* CE-02 Feedback
* Structure/Efficiency: **Poor** 0/10,
  + Unfortunately, all of the custom classes' variables should have been declared as private. Consider why public properties were implemented if the fields were already public. Remember to continually practice data encapsulation by preventing direct access to a class' data. To follow the OOP principle of data-encapsulation all variable / fields should be declared as **private** by default (with the exception of constant variables since their values cannot change). Doing so creates specific bottlenecks / throughputs for the data change. That allows the class to manage any invalid values attempting to be assigned to its members, and allows the developer to more easily debug the class as there are only certain locations where those values are accessed (get) and / or mutated (set).
  + The Student class appropriately declares the constructor for its base / parent class, but also performs the redundant operations as the base class. The "Person" (base) class performs the assignment operations from the arguments (values from the parameter variables) to the member fields of the class. As such, the child (Student) class of the Person class does *not* need to perform those some assignments in its (Student) constructor.
  + The Program class's Main method contained variables other than the "current" class type requested by the instructions. When declaring extra variables, think about if they are already present in the "current" variable and should be accessed from that object instance. Likewise, determine if the extra variable needs to exist in the Main method's scope (body brackets) or if it can be local to the menu option which instantiates and populates it. Typically it only needs to be a temporary reference to a particular menu case as it will be assigned to a member of the "current" object.
* Classes: **Excellent** 15/15,
  + Please review the comments above in the "structure & efficiency" category.
* Constructors: **Good** 11.25/15,
  + Please review the comments above in the "structure & efficiency" category.
* Menu: **Good** 18.75/25,
  + Unfortunately, the "*Add Students*" menu option does not instantiate new Student class object for each location available in the Course class' array of Student objects. The goal was to ask the user for the number of students the course can contain when the course was created, instantiate the array in the constructor with the provided size (it is expected the array will contain "null" values for its elements), then use the array's length to loop in the add students menu option. That loop would be used to create a new single student object each loop cycle - once for each element / item location in the course's student array.
  + The "CreateTeacher" method should assign the returned newly created teacher object to the current course, if the current course exists.
* Main: **Fair** 4.5/15,
  + Please review the comments above in the "*structure & efficiency*" category concerning the "current" variable usage. Unfortunately, the program exits when the user enters invalid input on the main menu. Review the "default" case of the main menu's switch-statement, and think about how the "return" keyword is used.
* Input Validation: **Fair** 6/20,
  + Unfortunately, empty and / or blank (all spaces) input was accepted. The user should always be required to enter something and something other than solely spaces. Research the [IsNullOrEmpty](https://docs.microsoft.com/en-us/dotnet/api/system.string.isnullorempty?view=netframework-4.7.2" \t "_blank) and [IsNullOrWhiteSpace](https://docs.microsoft.com/en-us/dotnet/api/system.string.isnullorwhitespace?view=netframework-4.7.2" \t "_blank) methods.