Problem 1 write-up: Richard

The naming scheme I decided to use involves cheese production where you have “HeadCheese” in general as a base class while the child classes “Cheddar” and “Swiss” will inherit from the HeadCheese class. Each child class “Cheddar” and “Swiss” will have two children of their own or grandchild classes. “Cheddar” will have “YellowCheddar” and “WhiteCheddar” as classes that’ll inherit from “Cheddar” class and “WholeSwiss” and “SliceSwiss” will inherit from “Swiss” class. The different naming scheme showcase different varieties of forms that each cheese can come from. The class diagram is called CheeseProduction.

When I added the class diagram to the problem 1 folder, I used the toolbox side menu to help create all the classes making sure you have the class diagram selected to use the toolbox option for the class diagram which is called CheeseProduction. To add a class to the project folder, all you must do is first click and hold on the left mouse button on the class object option and drag it to the class diagram window and let go of the left mouse button. Once you let go of the button, a popup window will appear giving you option to name your class, what kind of access you want it to have and it’s file name which it’s usually the same name of your class plus whether you want to create and new file or add it to an existing file you already have. All the classes will go through this process, but for the base class HeadCheeses you want the access to be abstract instead of public. Since ultimately all other classes share traits with HeadCheeses, we want to visually show the connection using inheritance. To do that we go back to the toolbox under class diagram, we select the option inheritance and this time we first click on the class that will retrieve traits from a different class and click on another class the traits will be coming from. In order words: first selecting the class that will be the child and the second class will be considered the parent. When you’re done, you should see a line starting from the first class you selected to the second class you selected the traits will be coming from with a pointed arrow pointing to.

The class that have the abstract keyword will have code that describe what data it has so when the child classes have access to that data, it can start off with that data and use their own data to help set their own unique behavior for themselves. Although the child classes can execute their parent’s code with codes of their own to set the behavior of the child class, the parent themselves can't execute their own code as it was never meant to do so. This can work on methods as well where the base class method with abstract on it. The base class can define the return type, any parameters and the method’s name while again the child class will have code explaining what the method does. However, the child class using their parent’s codes will need to have the override keyword for them to execute their code.

abstract public class HeadCheese

This shows that the class has the abstract keyword meaning this is considered the base class. This helps me decide what the child classes will be and what do I want them to do when I create the child classes.

public class Cheddar : HeadCheese

This shows one of the child classes Cheddar is the child of HeadCheese the parent. This shows me that Cheddar will have traits from the HeadCheese class, and I can add code to make this class Cheddar stand out from its parent like the child class Swiss.

public class WhiteCheddar : Cheddar

This show the WhiteCheddar class is the child of the Cheddar class even though Cheddar has HeadCheese as a parent which makes WhiteCheddar the grandchild. This can help me go one step farther in creating a class that not only share traits from the parent HeadCheese but it’s child class Cheddar as well. This helps me create a subclass that helps me expand upon creating a new class to make it more unique to the other classes that all share traits with their parent class.