Step 7 – Create a student class

* Create fields,
  + First Name - string
  + Last Name – string
  + Student Class – SchoolClass
* Create a constructor
  + Must take a first and last name ( a student can be enrolled without having classes )
* Create properties
* Create a Student object
* Display Student first and last name
* Add a class to the student
* Display student, class, and teacher information
* Create a student list
* Add two more students
* Add classes from Class list

Now that we’ve gotten comfortable creating classes, you are going to create another one for Student.

Fields

* Fields for first name and second name
* Constructor to take the first and second name
* Properties for both

Back in Program.cs

Create a student object, pass it your name.

Create a List<student> called students

Add the student to the list

Display the student information

* Ex. Student Name: Hannah Angel

Go back to Course.cs

Change your \_student type to Student

Change the property to a return type of student

Create a default Student object in the constructor so our code doesn’t throw errors.

Back in Program.cs

Add a student from your student list, to your first class.

Display your results.

Text

Description automatically generated

Before we move on I bet your main is started to look really cluttered with code all about creating new instances of our classes, and then adding them to list. Let’s go ahead and move all this code to a separate method in our Program.cs

Create a new method below our main, call it Preload() and give it no return type. Now copy and past all the code from our main that involves creating instances and assigning them to other classes.

public static void Preload()

{

Course course1 = new Course("Programming 122", "CSI\_122\_2");

courses.Add(course1);

course1 = new Course("Programming 120", "CSI\_120\_2");

courses.Add(course1);

courses.Add(new Course("Programming 252", "CSI\_252\_1"));

Teacher teacher1 = new Teacher("William", "Cram");

teachers.Add(teacher1);

teachers.Add(new Teacher("Josh", "Emery"));

teachers.Add(new Teacher("Dimpy", "Gill"));

Student student1 = new Student("Hannah", "Angel");

students.Add(student1);

students.Add(new Student("Kristyn", "Taniguchi"));

students.Add(new Student("Dylan", "Register"));

courses[0].Teacher = teachers[0];

courses[0].Student = students[0];

} // Preload()

Now just call your preload method at the top of main.

static void Main(string[] args)

{

Preload();

DisplayCourses();

Console.ReadKey();

} // Main

This makes our main a lot cleaner and works just the same. That’s because when main runs, it calls preload, which loads all of our lists just like we were doing. It’s common to create preload methods to load lists up with data. If we ever wanted to prevent this from happening we could just remove the method call in main, or comment it out.

By now you should be fairly comfortable creating classes. So let’s go ahead and refactor some of our code to be more real world.