DUBLIN CITY UNIVERSITY

NATIONAL INSTITUTE FOR DIGITAL LEARNING

OPEN EDUCATION

Module: C2: OBJECT ORIENTED PROGRAMMING WITH JAVA

Programme(s): BSC IN INFORMATION TECHNOLOGY
BSC IN MANAGEMENT OF INFO TECH/INFO SYST
CERTIFICATE IN COMPUTER PROGRAMMING
HIGHER DIPLOMA IN SOFTWARE DEVELOPMENT

Assignment 3 2020–2021

Question 1 **(20 marks)**

Create a screencast to demonstrate an understanding of each of the following (with the aid of examples of your own creation):

- Dynamic Binding
- Inheritance
- Polymorphism
- Data Abstraction

Recommended screencast duration: 4–6 minutes (1–1.5 minutes per topic)

Notes:

- You can use the Screencast-o-matic tool available at https://screencast-o-matic.com/ (or any screen-casting tool you may own) to create the screencast.

- The screencast must include your own narration relating to each chosen example (the addition of video from your webcam is optional).

- A perfect studio recording is not expected, if you make a mistake, acknowledge it and carry on!

Question 2 **(40 marks)**

Write a program to allow the user to do some simple addition and subtraction tests. The program should allow for three skill levels. Level 1 should test only addition of numbers less than 10 whose sum is less than 10. Level 2 should test addition of random single digit numbers. Level 3 should test subtraction of one-digit numbers whose difference is greater than or equal to zero.

The program should generate random numbers for each question, get the user's answer and check if it is correct. It should start at level 1. The user is allowed two tries per question. When the user scores three points (three correct answers in that level) the program should advance to the next skill level.

A Quiz class should be created to provide this functionality, separate to the main program class. The Quiz class should contain a method called askQuestion which returns true if the user got the answer correct within 2 tries or false if they did not. It should also manage the progression from each level to the next.

Example screen output (user interaction shown in green):

```
Starting the test, there are 3 levels
To advance to each level, you must get 3 questions right in the previous
level
4 + 5 = 9
3 + 3 = 7
Oops, that is not right, try again: 6
4 + 1 = 5
Congratulations, advancing to level 2
5 + 6 = 11
2 + 5 = 5
Oops, that is not right, try again: 7
1 + 4 = 5
Congratulations, advancing to level 3
3 - 1 = 2
8 - 4 = 4
8 - 7 = 4
Oops, that is not right, try again: 1
You completed the test, thank you!
```

Question 3 **(20 marks)**

Write Java code that prompts the user to enter an array of four single digit integers. The program should validate the user's entry and use recursion to replace each element of the integer array with the sum of every other element in the array.

Example screen output (user interaction shown in green):

```
Enter an integer in the range of 1 to 9 (inclusive): 55
Invalid entry, number must be in range.
Enter an integer in the range of 1 to 9 (inclusive): 5
Enter an integer in the range of 1 to 9 (inclusive): 3
Enter an integer in the range of 1 to 9 (inclusive): 22
Invalid entry, number must be in range.
Enter an integer in the range of 1 to 9 (inclusive): 2
Enter an integer in the range of 1 to 9 (inclusive): 7
Original array: [5, 3, 2, 7]
Modified array: [12, 14, 15, 10]
```

Question 4 **(20 marks)**

Write a program that processes a file containing the makes of different cars.

First create a Set of 5 allowed car makes.

A valid file contains car makes that are separated by whitespace, e.g. a space, a tab or a newline.

Each car make can be present more than once in the file.

Read the file and every time the make of a car is encountered, if that car is in the set of allowed car makes, increment a total in a Map using the car make as the key, and the counter as the value.

The program should output the contents of the file to the console.

Once the file has been read and closed, and the map fully updated, the program should write each key/value pair in the map to the console.

Example input file:
audi porsche mitsubishi audi bmw volvo porsche mitsubishi bmw mitsubishi

Example console output (assuming all the cars in the input file are in the allowed Set):

```
Contents of file:
audi porsche mitsubishi audi bmw volvo porsche mitsubishi bmw mitsubishi
porsche: 2
audi: 2
mitsubishi: 3
bmw: 2
volvo: 1
```

Deliverables:

A zip file containing:

Question 1: A screencast called *c2a3**q1**_2021_StudentName.mp4*
Question 2: A program called c*2a3**q2**_2021_StudentName.java*
A supporting class called Quiz*.java*
Question 3: A program called c*2a3**q3**_2021_StudentName.java*
Question 4: A file called c*2a3**q4**_2021_StudentName.java*