Overview

Dragon

Inheritance is an Object-Oriented Programming (OOP) feature that allows objects to behave *polymorphically*. *Polymorph* means "many forms"—in this case, a C# object may be treated as any of the types it has inherited, all the way back to `Object`. Generally, classes that inherit (or extend) another can be thought of as more *specialized* versions of their base classes.

In this lab, you will create a number of classes that form an inheritance hierarchy. You will generate a UML Class Diagram that outlines the relationships and members of each of your classes. You will then translate this diagram into C# code.

Background

Dungeons and Dragons® (D&D) provides descriptions and attribute information for hundreds of different monsters—nemeses of the player characters. These monsters are categorized by type, such as aberrations, dragons, and undead. Various editions of the game include a description of common characteristics for all monsters that share a common type. For example, in the 3rd edition of the game, any creature who is *undead* is immune to all mind-affecting effects, poison, sleep effects, paralysis, stunning, disease, and death effects and has darkvision.

Different subtypes of undead may have more resistances and features that extend beyond these.

Instructions

Your task is to create a class hierarchy, specifying the properties and methods of a number of D&D-style *monsters*. Each of these will be extensions of a monster type class, and the types in-turn will be subclasses of `Creature`. This means that your inheritance hierarchy will be at least three levels deep.

You will be given some starter code containing examples of a couple of monsters, and their associated classes. The starter code also includes a small driver program that includes a menu-driven monster creation routine. You will build upon this starter code to do the following:

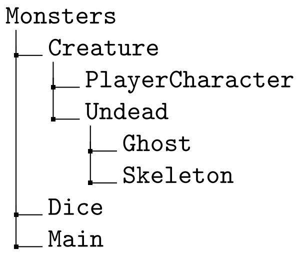
- Map out the starter code as given by creating a UML class diagram outlining the fields, properties, and methods of the given classes and the *relationships* among them.
- Specify at least **one** more *type* of monster (this is the second-level in the inheritance hierarchy, right below `Creature`, and **three** additional monsters. Note that the Roll20 5e Compendium has a listing of many of the standard monsters, their stats, and descriptions. This is a good starting point for planning.

  Each of the subclasses you create must include:

  - A constructor that calls the base class constructor and then initializes data unique to the subclass.
  - Override the properties where necessary (e.g., attributes, resistances, HP, Darkvision, ArmorClass)
  - At least one new method (preferably, an attack).
  - If the creature has an attack, override the `attack()` method.
  - Where appropriate, override the `ToString` method to include more information.

Feel free to edit the driver program (`Program.cs`), but **do not modify any of the other classes included in the starter code**! Additionally, there is a `Dice` class you should use to roll dice. You do not need to create or directly use a `Random` object.

The starter code classes are logically arranged as seen here:



Submission

You will fork the starter code given in the `repl.it` project, modify it, and share with me. The UML diagram should be submitted to Canvas as well, as an attached file or link.