

# Practical Task 2.2

(Pass Task)

Submission deadline: 10:00am Monday, August 2

Discussion deadline: 10:00pm Friday, August 13

## General Instructions

This simple practical task is the start of the series of exercises in which you will develop a small banking system applying all important OOP concepts. Your first task is to complete the Account class, one of the core classes of the future program. You will need to focus on the use of classes and objects and the ability to capture knowledge and behaviour within objects.

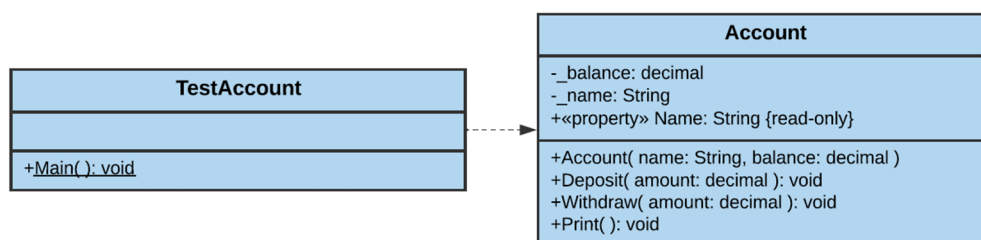
The banking system you are developing must have an account, which allows to withdraw and deposit money. In addition, it must be capable to print its balance and other details to the terminal. There are two auxiliary instance variables to support this functionality and capture the state of an object of the Account class:

- **\_balance** of decimal data type,
- **\_name** of string data type.

The following methods of the class must be made accessible to the user:

- **Account( String name, decimal balance )**  
Constructor. Initializes a new instance of the Account class and sets its initial **\_balance** and **\_name** attributes to the respective values given in the input.
- **void Deposit( decimal amount )**  
Adds funds to the account increasing the **\_balance** by the specified amount.
- **void Withdraw( decimal amount )**  
Withdraws funds from the account decreasing the **\_balance** by the specified amount.
- **void Print( )**  
Prints the contents of **\_balance** and **\_name** of the account to the terminal.
- **String Name**  
Property. Gets the **\_name** of the account.

The following UML diagram should help you to understand the design of the Account class.



Create a new C# Console Application project and write your code for the Account class. Rename the default Program class to TestAccount and write a test driver to examine the public methods of the Account class. Test your program and make sure that an object of the Account class behaves as expected.

Prepare to explain the following to your tutor:

- How **classes** are used to define **objects**.
- How **methods**, **fields (attributes)**, and **properties** all work together when you create a class.
- How fields give knowledge to each object created from a class.
- How methods give capabilities to each object created from a class.

## Further Notes

- Study the way to program and use classes and objects in C# by reading Sections 2.3-2.6 of SIT232 Workbook available in CloudDeakin → Learning Resources.

- The following links will give you more insights on classes/objects and relevant topics:

- <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/classes>
- <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/object-oriented-programming>
- <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/members>
- <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/methods>
- <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/constructors>
- <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/objects>

Because this information is quite detailed and addresses some advanced concepts in OOP, you may mainly focus on the parts you feel relevant to this practical task.

- If you seriously struggle with the remaining concepts used in this practical task, we recommend you to start reading the entire Sections 1 and 2 of SIT232 Workbook.
- In this unit, we will use Microsoft Visual Studio 2019 to develop C# programs. Find the instructions to install the community version of Microsoft Visual Studio 2019 available on the SIT232 unit web-page in CloudDeakin in Learning Resources → Software → Visual Studio Community 2019. You however are free to use another IDE, e.g. Visual Studio Code, if you prefer that.

## Submission Instructions and Marking Process

To get your task completed, you must finish the following steps strictly on time.

- Make sure your programs implement the required functionality. They must compile and have no runtime errors. Programs causing compilation or runtime errors will not be accepted as a solution. You need to test your programs thoroughly before submission. Think about potential errors where your programs might fail.
- **Submit** the expected code files as a solution to the task via OnTrack submission system. You must **record a short video explaining your solution** to the task. Upload the video to one of accessible resources and refer to it for the purpose of marking. You must provide a working link to the video to your marking tutor in OnTrack. Note that the video recording must be made in the **camera on mode**; that is, the tutor must see both the presenter and the shared screen.
- Once your solution is accepted by the tutor, you will be invited to **continue its discussion and answer relevant theoretical questions** through the Intelligent Discussion Service in OnTrack. Your tutor will record several audio questions. When you click on these, OnTrack will record your response live. You must answer straight away in your own words. As this is a live response, you should ensure you understand the solution to the task you submitted.  
Answer all additional questions that your tutor may ask you. Questions will cover the lecture notes; so attending (or watching) the lectures should help you with this **compulsory discussion part**. You should start the discussion as soon as possible as if your answers are wrong, you may have to pass another round, still before the deadline. Use available attempts properly.

Note that we will not accept your solution after **the submission deadline** and will not discuss it after **the discussion deadline**. If you fail one of the deadlines, you fail the task and this reduces the chance to pass the unit. Unless extended for all students, the deadlines are strict to guarantee smooth and on-time work throughout the unit.

Remember that this is your responsibility to keep track of your progress in the unit that includes checking which tasks have been marked as completed in the OnTrack system by your marking tutor, and which are still to be finalised. When grading your achievements at the end of the unit, we will solely rely on the records of the OnTrack system and feedback provided by your tutor about your overall progress and the quality of your solutions.