- Create a `Company` class that contains a `List<Department>` (Aggregation).
- Each `Department` contains a `List<Employee>`.
- Each `Employee` can be registered in multiple `Course`s (Association).
- A `Car` contains an `Engine` (Composition).

Requirement:
Write code to create a company with multiple departments, each department having employees, and each employee being enrolled in courses. Print the employee's name + department name + the courses they are enrolled in.

- Create a base class `Person` with common properties (`Name`, `Age`).
- Create `Instructor : Person` and `Student : Person`.
- `Instructor` has a method `TeachCourse()`.
- `Student` has a method `RegisterCourse()`.
- Create a `Course` class. When `Instructor.TeachCourse()` is called, assign the instructor to that course.

Requirement:
Override a method `Introduce()` → Instructor should introduce himself as a teacher, Student as a learner.

- Create an abstract class `Shape` with an abstract method `Area()`.
- Create `Circle` and `Rectangle` classes inheriting from it, implementing `Area()`.
- Create an interface `IDrawable` with `Draw()`.
- Each shape should implement `Draw()` to print a simple drawing in the console.

Requirement:
Create a `List<Shape>` with different shapes, call `Area()` and `Draw()` for each (demonstrating Polymorphism).

- Create a static class `IdGenerator` with a method `GenerateId()` that returns an auto-incremented number.
- Each new `Student` or `Instructor` should automatically get an ID from `IdGenerator` when created.
- Create a `Grade` class with a property `Value` (int).
- Overload the `+` operator to add two grades.
- Overload `==` and `!=` to compare two grades.

Requirement:
If a student has multiple grades, use `+` to calculate the total. Use `==`/`!=` to check if two grades are equal.

- Create an enum `CourseLevel { Beginner, Intermediate, Advanced }`.
- Each course has a level.
- When a student registers in a course, display a message depending on the level (e.g., Beginner → "Good luck starting out!", Advanced → "This will be challenging!").

System Simulation (Bring Everything Together)

- Create a **Company** with departments, each having employees and instructors.
- Create **Students**, register them in different courses with levels.
- Assign instructors to courses.
- Generate a report that shows:
    - Each student's name + enrolled courses + total grades.
    - Each instructor's name + courses they teach.
    - Each department's name + number of employees.