import java.util.Scanner;

public class Performance {

private int[] marks;

public Performance() {

marks = new int[10];

}

public void readMarks() {

System.out.println("Enter marks for 10 students:");

try (Scanner scanner = new Scanner(System.in)) {

for (int i = 0; i < 10; i++) {

System.out.print("Student " + (i + 1) + ": ");

marks[i] = scanner.nextInt();

// Validate marks (between 0 and 100)

while (marks[i] < 0 || marks[i] > 100) {

System.out.println("Invalid input! Marks should be between 0 and 100.");

System.out.print("Student " + (i + 1) + ": ");

marks[i] = scanner.nextInt();

}

}

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

}

public int highestMark() {

int highest = marks[0];

for (int i = 1; i < marks.length; i++) {

if (marks[i] > highest) {

highest = marks[i];

}

}

return highest;

}

public int leastMark() {

int least = marks[0];

for (int i = 1; i < marks.length; i++) {

if (marks[i] < least) {

least = marks[i];

}

}

return least;

}

public int getMode() {

int mode = -1;

int maxFrequency = 0;

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

int currentMark = marks[i];

int currentFrequency = 1;

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

if (marks[j] == currentMark) {

currentFrequency++;

}

}

if (currentFrequency > maxFrequency || (currentFrequency == maxFrequency && currentMark > mode)) {

mode = currentMark;

maxFrequency = currentFrequency;

}

}

return mode;

}

public int getFreqAtMode() {

int mode = getMode();

int frequency = 0;

for (int mark : marks) {

if (mark == mode) {

frequency++;

}

}

return frequency;

}

public void display() {

System.out.println("\nResults:");

System.out.println("Highest Mark: " + highestMark());

System.out.println("Least Mark: " + leastMark());

System.out.println("Mode: " + getMode());

System.out.println("Frequency at Mode: " + getFreqAtMode());

}

public static void main(String[] args) {

Performance performance = new Performance();

performance.readMarks();

performance.display();

}

}

**Output :**

java -cp /tmp/GKSVqZiZjW Performance

Enter marks for 10 students:

Student 1:2

Student 2: 6

Student 3: 8

Student 4: 7

Student 5: 6

Student 6: 8

Student 7: 5

Student 8: 6

Student 9: 4

Student 10: 9

Results:

Highest Mark: 9

Least Mark: 4

Mode: 6

Frequency at Mode: 3

**Q2:**

public class AlphabetWarGame {

// Default strengths for the letters

private static final int DEFAULT\_STRENGTH\_W = 4;

private static final int DEFAULT\_STRENGTH\_P = 3;

private static final int DEFAULT\_STRENGTH\_B = 2;

private static final int DEFAULT\_STRENGTH\_S = 1;

private static final int DEFAULT\_STRENGTH\_M = 4;

private static final int DEFAULT\_STRENGTH\_Q = 3;

private static final int DEFAULT\_STRENGTH\_D = 2;

private static final int DEFAULT\_STRENGTH\_Z = 1;

// Custom strengths for the letters

private int strengthW;

private int strengthP;

private int strengthB;

private int strengthS;

private int strengthM;

private int strengthQ;

private int strengthD;

private int strengthZ;

// Constructors for different scenarios

// Default constructor

public AlphabetWarGame() {

// Use default strengths

this(DEFAULT\_STRENGTH\_W, DEFAULT\_STRENGTH\_P, DEFAULT\_STRENGTH\_B, DEFAULT\_STRENGTH\_S,

DEFAULT\_STRENGTH\_M, DEFAULT\_STRENGTH\_Q, DEFAULT\_STRENGTH\_D, DEFAULT\_STRENGTH\_Z);

}

// Custom constructor

public AlphabetWarGame(int strengthW, int strengthP, int strengthB, int strengthS,

int strengthM, int strengthQ, int strengthD, int strengthZ) {

this.strengthW = strengthW;

this.strengthP = strengthP;

this.strengthB = strengthB;

this.strengthS = strengthS;

this.strengthM = strengthM;

this.strengthQ = strengthQ;

this.strengthD = strengthD;

this.strengthZ = strengthZ;

}

// Method to determine the winner when given a single word

public String alphabetWar(String word) {

int leftScore = calculateScore(word, "wpbs", strengthW, strengthP, strengthB, strengthS);

int rightScore = calculateScore(word, "mqdz", strengthM, strengthQ, strengthD, strengthZ);

return determineWinner(leftScore, rightScore);

}

// Method to determine the winner when given separate left and right words

public String alphabetWar(String leftWord, String rightWord) {

int leftScore = calculateScore(leftWord, "wpbs", strengthW, strengthP, strengthB, strengthS);

int rightScore = calculateScore(rightWord, "mqdz", strengthM, strengthQ, strengthD, strengthZ);

return determineWinner(leftScore, rightScore);

}

// Helper method to calculate the score based on the given word and letter strengths

private int calculateScore(String word, String letters, int strengthW, int strengthP, int strengthB, int strengthS) {

int score = 0;

for (char letter : word.toCharArray()) {

if (letters.indexOf(letter) != -1) {

switch (letter) {

case 'w':

score += strengthW;

break;

case 'p':

score += strengthP;

break;

case 'b':

score += strengthB;

break;

case 's':

score += strengthS;

break;

}

}

}

return score;

}

// Helper method to determine the winner based on scores

private String determineWinner(int leftScore, int rightScore) {

if (leftScore > rightScore) {

return "Left side wins!";

} else if (leftScore < rightScore) {

return "Right side wins!";

} else {

return "Let's fight again!";

}

}

// Example usage

public static void main(String[] args) {

AlphabetWarGame game = new AlphabetWarGame();

// Test cases

System.out.println(game.alphabetWar("z")); // Right side wins!

System.out.println(game.alphabetWar("zdqmwpbs")); // Let's fight again!

System.out.println(game.alphabetWar("wwwwwwz")); // Left side wins!

}

}

**Output :**

java -cp /tmp/GKSVqZiZjW AlphabetWarGame

Let's fight again!

Left side wins!

Left side wins!