Contents

UML Class Diagram

This lab serves multiple goals:

- To give you a general understanding of the purpose of UML diagrams,
- To help you read simple UML class diagrams,
- To help you match a UML diagram with its implementation,
- To transform an informal class description into a UML class diagram.

Introduction

Quoting wikipedia[1],

> The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

Stated differently, it is a tool for visually representing programs and their deployment, albeit abstractly. UML uses many different types of diagrams to represent different aspects of systems and software. In this lab, you will practice interpreting and creating one of them: a *class diagram*.

Interpreting a UML class diagram

Reading the diagram

Study the following diagram, then answer follow-up questions:

1. What is the name of this class?
2. How many attributes does this class have?
3. What is the data type of `balance`?
4. How many methods does this class have?

[1] https://www.wikiwand.com/en/Unified_Modeling_Language

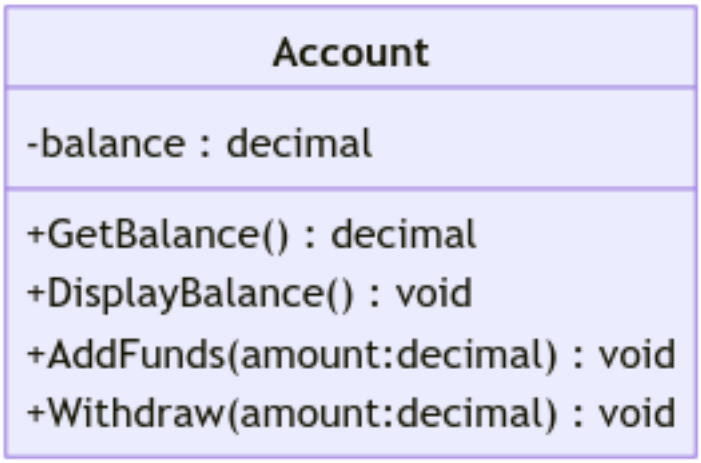

Figure 1: A UML diagram for the Account class (text version[2])

5. What is the significance of + and - in the diagram?
6. You will notice that there are two similar methods: `GetBalance` and `DisplayBalance`.
   - Based on the name, can you interpret the behavior of these methods?
   - Can you think of *why* we might need two similar methods such as these?

Solution:

- The name of the class is "Account".
- This class has only one attribute, called `balance` and of type `decimal`.
- This class has 4 methods.
- The symbol + is used to signify that the member (be it a method or an attribute) is public. The symbol - is used to indicate a private member. In general, in this course, we will let attributes be private and methods be public.
- The `GetBalance` method will return the value of the balance attribute. The `DisplayBalance` method will only display the balance, possibly via `Console.WriteLine`, but it will not return a value since its return type is `void`. There will be times such as this when we may need two different methods. One would be used to make computations (`GetBalance`), and the other would be used to display the information in a formatted fashion (`DisplayBalance`).

Implementing the class

Class diagrams provide a concise way to represent attributes and methods, but they do not describe the implementation of the methods.

Knowing that:

1. `GetBalance` returns the current value of balance.
2. `DisplayBalance` displays the current balance on the screen formatted as currency.

   for example:

   ```
   Your current balance is $1,000,000.00!
   ```

3. `AddFunds` increases the current balance by a specified `amount`.
4. `Withdraw` reduces the current balance by a specified `amount`.

Implement your version of this class in C#. After you are done, you should instantiate an object of the class and ensure it works as described.

Creating your own class diagram

In this next exercise, you will draw your own diagram on paper for practice. Draw the UML diagram of the following class:

1. The class is named `Rectangle`.
2. It has two attributes: `width` and `length`, both of type `int`
3. It has eight methods:
   - setters and getters for each of the two attributes,
   - a `ComputeArea` method to compute the area of a rectangle,
   - a `ComputePerimeter` method to compute the perimeter of a rectangle,
   - a `Swap` method to swap the length and the width of a rectangle, and
   - a `Multiply` method to multiply the length *and* width of a rectangle by a ratio given as an argument that is of type `int`.

Solution:

You can check your answer by referring back to the `Rectangle.cs` file from the Enriched Rectangle[3] project. The UML diagram for this class is indicated in the comments at the beginning of the file (but without the `Multiply` method).

[3] https:/princomp.github.io/code/projects/Enriched_Rectangle.zip

Pushing Further (Optional)

The following are independent tasks that you can perform to broaden your understanding of UML modeling concepts:

1. Class diagrams are one of many kinds of UML diagrams. Have a look at https://www.wikiwand.com/en/Unified_Modeling_Language#Diagrams. In which category are class diagrams: behavior or structure?
2. Besides modeling attributes and methods, class diagrams can also represent relationships between classes. Have a look at https://www.wikiwand.com/en/Class_diagram for more examples of class diagrams and their uses.
3. An Activity Diagram is another type of UML diagram for representing program actions. You will occasionally see activity diagrams in the lecture notes. Have a look at https://www.wikiwand.com/en/Activity_diagram and try to understand the example: "Activity diagram for a guided brainstorming process".