

CMPS 312 Mobile Application Development

LAB 2: Dart Fundamentals I

Objective

In this lab you will practice:

- Dart language fundamental constructs
- Higher order functions to search, filter, map and process collections of data

Overview

PART A : Warmup exercises on Dart Basics

PART B : You will solve a practice exercise.

PART A – Dart Basics

1. Write a program that displays all the even numbers from 1 to 100. You should display the results in the same format as shown below. [use for-in]

```
2 4 6 8 10
12 14 16 18 20
22 24 26 28 30
32 34 36 38 40
42 44 46 48 50
52 54 56 58 60
62 64 66 68 70
72 74 76 78 80
82 84 86 88 90
92 94 96 98 100
```

cdl ==>

2. Write and test `getLetterGrade` function that takes a numeric score and returns the corresponding letter grade.
e.g. If the score = 85, then the function should return B+. You can use the below table to identify the ranges for each letter grade. [Hint : use the **when** operator and **NOT if-else**]

Grade Symbol	Description	Percentage
A	Excellent	90 to 100
B+	Very Good	85 to < 90
B	Very Good	80 to < 85
C+	Good	75 to < 80
C	Good	70 to < 75
D+	Pass	65 to < 70
D	Pass	60 to < 65
F	Fail	Less than 60

3. Write a class **Friend** that has 3 properties: `firstname` , `lastname` and `gender`. The `gender` should have “M” as a default value.
 - Add a `toString` method to return a string representation of the object with Mr. title for male and Ms. title for female. E.g., Mr. Abdulahi Hassen or Ms. Fatima Hamza
 - Create a main function. Inside it declare a friends list and initialize with a list of friends shown the table below:

Firstname	Lastname	Gender
Abdulahi	Hassen	M
Fatima	Hamza	F
Fiona	Shrek	F
Abbas	Ibn Fernas	M

- Loop through the friends list and display their details

4. Create cities list and initialize it with "Doha", "Tokyo", "Delhi"
 - a. Add "Dhaka" to the list
 - b. Add "Beijing" to the list
 - c. Create and test a **display** extension function that extends a list to print the list elements.
 - d. Sort the cities list alphabetically then display it
 - e. Sort the cities list in alphabetically in reverse order then display it.
 - f. Remove Beijing from the list of cities

Output

```

43211234567891234 _____ cities _____
Doha
Tokyo
Delhi
_____ After adding Dhaka to the end _____
Doha
Tokyo
Delhi
Dhaka
_____ After adding Beijing to the beginning _____
Beijing
Doha
Tokyo
Delhi
Dhaka
_____ Sorted Cities by alphabetically _____
Beijing
Delhi
Dhaka
Doha
Tokyo
_____ Sorted Cities by alphabetically in reverse _____
Tokyo
Doha
Dhaka
Delhi
Beijing
_____ Cities after removing Beijing _____
Tokyo
Doha
Dhaka
Delhi

```

5. Create **nums** variable to hold a range of values from 5 to 50. [Hint use the List.generate .. operator]. Complete the following tasks using arrow functions and [built in list methods](#).
 - a. Display the elements in **nums**
 - b. Create and test **min** and **max** functions to return the minimum and maximum values in **nums**
 - c. Create and test **sum** function to return the sum of elements in **nums** [Use **reduce** or **fold** function]
 - d. Create and test **average** function to return the average of elements in **nums**
 - e. Cube every number in **nums** and save the result in **cubicNums**. Display the elements in cubNums.

- f. Display the sum of all even numbers.

PART B

Using the concepts you practiced in part A, solve the following questions.

Create a class called **Invoice** that a spare parts store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance variables-a **partNumber**(type String),a **partDescription**(type String),a **quantity** of the item being purchased (type int) and a **price** per item (double).

Your class should have a **constructor** that initializes the four instance variables. Provide a **set** and a **get** method for each instance variable. In addition, provide a method named **getInvoiceAmount** that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0.

Create an **InvoiceTest** application that showcases the functionality of the Invoice class. In this application, generate at least 10 random invoices using a loop. After creating these invoices, calculate and display the following statistics: the highest invoice amount, the lowest invoice amount, and the average invoice amount of all the invoices.