# U10787

—

# LAB 2 - Part B: More on Classes


| | |
|---|---|
| Author | **Dr. Alexios Louridas** |
| Last Update | **2023-02-13** |
| Number of pages | **2** |

# 1 Task - Modifying a class

Clone the code from: Task1 ClassRepo. Modify the class Account to provide a Withdraw method that withdraws money from an Account. Ensure that the withdrawal amount doesn't exceed the balance. If it does, the balance should not be changed and the method should display a message indicating "Withdrawal amount exceeded account balance." Modify the class test code (Before the class declaration) to test method Withdraw.

# 2 Task - Removing duplicate code

In the AccountTest class, at the start the class test code contains six statements that each display an Account object's Name and Balance. Study these statements and you'll notice that they differ only in the Account object being manipulated—account1 or account2. In this exercise, you'll define a new DisplayAccount method that contains one copy of that output statement. The method's parameter will be an Account object and the method will output the object's Name and Balance. You'll then replace the six duplicated statements with calls to DisplayAccount, passing as an argument the specific Account object to output.

Modify the output code to declare the following DisplayAccount method:

```
void DisplayAccount(Account accountToDisplay)
{
 // place the statement that displays
 // accountToDisplay's Name and Balance here
}
```

Listing 1: DisplayAccount Method Prototype

Replace the comment in the member function's body with a statement that displays accountToDisplay's Name and Balance.

Once you've completed DisplayAccount's declaration, modify your class test code to replace the statements that display each Account's Name and Balance with calls to DisplayAccount—each receiving as its argument the account1 or account2 object, as appropriate. Then, test the updated AccountTest class to ensure that it produces the same output as before.

# 3 Task - Make your own class

Create a new console project that contains a class called Training that includes four pieces of information as auto implemented properties—the training type (type string), the number of trainees (type int), class timings (type DateTime) and charges (type decimal). Your class should have a constructor that initializes the four automatic properties and assumes that the values provided are correct. Provide a method DisplayDetails that displays the training type, number of trainees, class timing and charges (in GBP) separated by tabs. Write a test global method named TrainingTest that demonstrates class Testing's capabilities.

# 4 Task - Make your own application using classes

Clone the solution from: Task4 ClassRepo. In this exercise, you will design a HealthRecord class for a person. The class should include members for the person's first name, last name, gender at

birth, date of birth, height and weight. Your class should have a constructor that receives all of these data. For each member provide a property with appropriate accessors. The class should contain a method to calculate and return the person's age in years and a method to calculate and return a person's BMI. Also the class should have a method to calculate the maximum heart rate depending on the age. The equation for obtaining the maximum heart rate is:

$$\text{Maximum Heart Rate at } 100\% = (220 - \text{age (in years)}) \text{beats per minute} \quad (1)$$

To test your class use the form provided in the project. You do not need to add any more controls. You only need to add functionality for the controls that already exist.