Codekata Report:

Name: Muskan

Email: kapoormuskan474@gmail.com

Specialization: School of Computing Science & Engineering

Completion Year: 2027

Section: Section-18

1. 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: Excellent75-89: Good50-74: Average35-49: Pass0-34: 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 -1Sample 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 n = scanner.nextInt();
while (true) {
```

```
int score = scanner.nextInt();
if (score == -1) {
break;
}
if (score >= 90 && score <= 100) {
System.out.println("Excellent");
} else if (score >= 75 && score <= 89) {
System.out.println("Good");
} else if (score >= 50 && score <= 74) {
System.out.println("Average");
} else if (score >= 35 && score <= 49) {
System.out.println("Pass");
} else if (score >= 0 && score <= 34) {
System.out.println("Fail");
} else {
System.out.println("Invalid Score");
}
}


//..... 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:

0.092s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Average
Average
Average

Compilation Status: Passed

Execution Time:

0.09s

2. 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:10159

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

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);

// Input
int a = scanner.nextInt();
int b = scanner.nextInt();
int c = scanner.nextInt();
// Transformations
//..... YOUR CODE STARTS HERE .....
if (a % 2 == 0) {
a += b;
}
if (b % 2 != 0) {
c *= 2;
}
if (c % 3 == 0) {
c += a;
}
if (a + b + c > 100) {
a -= 100;
b -= 100;
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:

0.116s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

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

Compilation Status: Passed

Execution Time:

0.116s

3. Complex Variable Initialization and Arithmetic Operations

Problem StatementWrite 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.

DescriptionYou 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 FormatFour integers, each on a new line.

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

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: PYTHON 3

Source Code:

```
# Read four integers from input
a = int(input())
b = int(input())
c = int(input())
d = int(input())

# Perform the operations
result1 = (a + b) * c
result2 = (b - d) // a
result3 = (a * d) + c
result4 = (a + b + c + d) // 2

# Print the results in the specified format
print(f"Result of operation 1: {result1}")
print(f"Result of operation 2: {result2}")
print(f"Result of operation 3: {result3}")
print(f"Result of operation 4: {result4}")
```


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:

0.017s

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:

0.017s

4. Bitwise Operations with Floating Point Variables

Problem StatementWrite 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.

DescriptionYou 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 FormatTwo floating-point numbers, each on a new line.

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

Completion Status: Completed

Concepts Included:

gu 27 3rd semester java programming

Language Used: PYTHON 3

Source Code:

```
# Function to perform bitwise operations
def bitwise_operations(a, b):
# Convert floating-point numbers to integers
int_a = int(a)
int_b = int(b)

# Perform bitwise AND
and_result = int_a & int_b

# Perform bitwise OR
or_result = int_a | int_b

# Perform bitwise XOR
xor_result = int_a ^ int_b

# Perform bitwise NOT on the first integer and AND with the second integer
not_and_result = (~int_a) & int_b

# Print the results in the required format
print(f"Bitwise AND result: {and_result}")
print(f"Bitwise OR result: {or_result}")
print(f"Bitwise XOR result: {xor_result}")
print(f"Bitwise NOT and AND result: {not_and_result}")

# Take two floating-point numbers as input
a = float(input())
b = float(input())

# Call the function to perform bitwise operations
bitwise_operations(a, b)
```


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:

0.011s

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:

0.011s

5. 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 -1Sample 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 n = scanner.nextInt();

int oddSum = 0;
int evenSum = 0;

for (int i = 0; i < n; i++){
int num = scanner.nextInt();
if (num == -1){
break;
}

if (num % 2 == 0){
evenSum += num;
} else{
oddSum += num;
}
}

if (oddSum > evenSum){
System.out.println("Odd Sum Greater");
}
else if (evenSum > oddSum){
System.out.println("Even Sum Greater");
}
else{
System.out.println("Equal");
}

scanner.close();

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

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Even Sum Greater

Compilation Status: Passed

Execution Time:

0.093s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Odd Sum Greater

Compilation Status: Passed

Execution Time:

0.093s

6. 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 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 t 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 Car {
private String make;
private String model;
private int year;

public Car(String make, String model, int year) {
this.make = make;
this.model = model;
this.year = year;
}

public int getCarAge(int currentYear) {
if (year > currentYear) {
System.out.println("Invalid year");
return -1;
}
return currentYear - year;
}
}

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

String make = scanner.nextLine();
String model = scanner.nextLine();
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);
}
}
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

9

Compilation Status: Passed

Execution Time:

0.088s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

6

Compilation Status: Passed

Execution Time:

0.09s

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:3Sample 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 scanner = new Scanner(System.in);

// Input the number of levels
int n = scanner.nextInt();

// Generate the pyramid pattern
for (int i = 1; i <= n; i++) {
// Determine the symbol to use for the current level
char symbol = (i % 2 == 1) ? '*' : '#';

// Print the symbol i times (i is the current level number)
for (int j = 1; j <= i; j++) {
System.out.print(symbol);
}
```

```
// Move to the next line after printing each level
System.out.println();
}

scanner.close();



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

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
*
##
***
```

Compilation Status: Passed

Execution Time:

0.092s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
*
##
***
####
```

Compilation Status: Passed

Execution Time:

0.089s

8. 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 -1Sample 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 month = scanner.nextInt();
int year = scanner.nextInt();

if (month == -1 && year == -1) {
break;
}

if (month < 1 || month > 12 || year < 0) {
System.out.println("Invalid input");
} else {
int days;
switch (month) {
```

```
case 1: days = 31; break;
case 2:
// Check for leap year
if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
days = 29;
} else {
days = 28;
}
break;
case 3: days = 31; break;
case 4: days = 30; break;
case 5: days = 31; break;
case 6: days = 30; break;
case 7: days = 31; break;
case 8: days = 31; break;
case 9: days = 30; break;
case 10: days = 31; break;
case 11: days = 30; break;
case 12: days = 31; break;
default: days = -1; break; // Invalid month
}

if (days == -1) {
System.out.println("Invalid month");
} else {
System.out.println(days);
}
}
}



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

scanner.close();
}
}
```


Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

29

Compilation Status: Passed

Execution Time:

0.087s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

28

Compilation Status: Passed

Execution Time:

0.091s

9. 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().

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 to initialize length and width
public Rectangle(double length, double width) {
this.length = length;
this.width = width;
}

@Override
public double area() {
return length * width;
}

// Method to calculate the perimeter of the rectangle
@Override
public double perimeter() {
return 2 * (length + width);
}



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

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

private double radius;

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

```
// Method to calculate the area of the circle
@Override
public double area() {
return Math.PI * radius * radius;
}

// Method to calculate the perimeter (circumference) of the circle
@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:

0.12s

10. 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 on502 Refrigerator 0.8 off3toggle 502calculateremove 501Sample Output:Appliance status toggledTotal Power Consumption: 2.3 kWAppliance 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 applianceID;
String name;
double powerRating;
```

```
boolean status;  // true for 'on', false for 'off'

// Parameterized constructor
public Appliance(int applianceID, String name, double powerRating, boolean status)
this.applianceID = applianceID;
this.name = name;
this.powerRating = powerRating;
this.status = status;
}

// Method to toggle the status
public void toggleStatus() {
this.status = !this.status;
}

// Method to get the power rating
public double getPowerRating() {
return this.powerRating;
}

// Method to check if the appliance is on
public boolean isOn() {
return this.status;
}

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

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

Appliance[] appliances;
int count;

// Constructor to initialize SmartHome with a max capacity of appliances
public SmartHome() {
appliances = new Appliance[100];  // Default maximum capacity is 100 appliances
count = 0;
}

// Method to add a new appliance
public void addAppliance(Appliance appliance) {
if (count < appliances.length) {
appliances[count++] = appliance;
}
}

// Method to remove an appliance by ID
public boolean removeAppliance(int applianceID) {
for (int i = 0; i < count; i++) {
if (appliances[i].applianceID == applianceID) {
// Shift the remaining appliances
for (int j = i; j < count - 1; j++) {
appliances[j] = appliances[j + 1];
```

```
}
count--;
return true;
}
}
return false;  // Appliance not found
}

// Method to toggle appliance status by ID
public boolean toggleApplianceStatus(int applianceID) {
for (int i = 0; i < count; i++) {
if (appliances[i].applianceID == applianceID) {
appliances[i].toggleStatus();
return true;
}
}
return false;  // Appliance not found
}

// Method to calculate total power consumption of appliances that are 'on'
public double calculateTotalPower() {
double totalPower = 0;
for (int i = 0; i < count; i++) {
if (appliances[i].isOn()) {
totalPower += appliances[i].getPowerRating();
}
}
return totalPower;
}

//..... 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:

0.121s

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:

0.127s

11. 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:20Sample 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 .....

public InvalidAgeException(String message) {
super(message);
}

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

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

int age = scanner.nextInt();

try {
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:

0.084s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Age must be 18 or older

Compilation Status: Passed

Execution Time:

0.09s

12. 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:102Sample Output:5Operation 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 .....

int a = 0;
int b = 0;

try {
a = Integer.parseInt(scanner.nextLine());
b = Integer.parseInt(scanner.nextLine());

int result = a / b;
System.out.println(result);
} catch (NumberFormatException e) {
System.out.println("Invalid input");
} catch (ArithmeticException e) {
System.out.println("Cannot divide by zero");
} finally {
System.out.println("Operation Completed");
}

scanner.close();

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


Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

5
Operation Completed

Compilation Status: Passed

Execution Time:

0.089s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Invalid input
Operation Completed

Compilation Status: Passed

Execution Time:

0.088s

13. 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 activityDISPLAY_ACTIVITIES userIdOutput 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:6ADD_ACTIVITY user1 loginADD_ACTIVITY user1 postADD_ACTIVITY user1 loginADD_ACTIVITY user2 loginDISPLAY_ACTIVITIES user1DISPLAY_ACTIVITIES user2Sample Output:loginpostlogin

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 .....

private Map<String, Set<String>> userActivities;

public UserActivityTracker() {
userActivities = new HashMap<>();
}

public void addActivity(String userId, String activity) {
userActivities.putIfAbsent(userId, new HashSet<>());
userActivities.get(userId).add(activity);
}

public void displayActivities(String userId) {
if (!userActivities.containsKey(userId) || userActivities.get(userId).isEmpty()) {
System.out.println("No activities found.");
} else {
List<String> activities = new ArrayList<>(userActivities.get(userId));
Collections.sort(activities);
for (String activity : activities) {
System.out.println(activity);
}
}
}

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

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

Scanner scanner = new Scanner(System.in);
UserActivityTracker tracker = new UserActivityTracker();

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

for (int i = 0; i < n; i++) {
String operation = scanner.nextLine();
String[] parts = operation.split(" ");

if (parts[0].equals("ADD_ACTIVITY")) {
String userId = parts[1];
String activity = parts[2];
tracker.addActivity(userId, activity);
} else if (parts[0].equals("DISPLAY_ACTIVITIES")) {
String userId = parts[1];
tracker.displayActivities(userId);
```

```
}
}

scanner.close();

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

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
login
post
login
```

Compilation Status: Passed

Execution Time:

0.094s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
login
post
```

Compilation Status: Passed

Execution Time:

0.094s

14. Dynamic Vehicle Registration System

Problem Statement:You need to design a dynamic Vehicle Registration System whe each vehicle can be of different types (Car, Motorcycle, or Truck) and should be abl to report its registration details. Implement a base class Vehicle and three subclass 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 10000kgSample 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 registrationNumber;
String ownerName;

// Constructor to initialize registration number and owner name
public Vehicle(String registrationNumber, String ownerName) {
this.registrationNumber = registrationNumber;
this.ownerName = ownerName;
}
```

```
// Method to display vehicle details
public void displayDetails() {
System.out.println("Registration Number: " + registrationNumber);
System.out.println("Owner: " + ownerName);
}

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

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

int numberOfDoors;

// Constructor to initialize Car attributes
public Car(String registrationNumber, String ownerName, int numberOfDoors) {
super(registrationNumber, ownerName);
this.numberOfDoors = numberOfDoors;
}

// Overriding displayDetails method to include Car-specific details
@Override
public void displayDetails() {
System.out.println("Vehicle: Car");
super.displayDetails();
System.out.println("Number of Doors: " + numberOfDoors);
}

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

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

String engineCapacity;

// Constructor to initialize Motorcycle attributes
public Motorcycle(String registrationNumber, String ownerName, String
engineCapacity) {
super(registrationNumber, ownerName);
this.engineCapacity = engineCapacity;
}

// Overriding displayDetails method to include Motorcycle-specific details
@Override
public void displayDetails() {
System.out.println("Vehicle: Motorcycle");
super.displayDetails();
System.out.println("Engine Capacity: " + engineCapacity);
}

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

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

String cargoCapacity;

// Constructor to initialize Truck attributes
public Truck(String registrationNumber, String ownerName, String cargoCapacity) {
super(registrationNumber, ownerName);
this.cargoCapacity = cargoCapacity;
}

// Overriding displayDetails method to include Truck-specific details
@Override
public void displayDetails() {
System.out.println("Vehicle: Truck");
super.displayDetails();
System.out.println("Cargo Capacity: " + cargoCapacity);
}

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

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

Scanner scanner = new Scanner(System.in);

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

// Process each registration operation
for (int i = 0; i < N; i++) {
String operation = scanner.nextLine();
String[] parts = operation.split(" ");

// Extract the operation details
String vehicleType = parts[1];
String registrationNumber = parts[2];
String ownerName = parts[3];
String specificAttribute = parts[4];

// Register the appropriate vehicle type
switch (vehicleType.toLowerCase()) {
case "car":
int numberOfDoors = Integer.parseInt(specificAttribute);
Car car = new Car(registrationNumber, ownerName, numberOfDoors);
car.displayDetails();
break;
case "motorcycle":
Motorcycle motorcycle = new Motorcycle(registrationNumber, ownerName,
specificAttribute);
motorcycle.displayDetails();
```

```
break;
case "truck":
Truck truck = new Truck(registrationNumber, ownerName, specificAttribute);
truck.displayDetails();
break;
default:
System.out.println("Unknown vehicle type!");
break;
}
}

scanner.close();

//..... 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:

0.112s

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:

0.111s

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 N 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 JohnSample 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 .....

private String name;
private double totalMarks;
private int marksCount;
private char grade;

public Student(String name) {
this.name = name;
this.totalMarks = 0;
this.marksCount = 0;
}

public void addMarks(double mark) {
totalMarks += mark;
marksCount++;
System.out.println("Marks added");
}

public void calculateGrade() {
if (marksCount > 0) {
double average = totalMarks / marksCount;
if (average >= 90) {
grade = 'A';
} else if (average >= 80) {
grade = 'B';
} else if (average >= 70) {
grade = 'C';
} else if (average >= 60) {
grade = 'D';
} else {
grade = 'F';
}
System.out.println("Grade calculated");
} else {
System.out.println("No marks available to calculate grade");
}
}

public void displayDetails() {
```

```
double average = marksCount > 0 ? totalMarks / marksCount : 0;
System.out.printf("%s: %.1f - %c%n", name, average, grade); // Changed to %.1f
}

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

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

Scanner scanner = new Scanner(System.in);
HashMap<String, Student> studentMap = new HashMap<>();

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

for (int i = 0; i < N; i++) {
String[] input = scanner.nextLine().split(" ");
String command = input[0];
String name = input[1];

Student student = studentMap.computeIfAbsent(name, Student::new);

if (command.equals("add")) {
double mark = Double.parseDouble(input[2]);
student.addMarks(mark);
} else if (command.equals("calculate")) {
student.calculateGrade();
} else if (command.equals("display")) {
student.displayDetails();
}
}

scanner.close();

//..... 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:

0.097s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Marks added
Grade calculated
Alice: 70.0 - C

Compilation Status: Passed

Execution Time:

0.099s

16. 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 HarryPotterSample 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;

public Book(String title, String author) {
this.title = title;
this.author = author;
this.isBorrowed = false;
}

public String getTitle() {
return title;
}

public boolean isBorrowed() {
return isBorrowed;
}

public void borrowBook() {
isBorrowed = true;
}

public void returnBook() {
isBorrowed = false;
}

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

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

private ArrayList<Book> books;

public Library() {
books = new ArrayList<>();
```

```
}

public void addBook(String title, String author) {
books.add(new Book(title, author));
System.out.println("Book added");
}

public void borrowBook(String title) {
for (Book book : books) {
if (book.getTitle().equals(title)) {
if (!book.isBorrowed()) {
book.borrowBook();
System.out.println("Book borrowed");
return;
} else {
System.out.println("Book not available");
return;
}
}
}
System.out.println("Book not found");
}

public void returnBook(String title) {
for (Book book : books) {
if (book.getTitle().equals(title)) {
if (book.isBorrowed()) {
book.returnBook();
System.out.println("Book returned");
return;
} else {
System.out.println("Book was not borrowed");
return;
}
}
}
System.out.println("Book not found");
}

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

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

Scanner scanner = new Scanner(System.in);
Library library = new Library();

int N = scanner.nextInt();
scanner.nextLine();  // Consume newline

for (int i = 0; i < N; i++) {
String operation = scanner.nextLine();
```

```
String[] parts = operation.split(" ");
String command = parts[0];

switch (command) {
case "add":
String title = parts[1];
String author = parts[2];
library.addBook(title, author);
break;
case "borrow":
title = parts[1];
library.borrowBook(title);
break;
case "return":
title = parts[1];
library.returnBook(title);
break;
default:
System.out.println("Invalid operation");
break;
}
}

scanner.close();


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


Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Book added
Book borrowed
Book returned

Compilation Status: Passed

Execution Time:

0.094s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Book added
Book borrowed
Book not available
Book returned

Compilation Status: Passed

Execution Time:

0.092s

17. 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:3LionElephantMonkeySample 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();
if (animalType.equalsIgnoreCase("Lion")) {
zoo.add(new Lion());
} else if (animalType.equalsIgnoreCase("Elephant")) {
zoo.add(new Elephant());
} else if (animalType.equalsIgnoreCase("Monkey")) {
zoo.add(new Monkey());
} else {
System.out.println("Unknown animal type");
}
}



//..... 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:

0.092s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Roar

Compilation Status: Passed

Execution Time:

0.09s

18. OOPs Features & Interface

Problem StatementImplement 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.

DescriptionDefine 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 FormatString 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 FormatString representing the employee details.Double representing the calculated salary.

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

Sample Input 2:PartTimeEmployeeJane15.0120Sample 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 to initialize the properties
public FullTimeEmployee(String name, double monthlySalary) {
this.name = name;
this.monthlySalary = monthlySalary;
}

// Implement the calculateSalary method
@Override
public double calculateSalary() {
return monthlySalary;
}

// Implement the getDetails method
@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 to initialize the properties
public PartTimeEmployee(String name, double hourlyRate, int hoursWorked) {
this.name = name;
this.hourlyRate = hourlyRate;
this.hoursWorked = hoursWorked;
}

// Implement the calculateSalary method
@Override
public double calculateSalary() {
return hourlyRate * hoursWorked;
}

// Implement the getDetails method
@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:

0.133s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Name: Jane, Salary: 1800.0

Compilation Status: Passed

Execution Time:

0.133s

19. 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:Alice25Sample Output 1:Name: AliceAge: 25

Sample Input 2:Diana0Sample 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 .....

private String name;
private int age;

// Method to set the name
public void setName(String name) {
this.name = name;
}

// Method to set the age, only allowing positive values
public void setAge(int age) {
if (age > 0) {
this.age = age;
} else {
System.out.println("Invalid age");
this.age = -1; // Set age to an invalid number to indicate error
}
}

// Method to get the name
public String getName() {
return name;
}

// Method to get the age
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:

0.107s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Invalid age

Compilation Status: Passed

Execution Time:

0.091s

20. 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 C (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 scanner = new Scanner(System.in);

// Input the number of registered voters
int registeredVoters = scanner.nextInt();

// Input the number of candidates
int numCandidates = scanner.nextInt();

// Input the number of votes
int numVotes = scanner.nextInt();

// Variable to count the number of processed votes
int votesCount = 0;

// Process each vote
for (int i = 0; i < numVotes; i++) {
int candidateID = scanner.nextInt();

// Check if the vote is valid
if (candidateID >= 1 && candidateID <= numCandidates && votesCount <
```

```
registeredVoters) {
System.out.println("VALID");
votesCount++;  // Increment the vote count
} else {
System.out.println("INVALID");
}
}

scanner.close();



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

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

VALID
INVALID
VALID
VALID

Compilation Status: Passed

Execution Time:

0.091s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

VALID

VALID
VALID
INVALID

Compilation Status: Passed

Execution Time:

0.092s

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 4Sample 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;

while (true) {
```

```
try {
String input = scanner.next();
try {
int number = Integer.parseInt(input);
sum += number;
} catch (NumberFormatException e) {
continue;
}
} catch (Exception e) {
break;
}
}

System.out.println("Sum of entered integers: " + sum);
scanner.close();

//..... 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:

0.101s

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

Sample Input 2:20Sample 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 .....

public UnderageException(String message) {
super(message);
}

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

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

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

Scanner scanner = new Scanner(System.in);
int age = scanner.nextInt();

try {
if (age < 18) {
throw new UnderageException("UnderageException: Age " + age + " is below the legal
age limit.");
} else {
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:

0.109s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Age verification successful.

Compilation Status: Passed

Execution Time:

0.092s

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

Sample Input 2:310 20 301Sample 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();
int[] array = new int[n];

for (int i = 0; i < n; i++) {
```

```
array[i] = scanner.nextInt();
}

int index = scanner.nextInt();

try {
System.out.println("Element at index " + index + ": " + array[index]);
} catch (ArrayIndexOutOfBoundsException e) {
System.out.println("ArrayIndexOutOfBoundsException: Index " + index + " out of
bounds 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:

0.112s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Element at index 1: 20

Compilation Status: Passed

Execution Time:

0.113s

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 1:102Sample Output 1:Result: 5

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

Sample Input 3:a 5Sample 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 .....

try {
String dividendInput = scanner.next();
String divisorInput = scanner.next();

int dividend = Integer.parseInt(dividendInput);
int divisor = Integer.parseInt(divisorInput);
```

```
int result = dividend / divisor;
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 .....
}
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Result: 5

Compilation Status: Passed

Execution Time:

0.103s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

ArithmeticException: Division by zero is not allowed.

Compilation Status: Passed

Execution Time:

0.083s

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_doeSample 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 .....

public InvalidUsernameException(String message) {
super(message);
}

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

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

public void registerUser(String username) throws InvalidUsernameException {
if (username.length() < 5) {
throw new InvalidUsernameException("Too short.");
}
if (!username.matches("[a-zA-Z0-9]+")) {
throw new InvalidUsernameException("Contains non-alphanumeric characters.");
```

```
}
System.out.println("Username registered successfully.");
}

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

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

Scanner scanner = new Scanner(System.in);
UserManager userManager = new UserManager();

String username = scanner.nextLine();

try {
userManager.registerUser(username);
} 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:

0.092s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Invalid username: Too short.

Compilation Status: Passed

Execution Time:

0.093s

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

Sample Input 2:200200Sample 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 .....
```

```
public InsufficientFundsException(String message) {
super(message);
}

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

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

private double balance;

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

public void withdraw(double amount) throws InsufficientFundsException {
if (amount > balance) {
throw new InsufficientFundsException("Withdrawal exceeds balance.");
}
balance -= amount;
System.out.println("Withdrawal successful. Remaining balance: " + balance);
}

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

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

Scanner scanner = new Scanner(System.in);

// Input for initial balance and withdrawal amount
double initialBalance = scanner.nextDouble();
double withdrawalAmount = scanner.nextDouble();

BankAccount account = new BankAccount(initialBalance);

try {
account.withdraw(withdrawalAmount);
} catch (InsufficientFundsException e) {
System.out.println("Insufficient funds: " + 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:

0.117s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Withdrawal successful. Remaining balance: 500.0

Compilation Status: Passed

Execution Time:

0.124s

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, □ (1 ≤ □ ≤ 100), denoting the number of elements in the list.The second line contains □ space-separated integers.Output Format:Print the reordered list, with all odd numbers appearing before all even numbers, maintaining their relative order.

Sample Input:54 3 2 1 5Sample 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 scanner = new Scanner(System.in);

int n = scanner.nextInt();
List<Integer> numbers = new ArrayList<>();

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

List<Integer> odds = new ArrayList<>();
List<Integer> evens = new ArrayList<>();

for (int number : numbers) {
if (number % 2 != 0) {
odds.add(number);
} else {
evens.add(number);
}
}

odds.addAll(evens);

for (int num : odds) {
System.out.print(num + " ");
}

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


Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

3 1 5 4 2

Compilation Status: Passed

Execution Time:

0.105s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

21 43 65 10 32 54

Compilation Status: Passed

Execution Time:

0.106s

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 helloSample 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 scanner = new Scanner(System.in);

String input = scanner.nextLine().toLowerCase();
String[] words = input.split("\\s+");

Set<String> uniqueWords = new HashSet<>(Arrays.asList(words));

System.out.println(uniqueWords.size());

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


Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

2

Compilation Status: Passed

Execution Time:

0.089s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

5

Compilation Status: Passed

Execution Time:

0.086s

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 ≤ n ≤ 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 ≤ m ≤ 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:41 2 3 433 4 5Sample Output:34

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();
Set<Integer> set1 = new HashSet<>();
for (int i = 0; i < n; i++) {
set1.add(scanner.nextInt());
```

```
}

int m = scanner.nextInt();
Set<Integer> set2 = new HashSet<>();
for (int i = 0; i < m; i++) {
set2.add(scanner.nextInt());
}

set1.retainAll(set2);

if (set1.isEmpty()) {
System.out.println("No common elements.");
} else {
for (int element : set1) {
System.out.println(element);
}
}

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


Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

3
4

Compilation Status: Passed

Execution Time:

0.089s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

200

Compilation Status: Passed

Execution Time:

0.093s

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 ≤ n ≤ 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:4apple banana apple grapeSample Output:banana 1apple 2grape 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);

int n = scanner.nextInt();
scanner.nextLine(); // Consume the newline character after the integer input
String input = scanner.nextLine();

String[] words = input.split("\\s+");
Map<String, Integer> frequencyMap = new HashMap<>();

for (String word : words) {
frequencyMap.put(word, frequencyMap.getOrDefault(word, 0) + 1);
}
```

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

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

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

banana 1
apple 2
grape 1

Compilation Status: Passed

Execution Time:

0.116s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

cat 1
fish 1
dog 3

Compilation Status: Passed

Execution Time:

0.119s

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 use 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 GrapesSample 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 .....

Scanner scanner = new Scanner(System.in);

int N = scanner.nextInt();
scanner.nextLine(); // Consume the newline character after the integer input
LinkedHashSet<String> shoppingList = new LinkedHashSet<>();

for (int i = 0; i < N; i++) {
String operation = scanner.nextLine();
String[] parts = operation.split(" ", 2);
String command = parts[0];
String item = parts.length > 1 ? parts[1] : "";
```

```
if (command.equals("ADD")) {
shoppingList.add(item);
} else if (command.equals("REMOVE")) {
shoppingList.remove(item);
}
}

System.out.println(String.join(", ", shoppingList));

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

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Apples, Grapes

Compilation Status: Passed

Execution Time:

0.092s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Apples, Bananas, Grapes

Compilation Status: Passed

Execution Time:

0.09s

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 be sorted in descending order of frequency, and in case of a tie, the words should b 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 appleSample Output:apple 3banana 2fruit 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 input = scanner.nextLine();
String[] words = input.split("\\W+"); // Split by non-word characters
Map<String, Integer> frequencyMap = new HashMap<>();

for (String word : words) {
if (!word.isEmpty()) {
frequencyMap.put(word, frequencyMap.getOrDefault(word, 0) + 1);
}
}

List<Map.Entry<String, Integer>> sortedEntries = new
ArrayList<>(frequencyMap.entrySet());
sortedEntries.sort((entry1, entry2) -> {
```

```
int freqCompare = entry2.getValue().compareTo(entry1.getValue());
return (freqCompare != 0) ? freqCompare :
entry1.getKey().compareTo(entry2.getKey());
});

for (Map.Entry<String, Integer> entry : sortedEntries) {
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:

0.111s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

hello 3
world 2

Compilation Status: Passed

Execution Time:

0.109s

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 ≤ n ≤ 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:5cat dog elephant rat batSample 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(); // Consume the newline character after the integer input
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:

0.115s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

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

Compilation Status: Passed

Execution Time:

0.114s

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 ther Use Java's List interface and iterators to solve this problem.

Input Format:The first line contains an integer, n (1 ≤ n ≤ 100), denoting the number 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:5applebananacherrydateguvaSample 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(); // Consume the newline character after the integer input

List<String> words = new ArrayList<>();
for (int i = 0; i < n; i++) {
words.add(scanner.nextLine()); // Read each word on a new line
}

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

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

0.09s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

mouse

Compilation Status: Passed

Execution Time:

0.095s

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 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 FormatA single integer which is the difference between the sum of even numbers and the product of odd numbers.

Sample Input:51 2 3 4 5Sample 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 scanner = new Scanner(System.in);

int n = scanner.nextInt();
List<Integer> numbers = new ArrayList<>();

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

// Lambda expression to compute the sum of even numbers
int sumOfEvens = numbers.stream()
.filter(num -> num % 2 == 0)
.mapToInt(Integer::intValue)
.sum();

// Lambda expression to compute the product of odd numbers
int productOfOdds = numbers.stream()
.filter(num -> num % 2 != 0)
.reduce(1, (a, b) -> a * b);
```

```
// Calculate the difference
int difference = sumOfEvens - productOfOdds;

System.out.println(difference);

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

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

-9

Compilation Status: Passed

Execution Time:

0.093s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

-3

Compilation Status: Passed

Execution Time:

0.096s

36. Functional InterfacesQuestionCreate 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.

DescriptionWrite 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 FormatAn integer n denoting the number of strings in the list.n space-separated strings representing the list elements.Output FormatA single integer which is the length of the longest string.

Sample Input:4applebananastrawberrykiwiSample 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 .....

String findLongest(List<String> strings);

//..... 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 finder = (strings) -> {
String longest = "";
for (String str : strings) {
if (str.length() > longest.length()) {
longest = str;
}
}
```

```
return longest;
};

String longestString = finder.findLongest(list);
System.out.println(longestString.length());

//..... 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:

8

Compilation Status: Passed

Execution Time:

0.089s

37. Local Variable Type Inference (var)

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

DescriptionWrite 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 FormatAn integer n denoting the number of elements in the list.n space-separated integers representing the list elements.Output FormatA single floating-point number which is the average of the list elements.

Sample Input:41 2 3 4Sample 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 .....

var average = list.stream()
.mapToInt(Integer::intValue)
.average()
.orElse(0.0);

System.out.printf("%.1f%n", average);

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


Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

2.5

Compilation Status: Passed

Execution Time:

0.105s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

20.0

Compilation Status: Passed

Execution Time:

0.104s

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 FormatThe sorted list of prime numbers in descending order, each number on a new line.

Sample Input:52 3 4 5 6Sample 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.stream()
.filter(Main::isPrime)  // Filter for prime numbers
.sorted(Comparator.reverseOrder()) // Sort in descending order
.forEach(System.out::println); // Print each prime number

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

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

if (num <= 1) return false; // 0 and 1 are not prime
for (int i = 2; i <= Math.sqrt(num); i++) {
if (num % i == 0) return false; // Found a divisor
}
return true; // It's prime

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


Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

5
3
2

Compilation Status: Passed

Execution Time:

0.095s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Compilation Status: Passed

Execution Time:

0.093s

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

QuestionDevelop 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.

DescriptionWrite 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 FormatAn integer n denoting the number of words in the list.n space-separated words representing the list elements.Output FormatThe resulting list of uppercase words, each word on a new line.

Sample Input:4apple banana pear figSample 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 .....

list.stream()
.filter(word -> word.length() < 5)
.map(String::toUpperCase)
.forEach(System.out::println);

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


Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

PEAR
FIG

Compilation Status: Passed

Execution Time:

0.091s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

ANT

Compilation Status: Passed

Execution Time:

0.098s

40. Lambda Expressions

QuestionWrite 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".

DescriptionWrite 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 FormatAn 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:4level racecar hello worldSample 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 .....

Predicate<String> isPalindrome = str -> str.equals(new
StringBuilder(str).reverse().toString());
boolean hasPalindrome = list.stream().anyMatch(isPalindrome);

System.out.println(hasPalindrome ? "Yes" : "No");

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

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Yes

Compilation Status: Passed

Execution Time:

0.097s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

No

Compilation Status: Passed

Execution Time:

0.091s

41. Local Variable Type Inference (var)

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

DescriptionWrite 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 FormatAn integer n denoting the number of elements in the list.n space-separated integers representing the list elements.Output FormatA single integer which is the second largest number in the list.

Sample Input:51 2 3 4 5Sample 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 .....
// Create a scanner object to read input
var scanner = new Scanner(System.in);

// Read the number of elements
var n = scanner.nextInt();

// Create a list to store the integers
var list = new ArrayList<Integer>();

// Read n integers and add them to the list
for (var i = 0; i < n; i++) {
list.add(scanner.nextInt());
}

// Sort the list in descending order
Collections.sort(list, Collections.reverseOrder());

// Find the second largest number
var largest = list.get(0);
var secondLargest = -1;
```

```
for (var num : list) {
if (num < largest) {
secondLargest = num;
break;
}
}

// Output the second largest number
System.out.println(secondLargest);


//..... 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:

0.091s

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 FormatThe resulting list of squares that are divisible by 3, each number on a new line.

Sample Input:41 2 3 4Sample 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 .....

// Stream API usage:
list.stream()
.map(x -> x * x) // square each number
.filter(x -> x % 3 == 0) // filter squares divisible by 3
.forEach(System.out::println); // print each number


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

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

9

Compilation Status: Passed

Execution Time:

0.093s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

36

Compilation Status: Passed

Execution Time:

0.095s

43. Custom Comparator with Lambda ExpressionProblem 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

FormatPrint 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 .....
// Custom comparator using a lambda expression
people.sort((p1, p2) -> {
// First compare by age in descending order
```

```
if (p1.age != p2.age) {
return Integer.compare(p2.age, p1.age); // Descending order
}
// If ages are the same, compare by name in ascending order
return p1.name.compareTo(p2.name); // Ascending order
});

// Print the sorted list


//..... 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:

0.123s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Eve 25

Dave 22

Compilation Status: Passed

Execution Time:

0.12s

44. Filtering Even Numbers using Functional Interface

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

DescriptionYou 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 FormatThe first line contains an integer n, the number of integers.The next line contains n integers separated by spaces.Output FormatPrint the list of odd numbers, each on a new line.

Sample Input:61 2 3 4 5 6Sample Output:135

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 number); // Method to test the condition



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

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

```
// Read the number of integers
int n = sc.nextInt();
List<Integer> numbers = new ArrayList<>();

// Read n integers into the list
for (int i = 0; i < n; i++) {
numbers.add(sc.nextInt());
}

// Create a Condition to filter out even numbers using a lambda expression
Condition isOdd = number -> number % 2 != 0;

// Filter the list using the condition
for (int number : numbers) {
if (isOdd.test(number)) { // Test if the number is odd
System.out.println(number); // Print odd numbers
}
}


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


Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

1
3
5

Compilation Status: Passed

Execution Time:

0.092s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

5

Compilation Status: Passed

Execution Time:

0.091s

45. Applying Function to a ListProblem StatementUse 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.

DescriptionYou 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 FormatPrint the transformed and sorted list of strings, each on a new line.

Sample Input:4helloworldjavalambda

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 .....

// Create a StringFunction to convert each string to uppercase
StringFunction toUpperCase = s -> s.toUpperCase();

// Transform the list using the StringFunction
List<String> transformedStrings = new ArrayList<>();
for (String s : strings) {
transformedStrings.add(toUpperCase.apply(s));
}

// Sort the transformed list in reverse alphabetical order
Collections.sort(transformedStrings, Collections.reverseOrder());

// Print the transformed and sorted list
for (String s : transformedStrings) {
System.out.println(s);
}


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


Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

WORLD
LAMBDA
JAVA
HELLO

Compilation Status: Passed

Execution Time:

0.092s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

WORLD
PROGRAMME
JAVA

Compilation Status: Passed

Execution Time:

0.093s

46. Combining Two Lists with Functional Interfaces

Problem StatementCombine 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:3123456Sample 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 .....
int combine(int a, int b); // Method to combine two integers


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

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

// Read the number of integers in each list
int n = sc.nextInt();

// Create two lists for the integers
List<Integer> list1 = new ArrayList<>();
List<Integer> list2 = new ArrayList<>();

// Read integers for the first list
for (int i = 0; i < n; i++) {
list1.add(sc.nextInt());
}

// Read integers for the second list
for (int i = 0; i < n; i++) {
list2.add(sc.nextInt());
}

// Create a Combiner to add two integers
Combiner adder = (a, b) -> a + b;

// Create a list to store the combined results
List<Integer> combinedList = new ArrayList<>();

// Combine corresponding elements from the two lists
for (int i = 0; i < n; i++) {
int combinedValue = adder.combine(list1.get(i), list2.get(i));
combinedList.add(combinedValue);
}

// Print the resulting combined list
for (int value : combinedList) {
System.out.println(value);
}


//..... 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

Execution Time:

0.092s

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 bot 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 2Sample Output:2110 -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 .....

// Read the input string of integers
String input = scanner.nextLine();

// Split the input string into an array of strings
String[] stringArray = input.split(" ");

// Convert the string array to an array of integers using Integer wrapper class
int[] intArray = new int[stringArray.length];
for (int i = 0; i < stringArray.length; i++) {
intArray[i] = Integer.parseInt(stringArray[i]); // Parse each string to an integer
}

// Compute the sum of the integers
int sum = 0;
for (int num : intArray) {
sum += num;
}

// Find the maximum and minimum values
int max = Integer.MIN_VALUE;
```

```
int min = Integer.MAX_VALUE;
for (int num : intArray) {
if (num > max) {
max = num; // Update max if the current number is greater
}
if (num < min) {
min = num; // Update min if the current number is lesser
}
}

// Compute the average of the integers
double average = (double) sum / intArray.length; // Cast to double for accurate
division

// Print the results
System.out.println(sum); // Print the sum
System.out.println(max + " " + min); // Print max and min
System.out.printf("%.2f\n", average); // Print average rounded to two decimal places

scanner.close(); // Close the scanner


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


Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
-15
-1 -5
-3.00
```

Compilation Status: Passed

Execution Time:

0.115s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

21
10 -3
4.20

Compilation Status: Passed

Execution Time:

0.111s

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:205Sample Output:Addition: 25Subtraction: 15Multiplication: 100Division: 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);

// Read two lines of input, each containing a string representation of an integer
String input1 = scanner.nextLine();
String input2 = scanner.nextLine();

// Parse the string inputs into integers using Integer wrapper class
int num1 = Integer.parseInt(input1);
int num2 = Integer.parseInt(input2);

// Perform arithmetic operations
int addition = num1 + num2;
int subtraction = num1 - num2;
int multiplication = num1 * num2;

// Handle division with zero check
String division;
if (num2 != 0) {
division = String.valueOf(num1 / num2); // Convert result to String
} else {
division = "Cannot divide by zero"; // Handle division by zero
}

// Print the results of each operation
System.out.println("Addition: " + addition);
System.out.println("Subtraction: " + subtraction);
System.out.println("Multiplication: " + multiplication);
System.out.println("Division: " + division);

scanner.close(); // Close the scanner


//..... 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

Compilation Status: Passed

Execution Time:

0.107s

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:15.753.14Sample Output:Converted Integer from Double: 15Converted Integer from Float: 3Note: 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);

// Read double value from input
double doubleValue = scanner.nextDouble();

// Convert double value to integer using Double.intValue() method
int doubleToInt = new Double(doubleValue).intValue();

// Read float value from input
float floatValue = scanner.nextFloat();

// Convert float value to integer using Float.intValue() method
int floatToInt = new Float(floatValue).intValue();

// Print the integer values and information about rounding or truncation
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.");
//..... 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:

0.12s

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:

0.117s

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:51 2 2 3 4Sample 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 scanner = new Scanner(System.in);

// Read the number of elements in the list
int n = scanner.nextInt();

// Create a HashSet to store unique integers
Set<Integer> uniqueIntegers = new HashSet<>();

// Read the list of integers
for (int i = 0; i < n; i++) {
int integer = scanner.nextInt();
uniqueIntegers.add(integer);
}

// Calculate the unique sum
int uniqueSum = 0;
for (int integer : uniqueIntegers) {
uniqueSum += integer;
}

// Print the unique sum
System.out.println(uniqueSum);

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


Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

10

Compilation Status: Passed

Execution Time:

0.085s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

5

Compilation Status: Passed

Execution Time:

0.089s

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 6Sample 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 scanner = new Scanner(System.in);

// Input: Reading the number of elements in the list
int N = scanner.nextInt();
scanner.nextLine(); // Consume the newline character

// Input: Reading the space-separated strings
String[] stringNumbers = scanner.nextLine().split(" ");

int product = 1; // To hold the product of even integers
boolean foundEven = false; // Flag to check if we found any even integers

// Process each string representation of a number
for (String str : stringNumbers) {
int number = Integer.parseInt(str); // Convert string to integer

// Check if the number is even
if (number % 2 == 0) {
product *= number; // Multiply to the product
foundEven = true; // Mark that we've found an even number
}
}

// Output the result
if (foundEven) {
System.out.println(product); // Print the product of even integers
} else {
System.out.println(0); // Print 0 if no even integers were found
}

// Close the scanner
scanner.close();


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


Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

48

Compilation Status: Passed

Execution Time:

0.09s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

0

Compilation Status: Passed

Execution Time:

0.087s

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:210 fSample 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 scanner = new Scanner(System.in);

// Input: Read the number of hexadecimal numbers
int N = scanner.nextInt();
scanner.nextLine(); // Consume the newline character

// Input: Read the space-separated hexadecimal strings
String[] hexNumbers = scanner.nextLine().split(" ");

int sum = 0; // Variable to hold the sum of hexadecimal numbers

// Convert each hexadecimal string to an integer and compute the sum
for (String hex : hexNumbers) {
sum += Integer.parseInt(hex, 16); // Convert hex string to integer
}

// Output the sum as a hexadecimal string (without 0x prefix)
System.out.println(Integer.toHexString(sum)); // Convert sum to hex string

// Close the scanner
scanner.close();


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


Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

1f

Compilation Status: Passed

Execution Time:

0.091s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

26

Compilation Status: Passed

Execution Time:

0.089s

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:51 2 2 3 3 3Sample Output:1: 12: 23: 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 scanner = new Scanner(System.in);

// Read the number of elements in the list
int n = scanner.nextInt();

// Read the list of integers
Map<Integer, Integer> frequencyMap = new TreeMap<>();
for (int i = 0; i < n; i++) {
int integer = scanner.nextInt();
frequencyMap.put(integer, frequencyMap.getOrDefault(integer, 0) + 1);
}

// Print the frequency of each integer
for (Map.Entry<Integer, Integer> entry : frequencyMap.entrySet()) {
System.out.println(entry.getKey() + ": " + entry.getValue());
}


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


Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

1: 1
2: 2
3: 2

Compilation Status: Passed

Execution Time:

0.112s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

5: 3
6: 1

Compilation Status: Passed

Execution Time:

0.111s

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.toBinaryString() 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:43 5 7 9Sample 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 scanner = new Scanner(System.in);
```

```
// Input: Read the number of integers
int N = scanner.nextInt();
scanner.nextLine(); // Consume the newline character

// Input: Read the space-separated integers
String[] stringNumbers = scanner.nextLine().split(" ");

int totalOnes = 0; // Variable to hold the total count of 1s

// Process each integer
for (String str : stringNumbers) {
int number = Integer.parseInt(str); // Convert string to integer
String binaryString = Integer.toBinaryString(number); // Convert to binary
representation
int countOfOnes = countOnes(binaryString); // Count the number of 1s

totalOnes += countOfOnes; // Add to the total count
}

// Calculate the average count of 1s
double averageOnes = (double) totalOnes / N; // Compute average

// Round to the nearest integer
int roundedAverage = (int) Math.round(averageOnes);

// Output the result
System.out.println(roundedAverage);

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

// Method to count the number of 1s in a binary string
private static int countOnes(String binaryString) {
int count = 0;
for (char bit : binaryString.toCharArray()) {
if (bit == '1') {
count++;
}
}
return count;


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


Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

2

Compilation Status: Passed

Execution Time:

0.089s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

1

Compilation Status: Passed

Execution Time:

0.092s

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 23Sample Output:Min: -10Max: 23-10: Negative0: Zero15: Positive23: Positive

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);

// Input: Read the number of integers
int N = scanner.nextInt();
scanner.nextLine(); // Consume the newline character

// Input: Read the space-separated integers
String[] stringNumbers = scanner.nextLine().split(" ");

// Initialize min and max values
int min = Integer.MAX_VALUE;
int max = Integer.MIN_VALUE;

// List to hold the integers
List<Integer> integers = new ArrayList<>();

// Process each integer
for (String str : stringNumbers) {
int number = Integer.parseInt(str); // Convert string to integer
integers.add(number); // Add the number to the list

// Update min and max values
if (number < min) {
min = number;
}
if (number > max) {
max = number;
}
}

// Output the minimum and maximum values
System.out.println("Min: " + min);
System.out.println("Max: " + max);
```

```
// Output the sign of each integer
for (int number : integers) {
String sign;
if (number > 0) {
sign = "Positive";
} else if (number < 0) {
sign = "Negative";
} else {
sign = "Zero";
}
System.out.println(number + ": " + sign);
}

// Close the scanner
scanner.close();


//..... 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:

0.122s

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:

0.119s

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:Integer100StringOne HundredSample 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 .....
public class Main {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
//  System.out.println("Choose the type of first value (Integer/String):");
String type1 = scanner.next();
//  System.out.println("Enter the first value:");
Object value1;
if (type1.equalsIgnoreCase("Integer")) {
value1 = scanner.nextInt();
} else {
scanner.nextLine(); // Consume the newline character
value1 = scanner.nextLine();
}

//  System.out.println("Choose the type of second value (Integer/String):");
String type2 = scanner.next();
//   System.out.println("Enter the second value:");
Object value2;
if (type2.equalsIgnoreCase("Integer")) {
value2 = scanner.nextInt();
} else {
scanner.nextLine(); // Consume the newline character
value2 = scanner.nextLine();
}

Pair<Object, Object> pair = new Pair<>(value1, value2);
pair.printPair("Original Pair");
pair.swap();
pair.printPair("Swapped Pair");
```

```
}
}


class Pair<T, U> {
private T first;
private U second;

public Pair(T first, U second) {
this.first = first;
this.second = second;
}

public void setFirst(T first) {
this.first = first;
}

public void setSecond(U second) {
this.second = second;
}

public T getFirst() {
return first;
}

public U getSecond() {
return second;
}

public void swap() {
Object temp = first;
first = (T) second;
second = (U) temp;
}

public void printPair(String message) {
System.out.println(message + ": (" + first + ", " + second + ")");
}
}

//..... 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:

0.122s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

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

Compilation Status: Passed

Execution Time:

0.114s

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, c
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 HellopeekpopprintexitSample 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 .....
private ArrayList<T> list = new ArrayList<>();

public void push(T item) {
list.add(item);
}

public T pop() {
if (isEmpty()) {
System.out.println("Stack is empty.");
return null;
}
return list.remove(list.size() - 1);
}

public T peek() {
if (isEmpty()) {
System.out.println("Stack is empty.");
return null;
}
return list.get(list.size() - 1);
}

public boolean isEmpty() {
return list.isEmpty();
}

public void printStack() {
System.out.println("Stack contents: " + list);
```

```
}

//..... 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 .....

boolean exit = false;

while (!exit) {
String input = scanner.nextLine();
String[] parts = input.split(" ", 2);
String command = parts[0];

switch (command) {
case "push":
if (parts.length > 1) {
stack.push(parts[1]);
} else {
System.out.println("Please provide a value to push.");
}
break;
case "pop":
Object popped = stack.pop();
if (popped != null) {
System.out.println("Popped value: " + popped);
}
break;
case "peek":
Object top = stack.peek();
if (top != null) {
System.out.println("Top of the stack: " + top);
}
break;
case "print":
stack.printStack();
break;
case "exit":
exit = true;
break;
default:
System.out.println("Invalid command.");
break;
}
}
scanner.close();

//..... 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:

0.095s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Stack contents: [42, World]

Compilation Status: Passed

Execution Time:

0.091s

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 list of integers.For String, input a list of strings.Output Format:Print the sorted array

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 .....

public void sort(T[] array, Comparator<T> comparator) {
Arrays.sort(array, comparator);
}
//..... YOUR CODE ENDS HERE .....
}

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

Scanner scanner = new Scanner(System.in);
//  System.out.println("Choose the type of array (Integer/String):");
String type = scanner.next();
scanner.nextLine(); // Add this line to consume the newline character

if (type.equalsIgnoreCase("Integer")) {
//  System.out.println("Enter the integers:");
Integer[] integers = Arrays.stream(scanner.nextLine().split("
")).map(Integer::parseInt).toArray(Integer[]::new);
Sorter<Integer> integerSorter = new Sorter<>();
integerSorter.sort(integers, Integer::compare);
System.out.println("Sorted Integer Array: " + Arrays.toString(integers));
} else if (type.equalsIgnoreCase("String")) {
//   System.out.println("Enter the strings:");
String[] strings = scanner.nextLine().split(" ");
Sorter<String> stringSorter = new Sorter<>();
stringSorter.sort(strings, String::compareTo);
```

```
System.out.println("Sorted String Array: " + Arrays.toString(strings));
}

//..... 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:

0.097s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

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

Compilation Status: Passed

Execution Time:

0.092s

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 5enqueue WorldprintexitSample Output 1:Queue contents: [5, World]

Sample Input 2:enqueue 5enqueue WorldpeekdequeueprintexitSample Output 2:Front of the queue: 5Dequeued value: 5Queue 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 .....
private LinkedList<T> list = new LinkedList<>();

public void enqueue(T item) {
list.addLast(item);
}

public T dequeue() {
return list.isEmpty() ? null : list.removeFirst();
}
```

```
public T peek() {
return list.isEmpty() ? null : list.getFirst();
}

public boolean isEmpty() {
return list.isEmpty();
}

public void printQueue() {
System.out.println("Queue contents: " + list);
}


//..... 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 .....
boolean exit = false;

while (!exit) {
String input = scanner.nextLine();
String[] parts = input.split(" ", 2);
String command = parts[0];

switch (command) {
case "enqueue":
if (parts.length > 1) {
queue.enqueue(parts[1]);
} else {
System.out.println("Please provide a value to enqueue.");
}
break;
case "dequeue":
Object dequeued = queue.dequeue();
System.out.println("Dequeued value: " + (dequeued != null ? dequeued : "Queue is
empty."));
break;
case "peek":
Object front = queue.peek();
System.out.println("Front of the queue: " + (front != null ? front : "Queue is empty."));
break;
case "print":
queue.printQueue();
break;
case "exit":
exit = true;
break;
default:
System.out.println("Invalid command.");
```

```
break;
}
}
scanner.close();


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

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Queue contents: [5, World]

Compilation Status: Passed

Execution Time:

0.089s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

Queue contents: [Hello, 20]
Dequeued value: Hello
Queue contents: [20]

Compilation Status: Passed

Execution Time:

0.098s

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 1:10apple10appleSample Output 1:The pairs are equal.

Sample Input 2:2java3codeSample 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 .....

private T first;
private U second;

public Pair(T first, U second) {
this.first = first;
this.second = second;
}

public T getFirst() {
return first;
}
```

```
public U getSecond() {
return second;
}

@Override
public boolean equals(Object obj) {
if (this == obj) return true;
if (obj == null || getClass() != obj.getClass()) return false;
Pair<?, ?> pair = (Pair<?, ?>) obj;
return first.equals(pair.first) && second.equals(pair.second);
}

public void printComparisonResult(Pair<T, U> other) {
if (this.equals(other)) {
System.out.println("The pairs are equal.");
} else {
System.out.println("The pairs are not equal.");
}
}


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

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


Integer firstValue1 = Integer.parseInt(scanner.nextLine());

// System.out.println("Enter the second value for the first pair (String):");
String secondValue1 = scanner.nextLine();

// System.out.println("Enter the first value for the second pair (Integer):");
Integer firstValue2 = Integer.parseInt(scanner.nextLine());

// System.out.println("Enter the second value for the second pair (String):");
String secondValue2 = scanner.nextLine();

Pair<Integer, String> pair1 = new Pair<>(firstValue1, secondValue1);
Pair<Integer, String> pair2 = new Pair<>(firstValue2, secondValue2);

pair1.printComparisonResult(pair2);

scanner.close();


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

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

The pairs are equal.

Compilation Status: Passed

Execution Time:

0.086s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

The pairs are not equal.

Compilation Status: Passed

Execution Time:

0.085s