Step 8 – Change Course to take a List of Students, Teacher to take a list of Courses, and Student to take a List of Courses

Tutorial:

Our code seems to be in working order. We have 3 classes that hold specific information, and we have some code to display information. But there’s a little flaw in our logic. Courses can have more than 1 student, teachers can teach more than 1 course, and students have more than 1 class.

Let’s refactor our code to account for this.

In our Course.cs

Fields

Change your Student object to be a List<student> object, also change the name from \_student to \_students. We want to make it plural to indicate it is a collection.

Constructor

And inside of the constructor, change your default new Student() object to be new List<Student>();

Property

Change your property to return a List<Student>. Delete the setter. We aren’t going to let the user change the list of students themselves. We create it in our constructor and that’s the only one they will work with.

Now for our two other classes we are going to ADD a List<Course> and write the constructor and property to work with them. It’s the same code for both Student.cs and Teacher.cs, because they can both have multiple classes.

Student.cs and Teacher.cs – your making the same exact changes

Field

* Add a new field: List<Course> \_courses;

Constructor

* Add a line that says \_courses = new List<Course>(). This is creating a new list specifically to hold courses for this instance of an object. Each new student or teacher will have a separate list of classes.

Property

* Create a property for Courses.
* The return type should be List<Course>
* Create a getter
* Do NOT create a setter. The users should not be able to change the attached list we made in the constructor.

We’ve now made our classes a bit more realistic. We’ve expanded them to all take more than one object in the logical fields. But now this has changed some of our code in main. Let’s take a look there.