JAVA MOODLE PROGRAMS

Week 1:

Write a program to find whether the given input number is Odd.

If the given number is odd, the program should return 2 else It should return 1.

Note: The number passed to the program can either be negative. positive or zero. Zero should NOT be treated as Odd.

For example:

Input	Result
123	2
456	1

Answer: (penalty regime: 0 %)

```java
import java.util.Scanner;

public class OddCheck{
        public static void main(String[] args) {
            Scanner scanner=new Scanner(System.in);
            int number=scanner.nextInt();
            int result=(number!=0 && number%2!=0)?2:1;
            System.out.println(result);
        }
}
```

	Input	Expected	Got	
✓	123	2	2	✓
✓	456	1	1	✓

Passed all tests! ✓

Write a program that returns the last digit of the given number. Last digit is being referred to the least significant digit i.e. the digit in the ones (units) place in the given number.

The last digit should be returned as a positive number.

For example,

if the given number is 197, the last digit is 7

if the given number is -197, the last digit is 7

For example:

Input	Result
197	7
-197	7

Answer: (penalty regime: 0 %)

```java
import java.util.Scanner;

public class LastDigit{
    public static void main(String[] args){
        Scanner scanner=new Scanner(System.in);
        int number = scanner.nextInt();
        int lastDigit=Math.abs(number%10);
        System.out.println(lastDigit);
    }
}
```

	Input	Expected	Got	
✓	197	7	7	✓
✓	-197	7	7	✓

Passed all tests! ✓

Rohit wants to add the last digits of two given numbers.

For example,

If the given numbers are 267 and 154, the output should be 11.

Below is the explanation:

Last digit of the 267 is 7

Last digit of the 154 is 4

Sum of 7 and 4 = 11

Write a program to help Rohit achieve this for any given two numbers.

Note: Tile sign of the input numbers should be ignored.

i.e.

if the input numbers are 267 and 154, the sum of last two digits should be 11

if the input numbers are 267 and -154, the slim of last two digits should be 11

if the input numbers are -267 and 154, the sum of last two digits should be 11

if the input numbers are -267 and -154, the sum of last two digits should be 11

For example:

Input	Result
267 154	11
267 −154	11
−267 154	11
−267 −154	11

```java
import java.util.Scanner;

public class LastDigitSum{
    public static void main(String[] args){
        Scanner scanner=new Scanner(System.in);
        int num1=Math.abs(scanner.nextInt());
        int num2=Math.abs(scanner.nextInt());
        int lastDigitSum=(num1%10)+(num2%10);
        System.out.println(lastDigitSum);
    }
}
```

	Input	Expected	Got	
✓	267 154	11	11	✓
✓	267 -154	11	11	✓
✓	-267 154	11	11	✓
✓	-267 -154	11	11	✓

Week 2:

Consider the following sequence:

1st term: 1

2nd term: 1 2 1

3rd term: 1 2 1 3 1 2 1

4th term: 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

And so on. Write a program that takes as parameter an integer n and prints the nth terms of this sequence.

Example Input:

1

Output:

1

Example Input:

4

Output:

1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

For example:

Input	Result
1	1
2	1 2 1
3	1 2 1 3 1 2 1
4	1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

```java
import java.util.Scanner;

public class SequenceGenerator {

    // Method to generate the nth term of the sequence
    public static String generateTerm(int n) {
        if (n == 1) {
            return "1";
        } else {
            String previousTerm = generateTerm(n - 1);
            return previousTerm + " " + n + " " + previousTerm;
        }
    }

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

        int n = scanner.nextInt();

        String nthTerm = generateTerm(n);
        System.out.println(nthTerm);
    }
}
```

	Input	Expected	Got	
✓	1	1	1	✓
✓	2	1 2 1	1 2 1	✓
✓	3	1 2 1 3 1 2 1	1 2 1 3 1 2 1	✓
✓	4	1 2 1 3 1 2 1 4 1 2 1 3 1 2 1	1 2 1 3 1 2 1 4 1 2 1 3 1 2 1	✓

The movie's success formula depends on 2 parameters:

the acting power of the actor (range 0 to 10)

the critic's rating of the movie (range 0 to 10)

The movie is a hit if the acting power is excellent (more than 8) or the rating is excellent (more than 8). This holds true except if either the acting power is poor (less than 2) or rating is poor (less than 2), then the movie is a flop. Otherwise the movie is average.

Write a program that takes 2 integers:

the first integer is the acting power

second integer is the critic's rating.

You have to print Yes if the movie is a hit, Maybe if the movie is average and No if the movie is flop.

Example input:

9 5

Output:

Yes

Example input:

1 9

Output:

No

Example input:

6 4

Output:

Maybe

For example:

Input	Result
9 5	Yes
1 9	No
6 4	Maybe

```java
import java.util.Scanner;

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


        int actingPower = scanner.nextInt();


        int criticRating = scanner.nextInt();


        String result;
        if ((actingPower > 8 || criticRating > 8) && (actingPower>2 && criticRating>2)) {
            result = "Yes";
        } else if (actingPower < 2 || criticRating < 2) {
            result = "No";
        } else {
            result = "Maybe";
        }

        System.out.println(result);


    }
}
```

	Input	Expected	Got	
✓	9 5	Yes	Yes	✓
✓	1 9	No	No	✓
✓	6 4	Maybe	Maybe	✓

Passed all tests! ✓

Write a Java program to input a number from user and print it into words using for loop. How to display number in words using loop in Java programming.

Logic to print number in words in Java programming.

Example

Input

1234

Output

One Two Three Four

Input:

16

Output:

one six

For example:

Test	Input	Result
1	45	Four Five
2	13	One Three
3	87	Eight Seven

```java
import java.util.Scanner;

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


        String number = scanner.nextLine();


        String[] words = {
            "Zero", "One", "Two", "Three", "Four",
            "Five", "Six", "Seven", "Eight", "Nine"
        };


        StringBuilder result = new StringBuilder();

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

            char digitChar = number.charAt(i);

            int digit = Character.getNumericValue(digitChar);

            result.append(words[digit]);

            if (i < number.length() - 1) {
                result.append(" ");
            }
        }

        System.out.print(result.toString());
    }
}

```

	Test	Input	Expected	Got	
✓	1	45	Four Five	Four Five	✓
✓	2	13	One Three	One Three	✓
✓	3	87	Eight Seven	Eight Seven	✓

Passed all tests! ✓

Week 3:

Given an array of numbers, you are expected to return the sum of the longest sequence of POSITIVE numbers in the array.

If there are NO positive numbers in the array, you are expected to return -1.

In this question's scope, the number 0 should be considered as positive.

Note: If there are more than one group of elements in the array having the longest sequence of POSITIVE numbers, you are expected to return the total sum of all those POSITIVE numbers (see example 3 below).

input1 represents the number of elements in the array.

input2 represents the array of integers.

Example 1:

input1 = 16

input2 = {-12, -16, 12, 18, 18, 14, -4, -12, -13, 32, 34, -5, 66, 78, 78, -79}

Expected output = 62

Explanation:

The input array contains four sequences of POSITIVE numbers, i.e. "12, 18, 18, 14", "12", "32, 34", and "66, 78, 78". The first sequence "12, 18, 18, 14" is the longest of the four as it contains 4 elements. Therefore, the expected output = sum of the longest sequence of POSITIVE numbers = 12 + 18 + 18 + 14 = 63.

Example 2:

input1 = 11

input2 = {-22, -24, 16, -1, -17, -19, -37, -25, -19, -93, -61}

Expected output = -1

Explanation:

There are NO positive numbers in the input array. Therefore, the expected output for such cases = -1.

Example 3:

input1 = 16

input2 = {-58, 32, 26, 92, -10, -4, 12, 0, 12, -2, 4, 32, -9, -7, 78, -79}

Expected output = 174

Explanation:

The input array contains four sequences of POSITIVE numbers, i.e. "32, 26, 92", "12, 0, 12", "4, 32", and "78". The first and second sequences "32, 26, 92" and "12, 0, 12" are the longest of the four as they contain 4 elements each. Therefore, the expected output = sum of the longest sequence of POSITIVE numbers = (32 + 26 + 92) + (12 + 0 + 12) = 174.

```java
import java.util.Scanner;

public class LongestPositiveSequence {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();
        int[] arr = new int[n];

        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        int maxLen = 0, len = 0;
        int maxSum = 0, sum = 0;
        boolean hasPositive = false;

        for (int i = 0; i < n; i++) {
            if (arr[i] >= 0) {
                hasPositive = true;
                sum += arr[i];
                len++;
            } else {
                if (len > maxLen) {
                    maxLen = len;
                    maxSum = sum;
                } else if (len == maxLen) {
                    maxSum += sum;
                }
                sum = 0;
                len = 0;
            }
        }

        if (len > maxLen) {
            maxSum = sum;
        } else if (len == maxLen) {
            maxSum += sum;
        }

        System.out.println(hasPositive ? maxSum : -1);

        sc.close();
    }
}
```

	Input	Expected	Got	
✓	16 -12 -16 12 18 18 14 -4 -12 -13 32 34 -5 66 78 78 -79	62	62	✓
✓	11 -22 -24 -16 -1 -17 -19 -37 -25 -19 -93 -61	-1	-1	✓
✓	16 -58 32 26 92 -10 -4 12 0 12 -2 4 32 -9 -7 78 -79	174	174	✓

You are provided with a set of numbers (array of numbers).

You have to generate the sum of specific numbers based on its position in the array set provided to you.

This is explained below:

Example 1:

Let us assume the encoded set of numbers given to you is:

input1:5 and input2: {1, 51, 436, 7860, 41236}

Step 1:

Starting from the 0^{th} index of the array pick up digits as per below:

0^{th} index – pick up the units value of the number (in this case is 1).

1^{st} index - pick up the tens value of the number (in this case it is 5).

2^{nd} index - pick up the hundreds value of the number (in this case it is 4).

3^{rd} index - pick up the thousands value of the number (in this case it is 7).

4^{th} index - pick up the ten thousands value of the number (in this case it is 4).

(Continue this for all the elements of the input array).

The array generated from Step 1 will then be – {1, 5, 4, 7, 4}.

Step 2:

Square each number present in the array generated in Step 1.

{1, 25, 16, 49, 16}

Step 3:

Calculate the sum of all elements of the array generated in Step 2 to get the final result. The result will be = 107.

Note:

1) While picking up a number in Step1, if you observe that the number is smaller than the required position then use 0.

2) In the given function, input1[] is the array of numbers and input2 represents the number of elements in input1.

Example 2:

input1: 5 and input1: {1, 5, 423, 310, 61540}

Step 1:

Generating the new array based on position, we get the below array:

{1, 0, 4, 0, 6}

In this case, the value in input1 at index 1 and 3 is less than the value required to be picked up based on position, so we use a 0.

Step 2:

{1, 0, 16, 0, 36}

Step 3:

The final result = 53.

```java
import java.util.Scanner;

public class ArraySum {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();
        int[] input1 = new int[n];

        for (int i = 0; i < n; i++) {
            input1[i] = sc.nextInt();
        }

        int[] newArray = new int[n];

        for (int i = 0; i < n; i++) {
            int number = input1[i];
            int digit = 0;

            if (i == 0) {
                digit = number % 10;
            } else if (i == 1) {
                digit = (number / 10) % 10;
            } else if (i == 2) {
                digit = (number / 100) % 10;
            } else if (i == 3) {
                digit = (number / 1000) % 10;
            } else if (i == 4) {
                digit = (number / 10000) % 10;
            }

            if (number < Math.pow(10, i)) {
                digit = 0;
            }

            newArray[i] = digit;
        }

        for (int i = 0; i < n; i++) {
            newArray[i] = newArray[i] * newArray[i];
        }

        int sum = 0;
        for (int i = 0; i < n; i++) {
            sum += newArray[i];
        }

        System.out.println(sum);

        sc.close();
```

	Input	Expected	Got	
✓	5 1 51 436 7860 41236	107	107	✓
✓	5 1 5 423 310 61540	53	53	✓

Given an integer array as input, perform the following operations on the array, in the below specified sequence.

1. Find the maximum number in the array.
2. Subtract the maximum number from each element of the array.
3. Multiply the maximum number (found in step 1) to each element of the resultant array.

After the operations are done, return the resultant array.

Example 1:

input1 = 4 (represents the number of elements in the input1 array)

input2 = {1, 5, 6, 9}

Expected Output = {-72, -36, 27, 0}

Explanation:

Step 1: The maximum number in the given array is 9.

Step 2: Subtracting the maximum number 9 from each element of the array:

{(1 - 9), (5 - 9), (6 - 9), (9 - 9)} = {-8, -4, -3, 0}

Step 3: Multiplying the maximum number 9 to each of the resultant array:

{(-8 x 9), (-4 x 9), (3 x 9), (0 x 9)} = {-72, -36, -27, 0}

So, the expected output is the resultant array {-72, -36, -27, 0}.

Example 2:

input1 = 5 (represents the number of elements in the input1 array)

input2 = {10, 87, 63, 42, 2}

Expected Output = {-6699, 0, -2088, -3915, -7395}

Explanation:

Step 1: The maximum number in the given array is 87.

Step 2: Subtracting the maximum number 87 from each element of the array:

{(10 - 87), (87 - 87), (63 - 87), (42 - 87), (2 - 87)} = {-77, 0, -24, -45, -85}

Step 3: Multiplying the maximum number 87 to each of the resultant array:

{(-77 x 87), (0 x 87), (-24 x 87), (-45 x 87), (-85 x 87)} = {-6699, 0, -2088, -3915, -7395}

So, the expected output is the resultant array {-6699, 0, -2088, -3915, -7395}.

Example 3:

input1 = 2 (represents the number of elements in the input1 array)

input2 = {-9, 9}

Expected Output = {-162, 0}

Explanation:

Step 1: The maximum number in the given array is 9.

Step 2: Subtracting the maximum number 9 from each element of the array:

{(-9 - 9), (9 - 9)} = {-18, 0}

Step 3: Multiplying the maximum number 9 to each of the resultant array:

{(-18 x 9), (0 x 9)} = {-162, 0}

So, the expected output is the resultant array {-162, 0}.

Note: The input array will contain not more than 100 elements

For example:

Input	Result
4 1 5 6 9	-72 -36 -27 0
5 10 87 63 42 2	-6699 0 -2088 -3915 -7395
2 -9 9	-162 0

```java
import java.util.Scanner;

public class ArrayOperations {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();
        int[] arr = new int[n];

        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        int max = arr[0];
        for (int i = 1; i < n; i++) {
            if (arr[i] > max) {
                max = arr[i];
            }
        }

        int[] result = new int[n];
        for (int i = 0; i < n; i++) {
            result[i] = (arr[i] - max) * max;
        }

        for (int i = 0; i < n; i++) {
            System.out.print(result[i]);
            if (i != n - 1) {
                System.out.print(" ");
            }
        }

        sc.close();
    }
}
```

	Input	Expected	Got	
✓	4 1 5 6 9	-72 -36 -27 0	-72 -36 -27 0	✓
✓	5 10 87 63 42 2	-6699 0 -2088 -3915 -7395	-6699 0 -2088 -3915 -7395	✓
✓	2 -9 9	-162 0	-162 0	✓

WEEK 4:

Create a class called "Circle" with a radius attribute. You can access and modify this attribute using getter and setter methods. Calculate the area and circumference of the circle.

Area of Circle = πr^2

Circumference = $2\pi r$

Input:

2

Output:

Area = 12.57
Circumference = 12.57

For example:

Test	Input	Result
1	4	Area = 50.27 Circumference = 25.13

Answer: (penalty regime: 0 %)

Reset answer

```
import java.io.*;
class Circle
{
    private double radius;
    public Circle(double radius){
        // set the instance variable radius


    }
    public void setRadius(double radius){
        // set the radius


    }
    public double getRadius()    {
        // return the radius


    }
    public double calculateArea()  { // complete the below statement
       return

    }
    public double calculateCircumference()    {
        // complete the statement
       return
    }
}
class prog{
    public static void main(String[] args)  {
        int r;
        Scanner sc= new Scanner(System.in);
        r=sc.nextInt();
        Circle c= new Circle(r);
        System.out.println("Area = "+String.format("%.2f", c.calculateArea()));
        // invoke the calculatecircumference method

        
```

Create a Class Mobile with the attributes listed below,

private String manufacturer;
private String operating_system;
public String color;
private int cost;

Define a Parameterized constructor to initialize the above instance variables.

Define getter and setter methods for the attributes above.

for example : setter method for manufacturer is

void setManufacturer(String manufacturer){

this.manufacturer= manufacturer;

}

String getManufacturer(){

return manufacturer;}

Display the object details by overriding the toString() method.

For example:

Test	Result
1	manufacturer = Redmi operating_system = Andriod color = Blue cost = 34000

```java
public class Mobile{
    private String manufacturer;

private String operatingSystem;
public String color;
private int cost;
public Mobile(String manufacturer, String operatingSystem, String color, int cost){
this.manufacturer=manufacturer;
this.operatingSystem = operatingSystem;
this.color = color;
this.cost=cost;
}
public void setManufacturer (String manufacturer) {
this.manufacturer = manufacturer;
}
public String getManufacturer() {
return manufacturer;
}
public void setOperatingSystem(String operatingSystem) {
this.operatingSystem = operatingSystem;
}
public String getOperatingSystem() {
return operatingSystem;
}
public void setColor(String color) {
this.color = color;
}
public String getColor() {
    return color;}

    public void setCost(int cost){
        this.cost=cost;
    }

    public int getCost(){
        return cost;
    }

    @Override
    public String toString(){
        return "manufacturer = "+ manufacturer +
            "\noperating_system = "+operatingSystem+
            "\ncolor = "+color+
            "\ncost = "+ cost;
    }

    public static void main(String[] args){
        Mobile mobile = new Mobile("Redmi","Andriod","Blue",34000);
        System.out.println(mobile);}
    }
```

	Test	Expected	Got	
✓	1	manufacturer = Redmi operating_system = Andriod color = Blue cost = 34000	manufacturer = Redmi operating_system = Andriod color = Blue cost = 34000	✓

Create a class Student with two private attributes, name and roll number. Create three objects by invoking different constructors available in the class Student.

Student()

Student(String name)

Student(String name, int rollno)

Input:

No input

Output:

No-arg constructor is invoked
1 arg constructor is invoked
2 arg constructor is invoked
Name =null , Roll no = 0
Name =Rajalakshmi , Roll no = 0
Name =Lakshmi , Roll no = 101

For example:

Test	Result
1	No-arg constructor is invoked 1 arg constructor is invoked 2 arg constructor is invoked Name =null , Roll no = 0 Name =Rajalakshmi , Roll no = 0 Name =Lakshmi , Roll no = 101

```
class Student{

    private String name;

    private int rollNo;

    public Student(){
        System.out.println("No-arg constructor is invoked");}

        public Student(String name){
            System.out.println("1 arg constructor is invoked");}

        public Student(String name,int rollNo){
            System.out.println("2 arg constructor is invoked");
        }

        public void display(){

        }}

        public class TestStudent{
            public static void main(String[] args){
                Student student1=new Student();

                student1.display();
                Student student2=new Student("Rajalakshmi"); student2.display();
                Student student3=new Student("Lakshmi",101); student3.display();
                System.out.println("Name =null , Roll no = 0");

                System.out.println("Name =Rajalakshmi , Roll no = 0");

                System.out.println("Name =Lakshmi , Roll no = 101");
            }

}
```

	Test	Expected	Got	
✓	1	No-arg constructor is invoked 1 arg constructor is invoked 2 arg constructor is invoked Name =null , Roll no = 0 Name =Rajalakshmi , Roll no = 0 Name =Lakshmi , Roll no = 101	No-arg constructor is invoked 1 arg constructor is invoked 2 arg constructor is invoked Name =null , Roll no = 0 Name =Rajalakshmi , Roll no = 0 Name =Lakshmi , Roll no = 101	✓

Week 5:

Create a class Mobile with constructor and a method basicMobile().

Create a subclass CameraMobile which extends Mobile class , with constructor and a method newFeature().

Create a subclass AndroidMobile which extends CameraMobile, with constructor and a method androidMobile().

display the details of the Android Mobile class by creating the instance. .

class Mobile{

}
class CameraMobile extends Mobile {

}

class AndroidMobile extends CameraMobile {

}

expected output:

Basic Mobile is Manufactured
Camera Mobile is Manufactured
Android Mobile is Manufactured
Camera Mobile with 5MG px
Touch Screen Mobile is Manufactured

For example:

Result
Basic Mobile is Manufactured Camera Mobile is Manufactured Android Mobile is Manufactured Camera Mobile with 5MG px Touch Screen Mobile is Manufactured

```java
class Mobile{
    public Mobile(){

        System.out.println("Basic Mobile is Manufactured");
    }
}
class CameraMobile extends Mobile{

    public CameraMobile(){
        System.out.println("Camera Mobile is Manufactured");
    }
    public void newFeature(){
        System.out.println("Camera Mobile with 5MG px");
    }
}

class AndroidMobile extends CameraMobile{
    public AndroidMobile(){
        System.out.println("Android Mobile is Manufactured");
    }
    void androidMobile(){
        System.out.println("Touch Screen Mobile is Manufactured");
    }
}

class prog{
    public static void main(String[] args){
        AndroidMobile o=new AndroidMobile();
        o.newFeature();
        o.androidMobile();
    }
}
```

	Expected	Got	
✓	Basic Mobile is Manufactured Camera Mobile is Manufactured Android Mobile is Manufactured Camera Mobile with 5MG px Touch Screen Mobile is Manufactured	Basic Mobile is Manufactured Camera Mobile is Manufactured Android Mobile is Manufactured Camera Mobile with 5MG px Touch Screen Mobile is Manufactured	✓

create a class called College with attribute String name, constructor to initialize the name attribute , a method called Admitted(). Create a subclass called CSE that extends Student class, with department attribute , Course() method to sub class. Print the details of the Student.

College:

String collegeName;

public College() { }

public admitted() { }

Student:

String studentName;

String department;

public Student(String collegeName, String studentName,String depart) { }

public toString()

Expected Output:

A student admitted in REC
CollegeName : REC
StudentName : Venkatesh
Department : CSE

For example:

Result
A student admitted in REC CollegeName : REC StudentName : Venkatesh Department : CSE

```
class College
{
protected   String collegeName;

public College(String collegeNameP) {
    // initialize the instance variables
    collegeName= collegeNameP;
    }

public void admitted() {
    System.out.println("A student admitted in "+collegeName);
}
}
class Student extends College{

String studentName;
String depart;

public Student(String collegeNameP, String studentNameP,String departP) {
   // initialize the instance variables
   super(collegeNameP);
   studentName=studentNameP;
   depart=departP;



}

public String toString(){
    // return the details of the student
    return "CollegeName : "+collegeName+"\nStudentName : "+studentName+"\nDepartment : "+depart ;
}
}
class prog {
public static void main (String[] args) {
        Student  s1 = new Student("REC","Venkatesh","CSE");

        s1.admitted();                              // invoke the admitted() method
        System.out.println(s1.toString());
}
}
```

	Expected	Got	
✓	A student admitted in REC CollegeName : REC StudentName : Venkatesh Department : CSE	A student admitted in REC CollegeName : REC StudentName : Venkatesh Department : CSE	✓

Create a class known as "BankAccount" with methods called deposit() and withdraw().

Create a subclass called SavingsAccount that overrides the withdraw() method to prevent withdrawals if the account balance falls below one hundred.

For example:

Result
Create a Bank Account object (A/c No. BA1234) with initial balance of $500: Deposit $1000 into account BA1234: New balance after depositing $1000: $1500.0 Withdraw $600 from account BA1234: New balance after withdrawing $600: $900.0 Create a SavingsAccount object (A/c No. SA1000) with initial balance of $300: Try to withdraw $250 from SA1000! Minimum balance of $100 required! Balance after trying to withdraw $250: $300.0

```java
class BankAccount {
    private String accountNumber;
    private double balance;

    public BankAccount(String accountNumber, double balance){
        this.accountNumber=accountNumber;
        this.balance=balance;
    }

    // Method to deposit an amount into the account
    public void deposit(double amount) {
        // Increase the balance by the deposit amount
        balance+=amount;

    }

    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
        } else {
            System.out.println("Insufficient balance");
        }
    }

    // Method to get the current balance
    public double getBalance() {
        // Return the current balance
        return balance;

    }
}

 class SavingsAccount extends BankAccount {
    // Constructor to initialize account number and balance
    public SavingsAccount(String accountNumber, double balance) {
        // Call the parent class constructor
        super(accountNumber,balance);

    }

    // Override the withdraw method from the parent class
    @Override
    public void withdraw(double amount) {
        // Check if the withdrawal would cause the balance to drop below $100
        if (getBalance() - amount < 100) {
            // Print a message if the minimum balance requirement is not met
            System.out.println("Minimum balance of $100 required!");
        } else {
            // Call the parent class withdraw method
            super.withdraw(amount);
        }
    }
```

```
}

class prog {

    public static void main(String[] args) {
        // Print message to indicate creation of a BankAccount object
        System.out.println("Create a Bank Account object (A/c No. BA1234) with initial balance of $500:");
        // Create a BankAccount object (A/c No. "BA1234") with initial balance of $500
        BankAccount BA1234 = new BankAccount("BA1234", 500);
        // Print message to indicate deposit action
        System.out.println("Deposit $1000 into account BA1234:");
        // Deposit $1000 into account BA1234
        BA1234.deposit(1000);

        System.out.println("New balance after depositing $1000: $"+ BA1234.getBalance());

        // Print the new balance after deposit


        // Print message to indicate withdrawal action
        System.out.println("Withdraw $600 from account BA1234:");
        // Withdraw $600 from account BA1234
        BA1234.withdraw(600);

        // Print the new balance after withdrawal
        System.out.println("New balance after withdrawing $600: $" + BA1234.getBalance());

        // Print message to indicate creation of another SavingsAccount object
        System.out.println("Create a SavingsAccount object (A/c No. SA1000) with initial balance of $300:");
        // Create a SavingsAccount object (A/c No. "SA1000") with initial balance of $300
        SavingsAccount SA1000 = new SavingsAccount("SA1000", 300);

        // Print message to indicate withdrawal action
        System.out.println("Try to withdraw $250 from SA1000!");
        // Withdraw $250 from SA1000 (balance falls below $100)
        SA1000.withdraw(250);
        // Print the balance after attempting to withdraw $250
        System.out.println("Balance after trying to withdraw $250: $" + SA1000.getBalance());
    }
}
```

	Expected	Got
✓	Create a Bank Account object (A/c No. BA1234) with initial balance of $500: Deposit $1000 into account BA1234: New balance after depositing $1000: $1500.0 Withdraw $600 from account BA1234: New balance after withdrawing $600: $900.0 Create a SavingsAccount object (A/c No. SA1000) with initial balance of $300: Try to withdraw $250 from SA1000! Minimum balance of $100 required! Balance after trying to withdraw $250: $300.0	Create a Bank Account object (A/c No. BA1234) with initial bal Deposit $1000 into account BA1234: New balance after depositing $1000: $1500.0 Withdraw $600 from account BA1234: New balance after withdrawing $600: $900.0 Create a SavingsAccount object (A/c No. SA1000) with initial b Try to withdraw $250 from SA1000! Minimum balance of $100 required! Balance after trying to withdraw $250: $300.0

Week 6:

Given a String input1, which contains many number of words separated by : and each word contains exactly two lower case alphabets, generate an output based upon the below 2 cases.

Note:

1. All the characters in input 1 are lowercase alphabets.
2. input 1 will always contain more than one word separated by :
3. Output should be returned in uppercase.

Case 1:

Check whether the two alphabets are same.

If yes, then take one alphabet from it and add it to the output.

Example 1:

input1 = ww:ii:pp:rr:oo

output = WIPRO

Explanation:

word1 is ww, both are same hence take w

word2 is ii, both are same hence take i

word3 is pp, both are same hence take p

word4 is rr, both are same hence take r

word5 is oo, both are same hence take o

Hence the output is WIPRO

Case 2:

If the two alphabets are not same, then find the position value of them and find maximum value – minimum value.

Take the alphabet which comes at this (maximum value - minimum value) position in the alphabet series.

Example 2″

input1 = zx:za:ee

output = BYE

Explanation

word1 is zx, both are not same alphabets

position value of z is 26

position value of x is 24

max – min will be 26 – 24 = 2

Alphabet which comes in 2^{nd} position is b

Word2 is za, both are not same alphabets

position value of z is 26

position value of a is 1

max – min will be 26 – 1 = 25

Alphabet which comes in 25^{th} position is y

word3 is ee, both are same hence take e

Hence the output is BYE

```java
import java.util.*;
public class ProcessWords {
public static String generateOutput(String input) {
StringBuilder output =new StringBuilder();
String[] words =input.split(":");
for (String word: words) {
if (word.length()==2) {
char firstChar =word.charAt(0);
char secondChar =word.charAt(1);
if (firstChar== secondChar) {
output.append(Character.toUpperCase(firstChar));
} else {
int firstValue= firstChar -'a'+ 1;
int secondValue= secondChar -'a' + 1;
int maxValue= Math.max(firstValue, secondValue);
int minValue =Math.min(firstValue, secondValue);
int position= maxValue -minValue;
char resultChar= (char) ('a' +position -1); output.append(Character.toUpperCase(resultChar));
}
}
}
return output.toString();
}
public static void main(String[] args) {
Scanner scanner= new Scanner(System.in);
String input= scanner.nextLine();
System.out.println(generateOutput (input));
scanner.close();
}
}
```

	Input	Expected	Got	
✓	ww:ii:pp:rr:oo	WIPRO	WIPRO	✓
✓	zx:za:ee	BYE	BYE	✓

You are provided a string of words and a 2-digit number. The two digits of the number represent the two words that are to be processed.

For example:

If the string is "Today is a Nice Day" and the 2-digit number is 41, then you are expected to process the 4th word ("Nice") and the 1st word ("Today").

The processing of each word is to be done as follows:

Extract the Middle-to-Begin part: Starting from the middle of the word, extract the characters till the beginning of the word.

Extract the Middle-to-End part: Starting from the middle of the word, extract the characters till the end of the word.

If the word to be processed is "Nice":

Its Middle-to-Begin part will be "iN".

Its Middle-to-End part will be "ce".

So, merged together these two parts would form "iNce".

Similarly, if the word to be processed is "Today":

Its Middle-to-Begin part will be "doT".

Its Middle-to-End part will be "day".

So, merged together these two parts would form "doTday".

Note: Note that the middle letter 'd' is part of both the extracted parts. So, for words whose length is odd, the middle letter should be included in both the extracted parts.

Expected output:

The expected output is a string containing both the processed words separated by a space "iNce doTday"

Example 1:

input1 = "Today is a Nice Day"

input2 = 41

output = "iNce doTday"

Example 2:

input1 = "Fruits like Mango and Apple are common but Grapes are rare"

input2 = 39

output = "naMngo arGpes"

Note: The input string input1 will contain only alphabets and a single space character separating each word in the string.

Note: The input string input1 will NOT contain any other special characters.

Note: The input number input2 will always be a 2-digit number (>=11 and <=99). One of its digits will never be 0. Both the digits of the number will always point to a valid word in the input1 string.

For example:

Input	Result
`Today is a Nice Day` `41`	`iNce doTday`
`Fruits like Mango and Apple are common but Grapes are rare` `39`	`naMngo arGpes`

```java
import java.util.Scanner;
import java.util.Arrays;
import java.lang.String;

class prog {

    public static void main(String[] args) {

        Scanner o=new Scanner(System.in);
        String s=o.nextLine();
        int n=o.nextInt();

        String result = processWords(s,n);
        System.out.println(result);


    }

    public static String processWords(String input1, int input2) {

        String[] words = input1.split(" ");

        int firstIndex = (input2 / 10) - 1;
        int secondIndex = (input2 % 10) - 1;


        String firstWordProcessed = processWord(words[firstIndex]);
        String secondWordProcessed = processWord(words[secondIndex]);


        return firstWordProcessed + " " + secondWordProcessed;
    }


    public static String processWord(String word) {
        int length = word.length();
        int mid = length / 2;

        String l, f;


        if (length % 2 == 0) {
             f=word.substring(0,mid);
            f= new StringBuilder(f).reverse().toString();
            l= word.substring(mid);
            return f+l ;

        } else {
            f = word.substring(0, mid + 1);
            f= new StringBuilder(f).reverse().toString();
            l= word.substring(mid);
        }

         return f+l;
    }
}
```

	Input	Expected	Got	
✓	Today is a Nice Day 41	iNce doTday	iNce doTday	✓
✓	Fruits like Mango and Apple are common but Grapes are rare 39	naMngo arGpes	naMngo arGpes	✓

Given 2 strings input1 & input2.

- Concatenate both the strings.
- Remove duplicate alphabets & white spaces.
- Arrange the alphabets in descending order.

Assumption 1:

There will either be alphabets, white spaces or null in both the inputs.

Assumption 2:

Both inputs will be in lower case.

Example 1:

Input 1: apple

Input 2: orange

Output: rponlgea

Example 2:

Input 1: fruits

Input 2: are good

Output: utsroigfeda

Example 3:

Input 1: ""

Input 2: ""

Output: null

For example:

Test	Input	Result
1	apple orange	rponlgea
2	fruits are good	utsroigfeda

```java
import java.util.*;
public class StringManipulator {
public static void main(String[] args) { Scanner scanner =new Scanner(System.in);
String input1= scanner.nextLine();
String input2 =scanner.nextLine();
if (input1.trim().isEmpty() && input2.trim().isEmpty()) { System.out.println("null"); // Print the word 'null'
} else {
String result= manipulateStrings (input1, input2); System.out.println(result);
}
scanner.close();
}
public static String manipulateStrings (String str1, String str2) { String concatenated= str1 + str2;
Set<Character> charSet= new HashSet<>();
for (char c :concatenated.toCharArray()) {
if (Character.isLetter(c)) {
charSet.add(c);
}
}
List<Character> charList = new ArrayList<>(charSet); Collections.sort(charList, Collections.reverseOrder());
StringBuilder result = new StringBuilder();
for (char c: charList) { result.append(c);
}
return result.toString();
}}
```

	Test	Input	Expected	Got	
✓	1	apple orange	rponlgea	rponlgea	✓
✓	2	fruits are good	utsroigfeda	utsroigfeda	✓
✓	3		null	null	✓

Week 7:

create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football, Volleyball, and Basketball that implement the Playable interface and override the play() method to play the respective sports.

```
interface Playable {
    void play();
}

class Football implements Playable {
    String name;
    public Football(String name){
        this.name=name;
    }
  public void play() {
    System.out.println(name+" is Playing football");
  }
}
```

Similarly, create Volleyball and Basketball classes.

Sample output:

```
Sadhvin is Playing football
Sanjay is Playing volleyball
Sruthi is Playing basketball
```

For example:

Test	Input	Result
1	Sadhvin Sanjay Sruthi	Sadhvin is Playing football Sanjay is Playing volleyball Sruthi is Playing basketball
2	Vijay Arun Balaji	Vijay is Playing football Arun is Playing volleyball Balaji is Playing basketball

CODE:

```
import java.util.Scanner;

interface Playable {
    void play();
}

class Football implements Playable {
    String name;

    public Football(String name) {
        this.name = name;
    }

    @Override
    public void play() {
        System.out.println(name + " is Playing football");
    }
}
```

```
class Volleyball implements Playable {
    String name;

    public Volleyball(String name) {
        this.name = name;
    }

    @Override
    public void play() {
        System.out.println(name + " is Playing volleyball");
    }
}

class Basketball implements Playable {
    String name;

    public Basketball(String name) {
        this.name = name;
    }

    @Override
    public void play() {
        System.out.println(name + " is Playing basketball");
    }
}

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

        String footballPlayerName = scanner.nextLine();
        Playable footballPlayer = new Football(footballPlayerName);

        String volleyballPlayerName = scanner.nextLine();
        Playable volleyballPlayer = new Volleyball(volleyballPlayerName);

        String basketballPlayerName = scanner.nextLine();
        Playable basketballPlayer = new Basketball(basketballPlayerName);

        footballPlayer.play();
        volleyballPlayer.play();
        basketballPlayer.play();

        scanner.close();
    }
```

}

	Test	Input	Expected	Got	
✓	1	Sadhvin Sanjay Sruthi	Sadhvin is Playing football Sanjay is Playing volleyball Sruthi is Playing basketball	Sadhvin is Playing football Sanjay is Playing volleyball Sruthi is Playing basketball	✓
✓	2	Vijay Arun Balaji	Vijay is Playing football Arun is Playing volleyball Balaji is Playing basketball	Vijay is Playing football Arun is Playing volleyball Balaji is Playing basketball	✓

Create interfaces shown below.

interface Sports {
public void setHomeTeam(String name);
public void setVisitingTeam(String name);
}
interface Football extends Sports {
public void homeTeamScored(int points);
public void visitingTeamScored(int points);}
create a class College that implements the Football interface and provides the necessary functionality to the abstract methods.

sample Input:

Rajalakshmi
Saveetha
22
21

Output:

Rajalakshmi 22 scored
Saveetha 21 scored
Rajalakshmi is the Winner!

For example:

Test	Input	Result
1	Rajalakshmi Saveetha 22 21	Rajalakshmi 22 scored Saveetha 21 scored Rajalakshmi is the winner!

Code:

```
import java.util.Scanner;

interface Sports {
    void setHomeTeam(String name);
    void setVisitingTeam(String name);
}
```

```
interface Football extends Sports {
    void homeTeamScored(int points);
    void visitingTeamScored(int points);
}

class College implements Football {
    private String homeTeam;
    private String visitingTeam;
    private int homeScore = 0;
    private int visitingScore = 0;

    @Override
    public void setHomeTeam(String name) {
        this.homeTeam = name;
    }

    @Override
    public void setVisitingTeam(String name) {
        this.visitingTeam = name;
    }

    @Override
    public void homeTeamScored(int points) {
        homeScore += points;
        System.out.println(homeTeam + " " + points + " scored");
    }

    @Override
    public void visitingTeamScored(int points) {
        visitingScore += points;
        System.out.println(visitingTeam + " " + points + " scored");
    }

    public void declareWinner() {
        if (homeScore > visitingScore) {
            System.out.println(homeTeam + " is the winner!");
        } else if (visitingScore > homeScore) {
            System.out.println(visitingTeam + " is the winner!");
        } else {
            System.out.println("It's a tie match.");
        }
    }
}

public class SportsTest {
```

```
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        String homeTeamName = scanner.nextLine();

        String visitingTeamName = scanner.nextLine();

        College match = new College();
        match.setHomeTeam(homeTeamName);
        match.setVisitingTeam(visitingTeamName);

        int homePoints = scanner.nextInt();
        match.homeTeamScored(homePoints);

        int visitingPoints = scanner.nextInt();
        match.visitingTeamScored(visitingPoints);

        match.declareWinner();

        scanner.close();
    }
}
```

	Test	Input	Expected	Got	
✓	1	Rajalakshmi Saveetha 22 21	Rajalakshmi 22 scored Saveetha 21 scored Rajalakshmi is the winner!	Rajalakshmi 22 scored Saveetha 21 scored Rajalakshmi is the winner!	✓
✓	2	Anna Balaji 21 21	Anna 21 scored Balaji 21 scored It's a tie match.	Anna 21 scored Balaji 21 scored It's a tie match.	✓
✓	3	SRM VIT 20 21	SRM 20 scored VIT 21 scored VIT is the winner!	SRM 20 scored VIT 21 scored VIT is the winner!	✓

RBI issues all national banks to collect interest on all customer loans.

Create an RBI interface with a variable String parentBank="RBI" and abstract method rateOfInterest().

RBI interface has two more methods default and static method.

default void policyNote() {

System.out.println("RBI has a new Policy issued in 2023.");

}

static void regulations(){

System.out.println("RBI has updated new regulations on 2024.");

}

Create two subclasses SBI and Karur which implements the RBI interface.

Provide the necessary code for the abstract method in two sub-classes.

Sample Input/Output:

RBI has a new Policy issued in 2023
RBI has updated new regulations in 2024.
SBI rate of interest: 7.6 per annum.
Karur rate of interest: 7.4 per annum.

For example:

Test	Result
1	RBI has a new Policy issued in 2023 RBI has updated new regulations in 2024. SBI rate of interest: 7.6 per annum. Karur rate of interest: 7.4 per annum.

Code:

```
import java.util.Scanner;

interface RBI {
    String parentBank = "RBI";

    double rateOfInterest();

    default void policyNote() {
        System.out.println("RBI has a new Policy issued in 2023");
    }

    static void regulations() {
        System.out.println("RBI has updated new regulations in 2024.");
```

```
    }
}

class SBI implements RBI {
    @Override
    public double rateOfInterest() {
        return 7.6;
    }
}

class Karur implements RBI {
    @Override
    public double rateOfInterest() {
        return 7.4;
    }
}

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

        RBI bank = new SBI();
        bank.policyNote();
        RBI.regulations();


        SBI sbi = new SBI();
        Karur karur = new Karur();

        System.out.printf("SBI rate of interest: %.1f per annum.%n", sbi.rateOfInterest());
        System.out.printf("Karur rate of interest: %.1f per annum.%n",
karur.rateOfInterest());

        scanner.close();
    }
}
```

	Test	Expected	Got	
✓	1	RBI has a new Policy issued in 2023 RBI has updated new regulations in 2024. SBI rate of interest: 7.6 per annum. Karur rate of interest: 7.4 per annum.	RBI has a new Policy issued in 2023 RBI has updated new regulations in 2024. SBI rate of interest: 7.6 per annum. Karur rate of interest: 7.4 per annum.	✓

create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football, Volleyball, and Basketball that implement the Playable interface and override the play() method to play the respective sports.

```
interface Playable {
    void play();
}
class Football implements Playable {
    String name;
    public Football(String name){
        this.name=name;
    }
  public void play() {
    System.out.println(name+" is Playing football");
   }
}
```

Similarly, create Volleyball and Basketball classes.

Sample output:

```
Sadhvin is Playing football
Sanjay is Playing volleyball
Sruthi is Playing basketball
```

For example:

Test	Input	Result
1	Sadhvin Sanjay Sruthi	Sadhvin is Playing football Sanjay is Playing volleyball Sruthi is Playing basketball
2	Vijay Arun Balaji	Vijay is Playing football Arun is Playing volleyball Balaji is Playing basketball

Code:

```
import java.util.Scanner;

interface Playable {
    void play();
}

class Football implements Playable {
    String name;

    public Football(String name) {
        this.name = name;
    }

    @Override
    public void play() {
        System.out.println(name + " is Playing football");
    }
}

class Volleyball implements Playable {
```

```
    String name;

    public Volleyball(String name) {
        this.name = name;
    }

    @Override
    public void play() {
        System.out.println(name + " is Playing volleyball");
    }
}

class Basketball implements Playable {
    String name;

    public Basketball(String name) {
        this.name = name;
    }

    @Override
    public void play() {
        System.out.println(name + " is Playing basketball");
    }
}

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

        String footballPlayerName = scanner.nextLine();
        Playable footballPlayer = new Football(footballPlayerName);

        String volleyballPlayerName = scanner.nextLine();
        Playable volleyballPlayer = new Volleyball(volleyballPlayerName);

        String basketballPlayerName = scanner.nextLine();
        Playable basketballPlayer = new Basketball(basketballPlayerName);

        footballPlayer.play();
        volleyballPlayer.play();
        basketballPlayer.play();

        scanner.close();
    }
}
```

	Test	Input	Expected	Got	
✓	1	Sadhvin Sanjay Sruthi	Sadhvin is Playing football Sanjay is Playing volleyball Sruthi is Playing basketball	Sadhvin is Playing football Sanjay is Playing volleyball Sruthi is Playing basketball	✓
✓	2	Vijay Arun Balaji	Vijay is Playing football Arun is Playing volleyball Balaji is Playing basketball	Vijay is Playing football Arun is Playing volleyball Balaji is Playing basketball	✓

WEEK 8:

As a logic building learner you are given the task to extract the string which has vowel as the first and last characters from the given array of Strings.

Step1: Scan through the array of Strings, extract the Strings with first and last characters as vowels; these strings should be concatenated.

Step2: Convert the concatenated string to lowercase and return it.

If none of the strings in the array has first and last character as vowel, then return no matches found

input1: an integer representing the number of elements in the array.

input2: String array.

Example 1:

input1: 3

input2: {“oreo”, “sirish”, “apple”}

output: oreoapple

Example 2:

input1: 2

input2: {“Mango”, “banana”}

output: no matches found

Explanation:

None of the strings has first and last character as vowel.

Hence the output is no matches found.

Example 3:

input1: 3

input2: {“Ate”, “Ace”, “Girl”}

output: ateace

For example:

Input	Result
3 oreo sirish apple	oreoapple
2 Mango banana	no matches found
3 Ate Ace Girl	ateace

Code:

```
import java.util.Scanner;

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


        int n = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character

        // Step 2: Check if the input size is valid
        if (n <= 0) {
            System.out.println("no matches found");
            return;
        }

        // Step 3: Read the array of strings
        String[] inputStrings = new String[n];
        String[] tokens = scanner.nextLine().split(" ");

        // Ensure we only take 'n' tokens from the input
        for (int i = 0; i < n; i++) {
            if (i < tokens.length) {
                inputStrings[i] = tokens[i];
            } else {
                System.out.println("no matches found");
                return;
            }
        }


        StringBuilder result = new StringBuilder();
        for (String str : inputStrings) {
            if (str.length() > 0 && isVowel(str.charAt(0)) && isVowel(str.charAt(str.length() -
1))) {
                result.append(str);
            }
        }


        if (result.length() == 0) {
            System.out.println("no matches found");
        } else {
            System.out.println(result.toString().toLowerCase());
```

```
        }

        scanner.close();
    }


    private static boolean isVowel(char c) {
        char lowerC = Character.toLowerCase(c);
        return lowerC == 'a' || lowerC == 'e' || lowerC == 'i' || lowerC == 'o' || lowerC == 'u';
    }
}
```

	Input	Expected	Got	
✓	3 oreo sirish apple	oreoapple	oreoapple	✓
✓	2 Mango banana	no matches found	no matches found	✓
✓	3 Ate Ace Girl	ateace	ateace	✓

1. Final Variable:

- Once a variable is declared `final`, its value cannot be changed after it is initialized.
- It must be initialized when it is declared or in the constructor if it's not initialized at declaration.
- It can be used to define constants

```
final int MAX_SPEED = 120;  // Constant value, cannot be changed
```

2. Final Method:

- A method declared `final` cannot be overridden by subclasses.
- It is used to prevent modification of the method's behavior in derived classes.

```
public final void display() {
   System.out.println("This is a final method.");
}
```

3. Final Class:

- A class declared as `final` cannot be subclassed (i.e., no other class can inherit from it).
- It is used to prevent a class from being extended and modified.
- public final class Vehicle {
    // class code
  }

Given a Java Program that contains the bug in it, your task is to clear the bug to the output.

you should delete any piece of code.

For example:

Test	Result
1	The maximum speed is: 120 km/h This is a subclass of FinalExample.

Code:

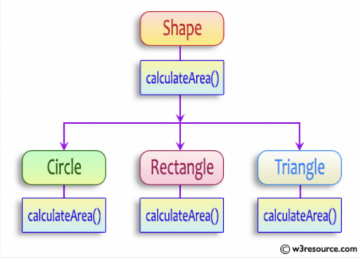
```
class FinalExample {

   // Final variable
            int maxSpeed = 120;

   // Final method
   public          void displayMaxSpeed() {
                     System.out.println("The maximum speed is: " + maxSpeed + " km/h");
   }
}

class SubClass extends FinalExample {
```

```
    public void displayMaxSpeed() {
        System.out.println("Cannot override a final method");
    }

    // You can create new methods here
    public void showDetails() {
        System.out.println("This is a subclass of FinalExample.");
    }
}

class prog {
    public static void main(String[] args) {
        FinalExample obj = new FinalExample();
        obj.displayMaxSpeed();

        SubClass subObj = new SubClass();
        subObj.showDetails();
    }
}
```

	Test	Expected	Got	
✓	1	The maximum speed is: 120 km/h This is a subclass of FinalExample.	The maximum speed is: 120 km/h This is a subclass of FinalExample.	✓

Create a base class Shape with a method called calculateArea(). Create three subclasses: Circle, Rectangle, and Triangle. Override the calculateArea() method in each subclass to calculate and return the shape's area.

In the given exercise, here is a simple diagram illustrating polymorphism implementation:




```
abstract class Shape {
   public abstract double calculateArea() ;
   }
}
```

System.out.printf("Area of a Triangle :%.2f%n",((0.5)*base*height)); // use this statement

sample Input :

4 // radius of the circle to calculate area PI*r*r

5 // length of the rectangle

6 // breadth of the rectangle to calculate the area of a rectangle

4 // base of the triangle

3 // height of the triangle

OUTPUT:

Area of a circle :50.27
Area of a Rectangle :30.00
Area of a Triangle :6.00

For example:

Test	Input	Result
1	4 5 6 4 3	Area of a circle: 50.27 Area of a Rectangle: 30.00 Area of a Triangle: 6.00
2	7 4.5 6.5 2.4	Area of a circle: 153.94 Area of a Rectangle: 29.25 Area of a Triangle: 4.32

Code:

```
import java.util.Scanner;

abstract class Shape {
    public abstract double calculateArea();
}

class Circle extends Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double calculateArea() {
        return Math.PI * radius * radius;
```

```
    }
}

class Rectangle extends Shape {
    private double length;
    private double breadth;

    public Rectangle(double length, double breadth) {
        this.length = length;
        this.breadth = breadth;
    }

    @Override
    public double calculateArea() {
        return length * breadth;
    }
}

class Triangle extends Shape {
    private double base;
    private double height;

    public Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    @Override
    public double calculateArea() {
        return 0.5 * base * height;
    }
}

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


        double radius = scanner.nextDouble();
        Shape circle = new Circle(radius);
        System.out.printf("Area of a circle: %.2f%n", circle.calculateArea());


        double length = scanner.nextDouble();
        double breadth = scanner.nextDouble();
        Shape rectangle = new Rectangle(length, breadth);
```

```
        System.out.printf("Area of a Rectangle: %.2f%n", rectangle.calculateArea());

        double base = scanner.nextDouble();
        double height = scanner.nextDouble();
        Shape triangle = new Triangle(base, height);
        System.out.printf("Area of a Triangle: %.2f%n", triangle.calculateArea());

        scanner.close();
    }
}
```

	Test	Input	Expected	Got	
✓	1	4 5 6 4 3	Area of a circle: 50.27 Area of a Rectangle: 30.00 Area of a Triangle: 6.00	Area of a circle: 50.27 Area of a Rectangle: 30.00 Area of a Triangle: 6.00	✓
✓	2	7 4.5 6.5 2.4 3.6	Area of a circle: 153.94 Area of a Rectangle: 29.25 Area of a Triangle: 4.32	Area of a circle: 153.94 Area of a Rectangle: 29.25 Area of a Triangle: 4.32	✓

Week 9:

Write a Java program to create a method that takes an integer as a parameter and throws an exception if the number is odd.

Sample input and Output:

```
82 is even.
Error: 37 is odd.
```

Fill the preloaded answer to get the expected output.

For example:

Result
82 is even. Error: 37 is odd.

Code:

```
class prog {
  public static void main(String[] args) {
    int n = 82;
    trynumber(n);
    n = 37;
    trynumber(n);
      }
  public static void trynumber(int n) {
    try {
        checkEvenNumber(n);

      System.out.println(n + " is even.");
    } catch (IllegalArgumentException e) {
      System.out.println("Error: " + e.getMessage());
    }
  }

  public static void checkEvenNumber(int number) {
    if (number % 2 != 0) {
      throw new IllegalArgumentException(number + " is odd.");
    }
  }}
```

	Expected	Got	
✓	82 is even. Error: 37 is odd.	82 is even. Error: 37 is odd.	✓

Write a Java program to handle ArithmeticException and ArrayIndexOutOfBoundsException.

Create an array, read the input from the user, and store it in the array.

Divide the 0th index element by the 1st index element and store it.

if the 1st element is zero, it will throw an exception.

if you try to access an element beyond the array limit throws an exception.

Input:

5

10 0 20 30 40

Output:

java.lang.ArithmeticException: / by zero
I am always executed

Input:

3
10 20 30

Output

java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3
I am always executed

For example:

Test	Input	Result
1	6 1 0 4 1 2 8	java.lang.ArithmeticException: / by zero I am always executed

Code:

```
import java.util.Scanner;
public class ExceptionHandlingExample{
    public static void main(String[] args){
```

```
        Scanner scanner = new Scanner(System.in);
        try{
            int size=scanner.nextInt();
            int[] arr=new int[size];
            for(int i=0;i<size;i++){
                arr[i]=scanner.nextInt();
            }
            int result=arr[0]/arr[1];
            System.out.println("Accessing out-of-bounds element: "+arr[3]);
        }
        catch(ArithmeticException e){
            System.out.println("java.lang.ArithmeticException: "+e.getMessage());
        }
        catch(ArrayIndexOutOfBoundsException e){
            System.out.println("java.lang.ArrayIndexOutOfBoundsException:
"+e.getMessage());
        }
        finally{
            System.out.println("I am always executed");
        }

    }
}
```

	Test	Input	Expected	Got
✓	1	6 1 0 4 1 2 8	java.lang.ArithmeticException: / by zero I am always executed	java.lang.ArithmeticException: / by zero I am always executed
✓	2	3 10 20 30	java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3 I am always executed	java.lang.ArrayIndexOutOfBoundsException: Index 3 out of I am always executed

In the following program, an array of integer data is to be initialized.
During the initialization, if a user enters a value other than an integer, it will throw an InputMismatchException exception.
On the occurrence of such an exception, your program should print “You entered bad data.”
If there is no such exception it will print the total sum of the array.

/* Define try-catch block to save user input in the array "name"
If there is an exception then catch the exception otherwise print the total sum of the array. */

Sample Input:

3
5 2 1

Sample Output:

8

Sample Input:

2

1 g

Sample Output:

```
You entered bad data.
```

For example:

Input	Result
3 5 2 1	8
2 1 g	You entered bad data.

Code:

```
import java.util.Scanner;
import java.util.InputMismatchException;
 public class ArraySumDemo {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int size=scanner.nextInt();
    int[] array=new int[size];
    try{
        for(int i=0;i<size;i++){
            array[i]=scanner.nextInt();
        }

        int sum=0;
        for(int num:array){
            sum+=num;
        }
        System.out.println(sum);
    }
```

```
    catch(InputMismatchException e){
        System.out.println("You entered bad data.");
    }
    finally{
        scanner.close();
    }
  }
}
```

	Input	Expected	Got	
✓	3 5 2 1	8	8	✓
✓	2 1 g	You entered bad data.	You entered bad data.	✓

Week 10:

Given an ArrayList, the task is to get the first and last element of the ArrayList in Java.

```
Input: ArrayList = [1, 2, 3, 4]
Output: First = 1, Last = 4

Input: ArrayList = [12, 23, 34, 45, 57, 67, 89]
Output: First = 12, Last = 89
```

Approach:

1. Get the ArrayList with elements.
2. Get the first element of ArrayList using the get(index) method by passing index = 0.
3. Get the last element of ArrayList using the get(index) method by passing index = size – 1.

Code:

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

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

```
        ArrayList<String> list = new ArrayList<>();


        int n = scanner.nextInt();

        scanner.nextLine();


        for (int i = 0; i < n; i++) {
            String element = scanner.nextLine();
            list.add(element);
        }

        if (!list.isEmpty()) {
            System.out.println("ArrayList: "+list);
            String firstElement = list.get(0);
            String lastElement = list.get(list.size() - 1);

            System.out.println("First : " + firstElement +", Last : "+lastElement);

        } else {
            System.out.println("The ArrayList is empty.");
        }

        scanner.close();
    }
}
```

	Test	Input	Expected	Got	
✓	1	6 30 20 40 50 10 80	ArrayList: [30, 20, 40, 50, 10, 80] First : 30, Last : 80	ArrayList: [30, 20, 40, 50, 10, 80] First : 30, Last : 80	✓
✓	2	4 5 15 25 35	ArrayList: [5, 15, 25, 35] First : 5, Last : 35	ArrayList: [5, 15, 25, 35] First : 5, Last : 35	✓

The given Java program is based on the ArrayList methods and its usage. The Java program is partially filled. Your task is to fill in the incomplete statements to get the desired output.

list.set();

list.indexOf());

list.lastIndexOf())

list.contains()

list.size());

list.add();

list.remove();

The above methods are used for the below Java program.

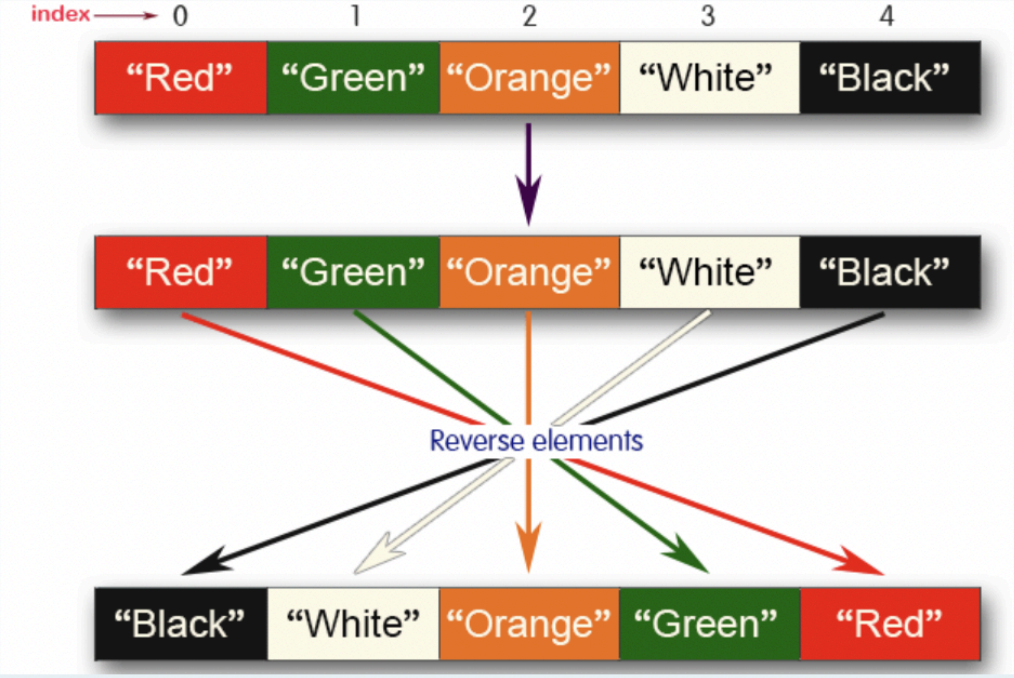
Code:

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

public class Prog {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        ArrayList<Integer> list = new ArrayList<Integer>();

        for (int i = 0; i < n; i++) {
            list.add(sc.nextInt());
        }

        System.out.println("ArrayList: " + list);

        if (list.size() > 1) {
            list.set(1, 100); // Replace element at index 1
        }

        System.out.println("Index of 100 = " + list.indexOf(100));

        System.out.println("LastIndex of 100 = " + list.lastIndexOf(100));

        System.out.println(list.contains(200));

        System.out.println("Size Of ArrayList = " + list.size());

        list.add(1, 500);

        if (list.size() > 3) {
            list.remove(3);
        }

        System.out.println("ArrayList: " + list);
```

```
        sc.close();
    }
}
```

	Test	Input	Expected	Got	
✓	1	5 1 2 3 100 5	ArrayList: [1, 2, 3, 100, 5] Index of 100 = 1 LastIndex of 100 = 3 false Size Of ArrayList = 5 ArrayList: [1, 500, 100, 100, 5]	ArrayList: [1, 2, 3, 100, 5] Index of 100 = 1 LastIndex of 100 = 3 false Size Of ArrayList = 5 ArrayList: [1, 500, 100, 100, 5]	✓

Write a Java program to reverse elements in an array list.




```
Sample input and Output:
Red
Green
Orange
White
Black
Sample output
List before reversing :
[Red, Green, Orange, White, Black]
List after reversing :
[Black, White, Orange, Green, Red]
```

Code:

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

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

        ArrayList<String> list = new ArrayList<>();

        int n = scanner.nextInt();

        scanner.nextLine();

        for (int i = 0; i < n; i++) {
            String element = scanner.nextLine();
            list.add(element);
        }
        System.out.println("List before reversing :");
        System.out.println(list);

        Collections.reverse(list);
        System.out.println("List after reversing :");
        System.out.println(list);

        scanner.close();
    }
}
```

	Test	Input	Expected	Got	
✓	1	5 Red Green Orange White Black	List before reversing : [Red, Green, Orange, White, Black] List after reversing : [Black, White, Orange, Green, Red]	List before reversing : [Red, Green, Orange, White, Black] List after reversing : [Black, White, Orange, Green, Red]	✓
✓	2	4 CSE AIML AIDS CYBER	List before reversing : [CSE, AIML, AIDS, CYBER] List after reversing : [CYBER, AIDS, AIML, CSE]	List before reversing : [CSE, AIML, AIDS, CYBER] List after reversing : [CYBER, AIDS, AIML, CSE]	✓