Agenda

1. Project Part A Discussion
2. Review: Information Hiding Principle
3. UML diagrams: Statechart, Class & Sequence diagram
4. Team formation & Idea Brainstorm

Project Part A

1. Guidelines released on CCLE
2. Sample report released on CCLE
3. Deadline: Thursday, 10/24 11:55pm

Information Hiding Principle

Module

A self contained piece of code that does one job or several related jobs.

Parnas: an independent work assignment that can be assigned to a single engineer

Information Hiding Principle

- A principle for breaking a program into modules
- Design decisions that are likely to change independently should be secrets of separate modules.
- The only assumptions that should appear in the interfaces between modules are those that are considered unlikely to change.

UML Diagrams & Exercises

Statechart diagram

Question: What is its purpose?

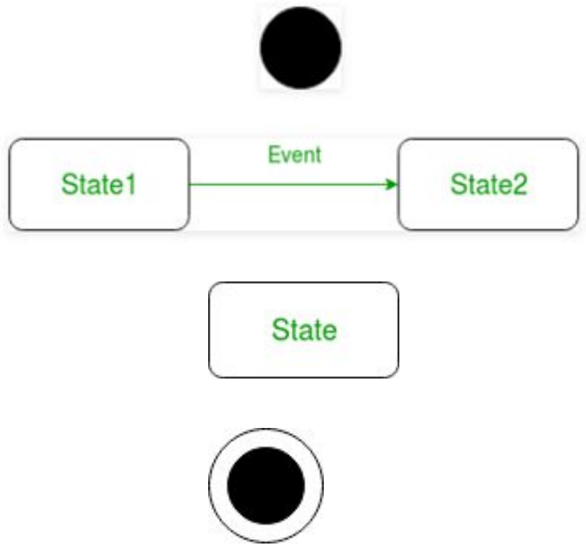
Answer:

1. To model the dynamic aspect of a system
2. To describe different states of an object during its life time
3. To depict the general flow in the system

Statechart diagram: Quick Recap

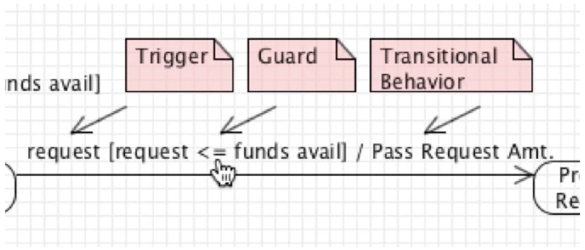
Building blocks:

1. **Initial starting point** – *solid circle*
2. **Transition between states** – *line with an open arrowhead*
3. **State** – *rectangle with rounded corners*
4. **One or more termination points** – *circle with a solid circle inside*



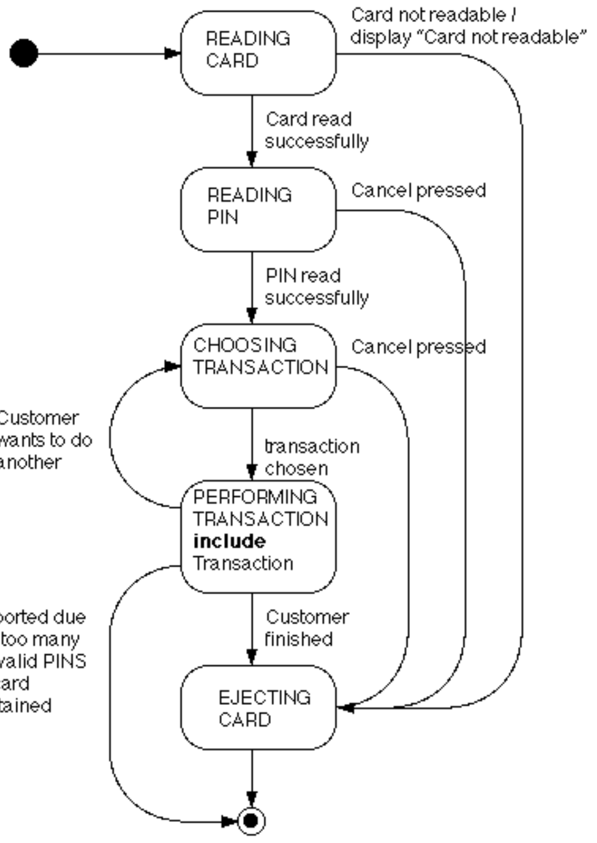
Example for State Chart Diagram

Q: Draw a statechart diagram for an ATM



Example of guard

State-Chart for One Session


READING CARD
Card not readable / display "Card not readable"
Card read successfully
READING PIN
Cancel pressed
PIN read successfully
CHOOSING TRANSACTION
Cancel pressed
Customer wants to do another
transaction chosen
PERFORMING TRANSACTION include Transaction
Aborted due to too many invalid PINS - card retained
Customer finished
EJECTING CARD

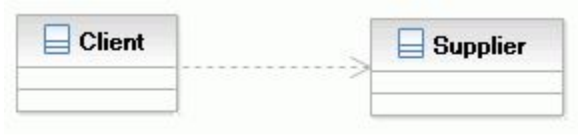
Conditional Branches

Class diagram

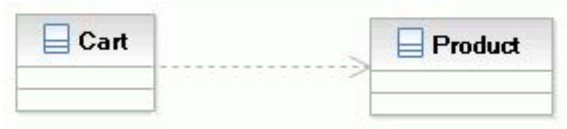
- Describes classes in the system
- Models static relationships
- Classes represent concepts in the system : Represented by Nouns
- Class relationships :
    - Dependency
    - Association
    - Aggregation
    - Composition
    - Generalization
    - Realization

Dependency

A dependency relationship is a relationship in which one element, the client, uses or depends on another element, the supplier.



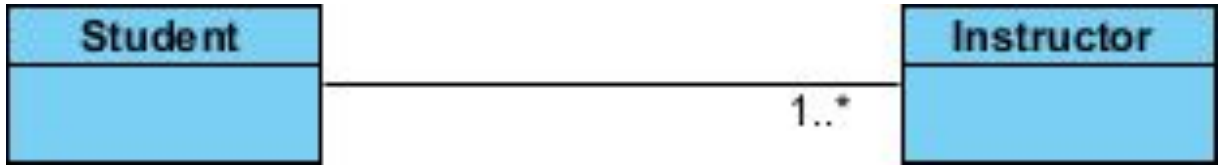
Example: In an e-commerce application, a Cart class depends on a Product class because the Cart class uses the Product class as a parameter for an add operation.



Association

If two classes in a model need to communicate with each other, there must be a link between them, and that can be represented by an association (connector).

We can indicate the multiplicity of an association by adding multiplicity adornments to the line denoting the association. The example indicates that a Student has one or more Instructors:

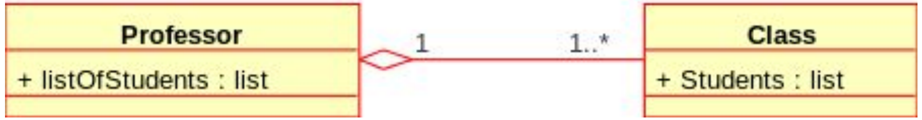


Aggregation

In UML models, an aggregation relationship shows a classifier as a part of or subordinate to another classifier.

An aggregation is a special type of association in which objects are assembled or configured together to create a more complex object

An aggregation association appears as a solid line with an unfilled diamond at the association end, which is connected to the classifier that represents the aggregate.

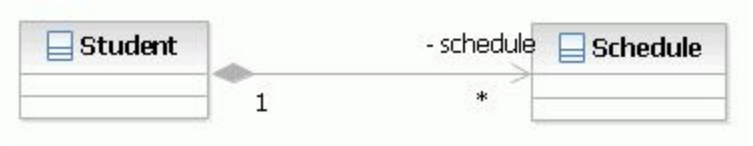


Composition

A composition association relationship represents a whole–part relationship and is a form of aggregation.

A composition association relationship specifies that the lifetime of the part classifier is dependent on the lifetime of the whole classifier.

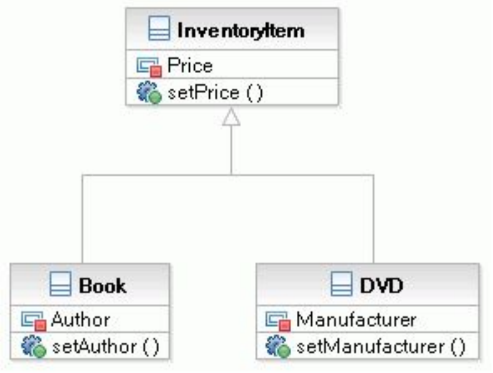
A composition association relationship appears as a solid line with a filled diamond at the association end, which is connected to the whole, or composite, classifier.



Generalization

A generalization relationship is a relationship in which one model element (the child) is based on another model element (the parent).

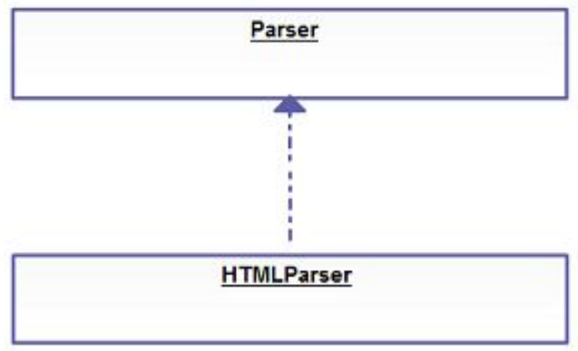
The parent model element can have one or more children, and any child model element can have one or more parents. It is more common to have a single parent model element and multiple child model elements.



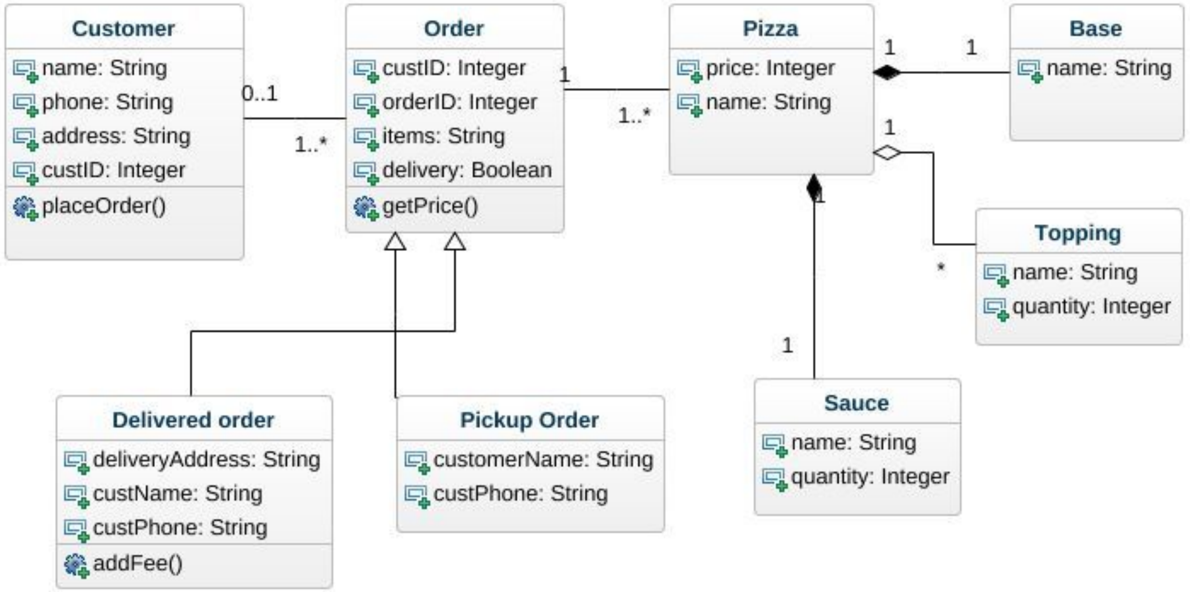
Realization

In a realization relationship, one entity (normally an interface) defines a set of functionalities as a contract and the other entity (normally a class) "realizes" the contract by implementing the functionality defined in the contract.

Realization shows subtyping.



Here’s an Example of a Class Diagram



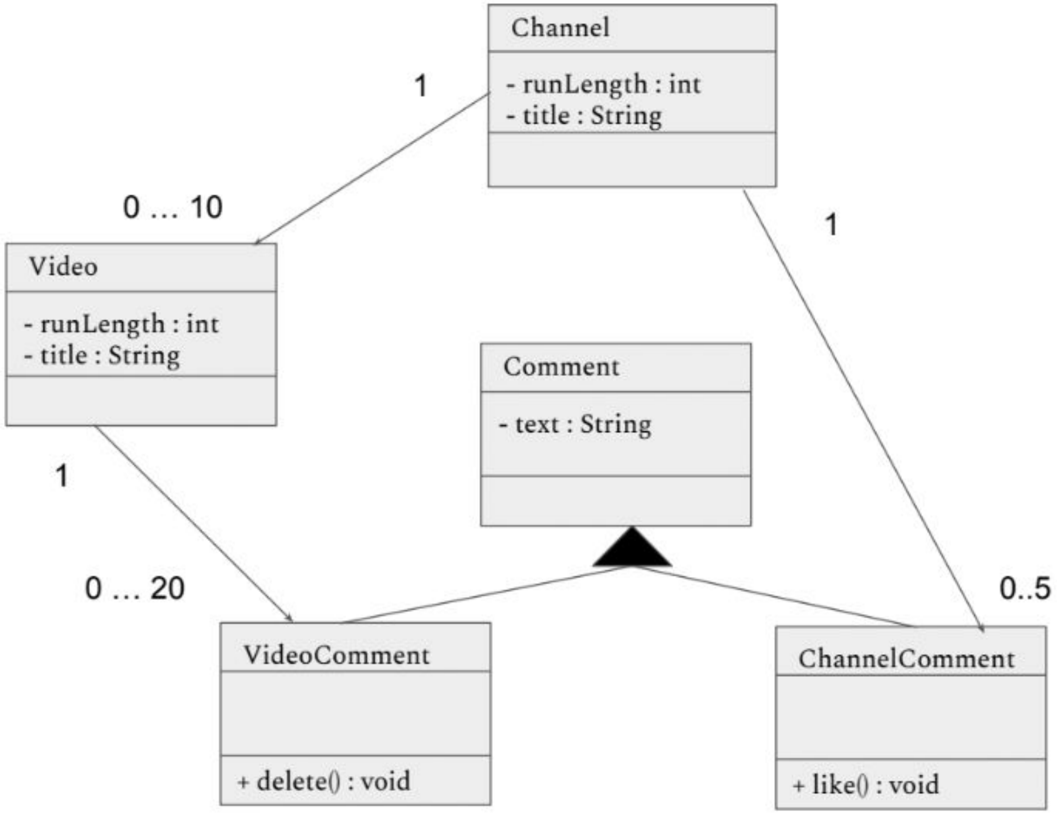
Let’s create a Class Diagram

YouCylinder

In this problem you're going to design a UML class diagram of a YouCylinder video.

- A YouCylinder video has a run length, a title.
- Each video also has a channel that it belongs to.
- Channels can have up to 10 videos, but does not need to have a video. Channels have a name.
- Comments contain a piece of text. YouCylinder video also can have a list of 20 VideoComments, but does not need to have a comment. VideoComments are Comments that have the added functionality that they can be deleted by the channel.
- Channels can also have up to 5 ChannelComments.
- ChannelComments are Comments that can be liked.

Draw a class diagram for this information, and be sure to label all associations with appropriate multiplicities in the diagram.


Channel
- runLength : int
- title : String
1
0 … 10
Video
- runLength : int
- title : String
1
Comment
- text : String
1
0 … 20
0..5
VideoComment
+ delete() : void
ChannelComment
+ like() : void

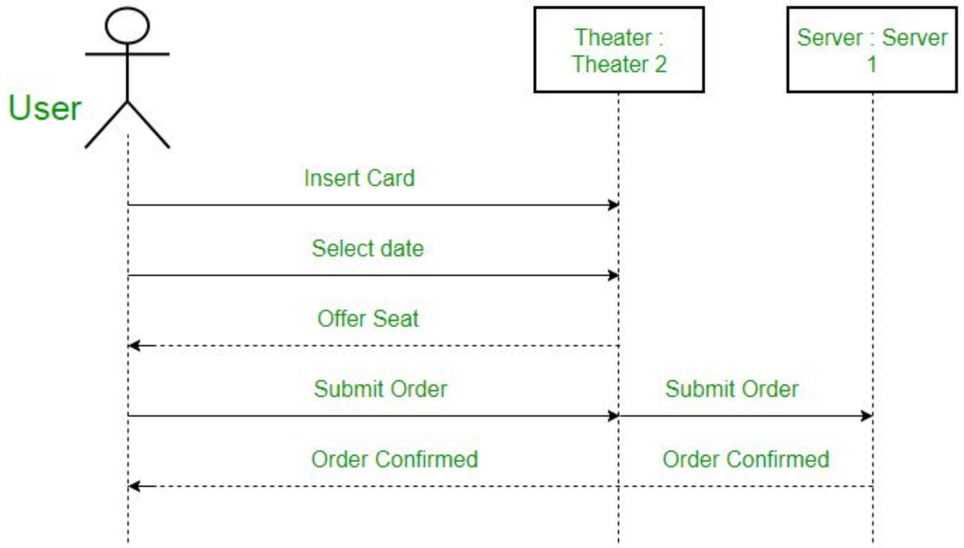
Sequence Diagram

What information do sequence diagrams provide?

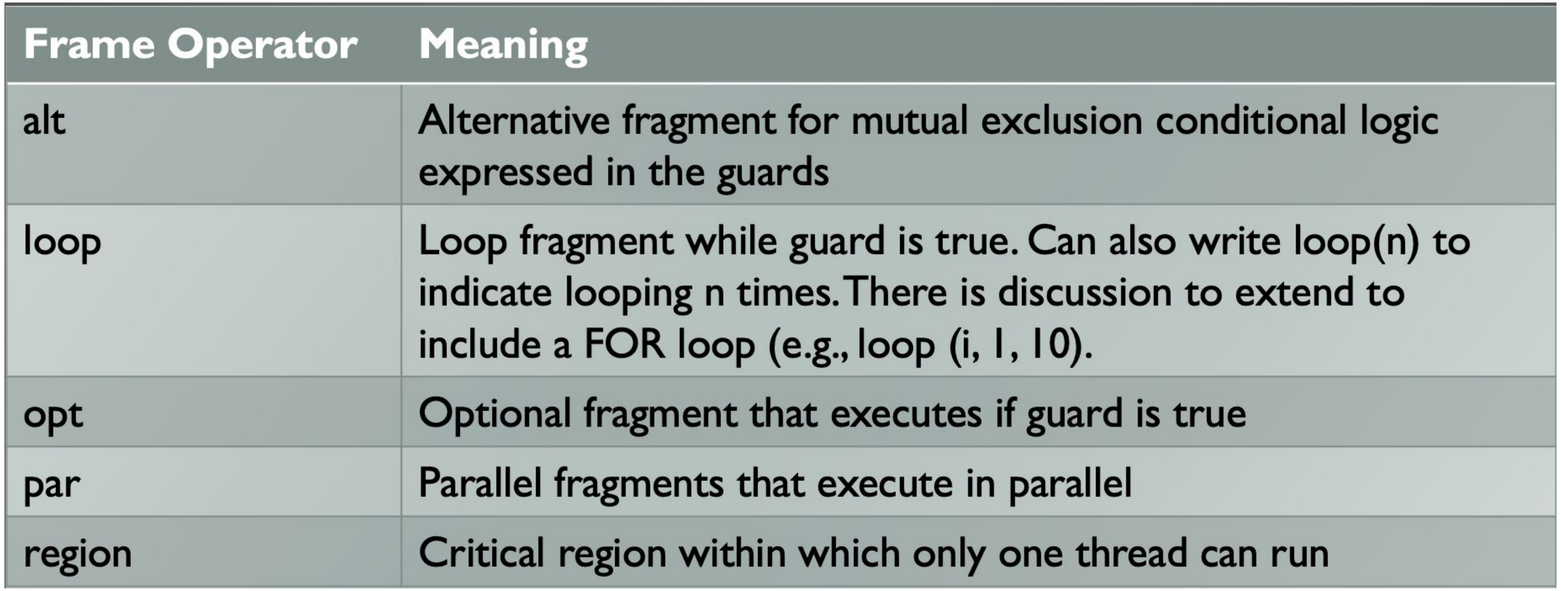
Sequence diagrams show a time-based view of messages between objects

Sequence diagrams describe how and in what order the objects in a system function

We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.

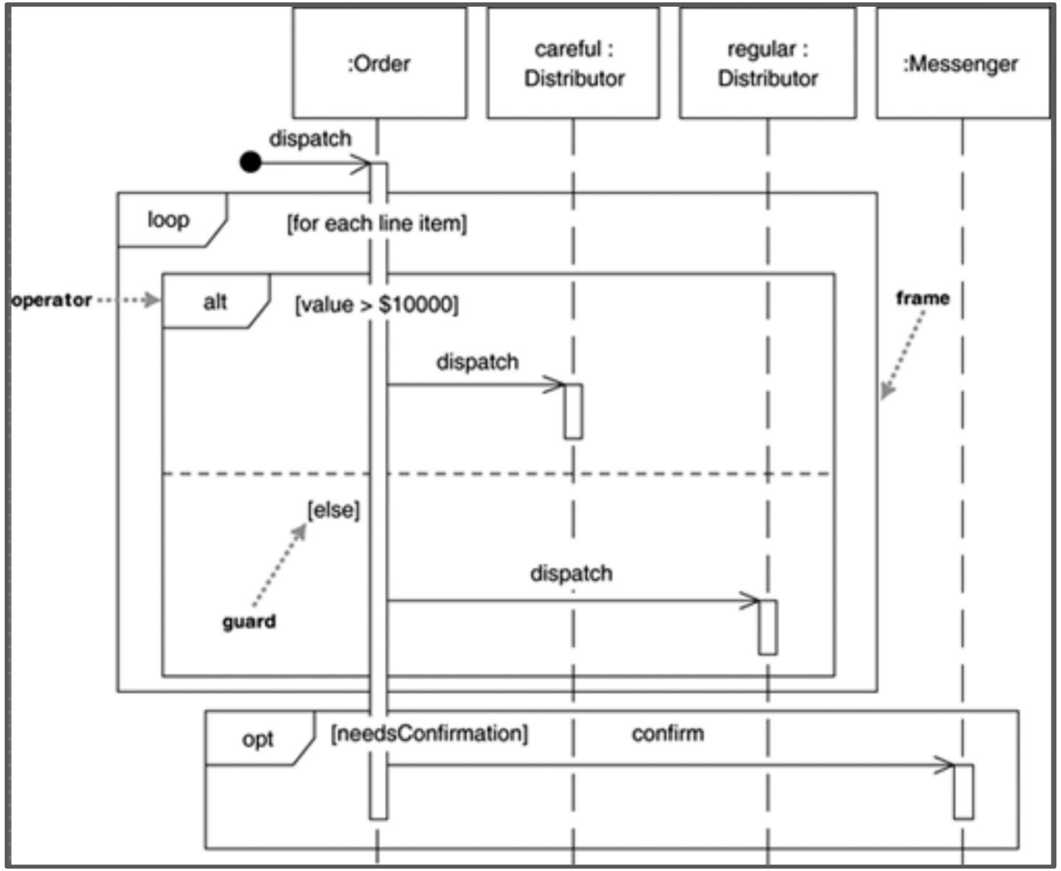


Frame Operators

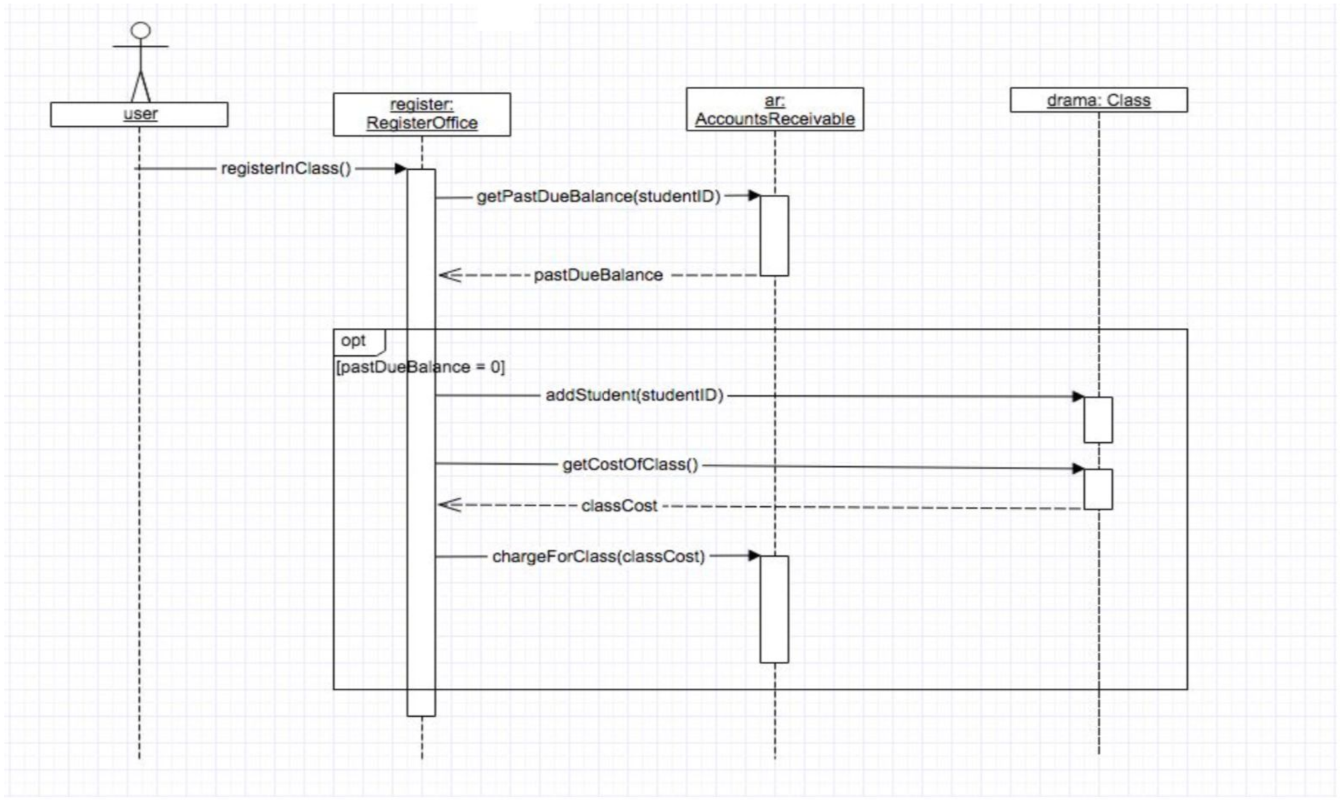


Frame Operator	Meaning
alt	Alternative fragment for mutual exclusion conditional logic expressed in the guards
loop	Loop fragment while guard is true. Can also write loop(n) to indicate looping n times. There is discussion to extend to include a FOR loop (e.g., loop (i, 1, 10).
opt	Optional fragment that executes if guard is true
par	Parallel fragments that execute in parallel
region	Critical region within which only one thread can run

An Example with Frame Operators



Q: Write Java code for the following sequence diagram



```
class RegisterOffice {
public void registerInClass() {
int pastDueBalance = ar.getPastDueBalance(studentID);
if(pastDueBalance == 0) {
    drama.addStudent(studentID);
    int classCost = drama.getCostOfClass();
   ar.chargeForClass(classCost);
}
}
}

class AccountReceivable {
public int getPastDueBalance(int studentID) { … }
public void chargeForClass();
}

class Class {
public void addStudent(int studentID) {...}
public int getCostOfClass() {...}

}

call to trigger this: register.registerInClass()
```

Team formation & Project Idea Discussion