Exercise: Namespaces

This document defines an in-class exercise from the "OOP" Course @ Software University.

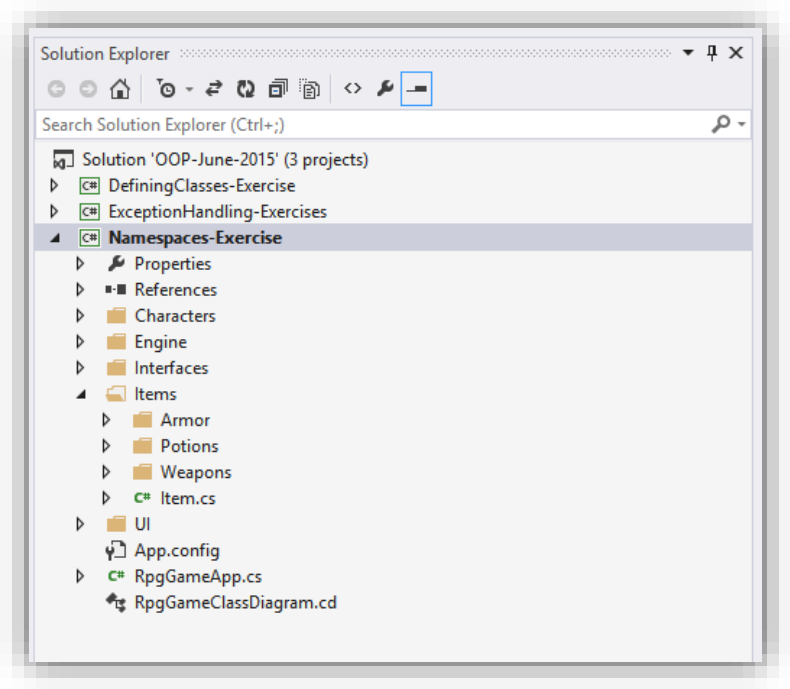
Problem 1. Structure of an RPG Game

When creating a larger project like a game, following a good structure is critical. You may need to create a lot of classes in your application and bundling them properly in **namespaces (with corresponding folders)** can help a great deal. For this problem, **you do not need to implement any classes**, just create the files and put them in appropriate namespaces.

Step 1. Overall Structure

In a game, there are characters, items, and engine taking care of all the interactions, as well as classes taking user input and printing info on the screen or the console. Good abstraction means having interfaces as well.
Create the following **folder structure** (these represent the different **namespaces** in the RPG game):

- **Interfaces** – will hold all interfaces in the application
- **Characters** – will hold all classes related to the game characters (mages, healers, warriors, etc.)
- **Items** – will hold all classes related to game items like weapons, shields, potions, etc.
    - Subfolder **Potions**
    - Subfolder **Armor**
    - Subfolder **Weapons**
- **UI** – will hold classes responsible for the user interface – input and output.
- **Engine** – will hold the game engine.



Step 2. Add Classes

For now, you do not need to bother with inheritance, just add the required classes and leave them unimplemented. The application should contain the following classes:

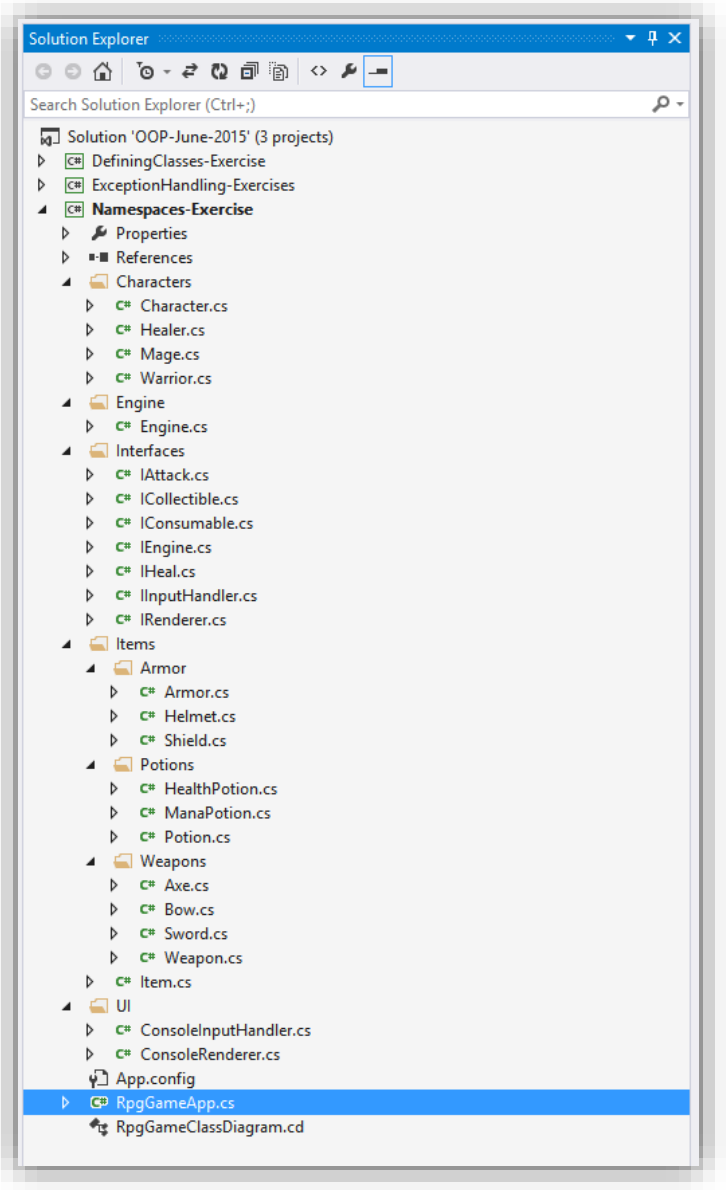
- Engine namespace
    - Engine
- Interfaces namespace (add interfaces here)

© Software University Foundation (softuni.org). This work is licensed under the CC-BY-NC-SA license.
Follow us:

    - IEngine
    - IRenderer
    - IInputHandler
    - IAttack
    - IHeal
    - ICollectible
    - IConsumable
- Characters namespace
    - Character – abstract class
    - Mage
    - Healer
    - Warrior
- Items namespace
    - Item – abstract class
    - Items.Potions namespace
        - Potion – abstract class
        - HealthPotion
        - ManaPotion
    - Items.Armor namespace
        - Armor – abstract class
        - Shield
        - Helmet
    - Items.Weapons namespace
        - Weapon – abstract class
        - Sword
        - Bow
        - Axe
- UI namespace
    - ConsoleRenderer
    - ConsoleInputHandler

Here is an example of how the structure should look in the end:

© Software University Foundation (softuni.org). This work is licensed under the CC-BY-NC-SA license.



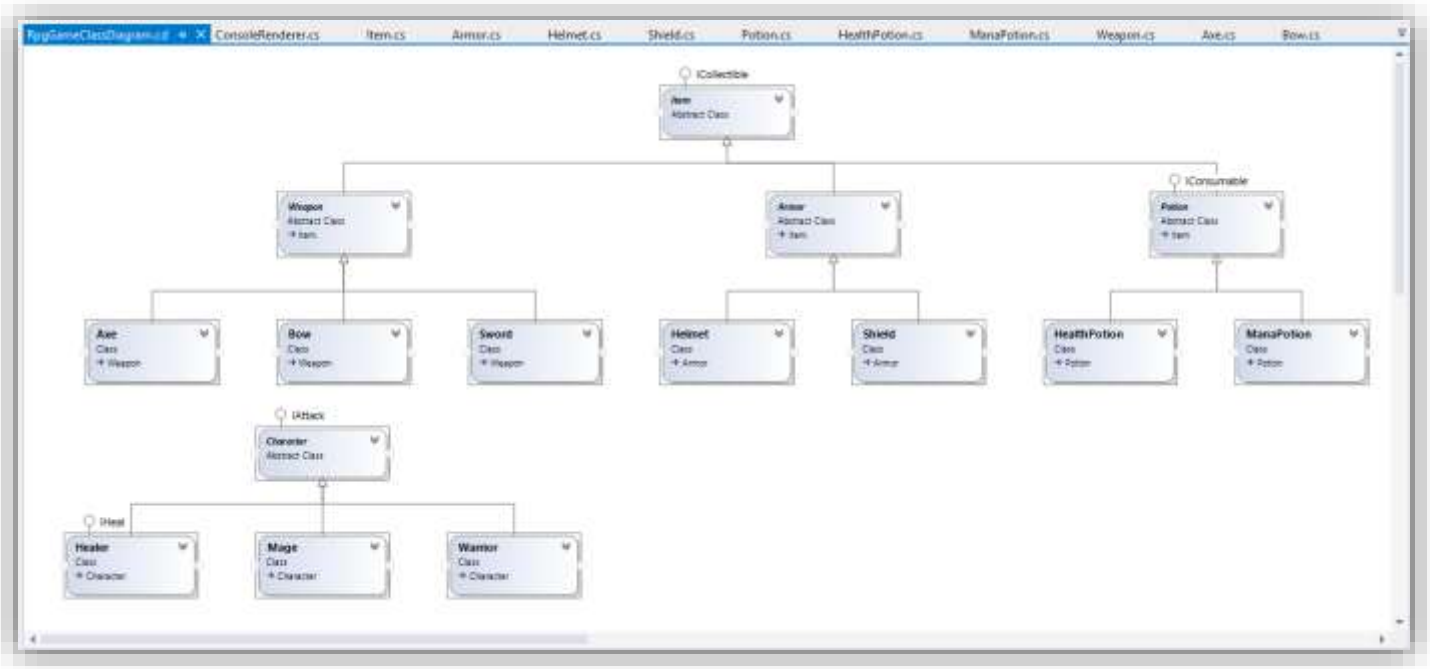
* Bonus – Inheritance

Complete the structure by creating a proper inheritance hierarchy. Here is what needs to be done:

- The Engine should implement the IEngine interface
- The abstract class Character should implement IAttack
- All characters should inherit the Character class
- The Healer should implement the IHeal interface
- The Item class should implement the ICollectible interface
- The Potion, Armor and Weapon abstract classes should inherit the Item class
- The Potion class should implement the IConsumable interface
- The HealthPotion and ManaPotion classes should inherit the Potion class
- The Shield and Helmet classes should inherit the Armor class
- The Sword, Bow and Axe classes should inherit the Weapon class
- The ConsoleRenderer class should implement the IRenderer interface
- The ConsoleInputHandler class should implement the IInputHandler interface

SOFTWARE UNIVERSITY FOUNDATION
© Software University Foundation (softuni.org). This work is licensed under the CC-BY-NC-SA license.
Follow us:

You can create a **class diagram** when you're done. Right-click the project's name, select **View -> View Class Diagram**. Check to see if the inheritance hierarchy is correct. Example:



© Software University Foundation (softuni.org). This work is licensed under the CC-BY-NC-SA license.