customerId;
private String customerName;
private long contactNumber;
private Address address;
public Customer(String customerId, String customerName, long contactNumber, Address address) {
this.customerId = customerId;
this.customerName = customerName;
this.contactNumber = contactNumber;
this.address = address;
}
public Customer(String customerName, long contactNumber, Address address) {
this.customerName = customerName;
this.contactNumber = contactNumber;
this.address = address;
}
public String getCustomerId() {
return customerId;
}
public void setCustomerId(String customerId) {
this.customerId = customerId;
}
public String getCustomerName() {
return customerName;
}
public void setCustomerName(String customerName) {
this.customerName = customerName;
}
public long getContactNumber() {
return contactNumber;
}
```

AGGREGATION ASSIGNMENT 1

You have already created the Customer and Order classes in the SwiftFood project. Modify the Customer and Order classes based on the class diagram given below.

```
// Food class
class Food2 {
private String name;
private double price;
public Food2(String name, double price) {
this.name = name;
this.price = price;
}
public String getName() { return name; }
public double getPrice() { return price; }
@Override
public String toString() {
return name + " (₹" + price + ")";
}
}
// Customer class
class Customer2 {
private String customerId;
private String customerName;
private long contactNumber;
private Address2 address;
```

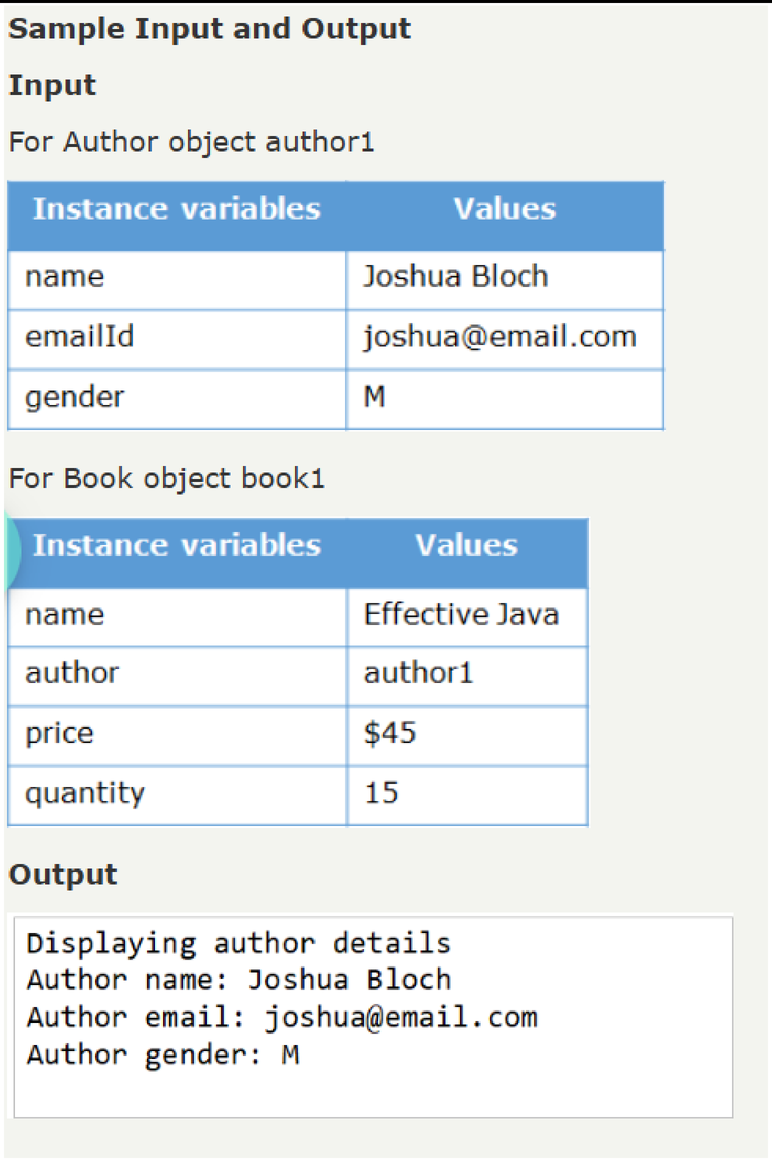
```
public Customer2() {
this.customerId = "C000";
this.customerName = "Default Name";
}
public Customer2(String customerId, String customerName, long contactNumber, Address2 address) {
this.customerId = customerId;
this.customerName = customerName;
this.contactNumber = contactNumber;
this.address = address;
}
public Customer2(String customerName, long contactNumber, Address2 address) {
this.customerId = "C000";
this.customerName = customerName;
this.contactNumber = contactNumber;
this.address = address;
}
public String getCustomerId() { return customerId; }
public String getCustomerName() { return customerName; }
public long getContactNumber() { return contactNumber; }
public Address2 getAddress() { return address; }
public void setCustomerId(String customerId) { this.customerId = customerId; }
public void setCustomerName(String customerName) { this.customerName = customerName; }
public void setContactNumber(long contactNumber) { this.contactNumber = contactNumber; }
public void setAddress(Address2 address) { this.address = address; }
public void displayCustomerDetails() {
System.out.println("*** Customer Details ***");
System.out.println("Customer ID: " + getCustomerId());
System.out.println("Customer Name: " + getCustomerName());
```

Aggregation - Assignment 2:

Implement the Book and Author classes based on the class diagram and description given below.

Author.Implement the getter and setter methods appropriately.

Book:Method Description:displayAuthorDetails().Display the details of the author for the book..Implement the getter and setter methods appropriately..Test your code using the Tester class.



Sample Input and Output

Input

For Author object author1

Instance variables	Values
name	Joshua Bloch
emailId	joshua@email.com
gender	M

For Book object book1

Instance variables	Values
name	Effective Java
author	author1
price	$45
quantity	15

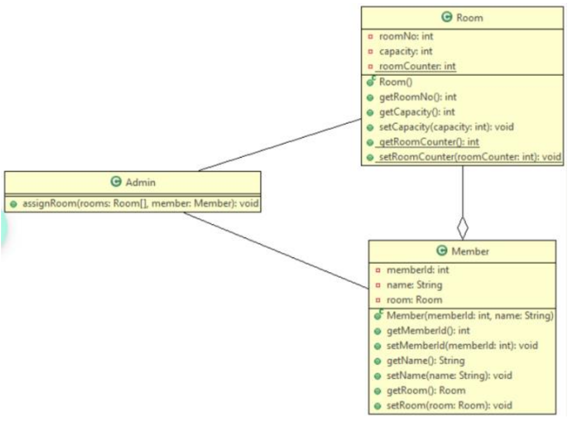
Output

```
Displaying author details
Author name: Joshua Bloch
Author email: joshua@email.com
Author gender: M
```

```
// Author class
class Author {
    private String name;
    private String email;
    private char gender;
    // Constructor
    public Author(String name, String email, char gender) {
        this.name = name;
        this.email = email;
        this.gender = gender;
    }
    // Getter and Setter methods
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public char getGender() {
        return gender;
    }
    public void setGender(char gender) {
        this.gender = gender;
    }
}
// Book class
class Book {
    private String title;
    private double price;
    private Author author;
    // Constructor
    public Book(String title, double price, Author author) {
        this.title = title;
        this.price = price;
        this.author = author;
    }
    // Method to display author details
    public void displayAuthorDetails() {
        System.out.println("Author Name: " + author.getName());
        System.out.println("Author Email: " + author.getEmail());
        System.out.println("Author Gender: " + author.getGender());
    }
    // Getter and Setter methods
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {
        this.price = price;
    }
    public Author getAuthor() {
        return author;
    }
    public void setAuthor(Author author) {
        this.author = author;
    }
}
// Tester class
class Tester {
    public static void main(String[] args) {
        Author author1 = new Author("J.K. Rowling", "jkrowling@example.com", 'F');
        Book book1 = new Book("Harry Potter", 500.0, author1);
        book1.displayAuthorDetails();
    }
}
```

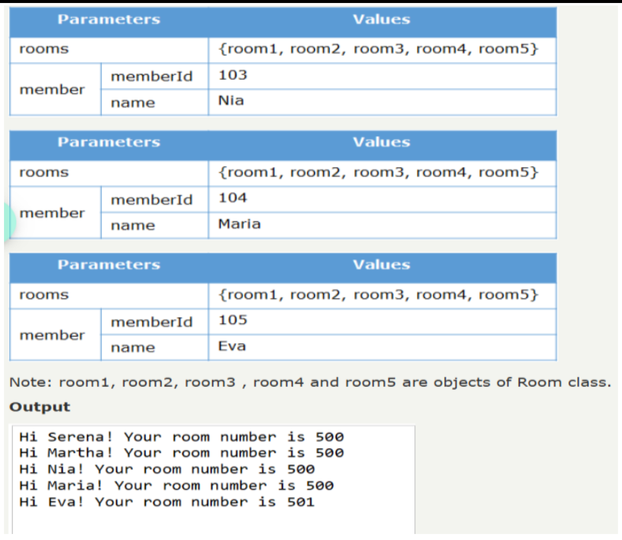
Aggregation - Assignment 3

Jasmine Girls Hostel wants you to create a small application for them to assign rooms to members. You need to implement the classes based on the class diagram and description given below.


Room
roomNo: int
capacity: int
roomCounter: int
Room()
getRoomNo(): int
getCapacity(): int
setCapacity(capacity: int): void
getRoomCounter(): int
setRoomCounter(roomCounter: int): void
Admin
assignRoom(rooms: Room[], member: Member): void
Member
memberId: int
name: String
room: Room
Member(memberId: int, name: String)
getMemberId(): int
setMemberId(memberId: int): void
getName(): String
setName(name: String): void
getRoom(): Room
setRoom(room: Room): void

Member

Member(int memberId, String name).Initialize the memberId and name instance variables appropriately with the values passed to the constructor..Implement the appropriate getter and setter methods.Room:Room()Generate the roomNo using the static variable roomCounter. The value of roomNo should start from 500 and should be incremented by 1 for the subsequent values. Initialize the roomNoCounter in static block.Initialize the capacity instance variable to 4.Implement the appropriate getter and setter methods. Admin :assignRoom(Room[] rooms, Member member).Assign a room to the member using the below conditions:One room can accommodate 4 members. Allocate the first room that is empty.Once a room is fully occupied, only then a new room should be assigned.Update the capacity of the allocated room accordingly.Test the functionalities using the provided Tester class.



Parameters		Values
rooms		{room1, room2, room3, room4, room5}
member	memberId	103
	name	Nia

Parameters		Values
rooms		{room1, room2, room3, room4, room5}
member	memberId	104
	name	Maria

Parameters		Values
rooms		{room1, room2, room3, room4, room5}
member	memberId	105
	name	Eva

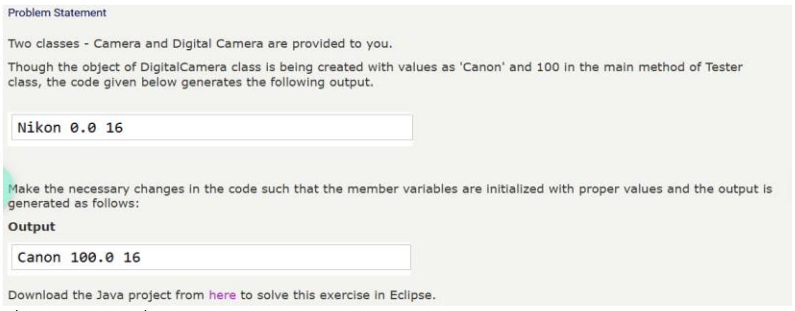
Note: room1, room2, room3 , room4 and room5 are objects of Room class.

Output

```
Hi Serena! Your room number is 500
Hi Martha! Your room number is 500
Hi Nia! Your room number is 500
Hi Maria! Your room number is 500
Hi Eva! Your room number is 501
```

```
class Room {
    private static int roomCounter = 500; // start from 500
    private int roomNo;
    private int capacity;
    public Room() {
        this.roomNo = roomCounter++;
        this.capacity = 4;
    }
    public int getRoomNo() { return roomNo; }
    public int getCapacity() { return capacity; }
    public void setCapacity(int capacity) { this.capacity = capacity; }
    // Uncomment this before verifying
    public String toString() {
        return "Room\nroomNo: "+this.roomNo+"\ncapacity: "+this.capacity;
    }
}
class Member {
    private int memberId;
    private String name;
    private Room room; // assigned room
    public Member(int memberId, String name) {
        this.memberId = memberId;
        this.name = name;
        this.room = null;
    }
    public int getMemberId() { return memberId; }
    public String getName() { return name; }
    public void setMemberId(int memberId) { this.memberId = memberId; }
    public void setName(String name) { this.name = name; }
    public Room getRoom() { return room; }

    public void setRoom(Room room) { this.room = room; }

    // Uncomment this before verifying

    public String toString() {

        return "Member\nmemberId: "+this.memberId+"\nname: "+this.name;

    }

}

class Admin {

    public void assignRoom(Room[] rooms, Member member) {

        for (Room room : rooms) {

            if (room.getCapacity() > 0) {

                member.setRoom(room);

                room.setCapacity(room.getCapacity() - 1);

                return; // assigned, stop searching

            }

        }member.setRoom(null); // no room available  }}
```

INHERITANCE EXERCISE 1



Problem Statement

Two classes - Camera and Digital Camera are provided to you.

Though the object of DigitalCamera class is being created with values as 'Canon' and 100 in the main method of Tester class, the code given below generates the following output.

```
Nikon 0.0 16
```

Make the necessary changes in the code such that the member variables are initialized with proper values and the output is generated as follows:

Output

```
Canon 100.0 16
```

Download the Java project from here to solve this exercise in Eclipse.

```
class Camera {

        private String brand;

        private double cost;

        public Camera() {

                this.brand = "Nikon";

        }

    public Camera(String brand, double cost){

        this.brand = brand;

        this.cost = cost;

    }

        public String getBrand() {

                return brand;

        }

        public void setBrand(String brand) {

                this.brand = brand;

        }

        public double getCost() {

                return cost;

        }
        public void setCost(double cost) {
                this.cost = cost;
        }
}
class DigitalCamera extends Camera {
        private int memory;
        public DigitalCamera(String brand, double cost) {
          super(brand, cost);
                this.memory = 16;
        }
  public int getMemory() {
                return memory;
        }
        public void setMemory(int memory) {
                this.memory = memory;
        }
}
class Tester {
        public static void main(String[] args) {
    DigitalCamera camera = new DigitalCamera("Canon",100);
    System.out.println(camera.getBrand()+" "+camera.getCost()+" "+camera.getMemory());
  }
}
```

Inheritance - Assignment 1

```
class Employee {
  private int employeeId;
  private String employeeName;
  private double salary;
  public Employee(int employeeId, String employeeName) {
    this.employeeId = employeeId;
    this.employeeName = employeeName;
    this.salary = 0;
  }
  // Getters
  public int getEmployeeId() { return employeeId; }
  public String getEmployeeName() { return employeeName; }
  public double getSalary() { return salary; }
  // Setters
  public void setEmployeeId(int employeeId) { this.employeeId = employeeId; }
  public void setEmployeeName(String employeeName) { this.employeeName = employeeName; }
  public void setSalary(double salary) { this.salary = salary; }
  // Uncomment before verifying
```

```
    public String toString(){
        return "Employee\nemployeeId: "+this.getEmployeeId()+"\nemployeeName: "+this.getEmployeeName()+"\nsalary: "+this.getSalary();
    }
}
class PermanentEmployee extends Employee {
    private double basicPay;
    private double hra;
    private float experience;
    public PermanentEmployee(int empId, String name, double basicPay, double hra, float experience) {
        super(empId, name);
        this.basicPay = basicPay;
        this.hra = hra;
        this.experience = experience;
    }
    public double getBasicPay() { return basicPay; }
    public double getHra() { return hra; }
    public float getExperience() { return experience; }
    public void setBasicPay(double basicPay) { this.basicPay = basicPay; }
    public void setHra(double hra) { this.hra = hra; }
    public void setExperience(float experience) { this.experience = experience; }
    public void calculateMonthlySalary() {
        double variableComponent = 0;
        if (experience < 3)
            variableComponent = 0.05 * basicPay;
        else if (experience >= 3 && experience <= 5)
            variableComponent = 0.10 * basicPay;
        else if (experience > 5)
            variableComponent = 0.15 * basicPay;
        double totalSalary = basicPay + hra + variableComponent;
        setSalary(Math.round(totalSalary));
    }
    // Uncomment before verifying
    public String toString(){
        return "PermanentEmployee\nemployeeId: "+this.getEmployeeId()+
                "\nemployeeName: "+this.getEmployeeName()+
                "\nsalary: "+this.getSalary()+
                "\nbasicPay: "+this.getBasicPay()+
                "\nhra: "+this.getHra()+
                "\nexperience: "+this.getExperience();
    }
}
class ContractEmployee extends Employee {
    private double wage;
    private float hoursWorked;
   public ContractEmployee(int empId, String name, double wage, float hoursWorked) {
        super(empId, name);
        this.wage = wage;
        this.hoursWorked = hoursWorked;
    }
    public double getWage() { return wage; }
    public float getHoursWorked() { return hoursWorked; }
    public void setWage(double wage) { this.wage = wage; }
    public void setHoursWorked(float hoursWorked) { this.hoursWorked = hoursWorked; }
    public void calculateSalary() {
        double totalSalary = wage * hoursWorked;
        setSalary(Math.round(totalSalary));
    }
    // Uncomment before verifying
    public String toString(){
        return "ContractEmployee\nemployeeId: "+this.getEmployeeId()+
                "\nemployeeName: "+this.getEmployeeName()+
                "\nsalary: "+this.getSalary()+
                "\nwage: "+this.getWage()+
                "\nhoursWorked: "+this.getHoursWorked();
    }
}
class Tester {
    public static void main(String[] args) {
        PermanentEmployee permanentEmployee = new PermanentEmployee(711211, "Rafael", 1855, 115, 3.5f);
        permanentEmployee.calculateMonthlySalary();
        System.out.println("Hi "+permanentEmployee.getEmployeeName()+", your salary is $"+permanentEmployee.getSalary());
        ContractEmployee contractEmployee = new ContractEmployee(102, "Jennifer", 16, 90);
        contractEmployee.calculateSalary();
        System.out.println("Hi "+contractEmployee.getEmployeeName()+", your salary is $"+contractEmployee.getSalary());
        // Additional testing
        PermanentEmployee emp2 = new PermanentEmployee(711212, "Alice", 2000, 200, 6);
        emp2.calculateMonthlySalary();
        System.out.println("Hi "+emp2.getEmployeeName()+", your salary is $"+emp2.getSalary());
        ContractEmployee emp3 = new ContractEmployee(103, "Bob", 20, 100);
        emp3.calculateSalary();
        System.out.println("Hi "+emp3.getEmployeeName()+", your salary is $"+emp3.getSalary());
    }
}
```

Method Overloading - Assignment 1(POLYMORPHISM)

```
class Point {
    private double xCoordinate;
    private double yCoordinate;
    public Point(double xCoordinate, double yCoordinate) {
        this.xCoordinate = xCoordinate;
        this.yCoordinate = yCoordinate;
    }

    public double calculateDistance() {
        double distance = Math.sqrt(xCoordinate * xCoordinate + yCoordinate * yCoordinate);
        distance = Math.round(distance * 100.0) / 100.0; // Round to 2 decimal places
        return distance;
    }

    public double calculateDistance(Point point) {
        double deltaX = this.xCoordinate - point.xCoordinate;
        double deltaY = this.yCoordinate - point.yCoordinate;
        double distance = Math.sqrt(deltaX * deltaX + deltaY * deltaY);
        distance = Math.round(distance * 100.0) / 100.0; // Round to 2 decimal places
        return distance;
    }

    public void setxCoordinate(double xCoordinate) {
        this.xCoordinate = xCoordinate;
    }

    public double getxCoordinate() {
        return xCoordinate;
    }

    public void setyCoordinate(double yCoordinate) {
        this.yCoordinate = yCoordinate;
    }

    public double getyCoordinate() {
        return yCoordinate;
    }
}
class Tester {
    public static void main(String[] args) {
        Point point1 = new Point(3.5, 1.5);
        Point point2 = new Point(6, 4);
        System.out.println("Distance of point1 from origin is " + point1.calculateDistance());
        System.out.println("Distance of point2 from origin is " + point2.calculateDistance());
        System.out.println("Distance between point1 and point2 is " + point1.calculateDistance(point2));
        // Additional test cases
        Point origin = new Point(0, 0);
        System.out.println("Distance of origin from itself is " + origin.calculateDistance());
        System.out.println("Distance of point1 from itself is " + point1.calculateDistance(point1));
        System.out.println("Distance of point2 from itself is " + point2.calculateDistance(point2));
    }
}
```

MATHODOVERRIDING ASSIGNMENT:

IndianValley University provides education in different courses and has more than 200 faculties working in various departments. The university wants to calculate the annual salary for each faculty member. Basic salary is provided for all faculties and an additional component is paid based on their designation or qualification. You need to help the university in creating an application for them by implementing the classes based on the class diagram and description given below.

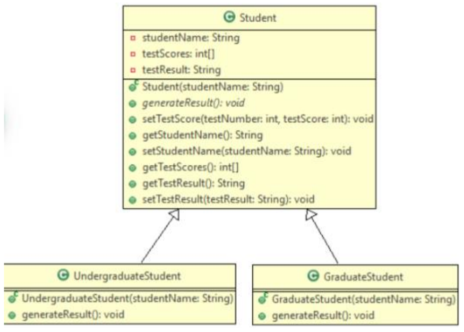
```
class User {
    private int id;
    private String userName;
    private String emailId;
    private double walletBalance;
    public User(int id, String userName, String emailId, double walletBalance) {
        this.id = id;
        this.userName = userName;
        this.emailId = emailId;
        this.walletBalance = walletBalance;
    }
    public boolean makePayment(double billAmount) {
        if (walletBalance >= billAmount) {
            walletBalance -= billAmount;
            return true;
        } else {
            return false;
        }
    }
    public void setId(int id) {
        this.id = id;
    }
    public int getId() {
        return this.id;
    }
    public void setUserName(String userName) {
        this.userName = userName;
    }
    public String getUserName() {
        return this.userName;
    }
    public void setEmailId(String emailId) {
        this.emailId = emailId;
    }
    public String getEmailId() {
        return this.emailId;
    }
    public void setWalletBalance(double walletBalance) {
        this.walletBalance = walletBalance;
    }
    public double getWalletBalance() {
        return this.walletBalance;
    }
}
class PremiumUser extends User {
    private int rewardPoints;
    public PremiumUser(int id, String userName, String emailId, double walletBalance) {
        super(id, userName, emailId, walletBalance);
        this.rewardPoints = 0;
    }
    public boolean makePayment(double billAmount) {
        boolean paymentSuccessful = super.makePayment(billAmount);
        if (paymentSuccessful) {
            rewardPoints += (int) (0.1 * billAmount);
        }
        return paymentSuccessful;
    }
    public void setRewardPoints(int rewardPoints) {
        this.rewardPoints = rewardPoints;
    }
    public int getRewardPoints() {
        return this.rewardPoints;
    }
}
class Tester {
    public static void main(String[] args) {
        User user = new User(101, "Joe", "joe@abc.com", 100);
        PremiumUser premiumUser = new PremiumUser(201, "Jill", "jill@abc.com", 300);
        processPayment(user, 70);
        processPayment(premiumUser, 150);
        processPayment(premiumUser, 80);
        processPayment(premiumUser, 120);
    }
    public static void processPayment(User user, double billAmount) {
        if (user.makePayment(billAmount)) {
            System.out.println("Congratulations " + user.getUserName() + ", payment of $" + billAmount + " was successful!");
        } else {
            System.out.println("Sorry " + user.getUserName() + ", you do not have enough balance to pay the bill!");
        }
        System.out.println("Your wallet balance is $" + user.getWalletBalance());
        if (user instanceof PremiumUser) {
            PremiumUser premiumUser = (PremiumUser) user;
            System.out.println("You have " + premiumUser.getRewardPoints() + " points!");
        }
        System.out.println();
    }
}
```

FINAL_KEYWORD EXERCISE 1:

```
class Student{
    //Implement your code here
    private final int STIPEND = 100;
    private int studentId;
    private int aggregateMarks;
    public void setStudentId(int studentId){
        this.studentId = studentId;
    }
    public int getStudentId(){
        return this.studentId ;
    }
    public void setAggregateMarks(int aggregateMarks){
        this.aggregateMarks = aggregateMarks;
    }
    public int getAggregateMarks(){
        return this.aggregateMarks ;
    }
    public int getSTIPEND(){
        return this.STIPEND ;
    }
    public double calculateTotalStipend(){
        if(aggregateMarks >= 85 && aggregateMarks < 90) return STIPEND + 10;
        else if(aggregateMarks >= 90 && aggregateMarks < 95)  return STIPEND + 15;
        else if(aggregateMarks >= 95 && aggregateMarks <= 100 ) return STIPEND + 20;
        else return STIPEND ;
    }
}
class Tester {
        public static void main(String[] args) {
                Student student1 = new Student();
                student1.setStudentId(1212);
                student1.setAggregateMarks(93);
                double totalStipend = student1.calculateTotalStipend();
                System.out.println("The final stipend of " + student1.getStudentId()+" is $" + totalStipend);
                Student student2 = new Student();
                student2.setStudentId(1222);
                student2.setAggregateMarks(84);
                totalStipend = student2.calculateTotalStipend();
                System.out.println("The final stipend of " + student2.getStudentId()+" is $" + totalStipend);
        }
}
```

ABSTRACT EXERCISE1:

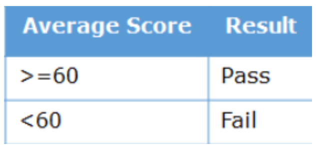
Anchor College offers both graduate and postgraduate programs. The college stores the names of the students, their test scores and the final result for each student. Each student has to take 4 tests in total. You need to create an application for the college by implementing the classes based on the class diagram and description given below.


Student
studentName: String
testScores: int[]
testResult: String
Student(studentName: String)
generateResult(): void
setTestScore(testNumber: int, testScore: int): void
getStudentName(): String
setStudentName(studentName: String): void
getTestScores(): int[]
getTestResult(): String
setTestResult(testResult: String): void
UndergraduateStudent
UndergraduateStudent(studentName: String)
generateResult(): void
GraduateStudent
GraduateStudent(studentName: String)
generateResult(): void

Method Description

Student

Student(String studentName).Initialize the instance variable studentName with the value passed to the constructor and other instance variables to the default values.:setTestScore(int testNumber, int testScore).Set the value of the testScore in the appropriate position of testScores array based on the testNumber.Implement the getter and setter methods appropriately.UndergraduateStudent.UndergraduateStudent(String studentName).Initialize the instance variable studentName with the value passed to the constructor and other instance variables to the default values.generateResult().Implement the abstract method of Student class by setting the value of testResult based on the below details.

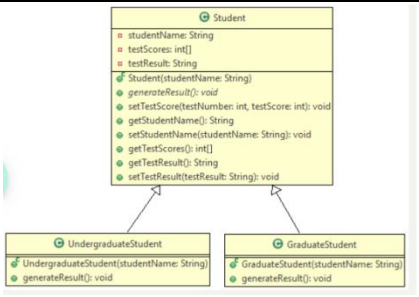


Average Score	Result
>=60	Pass
<60	Fail

```
abstract class Student{
    //Implement your code here
    private String studentName;
    private int[] testScores;
    private String testResult;
    public Student(String studentName){
        this.studentName = studentName;
    }
    public void setTestScore(int testNumber, int testScore){
        this.testScores[testNumber] = testScore;
    }
    public void setStudentName(String studentName){
        this.studentName = studentName;
    }
    public void setTestResult(String testResult){
        this.testResult = testResult;
    }
    public abstract void generateResult(){};

    public String getStudentName(){
        return this.studentName;
    }
    public String getTestResult(){
        return this.testResult;
    }
    public int[] getTestScores(){
        return this.testScores;
    }
}
class UndergraduateStudent{
    //Implment your code here
    public UndergraduateStudent(String studentName){
        this.studentName = studentName;
    }
    public void generateResult(){
        int averageScore;
        for(int score : testScores) averageScore += score;
        averageScore /= testScores.length();
        if(averageScore >= 60) this.testResult = "Pass";
        else this.testResult = "Fail";
    }
}

class GraduateStudent{
    //Implment your code here
    public GraduateStudent(String studentName){
        this.studentName = studentName;
    }
    public void generateResult(){
        int averageScore;
        for(int score : testScores) averageScore += score;
        averageScore /= testScores.length();
        if(averageScore >= 70) this.testResult = "Pass";
        else this.testResult = "Fail";
    }
}

class Tester {
    public static void main(String[] args) {
        UndergraduateStudent undergraduateStudent = new UndergraduateStudent("Philip");
        undergraduateStudent.setTestScore(0, 70);
        undergraduateStudent.setTestScore(1, 69);
        undergraduateStudent.setTestScore(2, 71);
        undergraduateStudent.setTestScore(3, 55);
        undergraduateStudent.generateResult();
        System.out.println("Student name: "+undergraduateStudent.getStudentName());
        System.out.println("Result: "+undergraduateStudent.getTestResult());
        System.out.println();
        GraduateStudent graduateStudent = new GraduateStudent("Jerry");
        graduateStudent.setTestScore(0, 70);
        graduateStudent.setTestScore(1, 69);
        graduateStudent.setTestScore(2, 71);
        graduateStudent.setTestScore(3, 55);
        graduateStudent.generateResult();
        System.out.println("Student name: "+graduateStudent.getStudentName());
        System.out.println("Result : "+graduateStudent.getTestResult());
        //Create more objects of the classes for testing your code
    }
}
```

ABSTRACT ASSIGNMENT 1

Anchor College offers both graduate and postgraduate programs. The college stores the names of the students, their test scores and the final result for each student. Each student has to take 4 tests in total. You need to create an application for the college by implementing the classes based on the class diagram and description given below.


Student
studentName: String
testScores: int[]
testResult: String
Student(studentName: String)
generateResult(): void
setTestScore(testNumber: int, testScore: int): void
getStudentName(): String
setStudentName(studentName: String): void
getTestScores(): int[]
getTestResult(): String
setTestResult(testResult: String): void
UndergraduateStudent
UndergraduateStudent(studentName: String)
generateResult(): void
GraduateStudent
GraduateStudent(studentName: String)
generateResult(): void

Method Description

Student

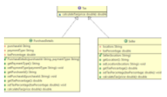
Student(String studentName).Initialize the instance variable studentName with the value passed to the constructor and other instance variables to the default values.setTestScore(int testNumber, int testScore).Set the value of the testScore in the appropriate position of testScores array based on the testNumber.Implement the getter and setter methods appropriately.UndergraduateStudentUndergraduateStudent(String studentName).Initialize the instance variable studentName with the value passed to the constructor and other instance variables to the default values.,generateResult().Implement the abstract method of Student class by setting the value of testResult based on the below details.

```
abstract class Payment {
    private int customerId;
    protected String paymentId;
    protected double serviceTaxPercentage;
    public abstract double payBill(double amount);
    public Payment(int customerId) {
        this.customerId = customerId;
    }
    // Getters and Setters
    public int getCustomerId() {
        return customerId;
    }
    public String getPaymentId() {
        return paymentId;
    }
    public void setPaymentId(String paymentId) {
        this.paymentId = paymentId;
    }
    public double getServiceTaxPercentage() {
        return serviceTaxPercentage;
    }
    public void setServiceTaxPercentage(double serviceTaxPercentage) {
        this.serviceTaxPercentage = serviceTaxPercentage;
    }
    public void setCustomerId(int customerId){
        this.customerId = customerId;
    }
}
class DebitCardPayment extends Payment {
```

```
private static int counter = 1000;
private double discountPercentage;
public DebitCardPayment(int customerId) {
    super(customerId);
    generatePaymentId();
}
private void generatePaymentId() {
    this.paymentId = "D" + counter++;
}
public double payBill(double amount) {
    if(amount <= 500){
        serviceTaxPercentage = 2.5;
        discountPercentage = 1;
    }
    else if(amount > 500 && amount <= 1000){
        serviceTaxPercentage = 4;
        discountPercentage = 2;
    }
    else if(amount > 1000){
        serviceTaxPercentage = 5;
        discountPercentage = 3;
    }
    // double serviceTaxAmount = (amount * serviceTaxPercentage) / 100;
    // double discountAmount = (amount * discountPercentage) / 100;


    double finalBillAmount = amount + serviceTaxAmount - discountAmount;
    return finalBillAmount;
}
// Getters and Setters
public String getPaymentId() {
    return paymentId;
}
public double getDiscountPercentage() {
    return discountPercentage;
}
public void setDiscountPercentage(double discountPercentage) {
    this.discountPercentage = discountPercentage;
}
public static int getCounter(){
    return DebitCardPayment.counter;
}
public static void setCounter(int counter){
        DebitCardPayment.counter = counter;
    }
}
class CreditCardPayment extends Payment {
    private static int counter = 1000;
    private String paymentId;
    private double rewardPoints;
    public CreditCardPayment(int customerId) {
        super(customerId);
        generatePaymentId();
    }
    private void generatePaymentId() {
        this.paymentId = "C" + counter++;
    }
    @Override
    public double payBill(double amount) {
       if(amount <= 500){
           serviceTaxPercentage = 3;
        }
        else if(amount > 500 && amount <= 1000){
           serviceTaxPercentage = 5;
        }
        else if(amount > 1000){
           serviceTaxPercentage = 6;
        }
        double serviceTaxAmount = (amount * serviceTaxPercentage) / 100;
        double discountAmount = (amount * discountPercentage) / 100;

        double finalBillAmount = amount + serviceTaxAmount;
        return finalBillAmount;
    }
    // Getters and Setters
    public String getPaymentId() {
        return paymentId;
    }
    public double getRewardPoints() {
        return rewardPoints;
    }
    public void setRewardPoints(double rewardPoints) {
        this.rewardPoints = rewardPoints;
    }
    public static int getCounter(){
        return CreditCardPayment.counter;
    }
    public static void setCounter(int counter){
        CreditCardPayment.counter = counter;
    }
}
class Tester {
    public static void main(String[] args) {
        DebitCardPayment payment1 = new DebitCardPayment(101);
        DebitCardPayment payment2 = new DebitCardPayment(102);
        double amount1 = 1000.0;
        double amount2 = 500.0;
        System.out.println("Payment ID: " + payment1.getPaymentId());
        System.out.println("Final Bill Amount: $" + payment1.payBill(amount1));
        System.out.println("Payment ID: " + payment2.getPaymentId());
        System.out.println("Final Bill Amount: $" + payment2.payBill(amount2));
    }
}
```

INTERFACE EXERCISE 1

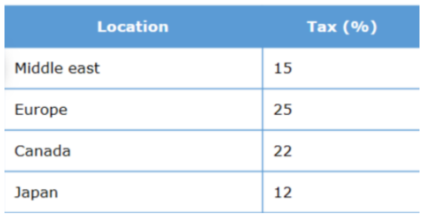
An e-commerce company has made it mandatory for all the customers to pay tax on every purchase and also for all the sellers to pay a certain amount of tax based on their location. You need to help the e-commerce company by implementing an application for paying tax based on the class diagram and description given below.



Method Description

PurchaseDetails

Purchase(String purchaseId, String paymentType):Initialize the purchaseId and paymentType instance variables using the values passed to the constructor.calculateTax(double price) .Initialize the value of the taxPercentage instance variable based on the details given below.



Location	Tax (%)
Middle east	15
Europe	25
Canada	22
Japan	12

- **Calculate and return the tax amount that needs to be paid.**

```
interface Tax {
    double calculateTax(double price);
}

class PurchaseDetails implements Tax {
    private String purchaseId;
    private String paymentType;
    private double taxPercentage;
    public PurchaseDetails(String purchaseId, String paymentType) {
        this.purchaseId = purchaseId;
        this.paymentType = paymentType;
    }
    @Override
    public double calculateTax(double price) {
        if (this.paymentType.equals("Debit Card")) this.taxPercentage = 2;
        else if (this.paymentType.equals("Credit Card")) this.taxPercentage = 3;
        else this.taxPercentage = 4;
        return price+(price*taxPercentage/100);
    }
    public void setPurchaseId(String purchaseId) {
        this.purchaseId = purchaseId;
    }
    public String getPurchaseId() {
        return this.purchaseId;
    }
    public void setPaymentType(String paymentType) {
        this.paymentType = paymentType;
    }
    public String getPaymentType() {
        return this.paymentType;
    }
    public void setTaxPercentage(double taxPercentage) {
        this.taxPercentage = taxPercentage;
    }
    public double getTaxPercentage() {
        return this.taxPercentage;
    }
}
class Seller implements Tax {
    private String location;
    private double taxPercentage;

    public Seller(String location) {
        this.location = location;
    }
    @Override
    public double calculateTax(double price) {
    }
    public void setLocation(String location) {
        this.location = location;
    }
    public String getLocation() {
        return this.location;
    }
    public void setTaxPercentage(double taxPercentage) {
        this.taxPercentage = taxPercentage;
    }
    public double getTaxPercentage() {
        return this.taxPercentage;
    }
}
class Tester {
    public static void main(String[] args) {
        System.out.println("Purchase Details\n***************");
        PurchaseDetails purchaseDetails = new PurchaseDetails("P1001", "Credit Card");
        System.out.println("Total purchase amount: " + Math.round(purchaseDetails.calculateTax(100) * 100) / 100.0);
        System.out.println("Tax percentage: " + purchaseDetails.getTaxPercentage());
        System.out.println("Seller Details\n***************");
        Seller seller = new Seller("Canada");
        System.out.println("Tax to be paid by the seller: " + Math.round(seller.calculateTax(100) * 100) / 100.0);
        System.out.println("Tax percentage: " + seller.getTaxPercentage());
        // Create more objects for testing your code
    }
}
```

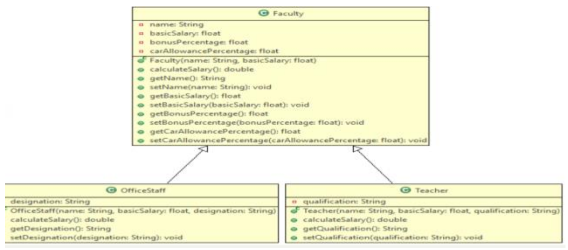
INTERFACE ASSIGNMENT 1

```
interface Testable {
    boolean testCompatibility();
}
class Mobile {
    private String name;
    private String brand;
    private String operatingSystemName;
    private String operatingSystemVersion;
    public Mobile(String name, String brand, String operatingSystemName, String operatingSystemVersion) {
        this.name = name;
        this.brand = brand;
        this.operatingSystemName = operatingSystemName;
        this.operatingSystemVersion = operatingSystemVersion;
    }
    public String getName() {
```

```java
        return name;
    }
    public String getBrand() {
        return brand;
    }
    public String getOperatingSystemName() {
        return operatingSystemName;
    }
    public String getOperatingSystemVersion() {
        return operatingSystemVersion;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setBrand(String brand) {
        this.brand = brand;
    }
    public void setOperatingSystemName(String operatingSystemName) {
        this.operatingSystemName = operatingSystemName;
    }
    public void setOperatingSystemVersion(String operatingSystemVersion) {
        this.operatingSystemVersion = operatingSystemVersion;
    }
}
class SmartPhone extends Mobile implements Testable {
    private String networkGeneration;

    public SmartPhone(String name, String brand, String operatingSystemName, String operatingSystemVersion, String networkGeneration) {
        super(name, brand, operatingSystemName, operatingSystemVersion);
        this.networkGeneration = networkGeneration;
    }
    public String getNetworkGeneration() {
        return networkGeneration;
    }
    public void setNetworkGeneration(String networkGeneration) {
        this.networkGeneration = networkGeneration;
    }
    @Override
    public boolean testCompatibility() {
        ArrayList<String> versionList = new ArrayList<>();
        if (this.operatingSystemName.equals("Saturn")) {
            if (this.networkGeneration.equals("3G")) Collections.addAll(versionList, "EXRT.1", "EXRT.2", "EXRU.1");
            else if (this.networkGeneration.equals("4G")) Collections.addAll(versionList, "EXRT.2", "EXRU.1");
            else if (this.networkGeneration.equals("5G")) Collections.addAll(versionList, "EXRU.1");
        } else if (this.operatingSystemName.equals("Gara")) {
            if (this.networkGeneration.equals("3G")) Collections.addAll(versionList, "1.1", "1.2", "1.3");
            else if (this.networkGeneration.equals("4G")) Collections.addAll(versionList, "1.2", "1.3");
            else if (this.networkGeneration.equals("5G")) Collections.addAll(versionList, "1.3");
        }
        String OSV = this.getOperatingSystemVersion();
        return versionList.contains(OSV);
    }
}
class Tester {
    public static void main(String args[]) {
        SmartPhone smartPhone = new SmartPhone("KrillinM20", "Nebula", "Saturn", "1.3", "5G");
        if (smartPhone.testCompatibility())
            System.out.println("The mobile OS is compatible with the network generation!");
        else
            System.out.println("The mobile OS is not compatible with the network generation!");
        // Create more objects for testing your code
    }
}
```

METHOD OVERRIDING ASSIGNMENT 1

IndianValley University provides education in different courses and has more than 200 faculties working in various departments. The university wants to calculate the annual salary for each faculty member. Basic salary is provided for all faculties and an additional component is paid based on their designation or qualification. You need to help the university in creating an application for them by implementing the classes based on the class diagram and description given below.

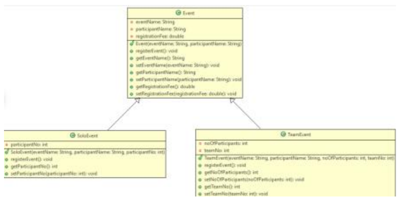


Faculty
name: String
basicSalary: float
bonusPercentage: float
carAllowancePercentage: float
Faculty(name: String, basicSalary: float)
calculateSalary(): double
getName(): String
setName(name: String): void
getBasicSalary(): float
setBasicSalary(basicSalary: float): void
getBonusPercentage(): float
setBonusPercentage(bonusPercentage: float): void
getCarAllowancePercentage(): float
setCarAllowancePercentage(carAllowancePercentage: float): void
OfficeStaff
designation: String
OfficeStaff(name: String, basicSalary: float, designation: String)
calculateSalary(): double
getDesignation(): String
setDesignation(designation: String): void
Teacher
qualification: String
Teacher(name: String, basicSalary: float, qualification: String)
calculateSalary(): double
getQualification(): String
setQualification(qualification: String): void

```
class Faculty {
    private String name;
    private float basicSalary;
    private float bonusPercentage;
    private float carAllowancePercentage;
    public Faculty(String name, float basicSalary) {
        this.name = name;
        this.basicSalary = basicSalary;
    }
    public double calculateSalary() {
        return basicSalary + (basicSalary * bonusPercentage / 100) + (basicSalary * carAllowancePercentage / 100);
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public float getBasicSalary() {
        return basicSalary;
    }
    public void setBasicSalary(float basicSalary) {
        this.basicSalary = basicSalary;
    }
    public float getBonusPercentage() {
        return bonusPercentage;
    }
    public void setBonusPercentage(float bonusPercentage) {
        this.bonusPercentage = bonusPercentage;
    }
    public float getCarAllowancePercentage() {
        return carAllowancePercentage;
    }
    public void setCarAllowancePercentage(float carAllowancePercentage) {
        this.carAllowancePercentage = carAllowancePercentage;
    }
}
class OfficeStaff extends Faculty {
    private String designation;
    public OfficeStaff(String name, float basicSalary, String designation) {
        super(name, basicSalary);
        this.designation = designation;
    }
    public void setDesignation(String designation) {
        this.designation = designation;
    }
    public String getDesignation() {
        return this.designation;
    }
    @Override
    public float calculateSalary() {
        float additionalPay = 0;
        if (this.designation.equals("Accountant")) {
            additionalPay = 10000f;
        } else if (this.designation.equals("Clerk")) {
            additionalPay = 7000f;
        } else if (this.designation.equals("Peon")) {
            additionalPay = 4500f;
        }
        return getBasicSalary() + (getBasicSalary() * getBonusPercentage() / 100) + (getBasicSalary() * getCarAllowancePercentage() / 100) +
additionalPay;
    }
       public double calculateSalary(){
    }
}
class Teacher extends Faculty {
    private String qualification;
    public Teacher(String name, float basicSalary, String qualification) {
        super(name, basicSalary);
        this.qualification = qualification;
    }
   @Override
    public float calculateSalary() {
        float additionalPay = 0;
        if (this.qualification.equals("Doctoral")) {
            additionalPay = 20000f;
        } else if (this.qualification.equals("Masters")) {
            additionalPay = 18000f;
        } else if (this.qualification.equals("Bachelors")) {
            additionalPay = 15500f;
        } else if (this.qualification.equals("Associate")) {
            additionalPay = 10000f;
        }
        return getBasicSalary() + (getBasicSalary() * getBonusPercentage() / 100) + (getBasicSalary() * getCarAllowancePercentage() / 100) +
additionalPay;
    }
    public double calculateSalary(){
    }
    public void setQualification(String qualification) {
        this.qualification = qualification;
    }
    public String getQualification() {
        return this.qualification;
    }
}
class Tester {
    public static void main(String[] args) {
        Teacher teacher = new Teacher("Caroline", 30500f, "Masters");
        teacher.setBonusPercentage(10f);
        teacher.setCarAllowancePercentage(5f);
        OfficeStaff officeStaff = new OfficeStaff("James", 24000f, "Accountant");
        officeStaff.setBonusPercentage(8f);
        officeStaff.setCarAllowancePercentage(4f);
        System.out.println("Teacher Details\n***************");
        System.out.println("Name: " + teacher.getName());
        System.out.println("Qualification: " + teacher.getQualification());
        System.out.printf("Total salary: $%.2f\n", teacher.calculateSalary());
        System.out.println();
        System.out.println("Office Staff Details\n***************");
        System.out.println("Name: " + officeStaff.getName());
        System.out.println("Designation: " + officeStaff.getDesignation());
        System.out.printf("Total salary: $%.2f\n", officeStaff.calculateSalary());
        // Create more objects for testing your code
    }
}
```

METHODOVERRIDING ASSIGNMENT 2

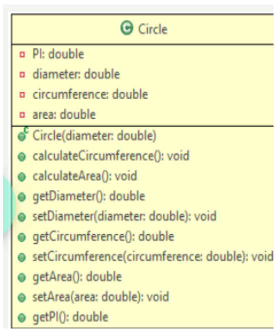
A college cultural event "Show Your Talent" is being conducted and the organizing committee has decided to open online registration for the same. The students can participate in solo events as well as team events. You need to develop an application for the online registration by implementing the classes based on the class diagram and description given below.


Event
eventName: String
participantName: String
registrationFee: double
Event(eventName: String, participantName: String)
registerEvent(): void
getEventName(): String
setEventName(eventName: String): void
getParticipantName(): String
setParticipantName(participantName: String): void
getRegistrationFee(): double
setRegistrationFee(registrationFee: double): void
SoloEvent
participantNo: int
SoloEvent(eventName: String, participantName: String, participantNo: int)
registerEvent(): void
getParticipantNo(): int
setParticipantNo(participantNo: int): void
TeamEvent
noOfParticipants: int
teamNo: int
TeamEvent(eventName: String, participantName: String, noOfParticipants: int, teamNo: int)
registerEvent(): void
getNoOfParticipants(): int
setNoOfParticipants(noOfParticipants: int): void
getTeamNo(): int
setTeamNo(teamNo: int): void

Method Description
Event
Event(String eventName, String participantName).Initialize the instance variables eventName and participantName using the values passed to the constructor.
registerEvent().Register participants for the event by setting the registrationFee with the help of the below table.
For any invalid event, set the registrationFee to 0.
Implement the appropriate getter and setter methods.
SoloEvent
SoloEvent(String eventName, String participantName, int participantNo).Initialize the instance variables eventName, participantName and participantNo using the values passed to the constructor.
registerEvent().Register participants for the event by setting the registrationFee with the help of the parent class method.
Implement the appropriate getter and setter methods.
TeamEvent
TeamEvent(String eventName, String participantName, int noOfParticipants, int teamNo).Initialize the instance variables eventName, participantName, noOfParticipants and teamNo using the values passed to the constructor.
registerEvent().Register participants for the event by setting the registrationFee with the help of the below table. The fee given in the table is for each member. The registrationFee should be calculated by considering the number of members in the team.
For any invalid event, set the registrationFee to 0.
Implement the appropriate getter and setter methods.
Test the functionalities using the provided Tester class.

```
class Event {
    private String eventName;
    private String participantName;
private double registrationFee;
public Event(String eventName, String participantName) {
    this.eventName = eventName;
    this.participantName = participantName;
    registerEvent();
}
public void registerEvent() {
    switch (eventName) {
        case "Singing":
            registrationFee = 8;
            break;
        case "Dancing":
            registrationFee = 10;
            break;
        case "DigitalArt":
            registrationFee = 12;
            break;
        case "Acting":
            registrationFee = 15;
            break;
        default:
            registrationFee = 0;
            break;
    }
}
public String getEventName() {
    return eventName;
}
public void setEventName(String eventName) {
    this.eventName = eventName;
}
public String getParticipantName() {
    return participantName;
}
public void setParticipantName(String participantName) {
    this.participantName = participantName;
}
public double getRegistrationFee() {
    return registrationFee;
}
public void setRegistrationFee(double registrationFee) {
    this.registrationFee = registrationFee;
    }
}
class SoloEvent extends Event {
    private int participantNo;
    public SoloEvent(String eventName, String participantName, int participantNo) {
        super(eventName, participantName);
        this.participantNo = participantNo;
    }
    public int getParticipantNo() {
        return participantNo;
    }
    public void setParticipantNo(int participantNo) {
        this.participantNo = participantNo;
    }
    public void registerEvent(){
         super.registerEvent();
    }
}
class TeamEvent extends Event {
    private int noOfParticipants;
    private int teamNo;
    public TeamEvent(String eventName, String participantName, int noOfParticipants, int teamNo) {
        super(eventName, participantName);
        this.noOfParticipants = noOfParticipants;
        this.teamNo = teamNo;
    }
    @Override
    public void registerEvent(){
    }
    protected void registerEvent() {
        super.registerEvent();
        if (getRegistrationFee() != 0) {
            setRegistrationFee(getRegistrationFee() * noOfParticipants);
        }
    }
    public int getNoOfParticipants() {
        return noOfParticipants;
    }

    public void setNoOfParticipants(int noOfParticipants) {
        this.noOfParticipants = noOfParticipants;
    }
    public int getTeamNo() {
        return teamNo;
    }
    public void setTeamNo(int teamNo) {
        this.teamNo = teamNo;
    }
}
class Tester {
    public static void main(String[] args) {
        SoloEvent soloEvent = new SoloEvent("Dancing", "Jacob", 1);
        if (soloEvent.getRegistrationFee() > 0) {
            System.out.println("Thank You " + soloEvent.getParticipantName()
                    + " for your participation! Your registration fee is $" + String.format("%.2f", soloEvent.getRegistrationFee()));
            System.out.println("Your participant number is " + soloEvent.getParticipantNo());
        } else {
            System.out.println("Please enter a valid event");
        }
        System.out.println();
        TeamEvent teamEvent = new TeamEvent("Acting", "Serena", 5, 1);
        if (teamEvent.getRegistrationFee() > 0) {
            System.out.println("Thank You " + teamEvent.getParticipantName()
                    + " for your participation! Your registration fee is $" + String.format("%.2f", teamEvent.getRegistrationFee()));
            System.out.println("Your team number is " + teamEvent.getTeamNo());
        } else {
            System.out.println("Please enter a valid event");
        }
    }
}
```

FINAL_ASSIGNMENT


Circle
PI: double
diameter: double
circumference: double
area: double
Circle(diameter: double)
calculateCircumference(): void
calculateArea(): void
getDiameter(): double
setDiameter(diameter: double): void
getCircumference(): double
setCircumference(circumference: double): void
getArea(): double
setArea(area: double): void
getPI(): double

Method Description

Circle(double diameter)Initialize the instance variable diameter with the value passed to the constructor.

calculateCircumference()Calculate and set the circumference of the circle using the formula given below.

circumference = πd, where d is the diameter

calculateArea()Calculate and set the area of the circle using the formula given below.

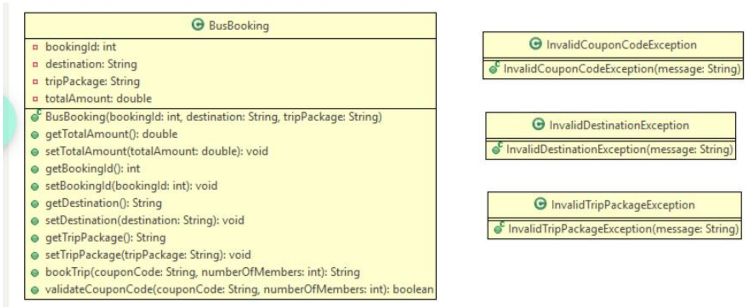
area=πr^2, where r is the radiusNote: PI is a final variable.

Implement the getter and setter methods appropriately.

```
class Circle {
    private final double PI = 3.14;
    private double diameter;
    private double circumference;
    private double area;
    public Circle(double diameter) {
        this.diameter = diameter;
    }
    // Getters
    public double getDiameter() {
        return diameter;
    }
    public double getCircumference() {
        return circumference;
    }
    public double getArea() {
        return area;
    }
    public double getPI(){
        return this.PI;
    }
    // Setters
    public void setDiameter(double diameter) {
        this.diameter = diameter;
    }
    public void setCircumference(double circumference) {
        this.circumference = circumference;
    }
    public void setArea(double area) {
        this.area = area;
    }
    // Calculate circumference and area based on diameter
    public void calculateCircumference() {
        this.circumference = PI * diameter;
    }
    public void calculateArea() {
        this.area = PI * Math.pow(diameter / 2, 2);
    }
}
class Tester {
    public static void main(String[] args) {
        Circle circle1 = new Circle(10.2);
        Circle circle2 = new Circle(5.7);
        // Create more objects of Circle class and add to the array given below for testing your code
        Circle[] circles = {circle1, circle2};
        for (Circle circle : circles) {
            circle.calculateCircumference();
            circle.calculateArea();
            System.out.println("Diameter of the circle is " + circle.getDiameter());
            System.out.println("Circumference of the circle is " + Math.round(circle.getCircumference() * 100) / 100.0);
            System.out.println("Area of the circle is " + Math.round(circle.getArea() * 100) / 100.0);
            System.out.println();
        }
    }
}
```

EXCEPTION ASSIGNMENT 1

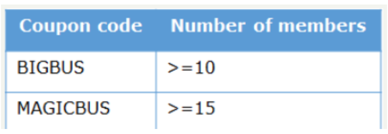
Jumbo Bus Service provides bus services from New York to various cities based on different kinds of packages. They want to ensure that appropriate messages are displayed for users while booking buses. You need to help the Jumbo Bus Service by developing an application based on the class diagram and description given below.


BusBooking
bookingId: int
destination: String
tripPackage: String
totalAmount: double
BusBooking(bookingId: int, destination: String, tripPackage: String)
getTotalAmount(): double
setTotalAmount(totalAmount: double): void
getBookingId(): int
setBookingId(bookingId: int): void
getDestination(): String
setDestination(destination: String): void
getTripPackage(): String
setTripPackage(tripPackage: String): void
bookTrip(couponCode: String, numberOfMembers: int): String
validateCouponCode(couponCode: String, numberOfMembers: int): boolean
InvalidCouponCodeException
InvalidCouponCodeException(message: String)
InvalidDestinationException
InvalidDestinationException(message: String)
InvalidTripPackageException
InvalidTripPackageException(message: String)

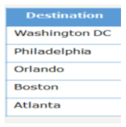
Method Description

BusBooking(int bookingId, String destination, String tripPackage).Initialize the bookingId, destination and tripPackage instance variables with the values passed to the constructor and initialize the totalAmount instance variable to 0.

validateCouponCode(String couponCode, int numberOfMembers).Validate the couponCode based on the table given below.



Coupon code	Number of members
BIGBUS	>=10
MAGICBUS	>=15



Destination
Washington DC
Philadelphia
Orlando
Boston
Atlanta

If the destination is not valid, throw InvalidDestinationException with the message "Invalid destination".Check if the tripPackage is either 'Regular' or 'Premium'.If the tripPackage is not valid, throw InvalidTripPackageException with the message "Invalid package".Check if the couponCode is valid or not by invoking the validateCouponCode() method.If all the values are valid, calculate and initialize the totalAmount instance variable based on the table given below.

```
class InvalidDestinationException extends Exception {
    public InvalidDestinationException(String message) {
        super(message);
    }
}
class InvalidCouponCodeException extends Exception {
    public InvalidCouponCodeException(String message) {
        super(message);
    }
}
class InvalidTripPackageException extends Exception {
    public InvalidTripPackageException(String message) {
        super(message);
    }
}
class BusBooking {
    private int bookingId;
    private String destination;
    private String tripPackage;
    private double totalAmount;
    public BusBooking(int bookingId, String destination, String tripPackage) {
        this.bookingId = bookingId;
        this.destination = destination;
        this.tripPackage = tripPackage;
    }
    public int getBookingId() {
        return bookingId;
    }
    public void setBookingId(int bookingId) {
        this.bookingId = bookingId;
    }
    public String getDestination() {
        return destination;
    }
    public void setDestination(String destination) {
        this.destination = destination;
    }
    public String getTripPackage() {
        return tripPackage;
    }
    public void setTripPackage(String tripPackage) {
        this.tripPackage = tripPackage;
    }
    public double getTotalAmount() {
        return totalAmount;
    }
    public void setTotalAmount(double totalAmount) {
        this.totalAmount = totalAmount;
    }
    public boolean validateCouponCode(String couponCode, int numberOfMembers) throws InvalidCouponCodeException {
        if ((couponCode.equals("BIGBUS") && numberOfMembers >= 10) || (couponCode.equals("MAGICBUS") && numberOfMembers >=
15)) {
            return true;
        } else {
            throw new InvalidCouponCodeException("Invalid coupon code");
        }
    }
public boolean validateDestination() throws InvalidDestinationException {
        if (this.destination.equals("Washington DC") ||
            this.destination.equals("Philadelphia") ||
            this.destination.equals("Orlando") ||
            this.destination.equals("Boston") ||
            this.destination.equals("Atlanta")) {
            return true;
        } else {
            throw new InvalidDestinationException("Invalid destination");
        }
    }
    public boolean validateTripPackage() throws InvalidTripPackageException {
        if (this.tripPackage.equals("Regular") || this.tripPackage.equals("Premium")) {
            return true;
        } else {
            throw new InvalidTripPackageException("Invalid package");
        }
    }
    public String bookTrip(String couponCode, int numberOfMembers) {
        try {
            if (validateCouponCode(couponCode, numberOfMembers) && validateTripPackage() && validateDestination()) {
                double price;
                if (this.tripPackage.equals("Regular")) {
                    price = 500;
                } else {
                    price = 800;
                }
                this.totalAmount = price * numberOfMembers;
                return "Booking successful";
            }
        } catch (InvalidCouponCodeException | InvalidDestinationException | InvalidTripPackageException e) {
            return e.getMessage();
        }
        return "Booking failed due to unknown error";
    }
}
class Tester {
    public static void main(String[] args) {
        BusBooking booking = new BusBooking(101, "Orlando", "Regular");
        String result = booking.bookTrip("BIGBUS", 11);
        if (result.equals("Booking successful")) {
            System.out.println(result);
            System.out.println("Total amount for the trip: " + booking.getTotalAmount());
        } else {
            System.out.println(result);
            System.out.println("Your booking was not successful, please try again!");
        }
    }
}
```

EXERCISE1

```
// Implement user defined exception classes
class Applicant {
    private String name;
    private String jobProfile;
    private int age;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getJobProfile() {
        return jobProfile;
    }
    public void setJobProfile(String jobProfile) {
        this.jobProfile = jobProfile;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}
```

```
class Validator{
   //Implement your code here
}
class Tester {
   public static void main(String[] args) {
      try {
         Applicant applicant= new Applicant();
         applicant.setName("Jenny");
         applicant.setJobProfile("Clerk");
         applicant.setAge(25);
         Validator validator = new Validator();
         validator.validate(applicant);
         System.out.println("Application submitted successfully!");
      }
      catch (InvalidNameException|InvalidJobProfileException|InvalidAgeException e) {
         System.out.println(e.getMessage());
      }
   }
}
```

RECURSION:
ASSIGNMNET 1:

Problem Statement:

Write a Java program to calculate the nth Fibonacci number using recursion.

```
class Tester {
   public static int findFibonacci(int n) {
      if (n == 0 || n == 1) return 0;
      else if (n == 2) return 1;
      else return findFibonacci(n - 1) + findFibonacci(n - 2);
   }
   public static void main(String[] args) {
      int n = 5; // Change this value to test different cases
      if (n < 0) {
         System.out.println("Please enter a non-negative value for n");
      } else {
         System.out.println("The " + n + "th Fibonacci number is: " + findFibonacci(n));
      }
   }
}
```

ASSIGNMENT 2:

Problem Statement:

Write a Java program to calculate the sum of the first n terms of a harmonic progression (HP) using recursion.

```
class Tester {
    public static double findHPSum(int num) {
        //Implement your code here and change the return value
        if (num == 1) {
            return 1.0;
        }
        return (1.0 / num) + findHPSum(num - 1);
    }
    public static void main(String args[]) {
        System.out.println(findHPSum(3));
    }
}
```

ASSIGNMENT 3:

Problem Statement:

Write a Java program to calculate the Greatest Common Divisor (GCD) of two numbers using recursion.

```
class Tester {
    public static int findGCD(int num1, int num2) {
        if (num2 == 0) {
            return num1;
        }
        return findGCD(num2, num1 % num2);
    }
    public static int findMax(int num1, int num2) {
        return (num1 > num2 ? num1 : num2);
    }
    public static int findMin(int num1, int num2) {
        return (num1 < num2 ? num1 : num2);
    }
    public static void main(String args[]) {
        System.out.println(findGCD(5, 10));
        System.out.println(findGCD(14, 21));
        System.out.println(findGCD(100, 200));
        System.out.println(findGCD(13, 13));
    }
}
```

ASSIGNMENT 4:

Problem Statement:

Write a Java program to reverse the elements of an integer array recursively.

```
class Tester {
    public static int[] reverseArray(int numbers[], int startIndex, int endIndex) {
        //Implement your code here and change the return value accordingly
        if(startIndex >= endIndex) return numbers;
        else{
            int temp = numbers[startIndex];
            numbers[startIndex] = numbers[endIndex];
            numbers[endIndex] = temp;
            return reverseArray(numbers,++startIndex,--endIndex);
        }
    }
    public static void main(String args[]) {
        int numbers[] = new int[] { 1, 2, 3, 4, 5, 6 };
        int reversedNumbers[] = reverseArray(numbers, 0, numbers.length - 1);
        System.out.println("Reversed array");
        for (int number : reversedNumbers) {
            System.out.println(number);
        }
    }
}
```

ASSIGNMENT 5:

Problem Statement:

Write a Java program to count the number of times a specific substring occurs in a given string recursively.

```
class Tester {
    public static int countSubstring(String inputString, String substring, int count) {
        //Implement your code here and change the return value
        if (inputString.length() < substring.length()) {
            return count;
        }
        if (inputString.startsWith(substring)) {
            count++;
            return countSubstring(inputString.substring(substring.length()), substring, count);
        } else {
            return countSubstring(inputString.substring(1), substring, count);
        }
    }
    public static void main(String[] args) {
        int count = countSubstring("I felt happy because I saw the others were happy and because I knew I should feel happy, but I
wasn't really happy", "happy", 0);
        System.out.println(count);
    }
}
```

EXERCISE:

Problem Statement:

Write a Java program to check whether a given integer is a palindrome using recursion.

A palindrome is a number that reads the same backward as forward.

```
class Tester{
    public static int findReverse(int num, int temp){
            //Implement your code here and change the return value accordingly
            if(num == 0) return temp;
            else{
                int n = num % 10;
                temp = temp*10 + n;
                num /= 10;
                return findReverse(num, temp);
            }
        }
        public static void main(String args[]){
            int num = 12321;
        int reverseNum = findReverse(num,0);
        if(num == reverseNum)
            System.out.println(num +" is a palindrome!");
        else
            System.out.println(num +" is not a palindrome!");
        }
}
```

REGULAR_EXPRESSION

ASSIGNMENT1

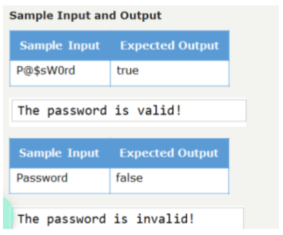
Problem Statement

Validate the password based on the below conditions using regular expression in the checkPasswordValidity() method.The length of the password should be between 8 and 20 characters (both inclusive).The password must contain minimum one lower case alphabet, one upper case alphabet, one digit and one special character.The password can contain only the following special characters.

!, @, #, $, %, &, *, _

Return true if the password is valid, else return false.

Test the functionalities using the main method of the Tester class



Sample Input and Output

Sample Input	Expected Output
P@$sW0rd	true

```
The password is valid!
```

Sample Input	Expected Output
Password	false

```
The password is invalid!
```

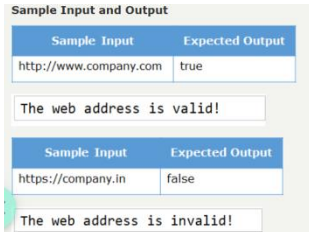
```
import java.util.regex.*;
class Tester {
    public static boolean checkPasswordValidity(String password) {
    String regex = "^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d)(?=.*[!@#$%&*_])[a-zA-Z\\d!@#$%&*_]{8,20}$";
    Pattern pattern = Pattern.compile(regex);
```

```
        Matcher matcher = pattern.matcher(password);

        return matcher.matches();

    }

    public static void main(String[] args) {

        String password = "P@$sW0rd";

        System.out.println("The password is " + (checkPasswordValidity(password) ? "valid!" : "invalid!"));

    }

}
```

ASSIGNMENT 2:

Validate the web address based on the below conditions using regular expression in the checkWebAddressValidity() method.The web address should start with http or httpshttp or https should be followed by ://.The web address may have 'www.'.The web address can be alphanumeric.After . (except the . after www), the web address should contain either "com" or "org" or "net"

Return true if the web address is valid, else return false.Test the functionalities using the main method of the Tester class.



Sample Input and Output

Sample Input	Expected Output
http://www.company.com	true

```
The web address is valid!
```

Sample Input	Expected Output
https://company.in	false

```
The web address is invalid!
```

```
class Tester {

    public static boolean checkWebAddressValidity(String webAddress) {

        //Implement your code here and change the return value accordingly

        String regex = "^(http|https):\\/\\/(www\\.)?[a-zA-Z0-9]+\\.(com|org|net)";

        return webAddress.matches(regex);

    }

    public static void main(String[] args) {

        //Change the value of webAddress for testing your code with different web addresses

        String webAddress = "http://www.company.com";

        System.out.println("The web address is "+ (checkWebAddressValidity(webAddress) ? "valid!" : "invalid!"));

    }

}
```

ASSIGNMENT 3:

Specific rules are followed by a company to set the product name, product Id and shipment tracker Id.

You need to validate them using regular expressions as mentioned below in the Shipment class.

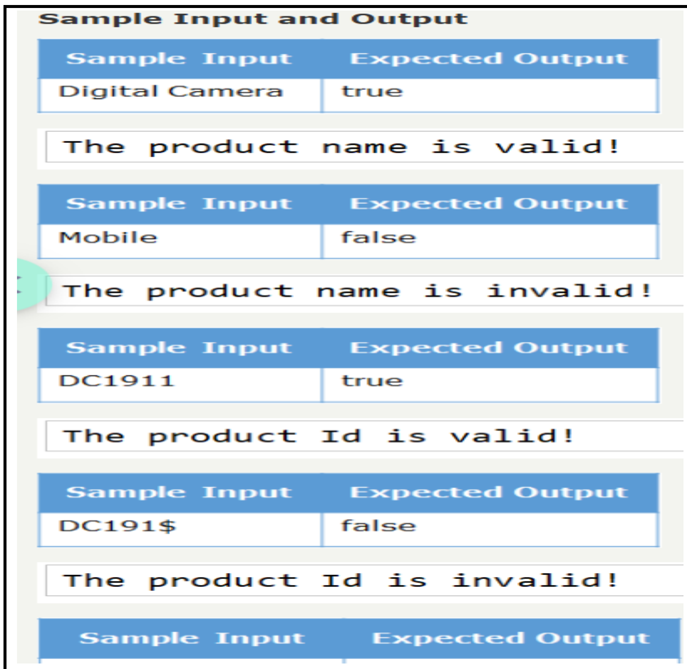
Validate the product name in the checkProductNameValidity() method based on the below conditions.Product name should contain minimum 2 words and maximum 3 words separated by spaces.The words in the product name should contain only alphabets.Return true if the product name is valid, else return false.Validate the product Id in the checkProductIdValidity() method based on the below conditions.The length of the product Id should be between 2 and 20 (both inclusive).The product Id can be alphanumeric.Return true if the product Id is valid, else return false.

Validate the shipment tracker id in the checkTrackerIdValidity() method based on the below conditions.

- **Shipment tracker Id should be in the below format**

<<1 upper case alphabet >>:<<4 upper case alphabets>>:<<3 lower case alphabets >>:<<2 digits>>

Return true if the shipment tracker Id is valid, else return false.



Sample Input and Output

Sample Input	Expected Output
Digital Camera	true

```
The product name is valid!
```

Sample Input	Expected Output
Mobile	false

```
The product name is invalid!
```

Sample Input	Expected Output
DC1911	true

```
The product Id is valid!
```

Sample Input	Expected Output
DC191$	false

```
The product Id is invalid!
```

Sample Input	Expected Output

```
class Shipment {
    public boolean checkProductNameValidity(String productName) {
        String regex = "^[A-Za-z]+( [A-Za-z]+){1,2}$";
        return productName.matches(regex);
    }
    public boolean checkProductIdValidity(String productId) {
        String regex = "^[a-zA-Z0-9]{2,20}$";
        return productId.matches(regex);
    }
    public boolean checkTrackerIdValidity(String trackerId) {
        String regex = "^[A-Z]:[A-Z]{4}:[a-z]{3}:[0-9]{2}$";
        return trackerId.matches(regex);
    }
}
class Tester {
    public static void main(String[] args) {
        Shipment shipment = new Shipment();
        String productName = "Digital Camera";
        System.out.println("The product name is " + (shipment.checkProductNameValidity(productName) ? "valid!" : "invalid!"));
        String productId = "DC1911";
        System.out.println("The product Id is " + (shipment.checkProductIdValidity(productId) ? "valid!" : "invalid!"));
        String trackerId = "D:CMDC:cmd:23";
        System.out.println("The tracker Id is " + (shipment.checkTrackerIdValidity(trackerId) ? "valid!" : "invalid!"));
    }
}
```