COM00141M

Department of Computer Science

**Algorithms and Data Structures**

# Assessment Background/Scenario

## Scenario

A department store needs an application for tracking sales, calculating sales targets, and producing reports. Each department submits its monthly sales as shown in Table 1. These sales are then presented to the company executive board in a quarterly report. Each quarter consist of 3 months starting with the first quarter being from January to March inclusively.

|  | **Electrical** | **Kitchen** | **Bathroom** | **Soft Furnishing** | **Accessories** |
| --- | --- | --- | --- | --- | --- |
| **April** | 67 | 65 | 63 | 18 | 16 |
| **May** | 63 | 67 | 63 | 24 | 23 |
| **June** | 78 | 56 | 65 | 22 | 21 |
| **July** | 78 | 45 | 71 | 19 | 19 |
| **August** | 104 | 56 | 73 | 17 | 20 |
| **September** | 103 | 72 | 69 | 16 | 19 |

Table 1: Sample data for 2nd & 3rd Quarter Sales per department (in £1,000)

# Task 1:

Write an algorithm in pseudocode for 3 of the above report outputs.

Pseudocode for 1.

**//a program to calculate the total sales for each department per quarter**

    // Define constants

Constants are data items with values that remain constant. They are used throughout a program, as opposed to variables, and should be named and declared as constants. [1] The four quarters of the year and the five departments are constants that will not change and thus can be declared as constants.

// Initialise total quarters, only data for 2 quarters has been provided so assign value 2 instead of 4

totalQuarters = 2

// Initialise array for departments

An array is a special data type in Java that can be thought of as a container to store a collection of items. These items are sometimes referred to as the elements of the array or items in the array. [2] The index is first initialised to zero (the first element in the array is indexed at 0), which is incremented with each loop and the loop condition will eventually become false when we have reached the end of the list [3] (the final element). If there are five departments in an array, there are five elements, and the indices would be 0, 1, 2, 3 and 4.

   departments = ["Electrical", "Kitchen", "Bathroom", "Soft Furnishing", "Accessories"]

// Initialise 2D array for total sales

Array initialisation means to set aside memory for the array, the value is set to 0, so that it can be incremented by the nested loop. This is to be read as “set the value of totalSales to zero” or alternatively as “totalSales becomes equal to zero”. [5]

    totalSales[totalQuarters][length(departments)] = 0

    // Initialise a 3D array for monthly sales

 A 3D array is used because there are three dimensions of data. The first dimension of the array represents the quarters, the second dimension of the array represents the departments, and the third dimension of the array represents the months within each quarter.

 // Calculate total sales for each department per quarter

Use nested loops, loops within loops with the innermost loop needing to be completed before the outermost loop can, to iterate through each quarter, each department, and each month in each quarter; q represents the quarter, d represents the department and m represents the month.

    // Loop through each quarter

    FOR each quarter q FROM 0 TO totalQuarters - 1:

    // Loop through each department

    FOR each department d FROM 0 TO length(departments) - 1:

        // Loop through each month

        FOR each month m FROM 0 TO 2:  // 3 months in a quarter

 Declare int numMonths which receives the actual number of months for the current department.

            int numMonths = monthlySales[q][d].length;

            // Accumulate monthly sales for each department per quarter

Accumulation operator += adds the value on the right to the value on the left.

            totalSales[q][d] += monthlySales[q][departments[d]][m]

 Display method is responsible for displaying the total sales for each department per quarter and the calculated tax for each quarter. Results will include total sales, tax for each quarter and currency formatting.

    // Display the total sales for each department per quarter

    FOR each quarter q FROM 0 TO totalQuarters - 1:

    FOR each department d FROM 0 TO length(departments) - 1:

    // Display the total sales for each department per quarter

        PRINT "Quarter " + (q + 1) + " total sales for " + departments[d] + ": £" + totalSales[q][d]

// End of code

  Pseudocode for 4.

**// A program for calculating tax on total sales for each department per quarter based on a 17% tax rate**

    // Define constants

    taxRate = 0.17 // 17%

    // Loop through each quarter

    FOR each quarter q FROM 0 TO totalQuarters - 1:

    // Loop through each department

    FOR each department d FROM 0 TO length(departments) - 1:

// Check if total sales for the first department is non-zero before calculating tax

IF totalSales[q][0] != 0:

        // Calculate tax for the quarter based on the total sales of each department

        taxForQuarter[q] = totalSales[q][d] \* taxRate

ELSE:

// Handle the case where total sales is zero to avoid division by zero

PRINT "Warning: Total sales for this department is zero for this quarter " + (q + 1)

taxForQuarter[q] = 0 // Set tax to zero or handle it according to your application's logic

    // Use a loop to display the tax for each quarter

    // Display the calculated tax for the quarter

    FOR each quarter q FROM 0 TO totalQuarters - 1:

    // Print the calculated tax for each quarter

    PRINT "Tax for Quarter " + (q + 1) + ": " + currencyFormat.format(taxForQuarter[q] \* 1000)

// End of code

    Pseudocode for 5.

**// A program for calculating a new sales target with a 12% increase**

    // Define constant for the increase percentage

    increasePercentage = 0.12 // 12%

    // Initialise arrays for average sales and new sales targets

    averageSales[length(departments)] = 0

    newSalesTargets[length(departments)] = 0

    // Loop through each department

    FOR each department d FROM 0 TO length(departments) - 1:

    // Calculate average sales for the last reported quarter

  averageSales[d] = calculateAverageSales(monthlySales, d)

    // Calculate new sales target with a 12% increase

    newSalesTargets[d] = averageSales[d] \* (1 + increasePercentage)

// Check if average sales are non-zero before calculating new sales target

// Exception handler with try and catch

The catch block will attempt to deal with the situation or report the problem if an exception arises from any statement within the associated try block [5]

IF averageSales[d] != 0:

// Calculate new sales target with a 12% increase

newSalesTargets[d] = averageSales[d] \* (1 + increasePercentage)

// Include new line so there is a break in the output between New Sales Target and the first quarter calculations for better presentation

ELSE:

// Handle the case where average sales is zero to avoid division by zero

PRINT "Warning: Average sales is zero for the " + departments[d] “department.”

newSalesTargets[d] = 0 // Set new sales target to zero

// Display the new sales target for the department

    PRINT "New sales target for " + departments[d] + ": £" + newSalesTargets[d]

// End of code

Final exam

The department store in the original brief has now received a full year’s sales reports. This is 12 months of sales, presented as quarterly reports, each for three months. Design an algorithm in pseudocode for the following requirements:

1. The total sales for each quarter across all departments.
2. The total sales for the year for the whole department store
3. The total tax at 17% for the whole department store.
4. Given the average for the quarterly sales across all departments (requirement 1 above), provide a new sales target for each quarter at an increase of 12%.

// Pseudocode for Full Year Sales Analysis

// Constants

totalQuarters = 4 // Full year with 4 quarters

departments = ["Electrical", "Kitchen", "Bathroom", "Soft Furnishing", "Accessories"]

taxRate = 0.17 // 17% tax rate

increasePercentage = 0.12 // 12% increase for new sales target

// Arrays and variables

int[][][] monthlySales // Monthly sales data for the full year

int[][] totalSales = new int[totalQuarters][length(departments)] // Total sales for each quarter

double[] taxForQuarter = new double[totalQuarters] // Total tax for each quarte

// Copy existing algorithms for quarterly calculations (if any)

// Algorithm for Full Year Sales Analysis

// Calculate total sales for each quarter across all departments

calculateTotalSales(monthlySales, departments, totalQuarters, totalSales)

// Calculate total sales for the year for the whole department store

totalSalesForYear = calculateTotalSalesForYear(totalSales, totalQuarters)

// Calculate total tax at 17% for the whole department store

totalTaxForYear = calculateTotalTaxForYear(totalSales, totalQuarters, taxRate, taxForQuarter)

// Provide new sales target for each quarter at a 12% increase

calculateNewSalesTarget(monthlySales, departments, increasePercentage)

// Display results for Full Year Sales Analysis

displayFullYearResults(totalSales, totalTaxForYear)

# References

[1] Q. Charatan & A Kans. Java in two semesters: featuring JavaFX 4th edition. Springer.2019. p25

[2] Q. Charatan & A Kans. Java in two semesters: featuring JavaFX 4th edition. Springer.2019. p120

[3] D. Barnes & M Kolling. Object first with Java 4th edition. Prentice Hall. 2022. p114

[4] Q. Charatan & A Kans. Java in two semesters: featuring JavaFX 4th edition. Springer.2019. p23

[5] D. Barnes & M Kolling. Object first with Java 4th edition. Prentice Hall. 2022. p433