**CSCI 2302**

**Inheritance & Polymorphism Chapter Program Assignment**

**Fictional Character, Antagonist, Being Classes**

**Purpose:** Work with inheritance and polymorphism

**Input:**  None, hard coded into the program

**Output:** To the screen

**Task**:

1. Define/Implement a Being class, which will be the super class for the Fictional Character class and the Antagonist class. The Being class has at least 2 states/fields/attributes; e.g. name. Being has at least 5 behaviors; e.g. 2 constructors, 2 getters methods, and toString method. Override the toString method from the Object class so it will describe the assets of the Beings. This is the super class for your fictional character and antagonist.
2. Define/Implement an Antagonist class that inherits from Being. This is the foe/opponent/archenemy/nemesis/villain… to your fictional character. There should be at least 3 states/fields/attributes and at least 2 behaviors, in addition to the constructors, mutators (as needed) and accessors. Hard code the data in.
3. Modify your Fictional Character class to inherit from Being.
   1. Your Fictional Character still has-a Tool
4. Your main program, MysfasuUsername\_InheritanceAndPolymorphismProgram.java, should:
   1. Instantiate your fictional character, make sure to invoke the method(s) that inform the user of your fictional character, including the description of the being.
   2. Instantiate an antagonist, make sure to invoke the method(s) that inform the user of the capabilities of the antagonist, including the description of the being.
   3. **Tell a scenario** of the fictional character (step a) and this antagonist (step b) in a plot against each other. Make sure to use at least two method from each object (the antagonist and the fictional character).
   4. Instantiate an implicit casting of a Being object with a fictional character subtype (yes – polymorphism).
   5. Instantiate an implicit casting of a Being object with an antagonist subtype (polymorphism).
   6. Define a method that accepts Being objects. In this method, invoke a method that is defined in the Being class.
      1. Invoke this method with the objects created from steps d and e.
   7. Define a method that accepts Being objects. In this method, invoke one of the behaviors that is specific to each object’s subtype. (Make sure to test prior to casting !)
      1. Invoke this method with the objects created from steps d and e.

Requirements: hard code the data in and print to the screen

Make sure to adhere to the requirements that have been stated in class for an object/class, notably the toString methods. Don’t forget to include the comments & the intro comments!

Submit the UML to the Dropbox in Brightspace by D2L and the .java files to the cs server.