6.10 (Sales Commissions)

import java.util.Scanner;

public class SalesCommissions {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int[] ranges = new int[9]; // For 9 salary ranges

System.out.print("Enter number of salespeople: ");

int numSalespeople = input.nextInt();

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

System.out.printf("Enter gross sales for salesperson %d: $", i + 1);

double sales = input.nextDouble();

int salary = 200 + (int)(0.09 \* sales);

if (salary >= 1000) {

ranges[8]++;

} else {

ranges[(salary / 100) - 2]++;

}

}

System.out.println("\nSalary Range\tNumber of Salespeople");

System.out.println("----------------------------------");

String[] rangeLabels = {

"$200-299", "$300-399", "$400-499", "$500-599",

"$600-699", "$700-799", "$800-899", "$900-999", "$1000+"

};

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

System.out.printf("%s\t\t%d%n", rangeLabels[i], ranges[i]);

}

input.close();

}

}

6.11 (One-Dimensional Array Operations)

public class ArrayOperations {

public static void main(String[] args) {

// Part a: Initialize counts array to zeros

int[] counts = new int[10];

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

counts[i] = 0;

}

// Part b: Increment each element in bonus array

int[] bonus = new int[15];

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

bonus[i] += 1;

}

// Part c: Display bestScores in column format

int[] bestScores = {95, 89, 91, 97, 100};

System.out.println("Best Scores:");

for (int score : bestScores) {

System.out.println(score);

}

}

}

// 6.12 (Duplicate Elimination)

import java.util.Scanner;

public class DuplicateElimination {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int[] numbers = new int[5];

int count = 0;

while (count < 5) {

System.out.printf("Enter number %d (10-100): ", count + 1);

int num = input.nextInt();

if (num < 10 || num > 100) {

System.out.println("Number must be between 10 and 100.");

continue;

}

boolean isDuplicate = false;

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

if (numbers[i] == num) {

isDuplicate = true;

break;

}

}

if (!isDuplicate) {

numbers[count] = num;

count++;

}

System.out.print("Unique values: ");

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

System.out.print(numbers[i] + " ");

}

System.out.println();

}

input.close();

}

}

// 6.13 (Two-Dimensional Array Initialization)

The order in which elements are set to zero is row-by-row:

1. sales[0][0], sales[0][1], sales[0][2], sales[0][3], sales[0][4]

2. sales[1][0], sales[1][1], sales[1][2], sales[1][3], sales[1][4]

3. sales[2][0], sales[2][1], sales[2][2], sales[2][3], sales[2][4]

// 6.14 (Variable-Length Argument List)

public class VariableLengthArguments {

public static int product(int... numbers) {

int result = 1;

for (int num : numbers) {

result \*= num;

}

return result;

}

public static void main(String[] args) {

System.out.println("Product of 1, 2, 3: " + product(1, 2, 3));

System.out.println("Product of 5, 10: " + product(5, 10));

System.out.println("Product of 2, 4, 6, 8: " + product(2, 4, 6, 8));

}

}

// 6.15 (Command-Line Arguments)

public class CommandLineArray {

public static void main(String[] args) {

int arraySize = 10; // default size

if (args.length > 0) {

arraySize = Integer.parseInt(args[0]);

}

int[] array = new int[arraySize];

System.out.printf("Array of size %d created.%n", arraySize);

}

}

// 6.16 (Enhanced for Statement)

public class EnhancedForSum {

public static void main(String[] args) {

double sum = 0;

for (String arg : args) {

sum += Double.parseDouble(arg);

}

System.out.println("Sum: " + sum);

}

}

// 6.17 (Dice Rolling)

import java.util.Random;

public class DiceRolling {

public static void main(String[] args) {

Random random = new Random();

int[] frequency = new int[13]; // sums range from 2 to 12

for (int i = 0; i < 36000000; i++) {

int die1 = 1 + random.nextInt(6);

int die2 = 1 + random.nextInt(6);

frequency[die1 + die2]++;

}

System.out.println("Sum\tFrequency");

for (int sum = 2; sum <= 12; sum++) {

System.out.printf("%d\t%d%n", sum, frequency[sum]);

}

}

}

// 6.18 (Airline Reservations System)

import java.util.Scanner;

public class AirlineReservation {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

boolean[] seats = new boolean[10]; // false means available

while (true) {

System.out.println("Please type 1 for First Class");

System.out.println("Please type 2 for Economy");

System.out.print("Choice: ");

int choice = input.nextInt();

if (choice == 1) {

assignSeat(seats, 0, 4, "First Class");

} else if (choice == 2) {

assignSeat(seats, 5, 9, "Economy");

} else {

System.out.println("Invalid choice. Try again.");

continue;

}

System.out.print("Another booking? (y/n): ");

if (!input.next().equalsIgnoreCase("y")) {

break;

}

}

input.close();

}

private static void assignSeat(boolean[] seats, int start, int end, String section) {

for (int i = start; i <= end; i++) {

if (!seats[i]) {

seats[i] = true;

System.out.printf("Boarding pass: Seat %d (%s)%n", i + 1, section);

return;

}

}

System.out.printf("%s is full. Would you like Economy instead? (y/n): ", section);

Scanner input = new Scanner(System.in);

if (input.next().equalsIgnoreCase("y")) {

String otherSection = section.equals("First Class") ? "Economy" : "First Class";

int otherStart = section.equals("First Class") ? 5 : 0;

int otherEnd = section.equals("First Class") ? 9 : 4;

assignSeat(seats, otherStart, otherEnd, otherSection);

} else {

System.out.println("Next flight leaves in 3 hours.");

}

}

}

// 6.19 (Total Sales)

import java.util.Scanner;

public class TotalSales {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

double[][] sales = new double[5][4]; // products x salespeople

System.out.print("Enter number of sales slips: ");

int numSlips = input.nextInt();

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

System.out.printf("Slip %d:%n", i + 1);

System.out.print("Salesperson (1-4): ");

int person = input.nextInt() - 1;

System.out.print("Product (1-5): ");

int product = input.nextInt() - 1;

System.out.print("Value: $");

double value = input.nextDouble();

sales[product][person] += value;

}

// Display table

System.out.printf("%10s%8s%8s%8s%8s%12s%n",

"Product", "SP1", "SP2", "SP3", "SP4", "Total");

double[] personTotals = new double[4];

double grandTotal = 0;

for (int product = 0; product < 5; product++) {

System.out.printf("%10d", product + 1);

double productTotal = 0;

for (int person = 0; person < 4; person++) {

System.out.printf("%8.2f", sales[product][person]);

productTotal += sales[product][person];

personTotals[person] += sales[product][person];

}

System.out.printf("%12.2f%n", productTotal);

grandTotal += productTotal;

}

System.out.printf("%10s", "Total");

for (double total : personTotals) {

System.out.printf("%8.2f", total);

}

System.out.printf("%12.2f%n", grandTotal);

input.close();

}

}

6.20 public class TurtleGraphics {

private static final int SIZE = 20;

private static final int[][] floor = new int[SIZE][SIZE];

private static int x = 0, y = 0;

private static boolean penDown = false;

private static int direction = 0; // 0=right, 1=down, 2=left, 3=up

public static void main(String[] args) {

int[] commands = {2, 5, 12, 3, 5, 12, 3, 5, 12, 3, 5, 12, 1, 6, 9};

executeCommands(commands);

}

private static void executeCommands(int[] commands) {

for (int cmd : commands) {

switch (cmd) {

case 1: penDown = false; break; // Pen up

case 2: penDown = true; break; // Pen down

case 3: direction = (direction + 1) % 4; break; // Turn right

case 4: direction = (direction + 3) % 4; break; // Turn left

case 5:

int steps = commands[++i];

move(steps);

break;

case 6: displayFloor(); break;

case 9: return; // End

}

}

}

private static void move(int steps) {

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

if (penDown) {

floor[y][x] = 1;

}

switch (direction) {

case 0: if (x < SIZE-1) x++; break;

case 1: if (y < SIZE-1) y++; break;

case 2: if (x > 0) x--; break;

case 3: if (y > 0) y--; break;

}

}

}

private static void displayFloor() {

for (int[] row : floor) {

for (int cell : row) {

System.out.print(cell == 1 ? "\*" : " ");

}

System.out.println();

}

}

}

6.21 (Knight's Tour)  
public class KnightsTour {

private static final int SIZE = 8;

private static final int[][] board = new int[SIZE][SIZE];

private static final int[] horizontal = {2, 1, -1, -2, -2, -1, 1, 2};

private static final int[] vertical = {-1, -2, -2, -1, 1, 2, 2, 1};

public static void main(String[] args) {

int currentRow = 3, currentCol = 4; // Start near center

board[currentRow][currentCol] = 1;

for (int move = 2; move <= 64; move++) {

int nextMove = findNextMove(currentRow, currentCol);

if (nextMove == -1) {

System.out.println("Tour ended at move " + (move-1));

break;

}

currentRow += vertical[nextMove];

currentCol += horizontal[nextMove];

board[currentRow][currentCol] = move;

}

printBoard();

}

private static int findNextMove(int row, int col) {

int minAccess = 9;

int bestMove = -1;

for (int move = 0; move < 8; move++) {

int newRow = row + vertical[move];

int newCol = col + horizontal[move];

if (isValid(newRow, newCol) && board[newRow][newCol] == 0) {

int access = countAccessibility(newRow, newCol);

if (access < minAccess) {

minAccess = access;

bestMove = move;

}

}

}

return bestMove;

}

private static int countAccessibility(int row, int col) {

int count = 0;

for (int move = 0; move < 8; move++) {

int newRow = row + vertical[move];

int newCol = col + horizontal[move];

if (isValid(newRow, newCol) && board[newRow][newCol] == 0) {

count++;

}

}

return count;

}

private static boolean isValid(int row, int col) {

return row >= 0 && row < SIZE && col >= 0 && col < SIZE;

}

private static void printBoard() {

for (int[] row : board) {

for (int cell : row) {

System.out.printf("%3d", cell);

}

System.out.println();

}

}

}

// 6.22 (Knight's Tour: Brute-Force Approaches)

import java.util.Random;

public class KnightsTourBruteForce {

private static final int SIZE = 8;

private static final int[][] board = new int[SIZE][SIZE];

private static final int[] horizontal = {2, 1, -1, -2, -2, -1, 1, 2};

private static final int[] vertical = {-1, -2, -2, -1, 1, 2, 2, 1};

private static Random random = new Random();

public static void main(String[] args) {

int fullTours = 0;

int attempts = 0;

while (fullTours < 1) { // Run until first full tour

attempts++;

if (attemptRandomTour() == 64) {

fullTours++;

System.out.println("Full tour found on attempt " + attempts);

printBoard();

}

resetBoard();

}

}

private static int attemptRandomTour() {

int currentRow = random.nextInt(SIZE);

int currentCol = random.nextInt(SIZE);

board[currentRow][currentCol] = 1;

for (int move = 2; move <= 64; move++) {

int nextMove = getRandomValidMove(currentRow, currentCol);

if (nextMove == -1) {

return move - 1;

}

currentRow += vertical[nextMove];

currentCol += horizontal[nextMove];

board[currentRow][currentCol] = move;

}

return 64;

}

private static int getRandomValidMove(int row, int col) {

int[] validMoves = new int[8];

int count = 0;

for (int move = 0; move < 8; move++) {

int newRow = row + vertical[move];

int newCol = col + horizontal[move];

if (newRow >= 0 && newRow < SIZE && newCol >= 0 && newCol < SIZE

&& board[newRow][newCol] == 0) {

validMoves[count++] = move;

}

}

if (count == 0) return -1;

return validMoves[random.nextInt(count)];

}

private static void resetBoard() {

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

for (int j = 0; j < SIZE; j++) {

board[i][j] = 0;

}

}

}

private static void printBoard() {

for (int[] row : board) {

for (int cell : row) {

System.out.printf("%3d", cell);

}

System.out.println();

}

}

}

// 6.24 (Eight Queens)

public class EightQueens {

private static final int SIZE = 8;

private static int[] queens = new int[SIZE]; // column position for each row

public static void main(String[] args) {

solve(0);

printBoard();

}

private static boolean solve(int row) {

if (row == SIZE) return true;

for (int col = 0; col < SIZE; col++) {

if (isSafe(row, col)) {

queens[row] = col;

if (solve(row + 1)) {

return true;

}

}

}

return false;

}

private static boolean isSafe(int row, int col) {

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

// Check column and diagonals

if (queens[i] == col ||

queens[i] - i == col - row ||

queens[i] + i == col + row) {

return false;

}

}

return true;

}

private static void printBoard() {

for (int row = 0; row < SIZE; row++) {

for (int col = 0; col < SIZE; col++) {

System.out.print(queens[row] == col ? "Q " : ". ");

}

System.out.println();

}

}

}

// 6.25 (Eight Queens: Brute-Force Approaches)

import java.util.Random;

public class EightQueensBruteForce {

private static final int SIZE = 8;

private static Random random = new Random();

public static void main(String[] args) {

int[] queens = new int[SIZE];

int attempts = 0;

while (true) {

attempts++;

randomPlacement(queens);

if (isValidSolution(queens)) {

System.out.println("Solution found after " + attempts + " attempts:");

printBoard(queens);

break;

}

}

}

private static void randomPlacement(int[] queens) {

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

queens[i] = random.nextInt(SIZE);

}

}

private static boolean isValidSolution(int[] queens) {

// Check all pairs of queens

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

for (int j = i + 1; j < SIZE; j++) {

// Same column or same diagonal

if (queens[i] == queens[j] ||

Math.abs(queens[i] - queens[j]) == Math.abs(i - j)) {

return false;

}

}

}

return true;

}

private static void printBoard(int[] queens) {

for (int row = 0; row < SIZE; row++) {

for (int col = 0; col < SIZE; col++) {

System.out.print(queens[row] == col ? "Q " : ". ");

}

System.out.println();

}

}

}

// 6.26 (Knight's Tour: Closed-Tour Test)

// Add this method to the Knight's Tour solution:

private static boolean isClosedTour(int startRow, int startCol, int endRow, int endCol) {

for (int move = 0; move < 8; move++) {

if (endRow + vertical[move] == startRow &&

endCol + horizontal[move] == startCol) {

return true;

}

}

return false;

}

// Call it after completing a full tour to check if it's closed.

// 6.27 (Sieve of Eratosthenes)

public class SieveOfEratosthenes {

public static void main(String[] args) {

boolean[] primes = new boolean[1000];

for (int i = 2; i < primes.length; i++) {

primes[i] = true;

}

for (int i = 2; i < Math.sqrt(primes.length); i++) {

if (primes[i]) {

for (int j = i \* i; j < primes.length; j += i) {

primes[j] = false;

}

}

}

System.out.println("Prime numbers between 2 and 999:");

for (int i = 2; i < primes.length; i++) {

if (primes[i]) {

System.out.print(i + " ");

}

}

}

}

// 6.28 (Tortoise and Hare Race)

import java.util.Random;

public class TortoiseAndHare {

public static void main(String[] args) {

Random random = new Random();

int tortoise = 1, hare = 1;

System.out.println("BANG !!!!!");

System.out.println("AND THEY'RE OFF !!!!!");

while (tortoise < 70 && hare < 70) {

// Tortoise move

int tMove = random.nextInt(10) + 1;

if (tMove <= 5) tortoise += 3; // Fast plod

else if (tMove <= 7) tortoise -= 6; // Slip

else tortoise += 1; // Slow plod

// Hare move

int hMove = random.nextInt(10) + 1;

if (hMove <= 2) ; // Sleep

else if (hMove <= 4) hare += 9; // Big hop

else if (hMove <= 5) hare -= 12; // Big slip

else if (hMove <= 8) hare += 1; // Small hop

else hare -= 2; // Small slip

// Ensure positions are within bounds

if (tortoise < 1) tortoise = 1;

if (hare < 1) hare = 1;

// Display current positions

displayPositions(tortoise, hare);

// Check for winner

if (tortoise >= 70 && hare >= 70) {

System.out.println("It's a tie!");

break;

} else if (tortoise >= 70) {

System.out.println("TORTOISE WINS!!! YAY!!!");

break;

} else if (hare >= 70) {

System.out.println("Hare wins. Yuch.");

break;

}

}

}

private static void displayPositions(int t, int h) {

for (int i = 1; i <= 70; i++) {

if (i == t && i == h) {

System.out.print("OUCH!!!");

i += 6; // Skip past the "OUCH!!!" text

} else if (i == t) {

System.out.print("T");

} else if (i == h) {

System.out.print("H");

} else {

System.out.print(" ");

}

}

System.out.println();

}

}

// 6.29 (Fibonacci Series)

import java.util.Scanner;

public class Fibonacci {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter n: ");

int n = input.nextInt();

System.out.printf("Fibonacci(%d) = %d%n", n, fibonacci(n));

// Part b: Find largest Fibonacci that fits in int

int maxIntFib = findMaxIntFibonacci();

System.out.println("Largest Fibonacci that fits in int: " + maxIntFib);

// Part c: Using double

System.out.printf("Fibonacci(%d) using double = %.0f%n",

100, fibonacciDouble(100));

input.close();

}

public static int fibonacci(int n) {

if (n <= 1) return n;

int a = 0, b = 1;

for (int i = 2; i <= n; i++) {

int temp = a + b;

a = b;

b = temp;

}

return b;

}

public static int findMaxIntFibonacci() {

int a = 0, b = 1;

while (true) {

if (Integer.MAX\_VALUE - a < b) {

return b;

}

int temp = a + b;

a = b;

b = temp;

}

}

public static double fibonacciDouble(int n) {

if (n <= 1) return n;

double a = 0, b = 1;

for (int i = 2; i <= n; i++) {

double temp = a + b;

a = b;

b = temp;

}

return b;

}

}// 6.29 (Fibonacci Series)

import java.util.Scanner;

public class Fibonacci {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter n: ");

int n = input.nextInt();

System.out.printf("Fibonacci(%d) = %d%n", n, fibonacci(n));

// Part b: Find largest Fibonacci that fits in int

int maxIntFib = findMaxIntFibonacci();

System.out.println("Largest Fibonacci that fits in int: " + maxIntFib);

// Part c: Using double

System.out.printf("Fibonacci(%d) using double = %.0f%n",

100, fibonacciDouble(100));

input.close();

}

public static int fibonacci(int n) {

if (n <= 1) return n;

int a = 0, b = 1;

for (int i = 2; i <= n; i++) {

int temp = a + b;

a = b;

b = temp;

}

return b;

}

public static int findMaxIntFibonacci() {

int a = 0, b = 1;

while (true) {

if (Integer.MAX\_VALUE - a < b) {

return b;

}

int temp = a + b;

a = b;

b = temp;

}

}

public static double fibonacciDouble(int n) {

if (n <= 1) return n;

double a = 0, b = 1;

for (int i = 2; i <= n; i++) {

double temp = a + b;

a = b;

b = temp;

}

return b;

}

}

// 6.33 (Polling)

import java.util.Scanner;

public class PollingProgram {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

String[] topics = {

"Climate Change", "Healthcare", "Education",

"Economic Inequality", "Immigration"

};

int[][] responses = new int[5][10]; // 5 topics, 10 ratings each

System.out.println("Rate each issue from 1 (least important) to 10 (most important)");

// Collect responses

boolean moreRespondents = true;

while (moreRespondents) {

System.out.println("\nNew Respondent:");

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

System.out.printf("%s: ", topics[i]);

int rating = input.nextInt();

while (rating < 1 || rating > 10) {

System.out.println("Rating must be 1-10. Try again:");

rating = input.nextInt();

}

responses[i][rating - 1]++;

}

System.out.print("Another respondent? (y/n): ");

moreRespondents = input.next().equalsIgnoreCase("y");

}

// Calculate and display results

System.out.println("\nPoll Results:");

System.out.printf("%-20s", "Topic");

for (int i = 1; i <= 10; i++) {

System.out.printf("%4d", i);

}

System.out.println(" Average");

int highestTotal = 0, lowestTotal = Integer.MAX\_VALUE;

String highestTopic = "", lowestTopic = "";

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

System.out.printf("%-20s", topics[i]);

int total = 0, count = 0;

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

System.out.printf("%4d", responses[i][j]);

total += (j + 1) \* responses[i][j];

count += responses[i][j];

}

double average = (double) total / count;

System.out.printf("%8.2f%n", average);

if (total > highestTotal) {

highestTotal = total;

highestTopic = topics[i];

}

if (total < lowestTotal) {

lowestTotal = total;

lowestTopic = topics[i];

}

}

System.out.println("\nHighest point total: " + highestTopic + " (" + highestTotal + ")");

System.out.println("Lowest point total: " + lowestTopic + " (" + lowestTotal + ")");

input.close();

}

}