CMPS310
Fall 2021

Lecture 7

Assigning Responsibilities to Classes

Objectives

- The quality of the **overall software design** **depends on how we assign *responsibilities* to the objects**.
  - What properties are important for which class?
  - What methods belong to which class?
  - Which objects should interact with which objects?
- Coupling
- Cohesion

Responsibilities and Methods

There are two types of Responsibilities of classes.

- **Knowing (properties/ attributes)**
  - about private encapsulated data
  - about related objects
  - about things it can derive or calculate
  - Attributes of class.

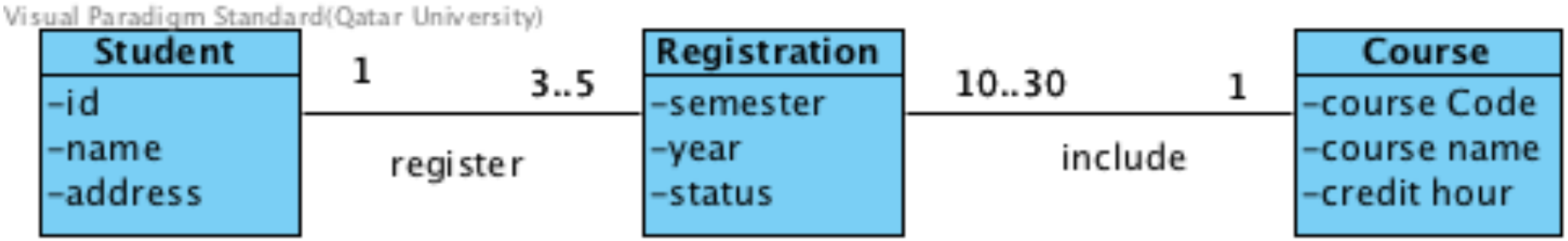
- **Doing (behaviour/ methods)**
  - doing something itself (recursive method)
  - initiating action in other objects (sending messages to other objects)
  - controlling and coordinating activities in other objects
  - methods of a class

Responsibility Driven Design

- Responsibilities implemented by methods
- Some methods act alone and do a task
- Some collaborate with other objects to fulfill their responsibility
  - Example: **Sale** class has **getTotal()** method,
  - **getTotal()** collaborates with another type of objects called **SalesLineItem** to fulfill the responsibility of computing total amount.
    - **SalesLineItem** objects have **getSubtotal()** methods
- Why assigning responsibilities to classes is important:
  - Reduce the degree of dependency of a software component on other.
    - We want less dependency (less coupling)
  - The classes should be focused, that means, a given class performs tasks which are very related to each other
  - The classes cannot do tasks those are unrelated.
- Assigning responsibility can be done using a **Problem-Solution** pair, sometimes it is called pattern.

Assigning a Method to Class

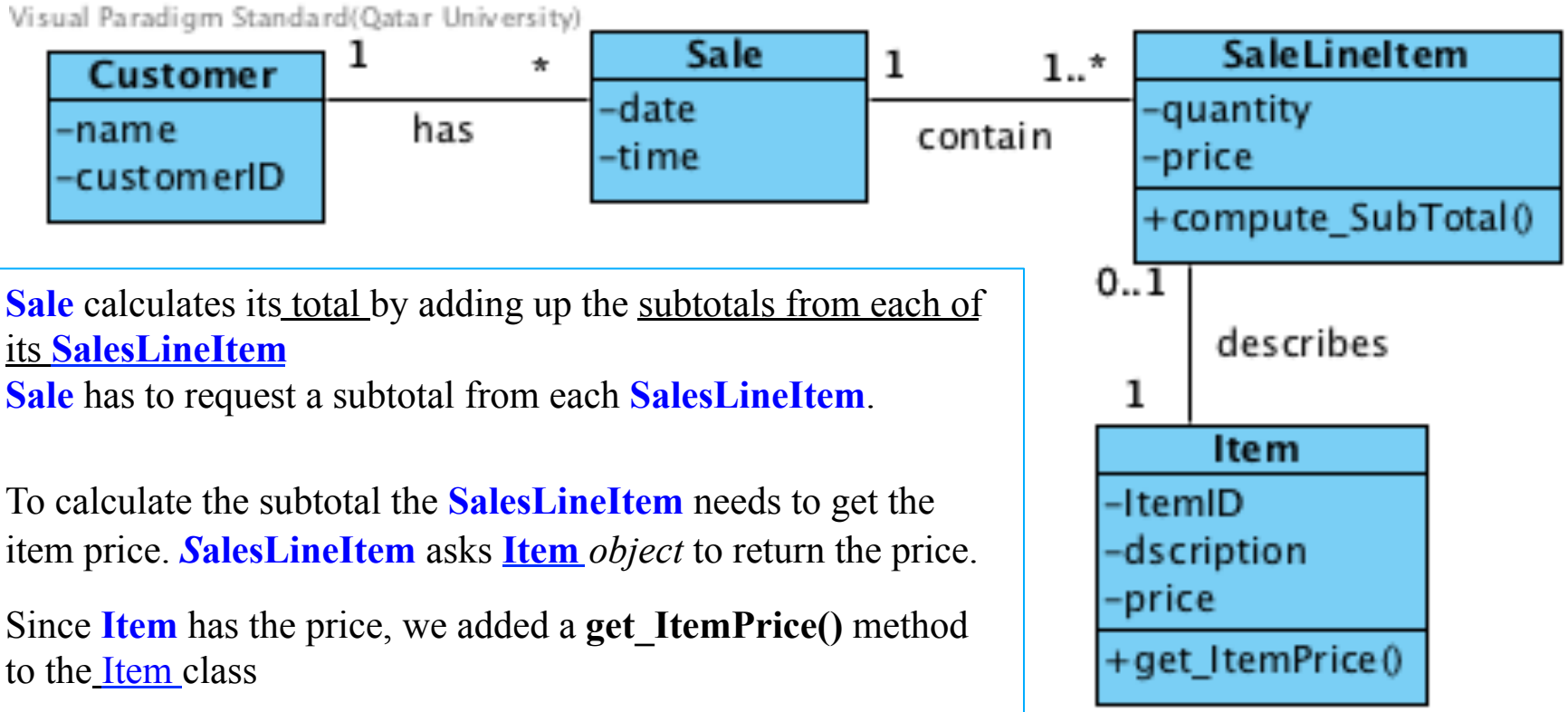
- **Problem:** What is a basic principle by which to assign a method (responsibilities) to objects?
- **Solution (advice):** Assign a method (responsibility) to the class that has the information needed to fulfill it
- **Example**: Which of the following classes will get the responsibilities to **find the credit hour** of a course? Student, Course, or Registration class?



- **Course** class will get this responsibility because it knows the credit hour
- "Objects **do things (method) related to the information they have**."

Example 1: Assigning Responsibilities

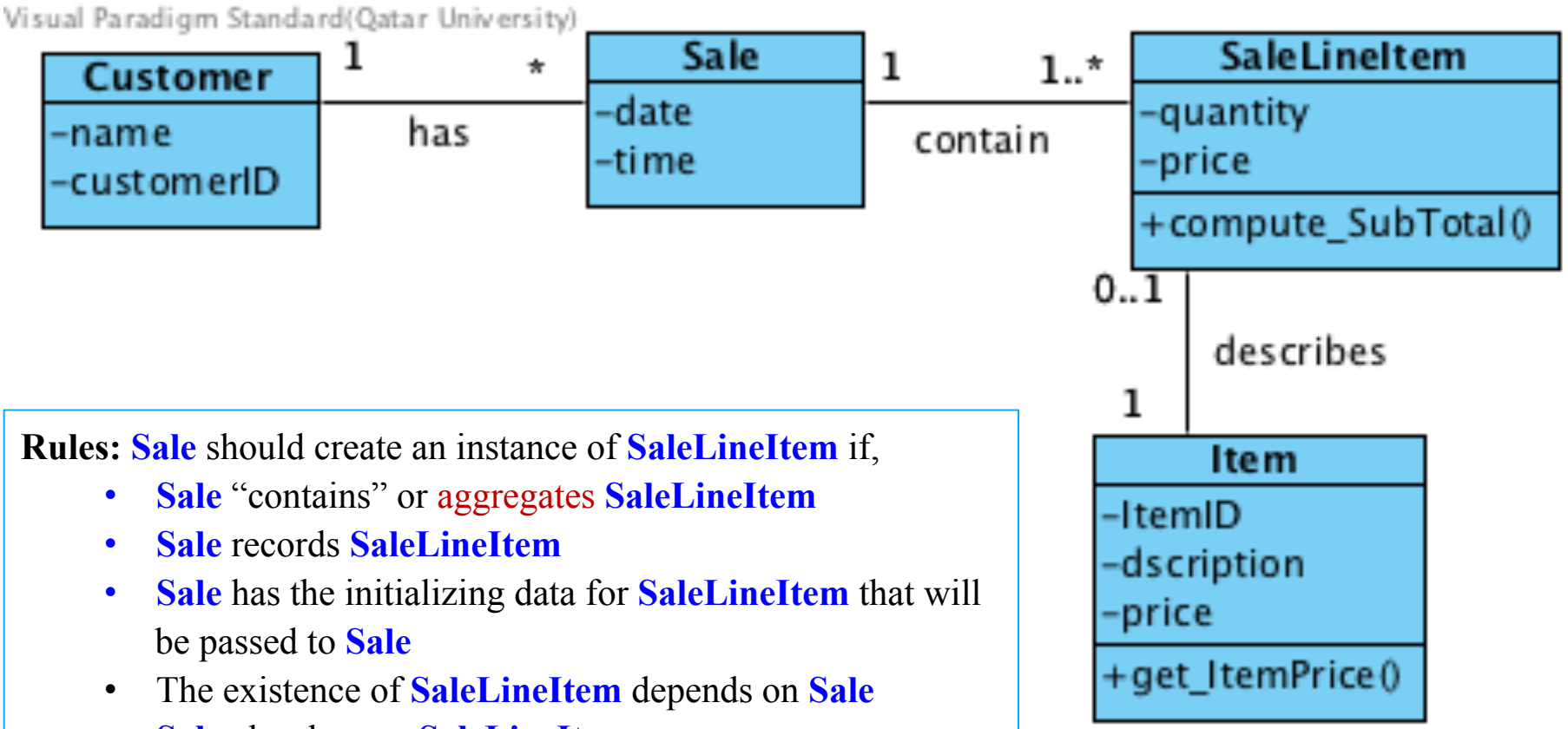
- Which of the following classes will be assigned the responsibilities to find the **total sale** for a customer:
- **Customer**, **Sale**, **SaleLineItem,** or **Item**? Why?



- **Sale** calculates its total by adding up the subtotals from each of its **SalesLineItem**
- **Sale** has to request a subtotal from each **SalesLineItem**.
- To calculate the subtotal the **SalesLineItem** needs to get the item price. ***S*alesLineItem** asks **Item** *object* to return the price.
- Since **Item** has the price, we added a **get_ItemPrice()** method to the Item class

Example 2: Assigning Responsibilities

- Which of the following classes will get the responsibility to **create SaleLineItem** object?
- **Customer, Sale, SaleLineItem**, or **Item**?

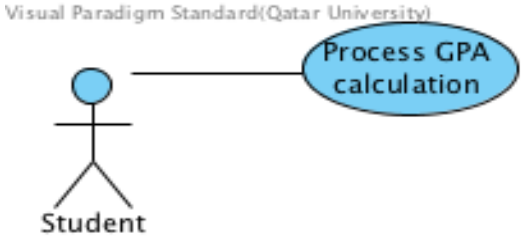


Rules: **Sale** should create an instance of **SaleLineItem** if,

- **Sale** "contains" or aggregates **SaleLineItem**
- **Sale** records **SaleLineItem**
- **Sale** has the initializing data for **SaleLineItem** that will be passed to **Sale**
- The existence of **SaleLineItem** depends on **Sale**
- **Sale** closely uses **SaleLineItem**

Example 3: Assigning Responsibilities
Use case – Class Diagram - Responsibilities

- Assume a student wants to know his/her GPA
- The followings are the use case diagram and the use case specification
- Your tasks:
  - Find the classes in this use case
  - Assign responsibilities to classes in order to compute GPA
  - How to do this?



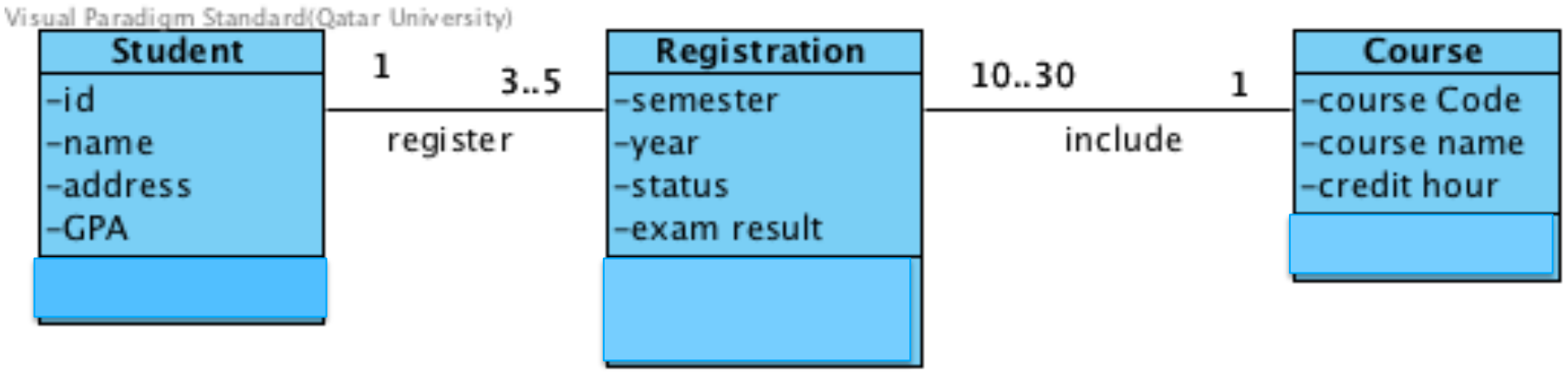
Actor Action	System Response
1. The Student wants to know his/her GPA	
2. The Student enters ID	3. Find exam result of each course that the student has registration
	4. Find credit hour of each course that the student has registration
	4. Compute GPA
	5. Store GPA
	6. Display GPA

Example 3:Finding Classes

Actor Action	System Response
1. The Student wants to know his/her GPA	
2. The Student enters ID	3. Find exam result of each **courses** that the **student** has **registration**
	4. Find credit hour of each **course** that the **student** has **registration**
	4. Compute GPA
	5. Store GPA
	6. Display GPA

- In the use case specification, we have three candidate classes: **Course**, **Student**, **Registration**
- Candidate attributes: **Exam results**, **credit hour**, **GPA**
- Candidate associations: **Student has Registration for Courses**
- Which class will have which attributes?

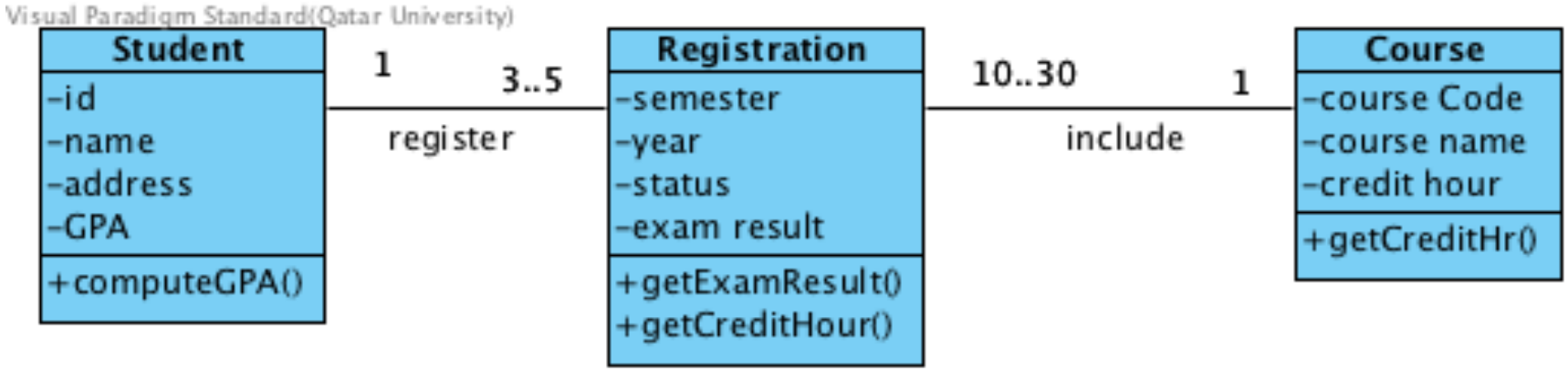
Example 3: Making Association



- Which information is needed to compute GPA?
- The system needs to find the exam result of each course that the student has registered and the credit hour of each course
- How to assign responsibilities to achieve the above?
  - Which class will have the responsibility to get the exam result of each course?
  - Which class should have the responsibility to find the credit hour of each course?
  - Finally, which class will have the responsibility to compute GPA?

Example 3: Assigning Responsibilities

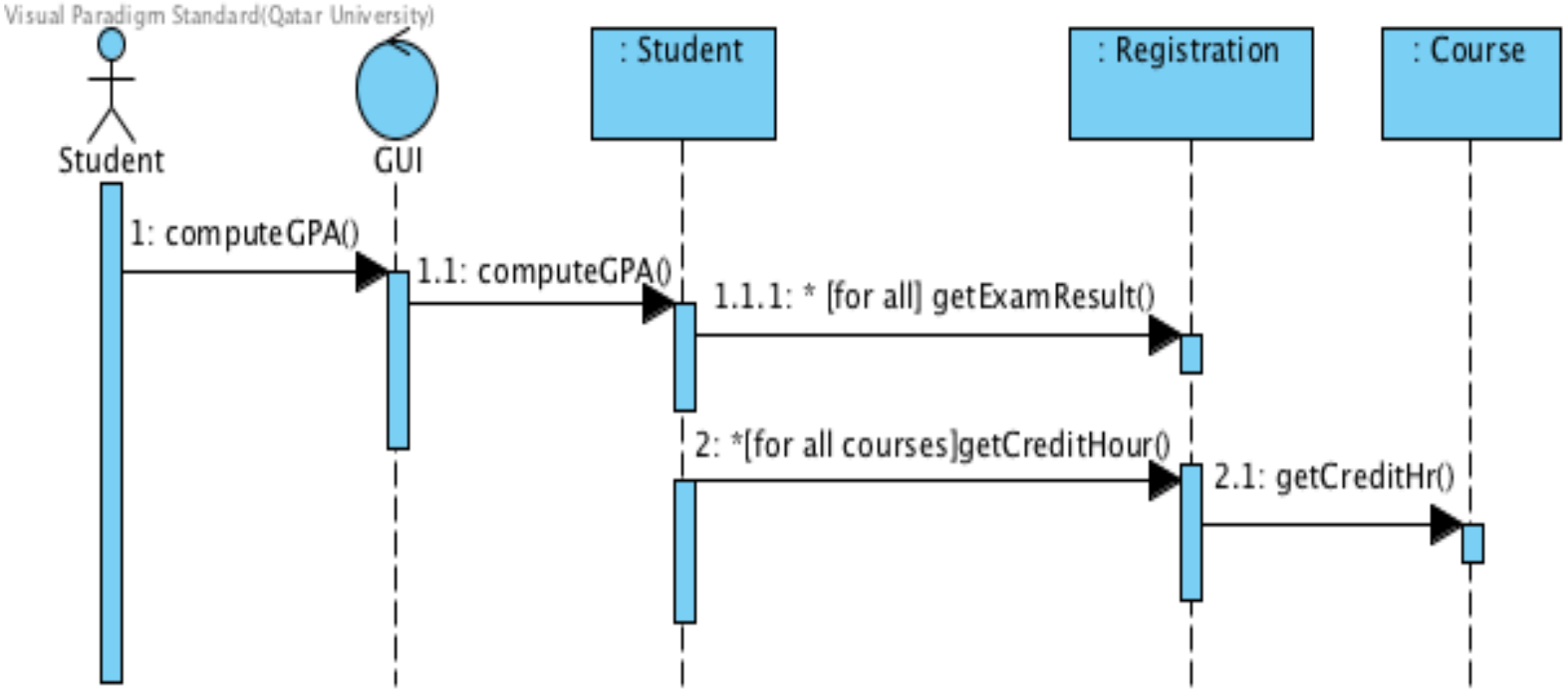
- The **student** object will ask the **registration** object to send the *exam result* of the registered **course** and the *credit hour* of each **course**
- The **registration** object does not have the *credit hour* of each **course** so it sends message to the **course** to get the *credit hour*
- The **registration** object sends the *exam result* and the *credit hour* of each **course** to the **student** object
- The **student** object now compute the *GPA* based on the *exam result* and *credit hour* of each registered **course**



Example 3: Lesson Learned

- Student is an actor and also is a class. Why?
- Use case “Process GPA Calculation” is not a single method, but a collection of methods.
- The use case needs many objects in order to achieve its goal
- Classes were assigned responsibilities based on having the required information in them.
  - Which responsibilities were assigned to which classes?
  - How and why?
- Objects collaborate with each other in a use case to fulfill the goal.
- How to develop and document such collaboration?

Example 3: Design Sequence Diagram



We will discuss Design Sequence Diagram in Lecture - 9

Assigning Responsibilities and Coupling

- Correct responsibilities to classes reduce coupling between classes
- *Coupling* is a measure of how strongly one class is connected to, has knowledge of, or relies upon other classes.
- Assign responsibilities to classes so that *coupling* remains low.
- Use this principle to evaluate alternatives.
- Strong Coupling is bad because classes are affected by changes in related classes
- Classes are harder for the designers/programmers to understand in isolation
- Hard to maintain classes
- Harder to re-use classes because it requires the presence of classes that it depends on
- In order to achieve low coupling, assign responsibiities such a way that a class is not dependent on many classes

Coupling Types

- Common Types of Coupling between two classes: X and Y)
  - Class X has an attribute type of class Y
  - Class X has a method that references object of class Y
    - Object of class Y is a method parameter
    - Class X has a local variable of class Y
    - A Method of Class X returns value an object of class Y
  - Class X is a direct/indirect subclass of class Y
  - Class X implements an interface of class Y.

Benefits of Low Coupling

- It reduces time, effort and defects involved in modifying software
- Classes are more
  - Independent
  - Easier to reuse
  - Easier to understand
  - Easier to maintain
- Coupling to stable class libraries is more acceptable
  - Such as Java libraries (more stable)
- Avoid coupling to frequent changes to your classes
- Manages classes
  - Can Avoid coupling to APIs for proprietary software
  - Can result in vendor lock in

Assigning Responsibilities and Cohesion

- Problem: How to keep objects focused, understandable, and manageable?
- Solution (advice): Assign responsibilities so that **cohesion remains high**. Use this principle to evaluate alternatives.

Cohesion measures the degree to which the tasks performed by a single class are **functionally related**.
=> **Single Responsibility Principle**

A class with high cohesion has a **relatively small number of methods, with highly related functionality, and does not do too much work**.

Levels of Cohesion

- Cohesion:
  - An object that has too many different responsibilities, probably has to collaborate with many other objects
  - How functionally are of a software component such as object, class, method?
  - How much related work is a software component doing?
  - **Low cohesion → High coupling, Both bad.**
- **Very Low cohesion**
  - Class has numerous **unrelated responsibilities**
- **Low cohesion**
  - Responsibilities for a complex task are in one class
- **Moderate (most common) cohesion**
  - Sole responsibilities in a few areas related to the class but not to each other
- **High cohesion**
  - Moderate responsibilities in one area and collaborates with others
- **Cohesive classes should have few methods.**

Benefits of High Cohesion

- **Benefits of high cohesion:**
  - Clarity and ease of comprehension of the design is increased.
  - Likelihood of reuse is increased
  - Maintenance and enhancements are simplified.
  - Low coupling is often supported.
- **Disadvantages of low cohesion:**
  - Increased difficulty in understanding classes.
  - Increased difficulty in maintaining a system, because application changes affect multiple classes, and because changes in one class require changes in related classes.
  - Increased difficulty in reusing a class because most applications won't need the arbitrary set of operations provided by a class.