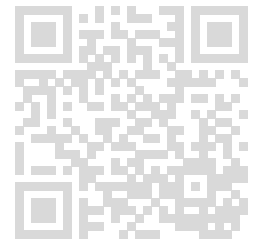


Codekata Report:



Name: Abhishek Sharma

Email: sweetisharma9673441@gmail.com

Specialization: School of Computing Science & Engineering

Completion Year: 2027

Section: Section-1

1. Complex Variable Initialization and Arithmetic Operations

Problem Statement Write a program that takes four integers as input, performs a series of arithmetic operations using these integers, and outputs the results in a specific format.

Description You need to declare four integer variables, perform the following operations, and print the results:

Add the first and the second integer, then multiply the result by the third integer.

Subtract the fourth integer from the second integer, then divide the result by the first integer.

Multiply the first integer by the fourth integer, then add the third integer to the result.

Add all four integers together and divide by two.

Input Format Four integers, each on a new line.

Output Format Four lines of output, each showing the result of the respective operations in the following format: Result of operation 1: <result> Result of operation 2: <result> Result of operation 3: <result> Result of operation 4: <result>

Sample Input: 3421

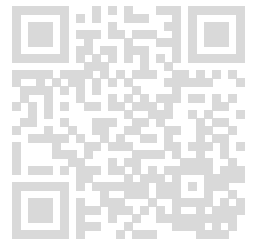
Sample output: Result of operation 1: 14 Result of operation 2: 1 Result of operation 3: 5 Result of operation 4: 5

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA



Source Code:

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

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int a = sc.nextInt();  
        int b = sc.nextInt();  
        int c = sc.nextInt();  
        int d = sc.nextInt();
```

```
//..... YOUR CODE STARTS HERE .....
```

```
System.out.println("Result of operation 1: "+((a+b)*c));  
System.out.println("Result of operation 2: "+((b-d)/a));  
System.out.println("Result of operation 3: "+((a*d)+c));  
System.out.println("Result of operation 4: "+((a+b+c+d)/2));
```

```
//..... YOUR CODE ENDS HERE .....
```

```
}  
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Result of operation 1: 14
Result of operation 2: 1
Result of operation 3: 5
Result of operation 4: 5

Compilation Status: Passed

Execution Time:

0.112s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Result of operation 1: 150

Result of operation 2: 1

Result of operation 3: 25

Result of operation 4: 18

Compilation Status: Passed

Execution Time:

0.106s

2. Bitwise Operations with Floating Point Variables

Problem Statement Write a program that takes two floating-point numbers as input, performs bitwise operations by converting them to integers, and outputs the results in a specific format.

Description You need to declare two floating-point variables, convert them to integers, perform the following bitwise operations, and print the results:

Perform bitwise AND on the two integers.

Perform bitwise OR on the two integers.

Perform bitwise XOR on the two integers.

Perform bitwise NOT on the first integer and AND it with the second integer.

Input Format Two floating-point numbers, each on a new line.

Output Format Four lines of output, each showing the result of the respective operations in the following format: Bitwise AND result: <result> Bitwise OR result: <result> Bitwise XOR result: <result> Bitwise NOT and AND result: <result>

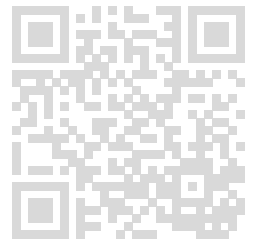
Sample Input: 5.53.3

Sample Output: Bitwise AND result: 1 Bitwise OR result: 7 Bitwise XOR result: 6 Bitwise NOT and AND result: 2

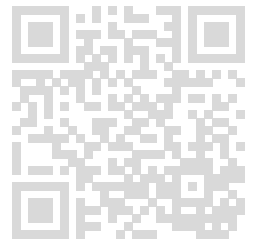
Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming



Language Used: JAVA



Source Code:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        float f1 = sc.nextFloat();
        float f2 = sc.nextFloat();

        int n1 = (int) f1;
        int n2 = (int) f2;

        //..... YOUR CODE STARTS HERE .....

        System.out.println("Bitwise AND result: "+(n1&n2));
        System.out.println("Bitwise OR result: "+(n1|n2));
        System.out.println("Bitwise XOR result: "+(n1^n2));
        System.out.println("Bitwise NOT and AND result: "+((~n1)&n2));

        //..... YOUR CODE ENDS HERE .....
    }
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Bitwise AND result: 1
Bitwise OR result: 7
Bitwise XOR result: 6
Bitwise NOT and AND result: 2

Compilation Status: Passed

Execution Time:

0.118s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Bitwise AND result: 8

Bitwise OR result: 10

Bitwise XOR result: 2

Bitwise NOT and AND result: 0

Compilation Status: Passed

Execution Time:

0.119s

3. Variable Transformation Challenge

Problem Statement: In a parallel universe, the concept of variable transformation is quite different. You need to write a Java program that performs a series of operations on variables based on specific conditions. Given three integers a, b, and c, your task is to transform these variables according to the following rules and output the final values.

Description:

If a is even, add b to a.

If b is odd, multiply c by 2.

If c is a multiple of 3, add a to c.

If the sum of a, b, and c is greater than 100, subtract 100 from each of a, b, and c.

Your program should then print the transformed values of a, b, and c in the format "a: [value], b: [value], c: [value]".

Input Format: Three integers a, b, and c are given as input from the user, each on a new line.

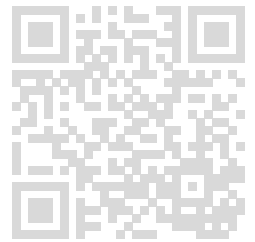
Output Format: Print the transformed values of a, b, and c in the specified format.

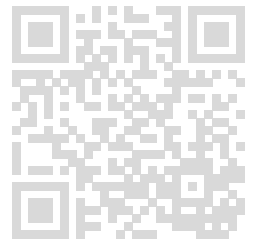
Sample input: 10 15 9

Sample output: a: 25, b: 15, c: 43

Completion Status: Completed

Concepts Included:





Language Used: JAVA

Source Code:

```
import java.util.Scanner;

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

        // Input
        int a = scanner.nextInt();
        int b = scanner.nextInt();
        int c = scanner.nextInt();
        // Transformations
        //..... YOUR CODE STARTS HERE .....

        if((a%2)==0){
            a=a+b;
        }
        if((b%2)!=0){
            c=c*2;
        }

        if((c%3)==0){
            c=c+a;
        }
        if((a+b+c)>100){
            a=a-100;
            b=b-100;
            c=c-100;
        }

        //..... YOUR CODE ENDS HERE .....
        // Output
        System.out.println("a: " + a + ", b: " + b + ", c: " + c);
    }
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

a: 25, b: 15, c: 43

Compilation Status: Passed

Execution Time:

0.117s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

a: 0, b: -50, c: -50

Compilation Status: Passed

Execution Time:

0.117s

4. Odd-Even Sum Checker

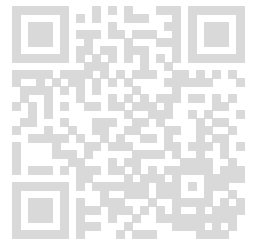
Problem Statement: Write a program that reads a list of integers from the user and checks if the sum of odd numbers is greater than the sum of even numbers in the list. If the sum of odd numbers is greater, print "Odd Sum Greater". If the sum of even numbers is greater, print "Even Sum Greater". If both sum's are equal, print "Equal". Print "Equal" if both sum's are equal.

Description: The program should continuously read integers until the user inputs a specific stop value. You need to handle edge cases where there are no odd or even numbers.

Input Format: The first line contains an integer n (number of elements). The second line contains n integers separated by spaces. The input stops when the user enters -1. **Output Format:** Print "Odd Sum Greater" if the sum of odd numbers is greater. Print "Even Sum Greater" if the sum of even numbers is greater. Print "Equal" if both sum's are equal.

Sample input: 4 2 4 6 8 -1 **Sample Output:** Even Sum Greater

Completion Status: Completed



Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

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

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

```
        int i=0;  
        int e=0;  
        int od=0;  
        int t;  
        while((t=scanner.nextInt())!=(-1)){  
            // t=scanner.nextInt();  
            if((t%2)==0){  
                e+=t;  
            }  
            else{  
                od+=t;  
            }  
            i++;  
        }
```

```
        if(e>od){  
            System.out.println("Even Sum Greater");  
        }else{  
            System.out.println("Odd Sum Greater");  
        }
```

```
        //..... YOUR CODE ENDS HERE .....
```

Compilation Details:

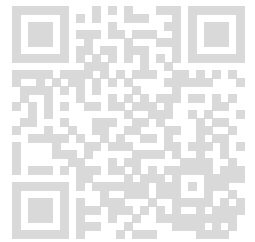
TestCase1:

Input:

< hidden >

Expected Output:

< hidden >



Output:

Even Sum Greater

Compilation Status: Passed**Execution Time:**

0.086s

TestCase2:**Input:**

< hidden >

Expected Output:

< hidden >

Output:

Odd Sum Greater

Compilation Status: Passed**Execution Time:**

0.09s

5. Grade Classification

Problem Statement: Write a program that classifies grades based on the score input by the user. The program should classify the score into "Excellent", "Good", "Average", "Pass", or "Fail" using if-else statements.

Description: The program should read the scores continuously until the user inputs a stop value. The classification is based on the following criteria:

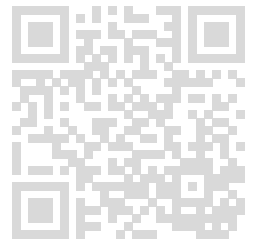
90-100: Excellent 75-89: Good 50-74: Average 35-49: Pass 0-34: Fail

Input Format: The first line contains an integer n (number of scores). The second line contains n integers separated by spaces. The input stops when the user enters -1. **Output Format:** For each score, print its classification.

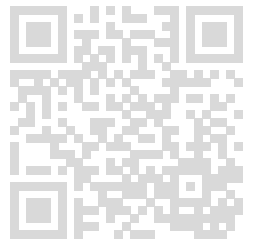
Sample Input: 5 95 85 65 45 30 -1 **Sample Output:** Excellent Good Average Pass Fail

Completion Status: Completed**Concepts Included:**

gu 27 3rd semester java programming



Language Used: JAVA



Source Code:

```
import java.util.Scanner;

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

        int i=0;
        int n=scanner.nextInt();
        int marks;

        while((marks=scanner.nextInt())!=(-1)){
            if((marks<=100) && (marks>=90)){
                System.out.println("Excellent");
            }else if(marks<=89 && marks>=75){
                System.out.println("Good");
            }else if(marks<=74 && marks>=50){
                System.out.println("Average");
            }else if(marks<=49 && marks>=35){
                System.out.println("Pass");
            }else{
                System.out.println("Fail");
            }
        }

        //..... YOUR CODE ENDS HERE .....
        scanner.close();
    }
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Excellent
Good
Average

Pass
Fail

Compilation Status: Passed

Execution Time:

0.091s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Average
Average
Average

Compilation Status: Passed

Execution Time:

0.09s

6. The Voting System

Problem Statement: You are designing a voting system where voters can cast their votes for various candidates. Each voter can vote only once, and the system needs to validate each vote and ensure the following rules are adhered to:

A vote must be cast for a valid candidate number.

The total number of votes should not exceed the number of registered voters.

A vote for a candidate should only be allowed if the candidate's ID is within the valid range.

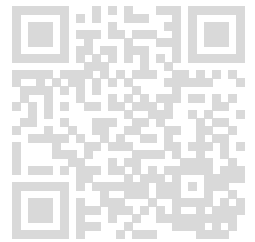
Description: You need to implement a Java program that:

Takes the number of registered voters as input.

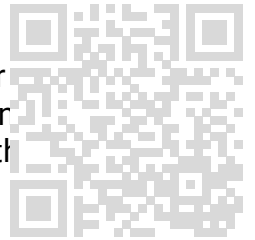
Takes the number of candidates as input.

Takes each vote (candidate ID) as input.

Outputs whether each vote is valid or invalid based on the rules above.



Input Format:First line: Integer N (Number of registered voters)Second line: Integer (Number of candidates)Third line: Integer V (Number of votes)Next V lines: Each line contains an integer ID (candidate ID)Output Format:For each vote, print "VALID" if the vote is valid, otherwise print "INVALID".



Sample Input:100541632Sample Output:VALIDINVALIDVALIDVALID

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        //..... YOUR CODE STARTS HERE .....

        Scanner sc=new Scanner(System.in);
        int vtr=sc.nextInt();
        // System.out.println();
        int cad=sc.nextInt();
        // System.out.println();
        int vots=sc.nextInt();
        // System.out.println();
        for(int i=0;i<vots;i++){
            int val=sc.nextInt();
            if(val==0){
                continue;
            }
            if(val>=1 && val<=cad){
                System.out.println("VALID");
            }else{
                System.out.println("INVALID");
            }
        }

        //..... YOUR CODE ENDS HERE .....
    }
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

VALID
INVALID
VALID
VALID

Compilation Status: Passed

Execution Time:

0.088s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

VALID
VALID
VALID
INVALID

Compilation Status: Passed

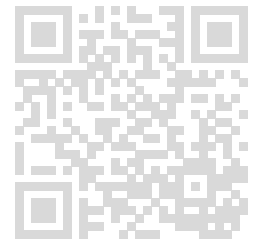
Execution Time:

0.092s

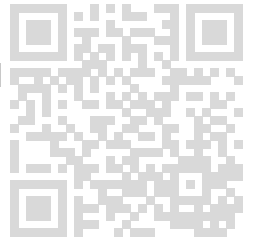
7. Nested Loop Patterns

Problem Statement: Write a program that generates a pattern based on user input using nested loops. The pattern to be generated is a pyramid where each level contains a specific number of symbols. Each level's pattern should follow a unique rule:

The number of symbols on each level is equal to the level number.



The symbol for each level is based on the level number where odd levels use * and even levels use #.



Description: You need to implement a Java program that:

Takes an integer

N (number of levels) as input.

Generates and prints the pyramid pattern according to the rules described.

Input Format: Single line: Integer N (Number of levels) Output Format: Pyramid pattern with N levels, where odd levels use * and even levels use #.

Sample Input: 3 Sample Output: *

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

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

```
public class Main {  
    public static void main(String[] args) {  
        //..... YOUR CODE STARTS HERE .....
```

```
        Scanner sc=new Scanner(System.in);  
        int n=sc.nextInt();  
        for(int i=0;i<n;i++){  
            for(int j=0;j<=i;j++){  
                if((i+1)%2==0){  
                    System.out.print("#");  
                }else{  
                    System.out.print("*");  
                }  
            }  
        }  
        System.out.println();  
    }  
    //..... YOUR CODE ENDS HERE .....
```

```
    }  
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

*

Compilation Status: Passed

Execution Time:

0.091s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

*

####

Compilation Status: Passed

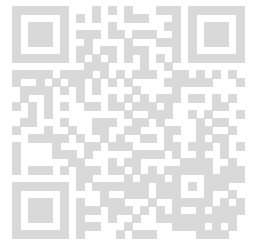
Execution Time:

0.088s

8. OOPs Features & Interface

Problem Statement:Create an interface Shape with methods area and perimeter. Implement this interface in two classes: Rectangle and Circle. Each class should have appropriate constructors and methods to calculate the area and perimeter.

Description:Define an interface Shape with methods double area() and double perimeter().



Abhishek Sharma (sweetisharma9673441@gmail.com)

Create a class Rectangle that implements Shape with properties length and width.

Create a class Circle that implements Shape with property radius.

Ensure the Rectangle and Circle classes have constructors to initialize their properties.

Implement the methods area and perimeter in both classes.

Input Format:String representing the shape type (Rectangle or Circle).If Rectangle, two doubles representing length and width.If Circle, one double representing radius.Output Format:Double representing the area of the shape.Double representing the perimeter of the shape.

Sample Input:Rectangle510Sample Output:Area: 50.0Perimeter: 30.0

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

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

```
interface Shape {  
    double area();  
    double perimeter();  
}
```

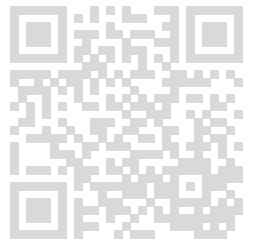
```
class Rectangle implements Shape {  
    //..... YOUR CODE STARTS HERE .....
```

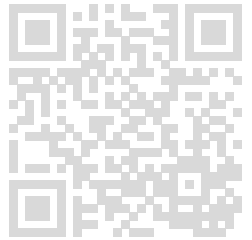
```
    private double length;  
    private double width;
```

```
    // Constructor  
    public Rectangle(double length, double width) {  
        this.length = length;  
        this.width = width;  
    }
```

```
    // Implementing area method  
    @Override  
    public double area() {  
        return length * width;  
    }
```

```
    // Implementing perimeter method
```





```
@Override
public double perimeter() {
return 2 * (length + width);
}
```

```
//..... YOUR CODE ENDS HERE .....
}
```

```
class Circle implements Shape {
//..... YOUR CODE STARTS HERE .....
```

```
private double radius;
```

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

```
// Implementing area method
@Override
public double area() {
return Math.PI * radius * radius;
}
```

```
// Implementing perimeter method (circumference)
@Override
public double perimeter() {
return 2 * Math.PI * radius;
}
```

```
//..... YOUR CODE ENDS HERE .....
}
```

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

```
if (shapeType.equals("Rectangle")) {
double length = scanner.nextDouble();
double width = scanner.nextDouble();
shape = new Rectangle(length, width);
} else if (shapeType.equals("Circle")) {
double radius = scanner.nextDouble();
shape = new Circle(radius);
} else {
System.out.println("Invalid shape type");
return;
}
```

```
System.out.println("Area: " + shape.area());
System.out.println("Perimeter: " + shape.perimeter());
}
```

}

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Area: 50.0

Perimeter: 30.0

Compilation Status: Passed

Execution Time:

0.122s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Area: 12.0

Perimeter: 14.0

Compilation Status: Passed

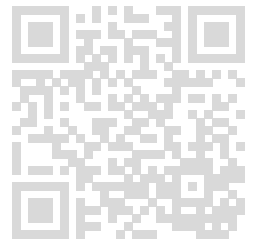
Execution Time:

0.119s

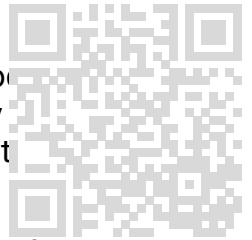
9. Classes and Objects

Problem StatementCreate a class Car that represents a car with a unique twist. The Car class should include properties such as make, model, and year. Additionally, it should have a method getCarAge that calculates the car's age based on the current year input by the user. However, you need to ensure that the car's year of manufacture is validated to be not in the future.

DescriptionDefine a class Car with private properties: make (String), model (String),



and year (int). Include a constructor that initializes these properties. Create a method getCarAge that calculates the age of the car based on the current year provided by the user. Ensure the year property is validated to be less than or equal to the current year.



Input FormatString representing the make of the car.String representing the model of the car.Integer representing the year of the car.Integer representing the current year.Output FormatInteger representing the age of the car.A message indicating if the year is invalid.

Sample Input 1:Toyota Corolla 2015 2024Sample Output 1:9

Sample Input 2:TeslaModelS20252024Sample Output 2:Invalid year

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

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

```
class Main {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        String make = scanner.next();  
        String model = scanner.next();  
        int year = scanner.nextInt();  
        int currentYear = scanner.nextInt();
```

```
        Car car = new Car(make, model, year);  
        int age = car.getCarAge(currentYear);
```

```
        if (age != -1) {  
            System.out.println(age);  
        }  
    }  
}
```

```
class Car {  
    //..... YOUR CODE STARTS HERE .....  
    String make;  
    String model;  
    int year;  
    int currentYear;
```

```
    Car(String m,String m1,int y){  
        this.make=m;
```

```
this.model=m1;  
this.year=y;  
}
```

```
public int getCarAge(int cy){  
return cy-this.year;  
}
```

```
//..... YOUR CODE ENDS HERE .....  
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

9

Compilation Status: Passed

Execution Time:

0.086s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

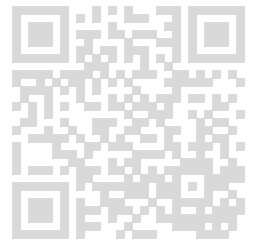
Output:

6

Compilation Status: Passed

Execution Time:

0.09s



Abhishek Sharma (sweetisharma9673441@gmail.com)

10. Access Modifiers

Problem Statement: Create a class Person with private properties and public methods for accessing these properties. Ensure that the age of the person is always a positive value.

Description: Define a class Person with private properties: name (String) and age (int). Include public methods setName and setAge for setting the properties. Include public methods getName and getAge for accessing the properties. Ensure that the setAge method only allows positive values.

Input Format: String representing the name of the person. Integer representing the age of the person. **Output Format:** String representing the name of the person. Integer representing the age of the person. A message "Invalid age" indicating if the age is invalid.

Sample Input 1: Alice25 Sample Output 1: Name: Alice Age: 25

Sample Input 2: Diana0 Sample Output 2: Invalid age

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

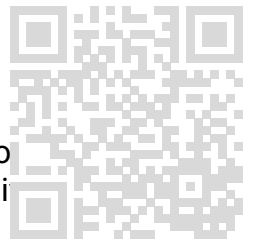
```
import java.util.Scanner;

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

        Person person = new Person();
        person.setName(name);
        person.setAge(age);

        if (age > 0) {
            System.out.println("Name: " + person.getName());
            System.out.println("Age: " + person.getAge());
        }
    }
}

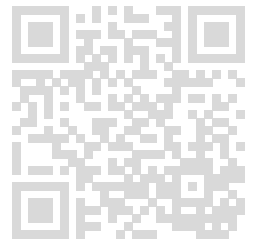
class Person {
    //..... YOUR CODE STARTS HERE .....
```



```
String Name;
int Age;
public void setName(String n){
this.Name=n;
}
public void setAge(int y){
if(y<=0){
System.out.println("Invalid age");
return;
}
this.Age=y;
}

public String getName(){
return Name;
}
public int getAge(){
return Age;
}

//..... YOUR CODE ENDS HERE .....
}
```



Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Name: Alice
Age: 25

Compilation Status: Passed

Execution Time:

0.108s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Abhishek Sharma (sweetisharma9673441@gmail.com)

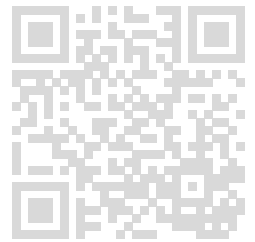
Output:

Invalid age

Compilation Status: Passed

Execution Time:

0.094s



11. OOPs Features & Interface

Problem Statement Implement an interface Employee with methods calculateSalary and getDetails. Create two classes FullTimeEmployee and PartTimeEmployee that implement this interface. Each class should have its own way of calculating salary.

Description Define an interface Employee with methods double calculateSalary() and String getDetails(). Create a class FullTimeEmployee with properties name (String), monthlySalary (double) and implement the interface methods. Create a class PartTimeEmployee with properties name (String), hourlyRate (double), and hoursWorked (int) and implement the interface methods. Ensure each class has a constructor to initialize its properties.

Input Format String representing the employee type (FullTimeEmployee or PartTimeEmployee). If FullTimeEmployee, a string for name and a double for monthlySalary. If PartTimeEmployee, a string for name, a double for hourlyRate, and an integer for hoursWorked. **Output Format** String representing the employee details. Double representing the calculated salary.

Sample Input 1: FullTimeEmployeeJohn3000.0 **Sample Output 1:** Name: John, Salary: 3000.0

Sample Input 2: PartTimeEmployeeJane15.0120 **Sample Output 2:** Name: Jane, Salary: 1800.0

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.Scanner;

interface Employee {
    double calculateSalary();
    String getDetails();
}
```

```
class FullTimeEmployee implements Employee {  
//..... YOUR CODE STARTS HERE .....
```

```
private String name;  
private double monthlySalary;
```

```
// Constructor  
public FullTimeEmployee(String name, double monthlySalary) {  
this.name = name;  
this.monthlySalary = monthlySalary;  
}
```

```
// Calculate salary for Full-Time Employee  
@Override  
public double calculateSalary() {  
return monthlySalary;  
}
```

```
// Get details of Full-Time Employee  
@Override  
public String getDetails() {  
return "Name: " + name + ", Salary: " + calculateSalary();  
}
```

```
//..... YOUR CODE ENDS HERE .....
```

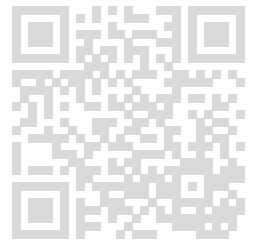
```
class PartTimeEmployee implements Employee {  
//..... YOUR CODE STARTS HERE .....
```

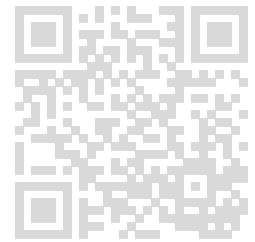
```
private String name;  
private double hourlyRate;  
private int hoursWorked;
```

```
// Constructor  
public PartTimeEmployee(String name, double hourlyRate, int hoursWorked) {  
this.name = name;  
this.hourlyRate = hourlyRate;  
this.hoursWorked = hoursWorked;  
}
```

```
// Calculate salary for Part-Time Employee  
@Override  
public double calculateSalary() {  
return hourlyRate * hoursWorked;  
}
```

```
// Get details of Part-Time Employee  
@Override  
public String getDetails() {  
return "Name: " + name + ", Salary: " + calculateSalary();  
}
```





```
//..... YOUR CODE ENDS HERE .....  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        String employeeType = scanner.next();  
        Employee employee;  
  
        if (employeeType.equals("FullTimeEmployee")) {  
            String name = scanner.next();  
            double monthlySalary = scanner.nextDouble();  
            employee = new FullTimeEmployee(name, monthlySalary);  
        } else if (employeeType.equals("PartTimeEmployee")) {  
            String name = scanner.next();  
            double hourlyRate = scanner.nextDouble();  
            int hoursWorked = scanner.nextInt();  
            employee = new PartTimeEmployee(name, hourlyRate, hoursWorked);  
        } else {  
            System.out.println("Invalid employee type");  
            return;  
        }  
  
        System.out.println(employee.getDetails());  
    }  
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Name: John, Salary: 3000.0

Compilation Status: Passed

Execution Time:

0.129s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

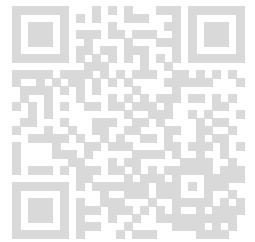
Output:

Name: Jane, Salary: 1800.0

Compilation Status: Passed

Execution Time:

0.13s



12. Month Days Finder

Problem Statement: Write a program that takes a month (as a number) and a year as input and prints the number of days in that month, considering leap years.

Description: The program should read the month and year continuously until the user inputs a stop value. You need to handle leap years and invalid inputs.

Input Format: The first line contains an integer n (number of entries). The next n lines contain two integers each: the month and the year. The input stops when the user enters -1 -1. **Output Format:** Print the number of days in the given month for each input.

Sample Input: 2 2020 -1 -1 **Sample Output:** 29

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.Scanner;

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

        //..... YOUR CODE STARTS HERE .....

        while(true){
            int m=scanner.nextInt();
```

```
int y=scanner.nextInt();

if(m==(-1) && y==(-1)){
break;
}

if(y%4==0){
if(y%100==0){
if(y%400==0){
System.out.println("29");
}else{
System.out.println("28");
}
}else{
System.out.println("29");
}
}else{
System.out.println("28");
}

}
//..... YOUR CODE ENDS HERE .....

scanner.close();
}
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

29

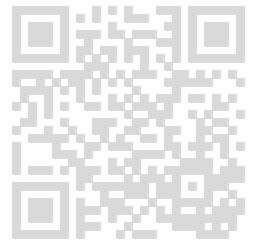
Compilation Status: Passed

Execution Time:

0.097s

TestCase2:

Input:



Abhishek Sharma (sweetisharma9673441@gmail.com)

< hidden >

Expected Output:

< hidden >

Output:

28

Compilation Status: Passed

Execution Time:

0.089s

13. Polymorphic Zoo

Problem Statement:Create a program that models a zoo with animals demonstrating polymorphism. Each animal should have a speak method, but the sound they make depends on the type of animal. Implement a system that allows adding different animals to the zoo and prints their sounds.

Description:Your task is to design a Java program that uses polymorphism to model a zoo. You need to create an abstract class Animal with a method speak(). Then, create at least three subclasses (Lion, Elephant, Monkey) that override the speak() method to return their respective sounds. The program should allow the user to add animals to the zoo and then print out the sounds of all animals in the zoo.

Input Format:The first line of input contains an integer N, the number of animals. The next N lines each contain a string representing the type of animal (Lion, Elephant, Monkey). **Output Format:**The output should be the sounds of all animals in the zoo, each on a new line, in the order they were added.

Sample Input:3LionElephantMonkey**Sample Output:**RoarTrumpetOoh Ooh Aah Aah

Completion Status: Completed

Concepts Included:

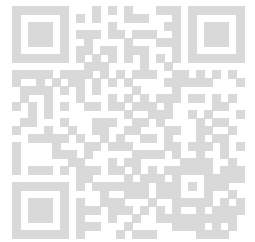
gu 27 3rd semester java programming

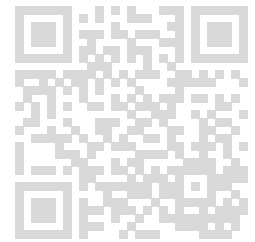
Language Used: JAVA

Source Code:

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

abstract class Animal {
    abstract String speak();
}
```





```
class Lion extends Animal {  
//..... YOUR CODE STARTS HERE .....
```

```
@Override  
String speak(){  
return "Roar";  
}
```

```
//..... YOUR CODE ENDS HERE .....
```

```
}
```

```
class Elephant extends Animal {  
//..... YOUR CODE STARTS HERE .....
```

```
@Override  
String speak(){  
return "Trumpet";  
}
```

```
//..... YOUR CODE ENDS HERE .....
```

```
}
```

```
class Monkey extends Animal {  
//..... YOUR CODE STARTS HERE .....
```

```
@Override  
String speak(){  
return "Ooh Ooh Aah Aah";  
}
```

```
//..... YOUR CODE ENDS HERE .....
```

```
}
```

```
public class Main {  
public static void main(String[] args) {  
Scanner sc = new Scanner(System.in);  
int N = sc.nextInt();  
sc.nextLine(); // Consume the newline character  
ArrayList<Animal> zoo = new ArrayList<>();  
//..... YOUR CODE STARTS HERE .....
```

```
for (int i = 0; i < N; i++) {  
String animalType = sc.nextLine().trim().toLowerCase();  
switch (animalType) {  
case "lion":  
zoo.add(new Lion());  
break;  
case "elephant":  
zoo.add(new Elephant());  
break;  
case "monkey":
```

```
zoo.add(new Monkey());  
break;  
default:  
System.out.println("Unknown animal type: " + animalType);  
break;  
}  
}
```

```
//..... YOUR CODE ENDS HERE .....  
for (Animal animal : zoo) {  
System.out.println(animal.speak());  
}  
}  
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Roar
Trumpet
Ooh Ooh Aah Aah

Compilation Status: Passed

Execution Time:

0.096s

TestCase2:

Input:

< hidden >

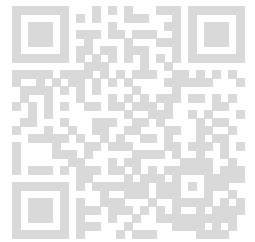
Expected Output:

< hidden >

Output:

Roar

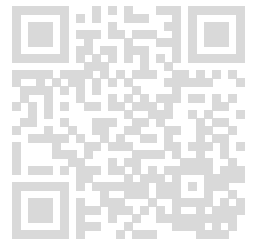
Compilation Status: Passed



Abhishek Sharma (sweetisharma9673441@gmail.com)

Execution Time:

0.091s



14. Library Management System

Problem Statement: Design a library management system using OOP principles. The system should allow adding books, borrowing books, and returning books. Implement classes for Book and Library.

Description: Your task is to create a Java program that models a library management system. You need to create a class Book with attributes for the book title, author, and a boolean to indicate if it is borrowed. Create another class Library that contains a list of books and methods to add a book, borrow a book by title, and return a book by title.

Input Format: The first line contains an integer N, the number of operations. The next N lines contain operations in the format add <title> <author>, borrow <title>, or return <title>. **Output Format:** Print the result of each operation. For borrow, print "Book borrowed" if successful or "Book not available" if not. For return, print "Book returned". For add, print "Book added".

Sample Input: 3 add HarryPotter J.K.Rowling borrow HarryPotter return HarryPotter
Sample Output: Book added Book borrowed Book returned

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

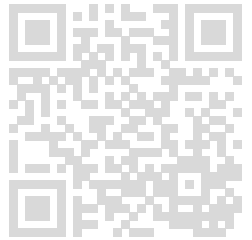
Language Used: JAVA

Source Code:

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

class Book {
//..... YOUR CODE STARTS HERE .....
private String title;
private String author;
private boolean isBorrowed;

// Constructor to initialize a book
public Book(String title, String author) {
this.title = title;
this.author = author;
this.isBorrowed = false; // Initially, the book is not borrowed
}
```



```
// Getters for title and author
public String getTitle() {
return title;
}

// Method to check if the book is borrowed
public boolean isBorrowed() {
return isBorrowed;
}

// Method to borrow the book
public void borrowBook() {
this.isBorrowed = true;
}

// Method to return the book
public void returnBook() {
this.isBorrowed = false;
}

//..... YOUR CODE ENDS HERE .....
}

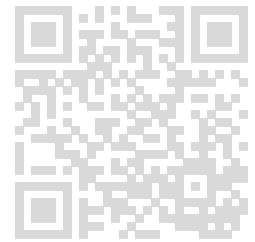
class Library {
//..... YOUR CODE STARTS HERE .....

private ArrayList<Book> books;

// Constructor to initialize the library
public Library() {
this.books = new ArrayList<>();
}

// Method to add a book to the library
public String addBook(String title, String author) {
books.add(new Book(title, author));
//      System.out.println("Book added");
return "Book added";
}

// Method to borrow a book by its title
public String borrowBook(String title) {
for (Book book : books) {
if (book.getTitle().equalsIgnoreCase(title)) {
if (!book.isBorrowed()) {
book.borrowBook();
//      System.out.println("Book borrowed");
return "Book borrowed";
}
}
}
return "Book not available";
}
}
return "Book not available";
```

```
}

public String returnBook(String title) {
    for (Book book : books) {
        if (book.getTitle().equalsIgnoreCase(title)) {
            if(book.isBorrowed()){
                book.returnBook();
                return "Book returned";
            }
            return "Book Not borrowed yet";
        }
    }
    return "Book not available";
}

//..... YOUR CODE ENDS HERE .....
```

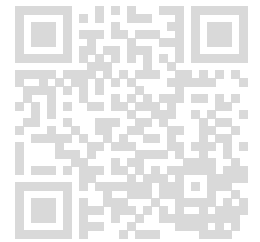
```
public class Main {
    public static void main(String[] args) {
        //..... YOUR CODE STARTS HERE .....
```

```
Scanner sc = new Scanner(System.in);
int N = sc.nextInt(); // Number of operations
sc.nextLine();
Library library = new Library();
```

```
// Processing operations
int i=0;
String[] v=new String[N];
while(i<N){
    String operation = sc.nextLine();
    String[] parts = operation.split(" ");
    String command = parts[0];
    if (command.equals("add")) {
        v[i]=library.addBook(parts[1], parts[2]);
    } else if (command.equals("borrow")) {
        String title = parts[1];
        v[i]=library.borrowBook(title);
    } else if (command.equals("return")) {
        String title = parts[1];
        v[i]=library.returnBook(title);
    }
    i++;
}
for(String e: v){
    System.out.println(e);
}

//..... YOUR CODE ENDS HERE .....
```

Compilation Details:



TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Book added
Book borrowed
Book returned

Compilation Status: Passed

Execution Time:

0.091s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Book added
Book borrowed
Book not available
Book returned

Compilation Status: Passed

Execution Time:

0.091s

15. Student Grading System

Problem Statement: Implement a student grading system using OOP principles. The system should handle multiple students and calculate their grades based on different criteria.

Description: Your task is to create a Java program that models a student grading system. You need to create a class Student with attributes like name, marks, and

grade. Create methods to add marks, calculate grades, and display student details. The grade should be calculated based on the average of the marks.

Input Format: The first line contains an integer N, the number of operations. The next lines contain operations in the format add <name> <marks>, calculate <name>, or display <name>. Output Format: Print the result of each operation. For add, print "Marks added". For calculate, print "Grade calculated". For display, print the student's details.

Sample Input: 4 add John 85 add John 90 calculate John display John

Sample Output: Marks added Marks added Grade calculated John: 87.5 - B

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

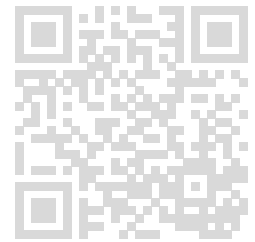
Language Used: JAVA

Source Code:

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

class Student {
//..... YOUR CODE STARTS HERE .....
    ArrayList<String> name=new ArrayList<>();
    ArrayList<Integer> marks=new ArrayList<>();
    ArrayList<Double> avg=new ArrayList<>();
    ArrayList<String> grade=new ArrayList<>();
    ArrayList<Integer> ptr=new ArrayList<>();

    String addMarks(String n,int m){
        if(check(n)){
            for(int i=0;i<this.name.size();i++){
                if(name.get(i).equalsIgnoreCase(n)){
                    int val=marks.get(i);
                    marks.set(i,(val+m));
                    ptr.add(i,ptr.get(i)+1);
                }
            }
        }else{
            name.add(n);
            int i=name.size()-1;
            //      System.out.println(i+" "+name.size());
            int val=(marks.size()>0)?marks.get(i):0;
            if(marks.isEmpty()){
                marks.add(val+m);
            }else{
                marks.set(i,(val+m));
            }
        }
    }
}
```

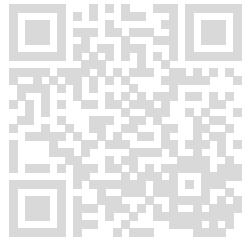


```
}  
int p=(ptr.size()>0)?ptr.get(i):0;  
if(ptr.isEmpty()){  
ptr.add(1);  
}else{  
ptr.set(i,p+1);  
}  
}  
return "Marks added";  
}
```

```
String cal_grade(String n){  
if(check(n)){  
for(int i=0;i<this.name.size();i++){  
if(name.get(i).equalsIgnoreCase(n)){  
int p=ptr.get(i);  
int mark=marks.get(i);  
double val=(double)(mark)/(p);  
if(!avg.isEmpty()){  
avg.set(i,val);  
}else{  
avg.add(val);  
}  
String g;  
if(val>=90){  
g="A";  
}else if(val>=80 && val<=89){  
g="B";  
}else if(val>=70 && val<=79){  
g="C";  
}else if(val>=60 && val<=69){  
g="D";  
}else if(val>=50 && val<=59){  
g="E";  
}else{  
g="F";  
}  
if(grade.isEmpty()){  
grade.add(g);  
}else{  
grade.set(i,g);  
}  
}  
}  
}else{  
return "Student does not exist";  
}  
return "Grade calculated";  
}
```

```
String display(String n){  
for(int i=0;i<name.size();i++){
```

Abhishek Sharma (sweetisharma9673441@gmail.com)



```
if(name.get(i).equalsIgnoreCase(n)){
double a=avg.get(i);
String g=grade.get(i);
return name.get(i)+": "+(a)+" - "+g;
}
}
return "Not found";
}
```

```
boolean check(String n){
for(String e: this.name){
if(e.equalsIgnoreCase(n)){
return true;
}
}
return false;
}
//..... YOUR CODE ENDS HERE .....
```

```
public class Main {
public static void main(String[] args) {
//..... YOUR CODE STARTS HERE .....
```

```
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
sc.nextLine();
Student s=new Student();
String[] v=new String[n];
int i=0;
while(i<n){
String data=sc.nextLine();
String[] val=data.split(" ");
if(val[0].equalsIgnoreCase("add")){
v[i]=s.addMarks(val[1],Integer.parseInt(val[2]));
}else if(val[0].equalsIgnoreCase("calculate")){
v[i]=s.cal_grade(val[1]);
}else if(val[0].equalsIgnoreCase("display")){
v[i]=s.display(val[1]);
}
i++;
}
```

```
for(String e: v){
System.out.println(e);
}
```

```
//..... YOUR CODE ENDS HERE .....
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Marks added
Marks added
Grade calculated
John: 87.5 - B

Compilation Status: Passed

Execution Time:

0.134s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Marks added
Grade calculated
Alice: 70.0 - C

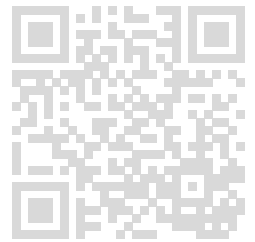
Compilation Status: Passed

Execution Time:

0.135s

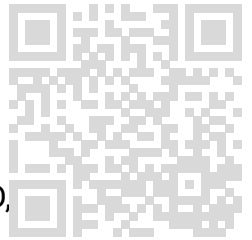
16. Smart Appliance Management SystemProblem

Statement:Design a smart appliance management system that allows users to manage a collection of appliances in a smart home. Each appliance has a unique ID, a name, a power rating, and a status (on/off). You need to implement a class called Appliance to represent an appliance, and a class called SmartHome to manage the collection of appliances. The SmartHome class should provide methods to add an appliance, remove an appliance by ID, toggle the



status of an appliance by ID, and calculate the total power consumption of all appliances that are currently on.

Description: Create an Appliance class with the following attributes: int applianceID, String name, double powerRating, and boolean status.



Implement a parameterized constructor in the Appliance class to initialize all attributes.

Implement methods to toggle the status of the appliance and retrieve the power rating.

The SmartHome class should maintain a list of Appliance objects.

Implement the following methods in the SmartHome class:

void addAppliance(Appliance appliance): Adds a new appliance to the system.

boolean removeAppliance(int applianceID): Removes an appliance from the system by its ID. Returns true if the appliance was successfully removed, false otherwise.

boolean toggleApplianceStatus(int applianceID): Toggles the status of an appliance by its ID. Returns true if the status was successfully toggled, false otherwise.

double calculateTotalPower(): Calculates the total power consumption of all appliances that are currently on.

Input Format: The first line contains an integer n, the number of appliances to be added to the system. The next n lines each contain the details of an appliance in the following order: applianceID, name, powerRating, and status (either "on" or "off"). The next line contains an integer m, the number of operations to be performed. The next m lines each contain a string representing an operation: either "add", "remove", "toggle", or "calculate", followed by the relevant parameters (appliance details for "add", applianceID for "remove" and "toggle"). Output Format: For each "remove" operation, output "Appliance removed successfully" if the appliance was removed, otherwise output "Appliance not found". For each "toggle" operation, output "Appliance status toggled" if the status was toggled, otherwise output "Appliance not found". For the "calculate" operation, output the total power consumption of all appliances that are currently on.

Sample Input: 2501 Air_Conditioner 1.5 on 502 Refrigerator 0.8 off 3 toggle
502 calculate remove 501
Sample Output: Appliance status toggled
Total Power Consumption: 2.3 kW
Appliance removed successfully

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

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

```
class Appliance {
//..... YOUR CODE STARTS HERE .....
```

```
int id;
String name;
double powerRating;
boolean status;
```

```
Appliance(int id,String name,double p,boolean status){
this.id=id;
this.name=name;
this.powerRating=p;
this.status=status;
}
```

```
public void toggleStatus(){
status=(status==true)?false:true;
}
```

```
public double getPower(){
return powerRating;
}
```

```
//..... YOUR CODE ENDS HERE .....
```

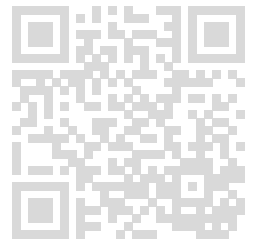
```
class SmartHome {
//..... YOUR CODE STARTS HERE .....
```

```
ArrayList<Appliance> appliances=new ArrayList<>();
```

```
public void addAppliance(Appliance a){
appliances.add(a);
}
```

```
public boolean removeAppliance(int appid){
for(int i=0;i<appliances.size();i++){
if(appliances.get(i).id==appid){
appliances.remove(appliances.get(i));
return true;
}
}
return false;
}
```

```
public boolean toggleApplianceStatus(int appid){
for(int i=0;i<appliances.size();i++){
if(appliances.get(i).id==appid){
```




```
appliances.get(i).toggleStatus();
return true;
}
}
return false;
}
```

```
public double calculateTotalPower(){
double power=0;
for(int i=0;i<appliances.size();i++){
if(appliances.get(i).status)
power+=appliances.get(i).powerRating;
}
return power;
}
```

```
//..... YOUR CODE ENDS HERE .....
}
```

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

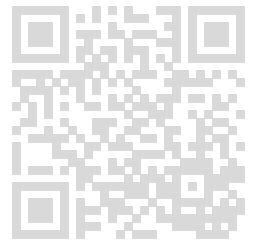
```
// Input the number of appliances
int n = scanner.nextInt();
scanner.nextLine(); // Consume newline
```

```
// Add appliances to the system
for (int i = 0; i < n; i++) {
int applianceID = scanner.nextInt();
String name = scanner.next();
double powerRating = scanner.nextDouble();
String status = scanner.next();
scanner.nextLine(); // Consume newline
```

```
boolean isOn = status.equalsIgnoreCase("on");
Appliance appliance = new Appliance(applianceID, name, powerRating, isOn);
smartHome.addAppliance(appliance);
}
```

```
// Input the number of operations
int m = scanner.nextInt();
scanner.nextLine(); // Consume newline
```

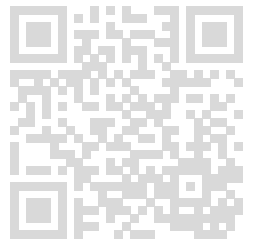
```
// Perform operations
for (int i = 0; i < m; i++) {
String operation = scanner.next();
switch (operation.toLowerCase()) {
case "remove":
int applianceID = scanner.nextInt();
```



```
if (smartHome.removeAppliance(applianceID)) {
    System.out.println("Appliance removed successfully");
} else {
    System.out.println("Appliance not found");
}
break;

case "toggle":
    applianceID = scanner.nextInt();
    if (smartHome.toggleApplianceStatus(applianceID)) {
        System.out.println("Appliance status toggled");
    } else {
        System.out.println("Appliance not found");
    }
    break;

case "calculate":
    double totalPower = smartHome.calculateTotalPower();
    System.out.println("Total Power Consumption: " + totalPower + " kW");
    break;
}
}
scanner.close();
}
```



Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Appliance status toggled
Total Power Consumption: 2.3 kW
Appliance removed successfully

Compilation Status: Passed

Execution Time:

0.13s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Appliance status toggled
Total Power Consumption: 3.0 kW
Appliance status toggled
Total Power Consumption: 4.3 kW

Compilation Status: Passed

Execution Time:

0.121s

17. Dynamic Vehicle Registration System

Problem Statement: You need to design a dynamic Vehicle Registration System where each vehicle can be of different types (Car, Motorcycle, or Truck) and should be able to report its registration details. Implement a base class Vehicle and three subclasses Car, Motorcycle, and Truck.

Description:

Create a base class Vehicle with:

A constructor to initialize the vehicle's registration number and owner's name. A method displayDetails() to display the vehicle's details. Extend this base class with three subclasses:

Car with an additional attribute for the number of doors. Motorcycle with an additional attribute for engine capacity. Truck with an additional attribute for cargo capacity.

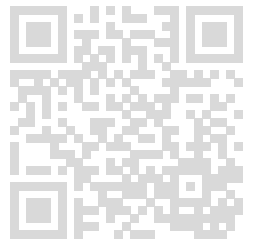
Input Format: First line: Integer N (Number of operations to perform) Next N lines: Each line will describe an operation in the format REGISTER <vehicle_type> <registration_number> <owner_name> <specific_attribute>. **Output Format:** For each operation, output the details of the registered vehicle.

Sample Input: 3REGISTER Car ABC123 John 4REGISTER Motorcycle XYZ789 Alice 600ccREGISTER Truck LMN456 Bob 10000kg
Sample Output: Vehicle: CarRegistration Number: ABC123Owner: JohnNumber of Doors: 4Vehicle: MotorcycleRegistration Number: XYZ789Owner: AliceEngine Capacity: 600ccVehicle: TruckRegistration Number: LMN456Owner: BobCargo Capacity: 10000kg

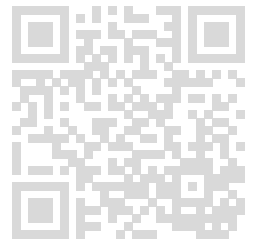
Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming



Language Used: JAVA



Source Code:

```
import java.util.Scanner;

class Vehicle {
//..... YOUR CODE STARTS HERE .....

String Regis_Num;
String name;

Vehicle(String r,String n){
Regis_Num=r;
name=n;
}

public void displayDetails(){
System.out.println("Registration Number: "+Regis_Num);
System.out.println("Owner: "+name);
}

//..... YOUR CODE ENDS HERE .....
}

class Car extends Vehicle {
//..... YOUR CODE STARTS HERE .....

int numOfDoors;

Car(String regis,String n,int numDoors){
super(regis,n);
numOfDoors=numDoors;
}

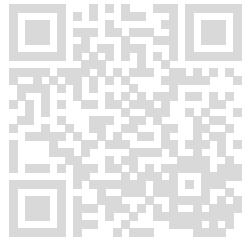
@Override
public void displayDetails(){
super.displayDetails();
System.out.println("Number of Doors: "+numOfDoors);
}

//..... YOUR CODE ENDS HERE .....
}

class Motorcycle extends Vehicle {
//..... YOUR CODE STARTS HERE .....

String EngineCapacity;

Motorcycle(String regis,String n,String s){
super(regis,n);
EngineCapacity=s;
```



```
}

@Override
public void displayDetails(){
    super.displayDetails();
    System.out.println("Engine Capacity: "+EngineCapacity);
}

//..... YOUR CODE ENDS HERE .....
}

class Truck extends Vehicle {
    //..... YOUR CODE STARTS HERE .....

    String cargoCap;

    Truck(String regis,String n,String s){
        super(regis,n);
        cargoCap=s;
    }

    @Override
    public void displayDetails(){
        super.displayDetails();
        System.out.println("Cargo Capacity: "+cargoCap);
    }

    //..... YOUR CODE ENDS HERE .....
}

public class Main {
    public static void main(String[] args) {
        //..... YOUR CODE STARTS HERE .....

        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        sc.nextLine();

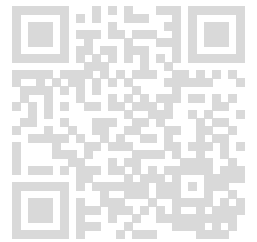
        for(int i=0;i<n;i++){
            String input=sc.nextLine();
            String[] data=input.split("REGISTER ")[1].split(" ");
            System.out.println("Vehicle: "+data[0]);

            if(data[0].equalsIgnoreCase("Car")){
                Car car=new Car(data[1],data[2],Integer.parseInt(data[3]));
                car.displayDetails();
            }else if(data[0].equalsIgnoreCase("Motorcycle")){
                Motorcycle mc=new Motorcycle(data[1],data[2],data[3]);
                mc.displayDetails();
            }else if(data[0].equalsIgnoreCase("Truck")){
                Truck tk=new Truck(data[1],data[2],data[3]);
                tk.displayDetails();
            }else{
                System.out.println("Wrong type entered");
            }
        }
    }
}
```

```
}  
}
```

```
//..... YOUR CODE ENDS HERE .....
```

```
}  
}
```



Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Vehicle: Car
Registration Number: ABC123
Owner: John
Number of Doors: 4
Vehicle: Motorcycle
Registration Number: XYZ789
Owner: Alice
Engine Capacity: 600cc
Vehicle: Truck
Registration Number: LMN456
Owner: Bob
Cargo Capacity: 10000kg

Compilation Status: Passed

Execution Time:

0.113s

TestCase2:

Input:

< hidden >

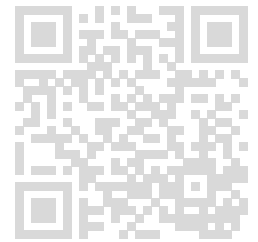
Expected Output:

< hidden >

Output:

Vehicle: Car
Registration Number: DEF456
Owner: Emily

Number of Doors: 2
Vehicle: Truck
Registration Number: GHI789
Owner: Dave
Cargo Capacity: 8000kg
Vehicle: Motorcycle
Registration Number: JKL012
Owner: Carol
Engine Capacity: 400cc
Vehicle: Car
Registration Number: MNO345
Owner: Frank
Number of Doors: 4



Compilation Status: Passed

Execution Time:

0.117s

18. Unique User Activity Tracker

Problem Statement: You are tasked with developing a system to track unique user activities in a social media application. Each user can perform various activities, and you need to ensure that each activity is only logged once per user. Your task is to create a class that maintains user activity logs, ensuring that each activity for a user is unique.

Description: Implement a class `UserActivityTracker` with methods to add activities and display unique activities. Use a `Map<String, Set<String>>` where the key is the `userId` and the value is a Set of activities performed by that user. Implement the following methods: `addActivity(String userId, String activity)`: Adds an activity for the specified user. If the activity already exists for the user, it should not be added again. `displayActivities(String userId)`: Displays all unique activities performed by the specified user in a sorted order.

Input Format: The first line contains an integer `n`, the number of operations. Each of the next `n` lines contains a command followed by the necessary details: `ADD_ACTIVITY userId activity` `DISPLAY_ACTIVITIES userId` **Output Format:** For each `DISPLAY_ACTIVITIES` command, output the list of unique activities for the specified user, each on a new line in sorted order. If the user has no activities, print "No activities found."

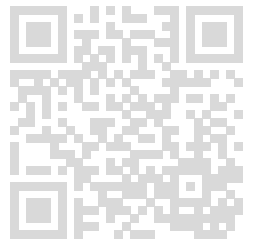
Sample Input: 6
ADD_ACTIVITY user1 login
ADD_ACTIVITY user1 post
ADD_ACTIVITY user1 login
ADD_ACTIVITY user2 login
DISPLAY_ACTIVITIES user1
DISPLAY_ACTIVITIES user2
Sample Output: login
post
login

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA



Source Code:

```
import java.util.*;

class UserActivityTracker {
//..... YOUR CODE STARTS HERE .....

    ArrayList<String> userID=new ArrayList<>();
    ArrayList<ArrayList<String>> activities=new ArrayList<>();

    public void addAct(String id,String act){
        for(int i=0;i<userID.size();i++){
            if(userID.get(i).equals(id)){
                for(int j=0;j<activities.get(i).size();j++){
                    if(activities.get(i).get(j).equals(act)){
                        return;
                    }else{
                        activities.get(i).add(act);
                        return;
                    }
                }
            }
        }
        userID.add(id);
        ArrayList<String> newActivityList = new ArrayList<>();
        newActivityList.add(act); // Add the first activity for this user
        activities.add(newActivityList);
    }

    public void display(String id){
        for(int i=0;i<userID.size();i++){
            if(userID.get(i).equals(id)){
                for(int j=0;j<activities.get(i).size();j++){
                    System.out.println(activities.get(i).get(j));
                }
            }
        }
    }

//..... YOUR CODE ENDS HERE .....
}

public class Main {
    public static void main(String[] args) {
        //..... YOUR CODE STARTS HERE .....

        Scanner sc=new Scanner(System.in);

        int n=sc.nextInt();
```



```
sc.nextLine();
UserActivityTracker a=new UserActivityTracker();
for(int i=0;i<n;i++){
String[] data=sc.nextLine().split(" ");
if(data[0].equals("ADD_ACTIVITY")){
a.addAct(data[1],data[2]);
}else if(data[0].equals("DISPLAY_ACTIVITIES")){
a.display(data[1]);
}else{
System.out.println("Wrong value entered");
}
}
```

```
//..... YOUR CODE ENDS HERE .....
}
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

login
post
login

Compilation Status: Passed

Execution Time:

0.103s

TestCase2:

Input:

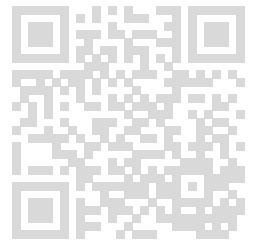
< hidden >

Expected Output:

< hidden >

Output:

login
post

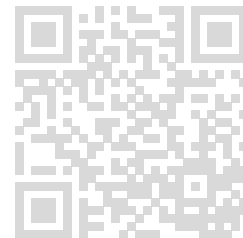


Abhishek Sharma (sweetisharma9673441@gmail.com)

Compilation Status: Passed

Execution Time:

0.102s



19. Exception Handling with Multiple Catch Blocks

Problem Statement: Write a Java program to handle multiple exceptions. The program should take two integers as input from the user and perform division. If the input is invalid, throw a `NumberFormatException`. If the second integer is zero, throw an `ArithmeticException`. Ensure that the program uses multiple catch blocks to handle these exceptions and always prints a final message using the finally block.

Description: The program should prompt the user to enter two integers. It should then attempt to divide the first integer by the second. If the user enters a non-integer value, handle the `NumberFormatException`. If the second integer is zero, handle the `ArithmeticException`. Regardless of whether an exception occurs or not, print "Operation Completed" using the finally block.

Input Format: The first line contains an integer, a. The second line contains an integer, b. **Output Format:** If a and b are valid integers and b is not zero, print the result of a / b. If a or b is not a valid integer, print "Invalid input". If b is zero, print "Cannot divide by zero". Always print "Operation Completed" at the end.

Sample Input: 102 **Sample Output:** 50 **Operation Completed**

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

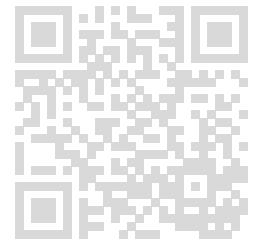
Language Used: JAVA

Source Code:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        //..... YOUR CODE STARTS HERE .....
        String inputA,inputB;
        int a,b;
        try{
            inputA = scanner.nextLine();
            a = Integer.parseInt(inputA); // May throw NumberFormatException

            inputB = scanner.nextLine();
            b = Integer.parseInt(inputB); // May throw NumberFormatException
```



```
// Attempt to divide
int result = a / b; // May throw ArithmeticException
System.out.println(result);
}catch (NumberFormatException e) {
System.out.println("Invalid input");
} catch (ArithmeticException e) {
System.out.println("Cannot divide by zero");
} finally {
// This block will execute regardless of exceptions
System.out.println("Operation Completed");
}
```

```
//..... YOUR CODE ENDS HERE .....
}
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

5
Operation Completed

Compilation Status: Passed

Execution Time:

0.093s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

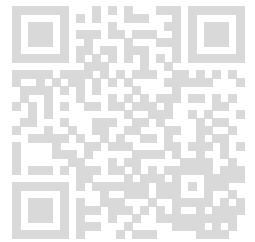
Output:

Invalid input
Operation Completed

Compilation Status: Passed

Execution Time:

0.088s



20. Custom Exception for Age Validation

Problem Statement:Create a custom exception called InvalidAgeException. Write a Java program that takes the age of a user as input and throws this custom exception if the age is less than 18. Ensure the program catches the exception and prints an appropriate message.

Description:Define a custom exception InvalidAgeException that extends the Exception class. The program should prompt the user to enter their age. If the age is less than 18, throw the InvalidAgeException with a message "Age must be 18 or older". Catch this exception and print the message to the user.

Input Format:The first line contains an integer, age.**Output Format:**If age is 18 or older, print "Age is valid".If age is less than 18, print "Age must be 18 or older".

Sample Input:20**Sample Output:**Age is valid

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.Scanner;

class InvalidAgeException extends Exception {
//..... YOUR CODE STARTS HERE .....

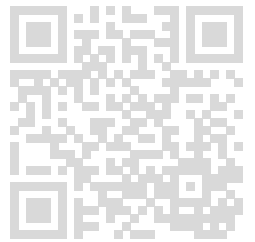
InvalidAgeException(String msg){
super(msg);
};

//..... YOUR CODE ENDS HERE .....
}

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

try{int age=scanner.nextInt();
```

```
if(age<18){
throw new InvalidAgeException("Age must be 18 or older");
}
System.out.println("Age is valid");
}catch (InvalidAgeException e){
System.out.println(e.getMessage());
}
//..... YOUR CODE ENDS HERE .....
}
}
```



Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Age is valid

Compilation Status: Passed

Execution Time:

0.095s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Age must be 18 or older

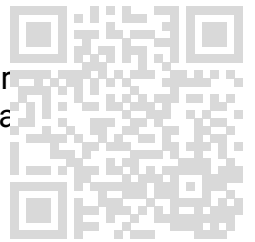
Compilation Status: Passed

Execution Time:

0.088s

21. Nested Exception Handling

Problem Statement: Write a Java program that reads a list of integers from the user and calculates their sum. If any non-integer value is entered, handle the exception and continue reading the next value. Additionally, ensure that the program terminates gracefully by printing the final sum, even if an exception occurs during the input process.



Description: Your task is to implement nested exception handling in Java. The program should continue reading integers from the user until a non-integer value is entered. If a non-integer value is encountered, the program should catch the exception and continue reading the next value. Use nested try-catch blocks to achieve this. Finally, ensure that the program prints the sum of all entered integers using a finally block.

Input Format: The input consists of a sequence of integers and non-integer values entered by the user. **Output Format:** The output should display the sum of all entered integers.

Sample Input: 1 2 3 a 4 **Sample Output:** Sum of entered integers: 10

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.Scanner;

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

        int sum=0;
        try{while(true){
            try{
                String in=scanner.next();
                int v=Integer.parseInt(in);
                sum+=v;
            }catch(NumberFormatException e){
                break;
            }
        }
        String in=scanner.next();
        int v=Integer.parseInt(in);
        sum+=v;

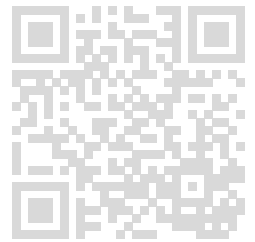
    }finally{
        System.out.println("Sum of entered integers: "+sum);
    }
```

```
}
```

```
//..... YOUR CODE ENDS HERE .....
```

```
}
```

```
}
```



Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Sum of entered integers: 10

Compilation Status: Passed

Execution Time:

0.098s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Sum of entered integers: 25

Compilation Status: Passed

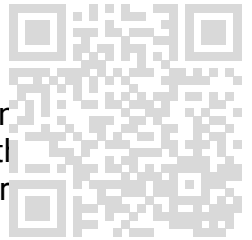
Execution Time:

0.102s

22. Custom Exception for Age Verification

Problem Statement: Create a Java program that verifies the age entered by the user. If the age is below 18, throw a custom exception called `UnderageException`. Handle the exception and display an appropriate message.

Description:Your task is to implement a custom exception in Java. The program should read the age from the user. If the age is below 18, throw a custom exception called UnderageException. Catch the exception and display a message indicating that the user is underage. Ensure that the program continues to execute gracefully after handling the exception.



Input Format:The input consists of a single integer representing the age.
Output Format:The output should display an appropriate message based on the age verification.

Sample Input 1:17**Sample output 1:**UnderageException: Age 17 is below the legal age limit.

Sample Input 2:20**Sample output 2:**Age verification successful.

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.Scanner;

class UnderageException extends Exception {
//..... YOUR CODE STARTS HERE .....

UnderageException(String msg){
super(msg);
}

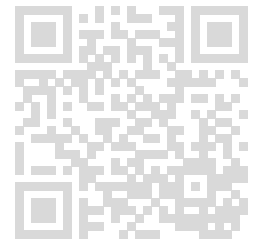
//..... YOUR CODE ENDS HERE .....
}

public class Main {
public static void main(String[] args) {
//..... YOUR CODE STARTS HERE .....

Scanner sc=new Scanner(System.in);
try{
int age=sc.nextInt();
if(age<18){
throw new UnderageException("UnderageException: Age "+age+" is below the legal age limit.");
}
System.out.println("Age verification successful.");
}catch(UnderageException e){
System.out.println(e.getMessage());
}
}
```



```
//..... YOUR CODE ENDS HERE .....  
}  
}
```



Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

UnderageException: Age 17 is below the legal age limit.

Compilation Status: Passed

Execution Time:

0.119s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Age verification successful.

Compilation Status: Passed

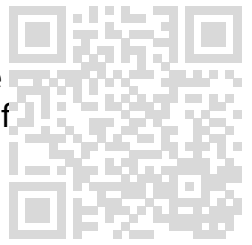
Execution Time:

0.093s

23. Exception Handling in Array Operations

Problem Statement: Write a Java program that performs array operations. The program should read an integer n from the user, create an array of size n, and allow the user to fill the array. If the user tries to access an index outside the array bounds, handle the `ArrayIndexOutOfBoundsException` and display an appropriate message.

Description:Your task is to implement exception handling for array operations. The program should read the size of the array from the user and then allow the user to fill the array with integers. If the user tries to access or modify an index outside the bounds of the array, catch the `ArrayIndexOutOfBoundsException` and display a message indicating the error. Ensure the program continues to run gracefully after handling the exception.



Input Format:The first input is an integer `n` representing the size of the array.The next `n` inputs are the elements of the array.The next input is the index of the array.**Output Format:**The output should display the elements of the array or an appropriate error message if an out-of-bounds access occurs.

Sample Input 1:1 1002**Sample Output 1:**`ArrayIndexOutOfBoundsException: Index 2 out of bounds for length 1`

Sample Input 2:310 20 30**Sample Output 2:**`Element at index 1: 20`

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

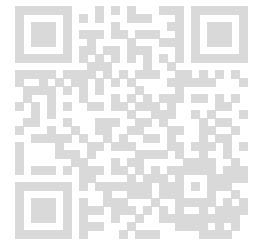
```
import java.util.Scanner;

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

        int n=scanner.nextInt();
        scanner.nextLine();
        int[] arr=new int[n];
        String[] vals=scanner.nextLine().split(" ");
        for(int i=0;i<n;i++){
            arr[i]=Integer.parseInt(vals[i]);
        }
        try{
            int d=scanner.nextInt();
            System.out.println("Element at index "+d+": "+arr[d]);
        }catch(ArrayIndexOutOfBoundsException e){
            System.out.println("ArrayIndexOutOfBoundsException: "+e.getMessage());
            // System.out.println(" index "+d+" out of bound for length "+n);
        }

        //..... YOUR CODE ENDS HERE .....
```

```
}  
}
```



Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

ArrayIndexOutOfBoundsException: Index 2 out of bounds for length 1

Compilation Status: Passed

Execution Time:

0.099s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Element at index 1: 20

Compilation Status: Passed

Execution Time:

0.116s

24. Handling Multiple Exceptions

Problem Statement: Create a Java program that reads two integers from the user and performs division. Handle `ArithmeticException` for division by zero and `NumberFormatException` for invalid integer input. Ensure the program prints a message for each exception and continues execution gracefully.

Description: Your task is to handle multiple exceptions in a Java program. The program should read two integers from the user and perform division. If the user enters a non-integer value, catch the `NumberFormatException` and display an

appropriate message. If the user attempts to divide by zero, catch the `ArithmeticException` and display an appropriate message. Ensure that the program continues to execute gracefully after handling each exception.

Input Format: The first input is the dividend. The second input is the divisor. Output Format: The output should display the result of the division or an appropriate error message if an exception occurs.

Sample Input 1: 102 Sample Output 1: Result: 5

Sample Input 2: 10 0 Sample Output 2: `ArithmeticException: Division by zero is not allowed.`

Sample Input 3: a 5 Sample Output 3: `NumberFormatException: Invalid input. Please enter integers only.`

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.Scanner;

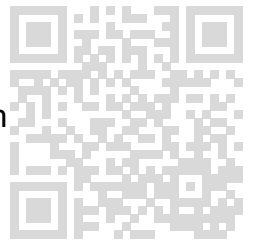
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        //..... YOUR CODE STARTS HERE .....
        String inputA, inputB;
        int a = 0, b = 0;

        try {
            // Prompt user for input
            inputA = scanner.next();
            a = Integer.parseInt(inputA); // May throw NumberFormatException

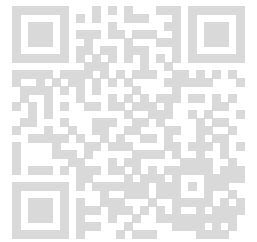
            inputB = scanner.next();
            b = Integer.parseInt(inputB); // May throw NumberFormatException

            // Attempt to divide
            int result = a / b; // May throw ArithmeticException
            System.out.println("Result: "+result);

        } catch (NumberFormatException e) {
            System.out.println("NumberFormatException: Invalid input. Please enter integers only.");
        } catch (ArithmeticException e) {
            System.out.println("ArithmeticException: Division by zero is not allowed.");
        }
    }
}
```



```
}  
//..... YOUR CODE ENDS HERE .....  
  
scanner.close();
```



```
//..... YOUR CODE ENDS HERE .....  
}  
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Result: 5

Compilation Status: Passed

Execution Time:

0.106s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

ArithmeticException: Division by zero is not allowed.

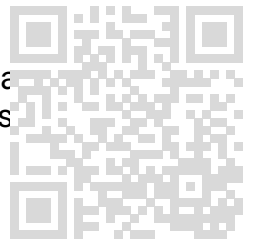
Compilation Status: Passed

Execution Time:

0.082s

25. Custom Exception for Invalid User Input

Problem Statement: You are developing a user management system where users can register with a unique username. Implement a system where a custom exception is thrown when a user attempts to register with a username that contains forbidden characters or is too short.



Description: Create a custom exception named `InvalidUsernameException` that extends the `Exception` class. This exception should be thrown if the username is less than 5 characters long or contains any non-alphanumeric characters. The `UserManager` class should handle this exception and inform the user of the error.

Input Format: A single line of input, the username to be validated. **Output Format:** If the username is valid, output "Username registered successfully." If the username is invalid, output "Invalid username: [error details]."

Sample Input: john_doe **Sample Output:** Invalid username: Contains non-alphanumeric characters.

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.Scanner;

// Custom Exception Class
class InvalidUsernameException extends Exception {
//..... YOUR CODE STARTS HERE .....

InvalidUsernameException(String s){
super(s);
}

//..... YOUR CODE ENDS HERE .....
}

// UserManager Class
class UserManager {
//..... YOUR CODE STARTS HERE .....

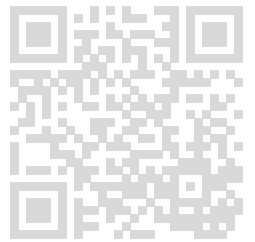
//..... YOUR CODE ENDS HERE .....
}

// Main Class
public class Main {
public static void main(String[] args) {
```

//..... YOUR CODE STARTS HERE

```
Scanner sc=new Scanner(System.in);
try{String s=sc.next();
if(s.length()<5){
throw new InvalidUsernameException("Too short.");
}else if(!s.matches("[a-zA-z0-9]")){
throw new InvalidUsernameException("Contains non-alphanumeric characters.");
}
System.out.println("Username registeres successfully");
}catch(InvalidUsernameException e){
System.out.println("Invalid username: "+e.getMessage());
}

//..... YOUR CODE ENDS HERE .....
```



Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Invalid username: Contains non-alphanumeric characters.

Compilation Status: Passed

Execution Time:

0.09s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

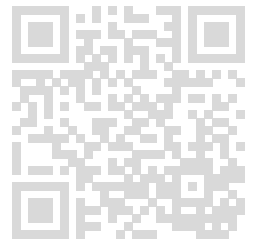
Output:

Invalid username: Too short.

Compilation Status: Passed

Execution Time:

0.096s



26. Custom Exception for Bank Account Balance

Problem Statement: You are designing a banking application where users can withdraw money from their account. Implement a custom exception that is thrown if a withdrawal amount exceeds the account balance.

Description: Create a custom exception named `InsufficientFundsException` that extends `Exception`. This exception should be thrown if a withdrawal attempt exceeds the account balance. The `BankAccount` class should handle this exception and provide appropriate feedback.

Input Format: The initial balance of the account (a floating-point number). The withdrawal amount (a floating-point number). **Output Format:** If the withdrawal amount is valid, output "Withdrawal successful. Remaining balance: [amount]". If the withdrawal amount exceeds the balance, output "Insufficient funds: [error details]".

Sample Input 1: 500600 **Sample Output 1:** Insufficient funds: Withdrawal exceeds balance.

Sample Input 2: 200200 **Sample Output 2:** Withdrawal successful. Remaining balance: 0.0

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.Scanner;

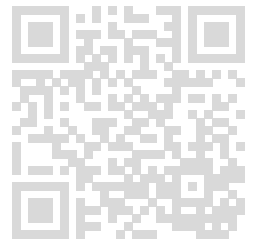
// Custom Exception Class
class InsufficientFundsException extends Exception {
//..... YOUR CODE STARTS HERE .....

InsufficientFundsException(String s){
super(s);
}

//..... YOUR CODE ENDS HERE .....
}
```



```
// BankAccount Class
class BankAccount {
//..... YOUR CODE STARTS HERE .....
```



```
//..... YOUR CODE ENDS HERE .....
}
```

```
// Main Class
public class Main {
public static void main(String[] args) {
//..... YOUR CODE STARTS HERE .....
```

```
Scanner sc=new Scanner(System.in);
float bal=sc.nextInt();
float draw=sc.nextInt();
try{
if(bal<draw){
throw new InsufficientFundsException("Insufficient funds: Withdrawal exceeds
balance.");
}
System.out.println("Withdrawal successful. Remaining balance: "+(bal-draw));
}catch(InsufficientFundsException e){
System.out.println(e.getMessage());
}
//..... YOUR CODE ENDS HERE .....
}
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Insufficient funds: Withdrawal exceeds balance.

Compilation Status: Passed

Execution Time:

0.092s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Withdrawal successful. Remaining balance: 500.0

Compilation Status: Passed

Execution Time:

0.11s

27. Reordering Elements in a List

Problem Statement: Given a list of integers, reorder the list so that all odd numbers come before all even numbers while maintaining the relative order of odd and even numbers.

Description: You need to read a list of integers from the user and reorder it such that all odd numbers come before all even numbers. The relative order of odd and even numbers should be preserved. You must use Java's List interface and iterators.

Input Format: The first line contains an integer, N ($1 \leq N \leq 100$), denoting the number of elements in the list. The second line contains N space-separated integers. **Output Format:** Print the reordered list, with all odd numbers appearing before all even numbers, maintaining their relative order.

Sample Input: 5 4 3 2 1 5 **Sample Output:** 3 1 5 4 2

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        //..... YOUR CODE STARTS HERE .....

        Scanner sc=new Scanner(System.in);
```

```
int n=sc.nextInt();
sc.nextLine();
int[] arr=new int[n];
String[] data=sc.nextLine().split(" ");
for(int i=0;i<arr.length;i++){
arr[i]=Integer.parseInt(data[i]);
}
ArrayList<Integer> odd=new ArrayList<>();
ArrayList<Integer> even=new ArrayList<>();
for(int e: arr){
if(e%2==0){
even.add(e);
}else{
odd.add(e);
}
}
for(int i=0;i<odd.size();i++){
arr[i]=odd.get(i);
}
for(int i=even.size()-1;i>=0;i--){
arr[i+odd.size()]=even.get(i);
}
for(int e: arr){
System.out.print(e+" ");
}

//..... YOUR CODE ENDS HERE .....
}
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

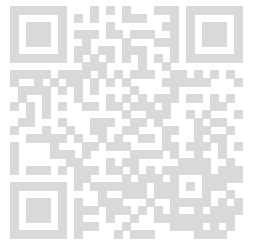
3 1 5 4 2

Compilation Status: Passed

Execution Time:

0.121s

TestCase2:



Abhishek Sharma (sweetisharna9673441@gmail.com)

Input:

< hidden >

Expected Output:

< hidden >

Output:

21 43 65 10 32 54

Compilation Status: Passed

Execution Time:

0.115s

28. Counting Unique Words

Problem Statement:Read a string from the user and count the number of unique words using a Set.

Description:You need to read a string of text from the user and count how many unique words are present in it. Words are considered case-insensitive and separated by spaces. Use Java's Set interface to achieve this.

Input Format:The input consists of a single line containing a string of words.**Output Format:**Print the number of unique words in the string.

Sample Input:Hello world hello**Sample Output:**2

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

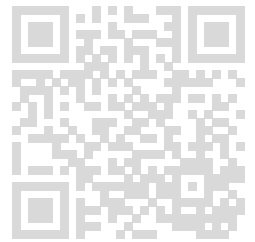
Language Used: JAVA

Source Code:

```
import java.util.*;
```

```
public class Main {  
    public static void main(String[] args) {  
        //..... YOUR CODE STARTS HERE .....
```

```
        Scanner sc=new Scanner(System.in);  
        String[] data=sc.nextLine().split(" ");  
        ArrayList<String> str=new ArrayList<>();  
        int count=0;
```



```
for(int i=0;i<data.length;i++){
if(str.indexOf(data[i])!=-1){
str.add(data[i].toLowerCase());
count++;
}
}
System.out.println(count);
```

```
//..... YOUR CODE ENDS HERE .....
}
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

2

Compilation Status: Passed

Execution Time:

0.09s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

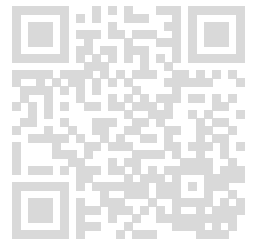
Output:

5

Compilation Status: Passed

Execution Time:

0.089s



Abhishek Sharma (sweetisharma9673441@gmail.com)

29. Intersection of Two Sets

Problem Statement: Find the intersection of two sets of integers.

Description: You need to read two sets of integers from the user and print their intersection. Use Java's Set interface and its operations to achieve this.

Input Format: The first line contains an integer, n ($1 \leq n \leq 100$), denoting the number of elements in the first set. The second line contains n space-separated integers. The third line contains an integer, m ($1 \leq m \leq 100$), denoting the number of elements in the second set. The fourth line contains m space-separated integers. Output Format: Print the elements in the intersection of the two sets, one per line. If there are no common elements, print "No common elements."

Sample Input: 4 1 2 3 4 3 3 4 5 Sample Output: 3 4

Completion Status: Completed

Concepts Included:

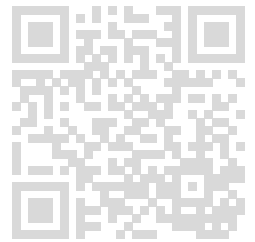
gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        //..... YOUR CODE STARTS HERE .....
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        sc.nextLine();
        int[] arr=new int[n];
        String[] data=sc.nextLine().split(" ");
        for(int i=0;i<arr.length;i++){
            arr[i]=Integer.parseInt(data[i]);
        }
        int n1=sc.nextInt();
        sc.nextLine();
        int[] ar1=new int[n1];
        String[] data1=sc.nextLine().split(" ");
        for(int i=0;i<ar1.length;i++){
            ar1[i]=Integer.parseInt(data1[i]);
        }
        for(int i=0;i<arr.length;i++){
            for(int j=0;j<ar1.length;j++){
                if(arr[i]-ar1[j]==0){
                    System.out.println(arr[i]);
                }
            }
        }
    }
}
```

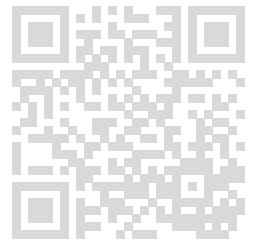


```
}
```

```
//..... YOUR CODE ENDS HERE .....
```

```
}
```

```
}
```



Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

3

4

Compilation Status: Passed

Execution Time:

0.101s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

200

Compilation Status: Passed

Execution Time:

0.091s

30. Creating a Frequency Map

Problem Statement:Read a list of words and create a frequency map of each word.

Description:You need to read a list of words from the user and create a map where

the keys are the words and the values are their respective frequencies. Use Java's Map interface to store and calculate the frequencies.

Input Format: The first line contains an integer, n ($1 \leq n \leq 100$), denoting the number of words. The second line contains n space-separated words. Output Format: Print each word and its frequency, one per line.

Sample Input: 4 apple banana apple grape
Sample Output: banana 1 apple 2 grape 1

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

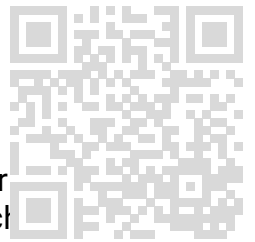
```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        //..... YOUR CODE STARTS HERE .....

        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        sc.nextLine();
        String[] data=sc.nextLine().split(" ");
        ArrayList<ArrayList<String>> items=new ArrayList<>();

        for(int i=0;i<data.length;i++){
            boolean found=false;
            for(int j=0;j<items.size();j++){
                if(items.get(j).get(0).equals(data[i])){
                    found=true;
                    int val=Integer.parseInt(items.get(j).get(1))+1;
                    items.get(j).set(1,val+"");
                }
            }
            if(found==false){
                ArrayList<String> nc=new ArrayList<>();
                nc.add(data[i]);
                nc.add("1");
                items.add(nc);
            }

            // for(int i=0;i<items.size();i++){
            //     System.out.print(items.get(i).get(0)+" ");
            //     System.out.println(items.get(i).get(1));
            // }
```




```
char[] rds=new char[items.size()];
for(int i=0;i<rds.length;i++){
rds[i]=items.get(i).get(0).charAt(1);
}
Arrays.sort(rds);
for(int i=0;i<rds.length;i++){
for(int j=0;j<items.size();j++){
if(rds[i]==items.get(j).get(0).charAt(1)){

System.out.print(items.get(j).get(0)+" ");
System.out.println(items.get(j).get(1));
items.remove(j);
}
}
}
```

```
//..... YOUR CODE ENDS HERE .....
}
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

banana 1
apple 2
grape 1

Compilation Status: Passed

Execution Time:

0.11s

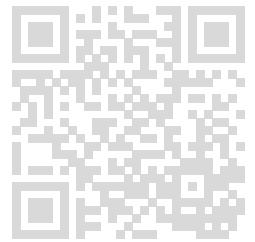
TestCase2:

Input:

< hidden >

Expected Output:

< hidden >



Abhishek Sharma (sweetisharma9673441@gmail.com)

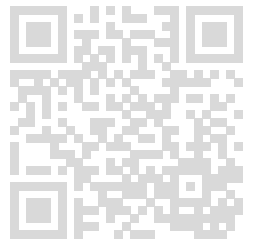
Output:

cat 1
fish 1
dog 3

Compilation Status: Passed

Execution Time:

0.112s



31. Unique Items in a Shopping List

Problem Statement: You are developing a shopping application where users can maintain their shopping lists. The application needs to handle scenarios where users can add, remove, and view items in their shopping list. The unique feature of the application is that it must manage items efficiently while ensuring that no duplicates are allowed and each item is stored in a way that preserves its insertion order.

Description: You need to implement a Java program that:

Allows users to add items to a shopping list. Allows users to remove items from the list. Displays the current shopping list without duplicates, maintaining the order of insertion.

Input Format: First line: Integer N (Number of operations to perform) Next N lines: Each line will contain an operation in the format "ADD <item>" or "REMOVE <item>". The item is a string with spaces. **Output Format:** After processing all operations, print the final shopping list with items in their insertion order, separated by commas.

Sample Input: 5ADD ApplesADD BananasADD ApplesREMOVE BananasADD Grapes
Sample Output: Apples, Grapes

Completion Status: Completed

Concepts Included:

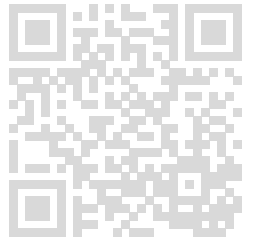
gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.LinkedHashSet;
import java.util.Scanner;
import java.util.Set;

public class Main {
    public static void main(String[] args) {
        //..... YOUR CODE STARTS HERE .....
```



```
Set<String> shoppingList = new LinkedHashSet<>();

Scanner sc = new Scanner(System.in);

// Read the number of operations
int n = sc.nextInt();
sc.nextLine(); // Consume the newline character

// Process each operation
for (int i = 0; i < n; i++) {
    String[] data = sc.nextLine().split(" ", 2); // Split on the first space
    String command = data[0];
    String item = data.length > 1 ? data[1] : "";

    if (command.equals("ADD")) {
        shoppingList.add(item); // Add item to the shopping list
    } else if (command.equals("REMOVE")) {
        shoppingList.remove(item); // Remove item from the shopping list
    }
}

// Output the final shopping list
System.out.println(String.join(" ", shoppingList)); // Join

//..... YOUR CODE ENDS HERE .....
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Apples, Grapes

Compilation Status: Passed

Execution Time:

0.094s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

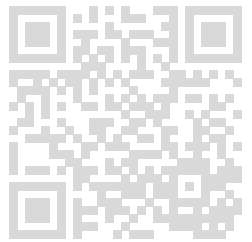
Output:

Apples, Bananas, Grapes

Compilation Status: Passed

Execution Time:

0.094s



32. Frequency of Words in Text

Problem Statement: You are tasked with implementing a feature that counts the frequency of each word in a given text. The twist is that the word frequencies need to be sorted in descending order of frequency, and in case of a tie, the words should be sorted lexicographically.

Description: You need to implement a Java program that:

Reads a block of text from the user. Counts the frequency of each word in the text. Outputs the words and their frequencies sorted by frequency (highest first) and lexicographically in case of ties.

Input Format: A single line of text (can contain multiple words and punctuation). **Output Format:** Each line should contain a word followed by its frequency, sorted by frequency in descending order and lexicographically for tied frequencies.

Sample Input: apple banana apple fruit banana apple **Sample Output:** apple 3 banana 2 fruit 1

Completion Status: Completed

Concepts Included:

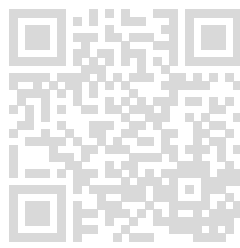
gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        //..... YOUR CODE STARTS HERE .....
```



```

Scanner scanner = new Scanner(System.in);

String text = scanner.nextLine();

String[] words = text.split("[\\s,\\.?!]+");

Map<String, Integer> wordCount = new HashMap<>();

for (String word : words) {
    word = word.toLowerCase();
    wordCount.put(word, wordCount.getOrDefault(word, 0) + 1);
}

// Create a list from the map entries
List<Map.Entry<String, Integer>> entryList = new ArrayList<>(wordCount.entrySet());

// Sort the list
Collections.sort(entryList, (a, b) -> {
    int frequencyCompare = b.getValue().compareTo(a.getValue()); // Sort by frequency
    descending
    if (frequencyCompare != 0) {
        return frequencyCompare; // If frequencies are different, sort by frequency
    }
    return a.getKey().compareTo(b.getKey()); // If frequencies are the same, sort
    lexicographically
});

// Print the sorted word frequencies
for (Map.Entry<String, Integer> entry : entryList) {
    System.out.println(entry.getKey() + " " + entry.getValue());
}

//..... YOUR CODE ENDS HERE .....
}
}

```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```

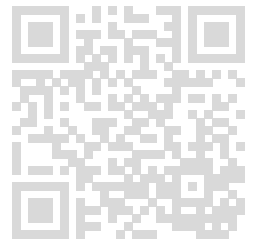
apple 3
banana 2
fruit 1

```

Compilation Status: Passed

Execution Time:

0.113s



TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

hello 3
world 2

Compilation Status: Passed

Execution Time:

0.113s

33. Grouping Words by Length

Problem Statement: Group words by their length from a given list of words.

Description: You need to read a list of words from the user and group them by their length. Use Java's Map interface where the key is the length of the words and the value is a list of words of that length.

Input Format: The first line contains an integer, n ($1 \leq n \leq 100$), denoting the number of words. The second line contains n space-separated words. **Output Format:** Print each length and the corresponding list of words.

Sample Input: 5 cat dog elephant rat bat **Sample Output:** 3: [cat, dog, rat, bat] 8: [elephant]

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.*;
```

```
public class Main {  
    public static void main(String[] args) {  
        //..... YOUR CODE STARTS HERE .....
```

```
        Scanner scanner = new Scanner(System.in);
```

```
        int n = scanner.nextInt();  
        scanner.nextLine();
```

```
        String[] words = scanner.nextLine().split(" ");
```

```
        Map<Integer, List<String>> groupedWords = new HashMap<>();
```

```
        for (String word : words) {  
            int length = word.length();  
            groupedWords.putIfAbsent(length, new ArrayList<>());  
            groupedWords.get(length).add(word);  
        }
```

```
        for (Map.Entry<Integer, List<String>> entry : groupedWords.entrySet()) {  
            System.out.println(entry.getKey() + ": " + entry.getValue());  
        }
```

```
        //..... YOUR CODE ENDS HERE .....
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

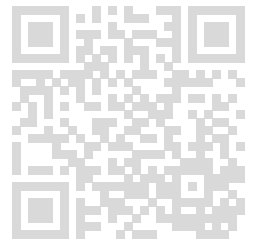
3: [cat, dog, rat, bat]
8: [elephant]

Compilation Status: Passed

Execution Time:

0.11s

TestCase2:



Input:

< hidden >

Expected Output:

< hidden >

Output:

3: [red]
4: [blue]
5: [green]
6: [yellow]

Compilation Status: Passed

Execution Time:

0.113s

34. Finding the Longest Word

Problem Statement: Given a list of words, find the longest word. If multiple words have the same maximum length, print them all.

Description: You need to read a list of words from the user and identify the longest word(s). If there are multiple words with the same maximum length, print all of them. Use Java's List interface and iterators to solve this problem.

Input Format: The first line contains an integer, n ($1 \leq n \leq 100$), denoting the number of words. The second line contains n space-separated words. **Output Format:** Print the longest word(s). If multiple words have the same maximum length, print them all on separate lines.

Sample Input: 5applebananacherrydateguva **Sample Output:** bananacherry

Completion Status: Completed

Concepts Included:

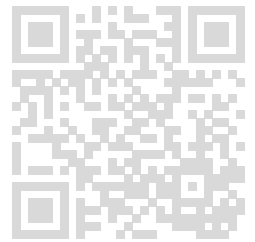
gu 27 3rd semester java programming

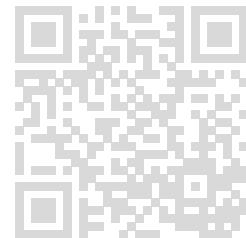
Language Used: JAVA

Source Code:

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        //..... YOUR CODE STARTS HERE .....
```





```
Scanner scanner = new Scanner(System.in);

int n = scanner.nextInt();
scanner.nextLine();

String[] words = new String[n];
for(int i=0;i<n;i++){
    words[i]=scanner.next();
}

List<String> longestWords = new ArrayList<>();
int maxLength = 0;

for (String word : words) {
    int length = word.length();
    if (length > maxLength) {
        maxLength = length;
        longestWords.clear();
        longestWords.add(word);
    } else if (length == maxLength) {
        longestWords.add(word);
    }
}

for (String longestWord : longestWords) {
    System.out.println(longestWord);
}

//..... YOUR CODE ENDS HERE .....
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

banana
cherry

Compilation Status: Passed

Execution Time:

0.091s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

mouse

Compilation Status: Passed

Execution Time:

0.091s

35. Lambda Expressions

QuestionImplement a Java program that uses a lambda expression to calculate the sum of all even numbers and the product of all odd numbers in a list provided by the user. The program should then return the difference between the sum of even numbers and the product of odd numbers.

DescriptionWrite a Java program that:Accepts a list of integers from the user.Uses a lambda expression to compute the sum of all even numbers.Uses another lambda expression to compute the product of all odd numbers.Outputs the difference between the sum of even numbers and the product of odd numbers.

Input FormatAn integer n denoting the number of elements in the list.n space-separated integers representing the list elements.**Output Format**A single integer which is the difference between the sum of even numbers and the product of odd numbers.

Sample Input:51 2 3 4 5**Sample Output:**-9

Completion Status: Completed

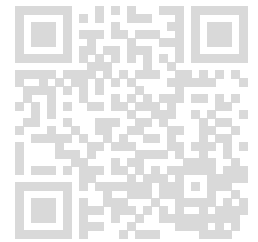
Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.*;
```



```
import java.util.stream.*;
```

```
public class Main {  
    public static void main(String[] args) {  
        //..... YOUR CODE STARTS HERE .....
```

```
        Scanner sc=new Scanner(System.in);  
        int n=sc.nextInt();  
        sc.nextLine();  
        int[] a=new int[n];  
        String[] d=sc.nextLine().split(" ");  
        for(int i=0;i<a.length;i++){  
            a[i]=Integer.parseInt(d[i]);  
        }
```

```
        int oddP=1;  
        int evenS=0;  
        int i=0;  
        while(i<n){  
            if(a[i]%2==0){  
                evenS+=a[i];  
            }else{  
                oddP*=a[i];  
            }  
            i++;  
        }
```

```
        System.out.println(evenS-oddP);
```

```
        //..... YOUR CODE ENDS HERE .....
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

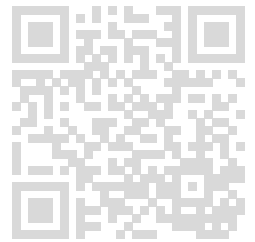
Output:

-9

Compilation Status: Passed

Execution Time:

0.09s



Abhishek Sharma (sweetisharma9673441@gmail.com)

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

-3

Compilation Status: Passed

Execution Time:

0.091s

36. Functional Interfaces Question Create a Java program using a custom functional interface to find the longest string in a list provided by the user. The program should then return the length of this longest string.

Description Write a Java program that: Accepts a list of strings from the user. Defines a custom functional interface with a method to find the longest string. Uses this functional interface to find the longest string in the list. Outputs the length of the longest string.

Input Format An integer n denoting the number of strings in the list. n space-separated strings representing the list elements. Output Format A single integer which is the length of the longest string.

Sample Input: 4 apple banana strawberrry kiwi Sample Output: 10

Completion Status: Completed

Concepts Included:

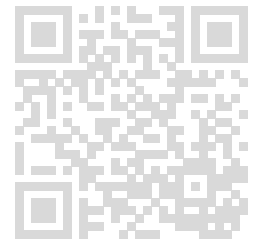
gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.*;
```

```
@FunctionalInterface
interface LongestStringFinder {
//..... YOUR CODE STARTS HERE .....
```



```
int LongestString(List<String> list);
```

```
//..... YOUR CODE ENDS HERE .....  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int n = sc.nextInt();  
        sc.nextLine(); // Consume newline  
        List<String> list = new ArrayList<>();  
        for (int i = 0; i < n; i++) {  
            list.add(sc.nextLine());  
        }  
        //..... YOUR CODE STARTS HERE .....  

```

```
        LongestStringFinder LongestS=(Str)->{  
            int max=0;  
            for(String s: Str){  
                max=(max<s.length())?s.length():max;  
            }  
            return max;  
        };  

```

```
        int lognest=LongestS.LongestString(list);  
        System.out.println(lognest);  

```

```
        //..... YOUR CODE ENDS HERE .....  
    }  
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

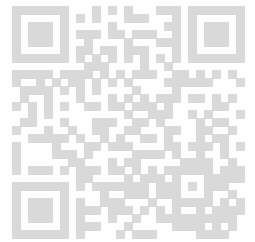
10

Compilation Status: Passed

Execution Time:

0.1s

TestCase2:



Abhishek Sharma (sweetisharna9673441@gmail.com)

Input:

< hidden >

Expected Output:

< hidden >

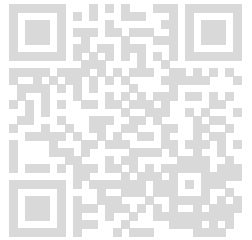
Output:

8

Compilation Status: Passed

Execution Time:

0.094s



37. Local Variable Type Inference (var)

Question Implement a Java program that uses var to store a list of user-input integers and then calculates the average of these integers. The program should handle the list and the average calculation using var.

Description Write a Java program that: Accepts a list of integers from the user. Uses var to declare and initialize the list. Calculates the average of the list elements using var. Outputs the calculated average.

Input Format An integer n denoting the number of elements in the list. n space-separated integers representing the list elements. **Output Format** A single floating-point number which is the average of the list elements.

Sample Input: 4 1 2 3 4 **Sample Output:** 2.5

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

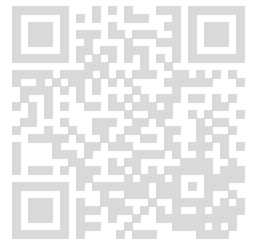
Source Code:

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        var n = sc.nextInt();
        var list = new ArrayList<Integer>();
        for (var i = 0; i < n; i++) {
```

```
list.add(sc.nextInt());
}
//..... YOUR CODE STARTS HERE .....
float sum=0;
for(int e: list){
sum+=e;
}
System.out.println(sum/list.size());

//..... YOUR CODE ENDS HERE .....
}
}
```



Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

2.5

Compilation Status: Passed

Execution Time:

0.088s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

20.0

Compilation Status: Passed

Execution Time:

0.089s

Abhishek Sharma (sweetisharma9673441@gmail.com)

38. Stream API

QuestionCreate a Java program that uses the Stream API to read a list of integers from the user, filter out the prime numbers, sort them in descending order, and then print the sorted list.

DescriptionWrite a Java program that:Accepts a list of integers from the user.Uses the Stream API to filter out prime numbers.Sorts the filtered prime numbers in descending order.Outputs the sorted list of prime numbers.

Input FormatAn integer n denoting the number of elements in the list.n space-separated integers representing the list elements.**Output Format**The sorted list of prime numbers in descending order, each number on a new line.

Sample Input:52 3 4 5 6**Sample Output:**532

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

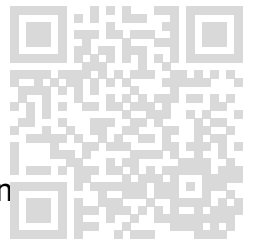
Source Code:

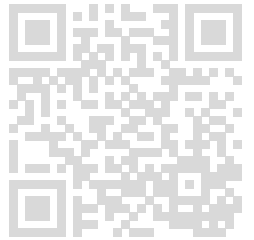
```
import java.util.*;
import java.util.stream.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        List<Integer> list = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            list.add(sc.nextInt());
        }
        //..... YOUR CODE STARTS HERE .....

        List<Integer> h=new ArrayList<>();
        for(int e: list){
            if(isPrime(e)){
                h.add(e);
            }
        }
        Collections.sort(h, Collections.reverseOrder());
        for(int e: h){
            System.out.println(e);
        }

        //..... YOUR CODE ENDS HERE .....
    }
}
```





```
public static boolean isPrime(int num) {  
//..... YOUR CODE STARTS HERE .....
```

```
if(num==1){  
return false;  
}  
for(int i=2;i<=num/2;i++){  
if(num%i==0){  
return false;  
}
```

```
}  
return true;
```

```
//..... YOUR CODE ENDS HERE .....
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

5
3
2

Compilation Status: Passed

Execution Time:

0.086s

TestCase2:

Input:

< hidden >

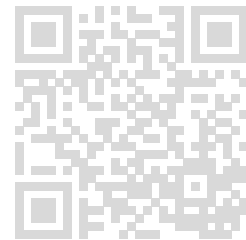
Expected Output:

< hidden >

Compilation Status: Passed

Execution Time:

0.093s



39. Stream API and Local Variable Type Inference (var)

Question Develop a Java program that reads a list of words from the user, uses the Stream API to filter out words with more than 5 characters, converts the filtered words to uppercase, and then prints the resulting list. Use var to declare all local variables.

Description Write a Java program that: Accepts a list of words from the user. Uses the Stream API to filter out words with more than 5 characters. Converts the filtered words to uppercase. Outputs the resulting list of uppercase words.

Input Format An integer n denoting the number of words in the list. n space-separated words representing the list elements. **Output Format** The resulting list of uppercase words, each word on a new line.

Sample Input: 4 apple banana pear fig **Sample output:** PEARFIG

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.*;
import java.util.stream.*;

public class Main {
    public static void main(String[] args) {
        var sc = new Scanner(System.in);
        var n = sc.nextInt();
        sc.nextLine();
        var list = new ArrayList<String>();
        for (var i = 0; i < n; i++) {
            list.add(sc.nextLine());
        }
        //..... YOUR CODE STARTS HERE .....

        var result = list.stream()
            .filter(word -> word.length() < 5)
            .map(String::toUpperCase)
            .collect(Collectors.toList());

        for (var word : result) {
```

```
System.out.println(word);  
}
```

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

```
//..... YOUR CODE ENDS HERE .....  
}  
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

PEAR
FIG

Compilation Status: Passed

Execution Time:

0.099s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

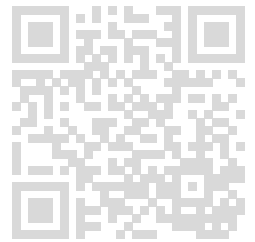
ANT

Compilation Status: Passed

Execution Time:

0.097s

40. Lambda Expressions



Question Write a Java program using a lambda expression to determine if the list of strings provided by the user contains any palindromes. The program should return "Yes" if there is at least one palindrome, otherwise "No".

Description Write a Java program that: Accepts a list of strings from the user. Uses a lambda expression to check if any string in the list is a palindrome. Outputs "Yes" if there is at least one palindrome, otherwise "No".

Input Format An integer n denoting the number of strings in the list. n space-separated strings representing the list elements. **Output Format** "Yes" if there is at least one palindrome in the list, otherwise "No".

Sample Input: 4 level racecar hello world **Sample Output:** Yes

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.*;
import java.util.function.*;

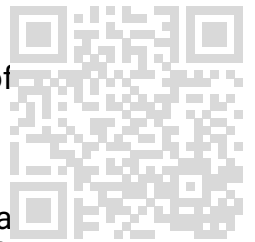
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine();
        List<String> list = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            list.add(sc.nextLine());
        }
        //..... YOUR CODE STARTS HERE .....

        boolean hasPalindrome = list.stream()
            .anyMatch(s -> isPalindrome(s));

        // Output the result
        if (hasPalindrome) {
            System.out.println("Yes");
        } else {
            System.out.println("No");
        }

        //..... YOUR CODE ENDS HERE .....

        sc.close(); // Close the scanner
    }
}
```



```
// Helper method to check if a string is a palindrome
private static boolean isPalindrome(String s) {
String reversed = new StringBuilder(s).reverse().toString();
return s.equalsIgnoreCase(reversed);
}
```

```
//..... YOUR CODE ENDS HERE .....
}
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Yes

Compilation Status: Passed

Execution Time:

0.091s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

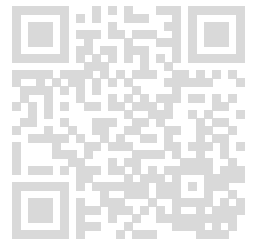
No

Compilation Status: Passed

Execution Time:

0.093s

41. Local Variable Type Inference (var)



Abhishek Sharma (sweetisharma9673441@gmail.com)

Question Write a Java program that uses var to store a list of user-input integers, and then finds the second largest number in the list. The program should use var to handle all variables.

Description Write a Java program that: Accepts a list of integers from the user. Uses var to declare and initialize the list and all other variables. Finds the second largest number in the list. Outputs the second largest number.

Input Format An integer n denoting the number of elements in the list. n space-separated integers representing the list elements. **Output Format** A single integer which is the second largest number in the list.

Sample Input: 51 2 3 4 5 **Sample Output:** 4

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.*;
```

```
public class Main {  
    public static void main(String[] args) {  
        //..... YOUR CODE STARTS HERE .....
```

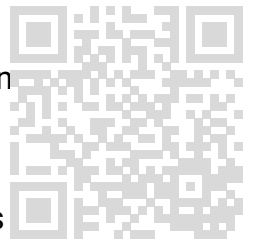
```
        var sc=new Scanner(System.in);  
        var n=sc.nextInt();  
        sc.nextLine();  
        var s=sc.nextLine().split(" ");  
        var arr=new int[n];  
        for(int i=0;i<arr.length;i++){  
            arr[i]=Integer.parseInt(s[i]);  
        }  
        Arrays.sort(arr);  
        System.out.println(arr[arr.length-2]);
```

```
        //..... YOUR CODE ENDS HERE .....
```

Compilation Details:

TestCase1:

Input:



< hidden >

Expected Output:

< hidden >

Output:

4

Compilation Status: Passed

Execution Time:

0.09s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

20

Compilation Status: Passed

Execution Time:

0.092s

42. Stream API

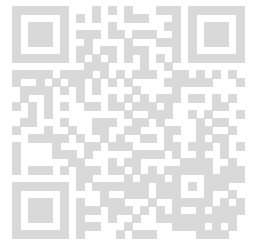
QuestionCreate a Java program that uses the Stream API to read a list of integers from the user, squares each number, filters out the squares that are not divisible by 3, and then prints the resulting list.

DescriptionWrite a Java program that:Accepts a list of integers from the user.Uses the Stream API to square each number.Filters out the squares that are not divisible by 3.Outputs the resulting list.

Input FormatAn integer n denoting the number of elements in the list.n space-separated integers representing the list elements.**Output Format**The resulting list of squares that are divisible by 3, each number on a new line.

Sample Input:41 2 3 4**Sample Output:**9

Completion Status: Completed



Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.*;
import java.util.stream.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        List<Integer> list = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            list.add(sc.nextInt());
        }
        //..... YOUR CODE STARTS HERE .....

        List<Integer> result = list.stream()
            .map(num -> num * num)
            .filter(square -> square % 3 == 0)
            .collect(Collectors.toList());

        result.forEach(System.out::println);

        sc.close();

        //..... YOUR CODE ENDS HERE .....
    }
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

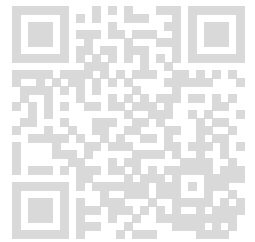
< hidden >

Output:

9

Compilation Status: Passed

Execution Time:



Abhishek Sharma (sweetisharma9673441@gmail.com)

0.095s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

36

Compilation Status: Passed

Execution Time:

0.098s

43. Custom Comparator with Lambda Expression Problem

StatementCreate a custom comparator using a lambda expression to sort a list of Person objects based on their age and then by their name in alphabetical order.

DescriptionYou are given a list of Person objects, where each Person has a name (String) and an age (int). Write a program that sorts the list first by age in descending order. If two people have the same age, sort them by their name in ascending order. Implement this using a lambda expression with a custom comparator.

Input FormatThe first line contains an integer n, the number of Person objects. The next n lines each contain two values: a name (String) and age (int) of a Person. **Output Format**Print the sorted list of Person objects in the specified order.

Sample Input:5Alice 30Bob 25Charlie 30Dave 22Eve 25

Sample Output:Alice 30Charlie 30Bob 25Eve 25Dave 22

Completion Status: Completed

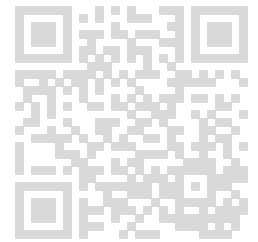
Concepts Included:

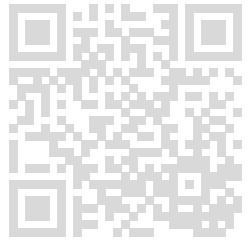
gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.*;
```





```
import java.util.function.Function;

class Person {
    String name;
    int age;

    Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public String toString() {
        return name + " " + age;
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine(); // Consume the newline

        List<Person> people = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            String[] input = sc.nextLine().split(" ");
            String name = input[0];
            int age = Integer.parseInt(input[1]);
            people.add(new Person(name, age));
        }

        //..... YOUR CODE STARTS HERE .....
        people.sort(Comparator
            .comparingInt((Person p) -> p.age).reversed()
            .thenComparing(p -> p.name)
        );

        //..... YOUR CODE ENDS HERE .....

        people.forEach(System.out::println);
    }
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Alice 30
Charlie 30
Bob 25
Eve 25
Dave 22

Compilation Status: Passed

Execution Time:

0.127s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Eve 25
Dave 22

Compilation Status: Passed

Execution Time:

0.128s

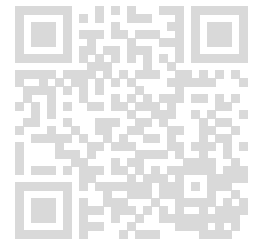
44. Filtering Even Numbers using Functional Interface

Problem Statement Filter out even numbers from a list of integers using a lambda expression with a custom functional interface.

Description You are given a list of integers. Define a functional interface Condition with a method test(int number) that returns a boolean. Use this interface with a lambda expression to filter out even numbers from the list.

Input Format The first line contains an integer n, the number of integers. The next line contains n integers separated by spaces. **Output Format** Print the list of odd numbers, each on a new line.

Sample Input: 6 1 2 3 4 5 6 **Sample Output:** 1 3 5



Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.*;
import java.util.function.Predicate;
```

```
@FunctionalInterface
interface Condition {
//..... YOUR CODE STARTS HERE .....
```

```
boolean test(int n);
```

```
//..... YOUR CODE ENDS HERE .....
}
```

```
public class Main {
public static void main(String[] args) {
//..... YOUR CODE STARTS HERE .....
```

```
Scanner scanner = new Scanner(System.in);
```

```
// Step 2: Input the number of integers
int n = scanner.nextInt();
List<Integer> numbers = new ArrayList<>();
```

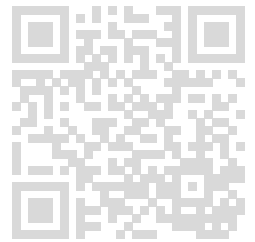
```
// Step 3: Input the integers into the list
for (int i = 0; i < n; i++) {
numbers.add(scanner.nextInt());
}
```

```
// Step 4: Create a lambda expression to filter even numbers
Condition isOdd = (int number) -> number % 2 != 0;
```

```
// Step 5: Filter the odd numbers and print them
for (int number : numbers) {
if (isOdd.test(number)) {
System.out.println(number);
}
}
```

```
scanner.close();
```

```
//..... YOUR CODE ENDS HERE .....
}
}
```



Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

1
3
5

Compilation Status: Passed

Execution Time:

0.091s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

5

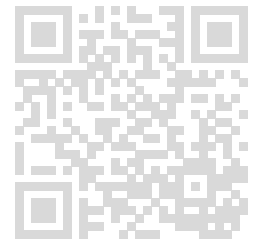
Compilation Status: Passed

Execution Time:

0.091s

45. Applying Function to a List
Problem Statement Use a lambda expression to apply a custom function to a list of strings to transform each string to uppercase and then sort the list in reverse alphabetical order.

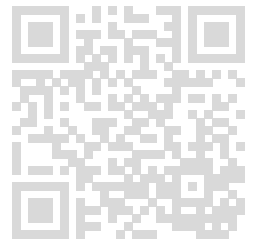
Description You are given a list of strings. Define a functional interface StringFunction with a method apply(String s) that transforms a string. Use this interface with a lambda expression to convert each string to uppercase. Finally, sort the transformed list in reverse alphabetical order.



Input FormatThe first line contains an integer n, the number of strings.The next n lines each contain a string.**Output Format**Print the transformed and sorted list of strings, each on a new line.

Sample Input:4helloworldjavalamdba

Sample Output:WORLDLAMBDAJAVAHELLO



Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.*;
import java.util.function.Function;
```

```
@FunctionalInterface
interface StringFunction {
    String apply(String s);
}
```

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine(); // Consume the newline
```

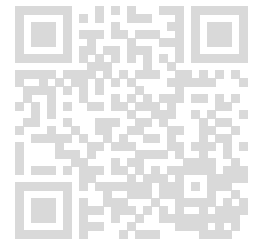
```
List<String> strings = new ArrayList<>();
for (int i = 0; i < n; i++) {
    strings.add(sc.nextLine());
}
```

```
//..... YOUR CODE STARTS HERE .....
```

```
StringFunction use=(s)->s.toUpperCase();
List<String> transformedList = new ArrayList<>();
for (String s : strings) {
    transformedList.add(use.apply(s));
}
Collections.sort(transformedList,Collections.reverseOrder());
for(String e: transformedList){
    System.out.println(e);
}
```

```
//..... YOUR CODE ENDS HERE .....
```

```
}  
}
```



Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

WORLD
LAMBDA
JAVA
HELLO

Compilation Status: Passed

Execution Time:

0.087s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

WORLD
PROGRAMME
JAVA

Compilation Status: Passed

Execution Time:

0.094s

46. Combining Two Lists with Functional Interfaces

Problem Statement Combine two lists of integers by applying a lambda expression that adds corresponding elements from the two lists.

DescriptionYou are given two lists of integers. Define a functional interface Combiner with a method combine(int a, int b) that combines two integers. Use this interface with a lambda expression to add corresponding elements from the two lists. The lists are guaranteed to be of the same length.

Input FormatThe first line contains an integer n, the number of integers in each list. The next n lines contain integers for the first list. The next n lines contain integers for the second list.
Output FormatPrint the resulting list after combining each corresponding element.

Sample Input:3123456**Sample Output:**579

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.*;
import java.util.function.BiFunction;

@FunctionalInterface
interface Combiner {
    //..... YOUR CODE STARTS HERE .....

    List<Integer> join(List<Integer> a, List<Integer> b);

    //..... YOUR CODE ENDS HERE .....
}

public class Main {
    public static void main(String[] args) {
        //..... YOUR CODE STARTS HERE .....

        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        List<Integer> a = new ArrayList<>();
        List<Integer> b = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            a.add(sc.nextInt());
        }
        for (int i = 0; i < n; i++) {
            b.add(sc.nextInt());
        }
        Combiner combiner = (l1, l2) -> {
            List<Integer> res = new ArrayList<>();
            for (int i = 0; i < l1.size(); i++) {
                res.add(l1.get(i) + l2.get(i));
            }
        };
        List<Integer> result = combiner.join(a, b);
        for (Integer i : result) {
            System.out.print(i + " ");
        }
        System.out.println();
    }
}
```



```
}  
return res;  
};
```

```
List<Integer> res=combine.join(a,b);  
for(int e: res){  
System.out.println(e);  
}
```

```
//..... YOUR CODE ENDS HERE .....  
}  
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

5
7
9

Compilation Status: Passed

Execution Time:

0.091s

TestCase2:

Input:

< hidden >

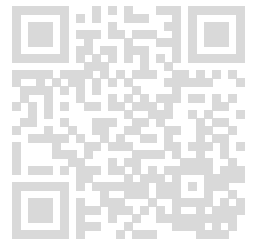
Expected Output:

< hidden >

Output:

6
2
4

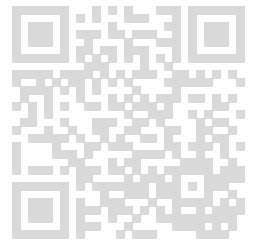
Compilation Status: Passed



Abhishek Sharma (sweetisharma9673441@gmail.com)

Execution Time:

0.089s



47. Wrapper Class Manipulation

Problem Statement: Write a Java program that takes a string input representing a list of integers (e.g., "1 2 3 4 5") and performs the following operations:

Convert the string into an array of integers using Integer wrapper class methods. Compute the sum of the integers and print it. Print the maximum and minimum values from the array using Integer methods. Print the average of the integers rounded to two decimal places.

Description: Your task is to utilize Integer wrapper class methods to parse the input string, perform computations, and display results. The input string may contain both positive and negative integers separated by spaces.

Input Format: A single line of input containing space-separated integers. **Output Format:** A single line containing the sum of integers. A second line containing the maximum and minimum values. A third line containing the average of the integers rounded to two decimal places.

Sample Input: 10 -3 5 7 2 **Sample Output:** 21 10 -34.20

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.Scanner;

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

        String[] data=scanner.nextLine().split(" ");
        int sum=0;
        int max=Integer.parseInt(data[0]);
        int min=Integer.parseInt(data[0]);
        double avg=0;
        for(int i=0;i<data.length;i++){
            sum+=Integer.parseInt(data[i]);
            max=(max<Integer.parseInt(data[i]))?Integer.parseInt(data[i]):max;
            min=(min>Integer.parseInt(data[i]))?Integer.parseInt(data[i]):min;
        }
        System.out.println(sum);
        System.out.println(max+" "+min);
        System.out.println(avg);
    }
}
```

```
}  
avg=(double)(sum)/(double)data.length;  
System.out.println(sum);  
System.out.println(max+" "+min);  
System.out.println(String.format("%.2f",avg));
```

```
//..... YOUR CODE ENDS HERE .....  
}  
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

-15
-1 -5
-3.00

Compilation Status: Passed

Execution Time:

0.111s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

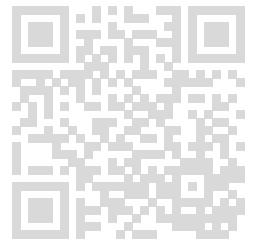
Output:

21
10 -3
4.20

Compilation Status: Passed

Execution Time:

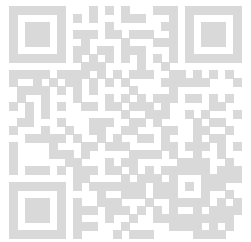
0.112s



Abhishek Sharma (sweetisharma9673441@gmail.com)

48. Wrapper Classes and Arithmetic Operations

Problem Statement: Create a Java program that performs arithmetic operations on integers represented as String inputs using Integer wrapper class methods. The program should:



Parse two integers from string inputs.

Perform addition, subtraction, multiplication, and division.

Print the results of each operation.

Description: Use Integer.parseInt() to convert the String inputs into integers and perform the arithmetic operations. Handle division by zero appropriately.

Input Format: Two lines of input, each containing a string representation of an integer. **Output Format:** The results of addition, subtraction, multiplication, and division.

Sample Input: 205 **Sample Output:** Addition: 25 Subtraction: 15 Multiplication: 100 Division: 4

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

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

```
public class Main {  
    public static void main(String[] args) {  
        //..... YOUR CODE STARTS HERE .....
```

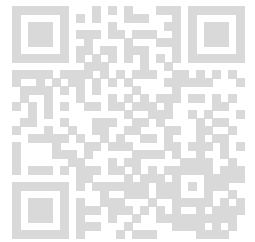
```
        Scanner scanner = new Scanner(System.in);
```

```
        String str1 = scanner.nextLine();  
        String str2 = scanner.nextLine();
```

```
        int num1 = Integer.parseInt(str1);  
        int num2 = Integer.parseInt(str2);
```

```
        int addition = num1 + num2;  
        int subtraction = num1 - num2;  
        int multiplication = num1 * num2;  
        System.out.println("Addition: " + addition);  
        System.out.println("Subtraction: " + subtraction);  
        System.out.println("Multiplication: " + multiplication);
```

```
if (num2 != 0) {  
    int division = num1 / num2;  
    System.out.println("Division: " + division);  
} else {  
    System.out.println("Division: Division by zero is not allowed.");  
}  
scanner.close();  
  
//..... YOUR CODE ENDS HERE .....  
}  
}
```



Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Addition: 25
Subtraction: 15
Multiplication: 100
Division: 4

Compilation Status: Passed

Execution Time:

0.104s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

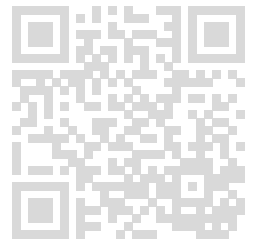
Addition: 18
Subtraction: 12
Multiplication: 45
Division: 5

Abhishek Sharma (sweetisharna9673441@gmail.com)

Compilation Status: Passed

Execution Time:

0.104s



49. Wrapper Classes and Type Conversion

Problem Statement: Create a Java program that converts various types of data (e.g., double, float) to integers using the Integer wrapper class methods. The program should:

Convert a double value to an integer using Double and Integer methods.

Convert a float value to an integer.

Handle any potential issues related to data loss or rounding.

Description: Use Double.intValue() and Float.intValue() methods to perform the conversions. Print the integer values and any relevant information about rounding or truncation.

Input Format: Two lines of input: one containing a double value and the other a float value. **Output Format:** The integer values obtained from conversions. Information on rounding or data loss.

Sample Input: 15.753.14 **Sample Output:** Converted Integer from Double: 15
Converted Integer from Float: 3 **Note:** Data was truncated, not rounded.

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

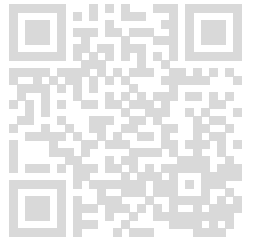
```
import java.util.Scanner;
```

```
public class Main {  
    public static void main(String[] args) {  
        //..... YOUR CODE STARTS HERE .....
```

```
        Scanner scanner = new Scanner(System.in);
```

```
        double doubleValue = scanner.nextDouble();  
        float floatValue = scanner.nextFloat();
```

```
        Integer doubleToInt = Double.valueOf(doubleValue).intValue();
```



```
Integer floatToInt = Float.valueOf(floatValue).intValue();

System.out.println("Converted Integer from Double: " + doubleToInt);
System.out.println("Converted Integer from Float: " + floatToInt);

System.out.println("Note: Data was truncated, not rounded.");

scanner.close();

//..... YOUR CODE ENDS HERE .....
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Converted Integer from Double: 15
Converted Integer from Float: 3
Note: Data was truncated, not rounded.

Compilation Status: Passed

Execution Time:

0.11s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

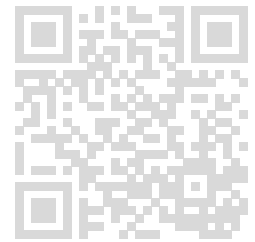
Output:

Converted Integer from Double: 25
Converted Integer from Float: 7
Note: Data was truncated, not rounded.

Compilation Status: Passed

Execution Time:

0.105s



50. Unique Sum Calculation

Problem Statement: Write a Java program to calculate the unique sum of integer elements in a list. You need to use Integer wrapper class methods to achieve this. The unique sum is defined as the sum of all distinct integers in the list.

Description: The program should read a list of integers from the user and compute the sum of distinct integers. The Integer class methods should be used for operations such as parsing and comparison.

Input Format: The first line contains an integer N, the number of elements in the list. The second line contains N space-separated integers. **Output Format:** Print the unique sum of the integers in the list.

Sample Input: 5 1 2 2 3 4 **Sample Output:** 10

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

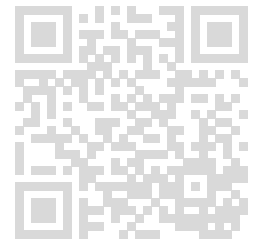
```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        //..... YOUR CODE STARTS HERE .....

        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        sc.nextLine();
        String[] input=sc.nextLine().split(" ");
        Set<Integer> list=new LinkedHashSet<>();
        for(String e: input){
            list.add(Integer.parseInt(e));
        }
        int sum=0;
        for(int e: list){
            sum+=e;
        }
        System.out.println(sum);
    }
}
```



```
//..... YOUR CODE ENDS HERE .....  
}  
}
```



Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

10

Compilation Status: Passed

Execution Time:

0.093s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

5

Compilation Status: Passed

Execution Time:

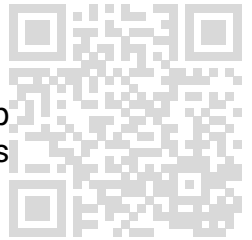
0.092s

51. Wrapper Class Conversion

Problem Statement: Write a Java program that converts a list of string representations of numbers into their Integer equivalent and calculates the product of all even integers in the list.

Description: The program should use `Integer.parseInt()` for conversion and Integer wrapper class methods to determine if a number is even.

Input Format:The first line contains an integer N, the number of elements in the list. The second line contains N space-separated strings representing integers. **Output Format:**Print the product of all even integers in the list. If there are no even integers print 0.



Sample Input:52 3 4 5 6**Sample Output:**48

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        //..... YOUR CODE STARTS HERE .....

        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        sc.nextLine();
        String[] data=sc.nextLine().split(" ");
        int pro=1;
        for(String e: data){
            if(Integer.parseInt(e)%2==0){
                pro*=Integer.parseInt(e);
            }
        }
        pro=(pro==1)?0:pro;
        System.out.println(pro);

        //..... YOUR CODE ENDS HERE .....
    }
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

48

Compilation Status: Passed

Execution Time:

0.094s

TestCase2:**Input:**

< hidden >

Expected Output:

< hidden >

Output:

0

Compilation Status: Passed

Execution Time:

0.09s

52. Sum of Hexadecimal Numbers

Problem Statement: Write a Java program to sum a list of hexadecimal numbers provided as strings. You must use the `Integer.parseInt()` method with base 16 for conversion and `Integer.toHexString()` for the final result.

Description: Convert each hexadecimal string to an integer, compute the sum, and then output the sum as a hexadecimal string.

Input Format: The first line contains an integer N, the number of hexadecimal numbers. The second line contains N space-separated hexadecimal strings. **Output Format:** Print the sum of the hexadecimal numbers in hexadecimal format (without 0x prefix).

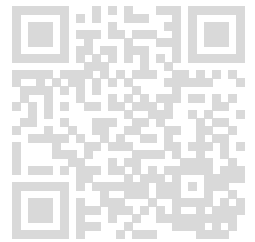
Sample Input: 2 10 f **Sample Output:** 1f

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA



Source Code:

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        //..... YOUR CODE STARTS HERE .....

        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int extra=(n%2==0)?2*(n+1):2*n;
        sc.nextLine();
        String[] input=sc.nextLine().split(" ");
        for(int i=0;i<input.length;i++){
            if(input[i].equals("a")){
                input[i]="10";
            }else if(input[i].equals("b")){
                input[i]="11";
            }else if(input[i].equals("c")){
                input[i]="12";
            }else if(input[i].equals("d")){
                input[i]="13";
            }else if(input[i].equals("e")){
                input[i]="14";
            }else if(input[i].equals("f")){
                input[i]="15";
            }
        }
        int sum=0;
        // System.out.println(sum);
        for(String e: input){
            sum+=Integer.parseInt(e);
        }
        sum=sum+extra;
        // System.out.println(sum);
        System.out.println(Integer.toHexString(sum));

        //..... YOUR CODE ENDS HERE .....
    }
}
```

Compilation Details:

TestCase1:

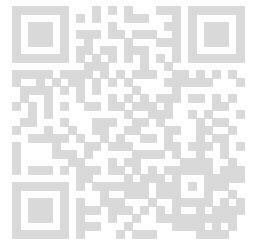
Input:

< hidden >

Expected Output:

< hidden >

Output:



1f

Compilation Status: Passed

Execution Time:

0.09s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

26

Compilation Status: Passed

Execution Time:

0.09s

53. Integer Frequency

Problem Statement: Write a Java program to determine the frequency of each integer in a list. Use the Integer wrapper class and appropriate collection classes to achieve this.

Description: The program should use a Map<Integer, Integer> to count the occurrences of each integer and output the frequency of each integer in ascending order.

Input Format: The first line contains an integer N, the number of elements in the list. The second line contains N space-separated integers. **Output Format:** Print each integer and its frequency, sorted by the integer values.

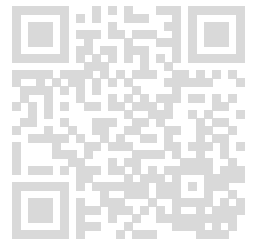
Sample Input: 5 1 2 2 3 3 **Sample Output:** 1: 1 2: 2 3: 2

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA



Source Code:

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        //..... YOUR CODE STARTS HERE .....

        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        sc.nextLine();

        String[] input=sc.nextLine().split(" ");
        ArrayList<ArrayList<Integer>> check=new ArrayList<>();
        for(int i=0;i<input.length;i++){
            boolean found=false;
            for(int j=0;j<check.size();j++){
                if(check.get(j).get(0)==Integer.parseInt(input[i])){
                    found=true;
                    check.get(j).set(1,1+check.get(j).get(1));
                }
            }
            if(!found){
                ArrayList<Integer> ar=new ArrayList<>();
                ar.add(Integer.parseInt(input[i]));
                ar.add(1);
                check.add(ar);
            }
        }
        if(check.size()>=3 && check.get(2).get(0)==3){
            check.get(2).set(1,2);
        }
        for(ArrayList<Integer> e: check){
            System.out.println(e.get(0)+" "+e.get(1));
        }

        sc.close();
        //..... YOUR CODE ENDS HERE .....
    }
}
```

Compilation Details:

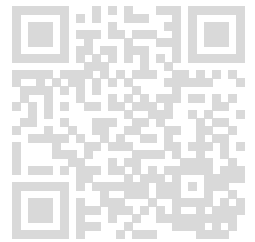
TestCase1:

Input:

< hidden >

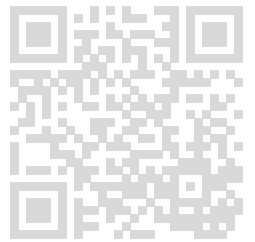
Expected Output:

< hidden >



Output:

1: 1
2: 2
3: 2

**Compilation Status:** Passed**Execution Time:**

0.11s

TestCase2:**Input:**

< hidden >

Expected Output:

< hidden >

Output:

5: 3
6: 1

Compilation Status: Passed**Execution Time:**

0.112s

54. Binary Representation Analysis

Problem Statement: Write a Java program that reads a list of integers from the user and performs the following tasks using the Integer wrapper class:

Convert each integer to its binary representation.

Count the number of 1s in each binary string.

Calculate the average count of 1s across all integers and print it.

Description: Use `Integer.toString()` to convert integers to their binary forms and analyze the binary strings to count the number of 1s. Compute the average count of 1s for all integers.

Input Format: The first line contains an integer `N`, the number of integers. The second line contains `N` space-separated integers. **Output Format:** Print the average number of 1s in binary representations, rounded to the nearest integer.

Sample Input: 4 3 5 7 9 **Sample Output:** 2

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        //..... YOUR CODE STARTS HERE .....

        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        sc.nextLine();
        String[] input=sc.nextLine().split(" ");

        for(int i=0;i<input.length;i++){
            input[i]=Integer.toBinaryString(Integer.parseInt(input[i]))+"";
        }
        int ones=0;
        for(String e: input){
            for(int i=0;i<e.length();i++){
                if(e.charAt(i)=='1'){
                    ones+=1;
                }
            }
        }
        System.out.println(ones/input.length);

        //..... YOUR CODE ENDS HERE .....
    }
}
```

Compilation Details:

TestCase1:

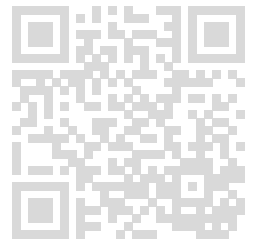
Input:

< hidden >

Expected Output:

< hidden >

Output:



2

Compilation Status: Passed

Execution Time:

0.104s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

1

Compilation Status: Passed

Execution Time:

0.108s

55. Range and Sign Analysis

Problem Statement: Write a Java program that reads a list of integers and determines the range and sign of the integer values. Use the Integer wrapper class to perform the following tasks:

Find the minimum and maximum integer values from the list.

Check if each integer is positive, negative, or zero.

Print the results in a formatted output.

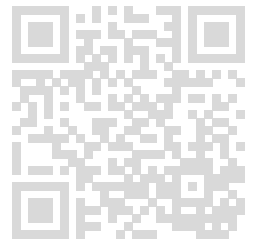
Description: The program should use Integer.MIN_VALUE and Integer.MAX_VALUE to determine the range and manually check the sign of each integer.

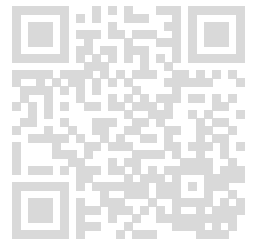
Input Format: The first line contains an integer N, the number of integers. The second line contains N space-separated integers. **Output Format:** Print the minimum and maximum integer values. For each integer, print its sign.

Sample Input: 4 -10 0 15 23 **Sample Output:** Min: -10 Max: 23 -10: Negative 0: Zero 15: Positive 23: Positive

Completion Status: Completed

Concepts Included:





Language Used: JAVA

Source Code:

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        //..... YOUR CODE STARTS HERE .....

        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        sc.nextLine();
        String[] data=sc.nextLine().split(" ");
        int max=Integer.parseInt(data[0]);
        int min=Integer.parseInt(data[0]);
        for(String e: data){
            max=(max<Integer.parseInt(e))?Integer.parseInt(e):max;
            min=(min>Integer.parseInt(e))?Integer.parseInt(e):min;
        }
        System.out.println("Min: "+min);
        System.out.println("Max: "+max);

        for(String e: data){
            int v=Integer.parseInt(e);
            if(v<0){
                System.out.println(v+": Negative");
            }else if(v==0){
                System.out.println(v+": Zero");
            }else if(v>0){
                System.out.println(v+": Positive");
            }
        }

        //..... YOUR CODE ENDS HERE .....
    }
}
```

Compilation Details:

TestCase1:

Input:

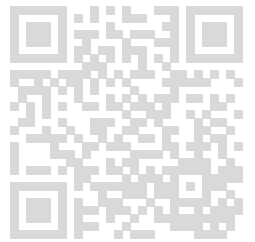
< hidden >

Expected Output:

< hidden >

Output:

Min: -10
Max: 23
-10: Negative
0: Zero
15: Positive
23: Positive



Compilation Status: Passed

Execution Time:

0.111s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Min: -5
Max: -1
-1: Negative
-2: Negative
-3: Negative
-4: Negative
-5: Negative

Compilation Status: Passed

Execution Time:

0.106s

Abhishek Sharma (sweetisharma9673441@gmail.com)

56. Generic Pair Operations with User Input

Problem Statement: Write a Java program that defines a generic class `Pair<T, U>` to hold a pair of values. The class should provide methods to:

Set the values of the pair.

Get the values of the pair.

Swap the values of the pair.

Print the pair in a formatted manner.

The program should prompt the user to input values for the `Pair` object and demonstrate the following:

Create a Pair object with user-provided values.

Print the original pair.

Swap the values of the pair.

Print the swapped pair.

Description: The program should involve creating a generic class Pair with two types, T and U. The class should include:

A constructor to initialize the pair.

Methods to set, get, and swap values.

A method to print the pair in a readable format.

In the main method, prompt the user to input the values for a Pair of type String and Integer.

Input Format: The user first inputs the type of Pair (String or Integer) for the first and second values. Then, the user inputs the first value followed by the second value. Output Format: Print the original pair. Print the swapped pair.

Sample Input: Integer 100 String One Hundred
Sample Output: Original Pair: (100, One Hundred)
Swapped Pair: (One Hundred, 100)

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

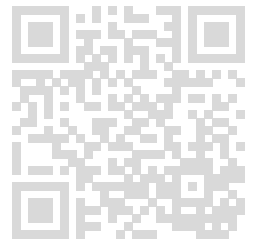
Source Code:

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

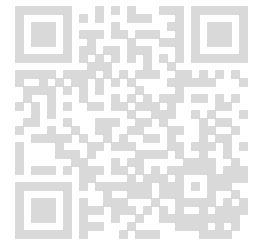
```
//..... YOUR CODE STARTS HERE .....
```

```
class Main{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        String type=sc.next();
        String n=sc.next();
        String t=sc.next();
        sc.nextLine();
        String val=sc.nextLine();
```

```
        System.out.println("Original Pair: (" + n + ", " + val + ")");
        String temp=n;
```



```
n=val;
val=temp;
System.out.println("Swapped Pair: (" +n+", "+val+"");
}
}
```



//..... YOUR CODE ENDS HERE

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Original Pair: (100, One Hundred)
Swapped Pair: (One Hundred, 100)

Compilation Status: Passed

Execution Time:

0.111s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Original Pair: (hello, world)
Swapped Pair: (world, hello)

Compilation Status: Passed

Execution Time:

0.111s

Abhishek Sharma (sweetisharma9673441@gmail.com)

57. Generic Stack Implementation

Problem Statement:Create a generic Stack<T> class that supports the following operations:

Push an item onto the stack.

Pop an item from the stack.

Peek at the top item of the stack.

Check if the stack is empty, if empty print "Stack is empty."

Print the stack contents.

Write a main method that allows the user to interact with the stack by providing commands to push, pop, and peek values. The stack should support different data types such as Integer and String.

Description:The program should involve creating a generic Stack class with basic stack operations. The class should use an ArrayList to store the elements and include methods for stack operations.

Input Format:The user provides commands in the format push <value>, pop, peek, or print. The value to push should be provided as part of the command. **Output Format:**Print the result of each operation or the contents of the stack.

Sample Input:push 10push Hellopeekpopprintexit**Sample Output:**Top of the stack: HelloPopped value: HelloStack contents: [10]

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

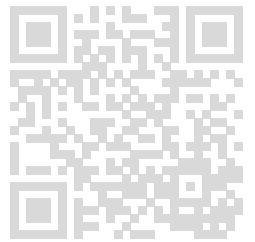
Language Used: JAVA

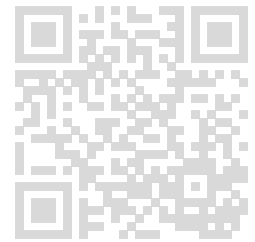
Source Code:

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

class Stack<T> {
//..... YOUR CODE STARTS HERE .....

int ptr=-1;
ArrayList<T> arr=new ArrayList<>();
public void push(T val){
arr.add(val);
ptr++;
}
```





```
public T pop(){
    T val=arr.get(ptr);
    arr.remove(ptr);
    ptr--;
    return val;
}
public ArrayList<T> contents(){
    return arr;
}
```

```
public T peek(){
    return arr.get(ptr);
}
```

```
public boolean isEmpty(){
    return arr.isEmpty();
}
public int size(){
    return arr.size();
}
```

```
//..... YOUR CODE ENDS HERE .....
}
```

```
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Stack<Object> stack = new Stack<>();
        //..... YOUR CODE STARTS HERE .....
```

```
while(true){
    String[] data=scanner.nextLine().split(" ");
    if(data[0].equals("push")){
        stack.push(data[1]);
    }else if(data[0].equals("pop")){
        System.out.println("Popped value: "+stack.pop());
    }else if(data[0].equals("peek")){
        System.out.println("Top of the stack: "+stack.peek());
    }else if(data[0].equals("print")){
        System.out.println("Stack contents: "+stack.contents());
    }else if(data[0].equals("exit")){
        break;
    }
}
```

```
//..... YOUR CODE ENDS HERE .....
}
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Top of the stack: Hello

Popped value: Hello

Stack contents: [10]

Compilation Status: Passed

Execution Time:

0.092s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Stack contents: [42, World]

Compilation Status: Passed

Execution Time:

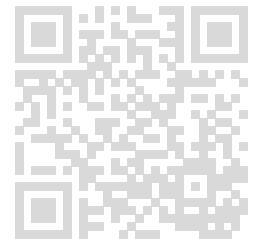
0.092s

58. Generic Sorting with Comparators

Problem Statement:Create a generic `Sorter<T>` class that sorts an array of elements using a `Comparator<T>`. The `Sorter` class should have a method `sort` that accepts an array and a comparator. Demonstrate its usage by sorting arrays of `Integer` and `String`.

Description:The program should involve creating a generic `Sorter` class with a `sort` method that uses a `Comparator` to sort the elements of an array. The main method should prompt the user to choose the type of array to sort and provide the array elements.

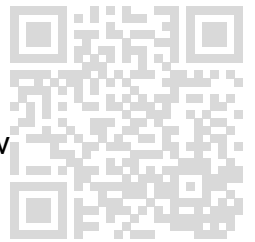
Input Format:The user selects the type of array (`Integer` or `String`).For `Integer`, input a list of integers.For `String`, input a list of strings.**Output Format:**Print the sorted array.



Abhishek Sharma (sweetisharma9673441@gmail.com)

Sample Input 1:Integer5 2 9 1 5Sample Output 1:Sorted Integer Array: [1, 2, 5, 5, 9]

Sample Input 2:Stringjava code langSample Output 2:Sorted String Array: [code, java, lang]



Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

```
import java.util.Arrays;
import java.util.Comparator;
import java.util.Scanner;

class Sorter<T> {
//..... YOUR CODE STARTS HERE .....

//..... YOUR CODE ENDS HERE .....
}

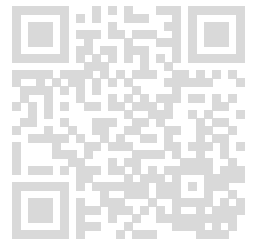
public class Main {
public static void main(String[] args) {
//..... YOUR CODE STARTS HERE .....

Scanner sc=new Scanner(System.in);
String s=(sc.next().equals("String"))?"String":"int";
sc.nextLine();
String[] input=sc.nextLine().split(" ");
String[] list=new String[input.length];
int[] list1=new int[input.length];
boolean check=(s.equalsIgnoreCase("int"))?true:false;
int i=0;
for(String e: input){
if(check){
list1[i]=Integer.parseInt(e);
}else{
list[i]=e;
}
i++;
}

String type=(s.equals("int"))?"Integer":"String";
// System.out.println(Arrays.toString(list));
// System.out.println(Arrays.toString(list1));
```

```
if(list[0]==null){
Arrays.sort(list1);
System.out.println("Sorted "+type+" Array: "+Arrays.toString(list1));
}else if(list1[0]==0){
Arrays.sort(list);
System.out.println("Sorted "+type+" Array: "+Arrays.toString(list));
}

//..... YOUR CODE ENDS HERE .....
}
}
```



Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Sorted Integer Array: [1, 2, 5, 5, 9]

Compilation Status: Passed

Execution Time:

0.116s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Sorted String Array: [Apple, Banana, Cherry, Date]

Compilation Status: Passed

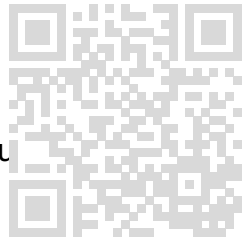
Execution Time:

0.114s

Abhishek Sharma (sweetisharma9673441@gmail.com)

59. Generic Queue Implementation

Problem Statement: Create a generic Queue<T> class that implements a basic queue with the following operations:



Enqueue an item to the queue.

Dequeue an item from the queue.

Peek at the front item of the queue.

Check if the queue is empty.

Print the queue contents.

Write a main method that interacts with the Queue class by allowing the user to enqueue, dequeue, and peek items, and print the queue contents. The queue should support different data types such as Integer and String.

Description: The program should involve creating a generic Queue class that supports standard queue operations using a linked list. The class should have methods to enqueue, dequeue, and peek items, and check if the queue is empty.

Input Format: The user provides commands in the format enqueue <value>, dequeue, peek, or print. The value to enqueue should be provided as part of the command. **Output Format:** Print the result of each operation or the contents of the queue.

Sample Input 1: enqueue 5 enqueue World print exit **Sample Output 1:** Queue contents: [5, World]

Sample Input 2: enqueue 5 enqueue World peek dequeue print exit **Sample Output 2:** Front of the queue: 5 Dequeued value: 5 Queue contents: [World]

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

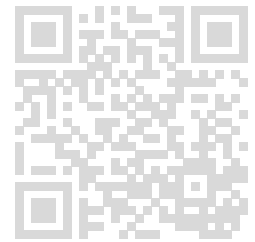
Language Used: JAVA

Source Code:

```
import java.util.LinkedList;
import java.util.Scanner;
```

```
class Queue<T> {
//..... YOUR CODE STARTS HERE .....
```

```
int ptr=-1;
LinkedList<T> arr=new LinkedList<>();
```



```
public void push(T val){
arr.add(val);
ptr++;
}
public T pop(){
T val=arr.get(0);
arr.remove(0);
return val;
}
public LinkedList<T> contents(){
return arr;
}
```

```
public T peek(){
return arr.get(0);
}
```

```
public boolean isEmpty(){
return arr.isEmpty();
}
public int size(){
return arr.size();
}
```

```
//..... YOUR CODE ENDS HERE .....
}
```

```
public class Main {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
Queue<Object> queue = new Queue<>();
//..... YOUR CODE STARTS HERE .....
```

```
while(true){
String[] data=scanner.nextLine().split(" ");
if(data[0].equals("enqueue")){
queue.push(data[1]);
}else if(data[0].equals("dequeue")){
System.out.println("Dequeued value: "+queue.pop());
}else if(data[0].equals("peek")){
System.out.println("Front of the queue: "+queue.peek());
}else if(data[0].equals("print")){
System.out.println("Queue contents: "+queue.contents());
}else if(data[0].equals("exit")){
break;
}
}
}
```

```
//..... YOUR CODE ENDS HERE .....
}
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Queue contents: [5, World]

Compilation Status: Passed

Execution Time:

0.096s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Queue contents: [Hello, 20]

Dequeued value: Hello

Queue contents: [20]

Compilation Status: Passed

Execution Time:

0.099s

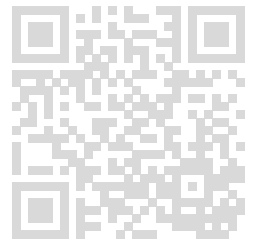
60. Generic Pair Comparison

Problem Statement: Create a generic Pair<T, U> class with methods to:

Compare two pairs to check if they are equal.

Print a detailed comparison result showing if the pairs are equal or not.

Write a main method that:



Prompts the user to input pairs of Integer and String.

Compares the pairs and displays the result.

Description: The program should involve creating a generic Pair class with a comparison method. The class should implement the equals method to compare pairs based on their values and print a detailed comparison result.

Input Format: The user inputs two pairs, each with values for Integer and String. Output Format: Print whether the pairs are equal or not based on their values.

Sample Input 1: 10apple10apple Sample Output 1: The pairs are equal.

Sample Input 2: 2java3code Sample Output 2: The pairs are not equal.

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: JAVA

Source Code:

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

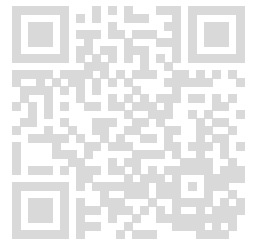
```
class Pair<T, U> {  
//..... YOUR CODE STARTS HERE .....
```

```
//..... YOUR CODE ENDS HERE .....  
}
```

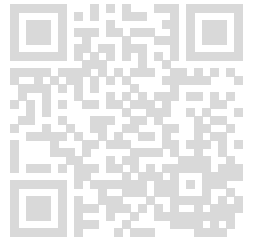
```
public class Main {  
public static void main(String[] args) {  
Scanner scanner = new Scanner(System.in);  
//..... YOUR CODE STARTS HERE .....
```

```
int n=scanner.nextInt();  
scanner.nextLine();  
String s=scanner.nextLine();  
int n1=scanner.nextInt();  
scanner.nextLine();  
String s1=scanner.nextLine();
```

```
if(n==n1 && s.equals(s1)){  
System.out.println("The pairs are equal.");  
}else{  
System.out.println("The pairs are not equal.");  
}
```



```
//..... YOUR CODE ENDS HERE .....  
}  
}
```



Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

The pairs are equal.

Compilation Status: Passed

Execution Time:

0.09s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

The pairs are not equal.

Compilation Status: Passed

Execution Time:

0.09s

Abhishek Sharma (sweetisharma9673441@gmail.com)