**LAB 1**

1)import java.util.Scanner;

public class Performance {

private int[] marks; // array to store marks

// Constructor to initialize the array

public Performance() {

marks = new int[10];

}

// Method to read marks into the array

public void readMarks() {

Scanner scanner = new Scanner(System.in);

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

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

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

marks[i] = scanner.nextInt();

}

}

// Method to return the highest mark scored in the class

public int highestMark() {

int highest = marks[0];

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

if (marks[i] > highest) {

highest = marks[i];

}

}

return highest;

}

// Method to return the least mark scored in the class

public int leastMark() {

int least = marks[0];

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

if (marks[i] < least) {

least = marks[i];

}

}

return least;

}

// Method to return the mode

public int getMode() {

int mode = marks[0];

int maxFrequency = 1;

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

int frequency = 1;

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

if (marks[i] == marks[j]) {

frequency++;

}

}

if (frequency > maxFrequency || (frequency == maxFrequency && marks[i] > mode)) {

mode = marks[i];

maxFrequency = frequency;

}

}

return mode;

}

// Method to return the frequency at mode

public int getFreqAtMode() {

int mode = getMode();

int frequency = 0;

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

if (marks[i] == mode) {

frequency++;

}

}

return frequency;

}

// Method to display the result

public void display() {

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();

}

}

2) public class AlphabetWarGame {

// Default strengths

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

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 with default and custom strengths

public AlphabetWarGame() {

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);

}

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 passing a single word

public void determineWinner(String word) {

int leftScore = 0;

int rightScore = 0;

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

switch (c) {

case 'w':

leftScore += strengthW;

break;

case 'p':

leftScore += strengthP;

break;

case 'b':

leftScore += strengthB;

break;

case 's':

leftScore += strengthS;

break;

case 'm':

rightScore += strengthM;

break;

case 'q':

rightScore += strengthQ;

break;

case 'd':

rightScore += strengthD;

break;

case 'z':

rightScore += strengthZ;

break;

}

}

printWinner(leftScore, rightScore);

}

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

public void determineWinner(String leftWord, String rightWord) {

int leftScore = calculateScore(leftWord, true);

int rightScore = calculateScore(rightWord, false);

printWinner(leftScore, rightScore);

}

private int calculateScore(String word, boolean isLeft) {

int score = 0;

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

if (isLeft) {

switch (c) {

case 'w':

score += strengthW;

break;

case 'p':

score += strengthP;

break;

case 'b':

score += strengthB;

break;

case 's':

score += strengthS;

break;

}

} else {

switch (c) {

case 'm':

score += strengthM;

break;

case 'q':

score += strengthQ;

break;

case 'd':

score += strengthD;

break;

case 'z':

score += strengthZ;

break;

}

}

}

return score;

}

private void printWinner(int leftScore, int rightScore) {

if (leftScore > rightScore) {

System.out.println("Left side wins!");

} else if (leftScore < rightScore) {

System.out.println("Right side wins!");

} else {

System.out.println("Let's fight again!");

}

}

public static void main(String[] args) {

AlphabetWarGame defaultGame = new AlphabetWarGame();

defaultGame.determineWinner("z");

AlphabetWarGame customGame = new AlphabetWarGame(5, 3, 2, 1, 4, 3, 2, 1);

customGame.determineWinner("zdqmwpbs");

AlphabetWarGame anotherCustomGame = new AlphabetWarGame();

anotherCustomGame.determineWinner("wwwwwwz");

}

}