# JIT121 Programming Principles Tutorial 7

## Learning Objectives

1. Learn about Inheritance and its advantages by creating subclasses of a given class
2. Learn about an **abstract** class and making subclasses from that class
3. Implement a class according to the specification given by a UML class diagram and a written description of the functionality of the methods in the class.
4. Learn about subclasses and their interaction with their base (or 'parent', or 'super') class.
5. Become familiar with simple Exception Handling techniques

## Assumptions

That you are already familiar with:

* Lectures 1 – 8 and Tutorials 1 – 7

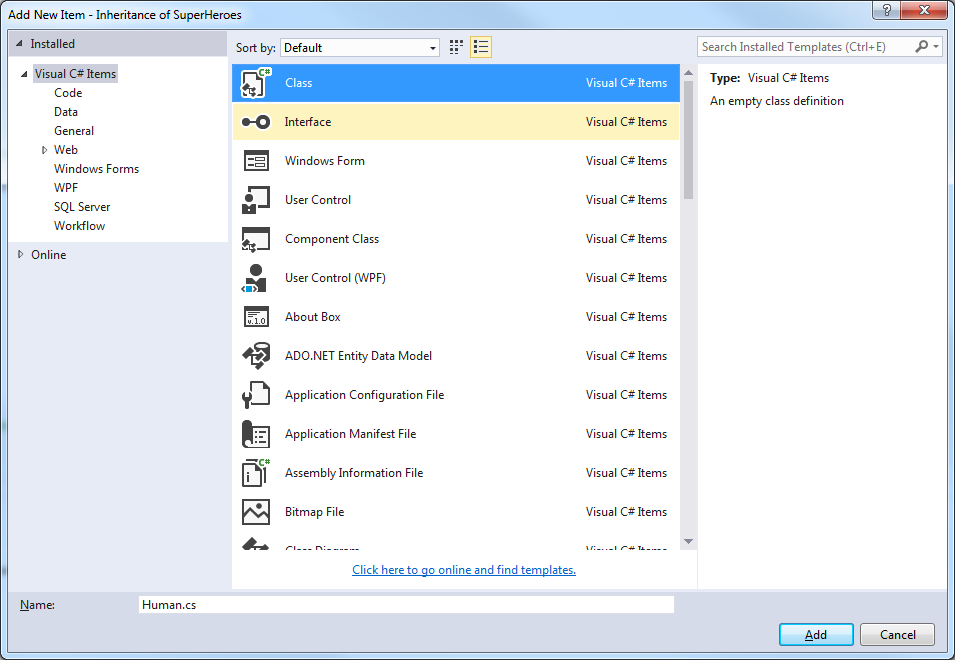
## Activity Overview

1. Creating a Subclass
2. Adding Exception Handling to the Currency Converter GUI

### Activity 1: Creating subclasses

This activity involves creating a subclass of a given base class. The base class is an **abstract** class named **SuperHero.**

From the zip file **SuperHeroes.zip** extract the project folder and open the solution file, **Inheritance of SuprHeroes.sln**. Look at the code of **SuperHero.cs**. It is nearly complete apart from missing the body of the method **SwitchIdentity**. You will complete this method later in this activity.

The focus at the moment is to add a subclass of **SuperHero** to this project. Open the **Solution Explorer** window, select the project name, **Inheritance of SuperHeroes,** right click and select **Add > New Item,** then select **Class** and change itsnameto **Human.cs** andclick the **Add** button.

Figure

### Change the class to be a subclass of SuperHero by adding ": SuperHero" to the class name as follows:

**class** Human : SuperHero {

Ignore the error message for the moment. Add a constructor to this class by copying the following declaration:

**public** Human(**string** name, **string** secretId) :

**base**(name, secretId) {

}

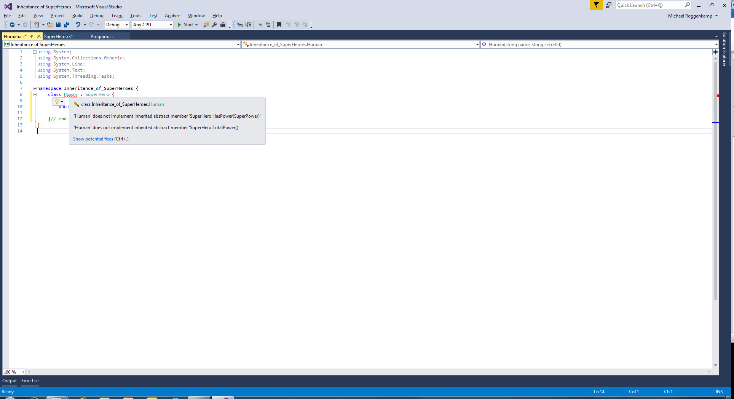


Figure 2

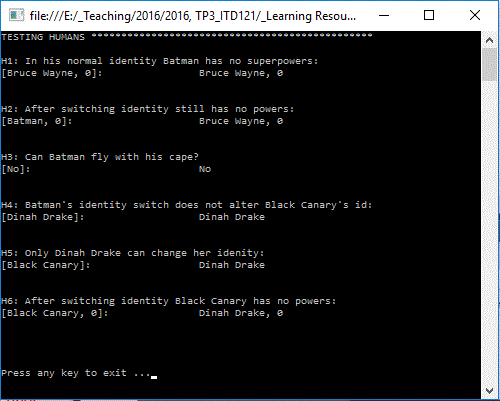
### This constructor calls the base constructor via base(name, secretId) to initialise the two string instance variables in the base class.

### Hover over the class name Human, read the error message, click on Show potential fixes, click Preview changes, then click Apply. See Figure 2. This is an example of refactoring support provided by VS.

The two abstract methods of SuperHeroare now implemented albeit with a simple exception **throw** statement. As a Human does not have any super powers replace body of HasPower with a simple **return false;** statement. Likewise change the body of TotalPower to returning the value 0.

In **Main** of **Program.cs** delete the end of line comment slashes on the call to the method TestHumansas well as the block comments surrounding the method TestHumansand run the program. Correct any compiler errors and/or run-time errors and you will eventually see the output similar to Figure 3:

Figure 3



Note that the contents of “**[...]**” represents the expected output and to the right of that is the actual output. H2: expected output, "**Batman, 0**" does not match the actual output of "**Bruce Wayne, 0**".

Why? Because the method SwitchIdentity in SuperHerohas not been implemented yet. Write the body of SwitchIdentity then run the code again.

Finally convince yourself that there is a one-to-one correspondence between the following UML diagram and your Human class.

|  |
| --- |
| Human extends SuperHero |
|  |
| +Human(string name, string secretId)  +HasPower(SuperPower whatPower) : bool  +TotalPower(): int |

Next we will add two additional subclasses to the **Inheritance of SuperHeroes** project.

1. Add a new class to the project and change its default name to **Enhanced Human.cs**

Make this class a subclass of SuperHerothen add to the body of the class using the following UML class diagram, and the additional information following the diagram.

|  |
| --- |
| Enhanced\_Human extends SuperHero |
| -sumOfPowers: int  -enhanced: bool  -powerSet: List of SuperPower |
| +Enhanced\_Human(string name, string secretId,  List of SuperPower myPowers)  +SwitchIdentity()  +HasPower(SuperPower whatPower) : bool  +TotalPower(): int |

Use the following to declare instance variable powerSet

**private** **List**<SuperPower> powerSet;

The only method of the **List** type that you will need to use in this class is:

**bool** **Contains**(*T* item)

which returns **true** if item is in the **List** and **false** if item is not in the **List**.

The instance variable enhancedis initially **false** as an Enhanced\_Human is born as a mere mortal but later can gain the ability to turn into a SuperHerothrough some magical or science fiction plot device. For example Billy Batson becomes Captain Marvel by saying the word "Shazam".

So when an Enhanced\_Humanswitches identity to their secret super power identity they also acquire a set of super powers and lose them when they change back to their real non-super power identity. Use the value of the instance variable enhancedto indicate if theEnhanced\_Humancurrently has super powers or not.

The constructor will call the base constructor similar to what was in the class Humanand in addition will assign theListparameter to powerSet and will set sumOfPowers to be the sum of the individualSuperPowercontained in powerSet.

Implement the method HasPower to return true if the Enhanced\_Humanis currently using their “super power identity” and has whatPower otherwise return **false**.

Implement the method TotalPowerto returnsumOfPowersifthe Enhanced\_Humanis currently using their “super power identity” otherwise return zero.

Build your solution and remove any compiler errors.

In **Main** of **Program.cs** delete the end of line comment slashes on the call to the method TestEnhancedHumansas well as the block comments surrounding the method TestEnhancedHumansand run the program (correcting any run-time errors). Check that the expected output matches the actual output.

1. Add another class to the project and change its default name to **Super Human.cs**

Make this class a subclass of SuperHerothen add to the body of the class using the following UML class diagram, and the additional information following diagram.

|  |
| --- |
| Super\_Human extends SuperHero |
| -sumOfPowers: int  -powerSet: List of SuperPower |
| +Super\_Human(string name, string secretId,  List of SuperPower myPowers)  + HasPower(SuperPower whatPower) : bool  +TotalPower(): int  +AddSuperPower(SuperPower newPower)  +LoseSinglePower(SuperPower power)  +LoseAllSuperPowers() |

The only methods of the **List** type that you will need to use in this class are

* + **bool** Contains(*T* item)
  + **void** Add(*T* item)which addsitemto the end of the **List.**
  + **bool** Remove(*T* item**)** which returns **true** ifitemis removed from the **List and false** ifitemis not in the **List.**

ASuper\_Humanis born with "powers and abilities far beyond those of mortal men", but they usually adopt a mild-mannered secret identity. For example Wonder Women assumes the identity of Diana Prince unless a crisis arises. However under certain circumstances they may lose one of their super powers or worse lose them all e.g., Superman when exposed to kryptonite.

The constructor is similar to that of Enhanced\_Humanclass.

The bodies of the other methods are straight forward and behave as suggested by their name with the following additional requirements:

* + AddSuperPowerwill only add newPower if that SuperPower is not already in the powerSet. If the SuperPower is added be sure to increase sumOfPowers by the appropriate value.
  + LoseSinglePowerremovespowerand adjustssumofPowersaccordingly ifpoweris inpowerSet**.**

Build Solution and remove any compiler errors.

In **Main** of **Program.cs** delete the end of line comment slashes on the call to the method TestSuperHumansas well as the block comments surrounding the method TestSuperHumansand run the program (correcting any compiler errors and/or run-time errors). Check that the expected output matches the actual output.

Feel free to add your own "super heroes" and create additional test data to the appropriate "test" method.

1. At the moment when Superman loses his power there is no means for him to regain his lost powers once the kryptonite is removed from his immediate surroundings. Add a method to Super\_Humanclasswhich reinstates lost powers.

This is purely a design exercise, there is no right or wrong way to do this. You are free to implement this method in whatever way that comes to mind which apart from adding the new method with a sensible name with or without parameters, may or may not involve adding an additional instance variable(s), and/or changing existing methods.

Once you have written the method, add a call to your new method to the bottom of TestSuperHumans method before the currently commented out last **Console.WriteLine** call. Uncomment this statement and run your program.

**Activity 2: Add Exception Handling to Currency Converter (Tutorial 5)**

Add defensive mechanisms to your solution from Tutorial 5 to handle cases where the user enters invalid input (input checking):

* + Add code to the event handler for the **Button** to generate a **MessageBox** if the user enters a non-numeric value for an amount in AUD to be converted.

