

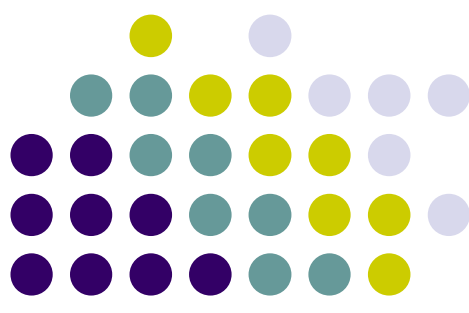
# ICS 52: Introduction to Software Engineering

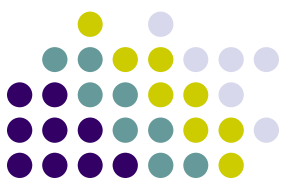
---

Instructor: Prof. Dan Frost

TA: Derek  
[dpfister@uci.edu](mailto:dpfister@uci.edu)

Oct. 31, 2008

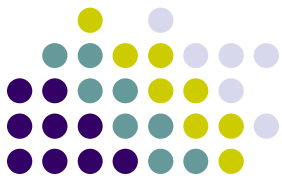




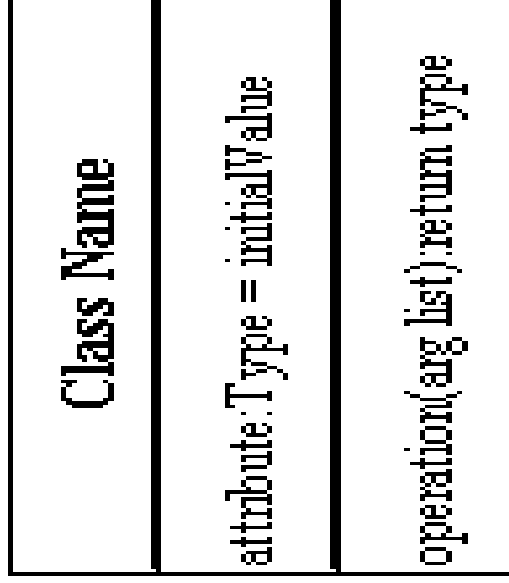
# Review from last discussion

- ***class diagram:***
  - Class (Class name, Attribute, Operation)
  - Relationships
    - Generalization
    - Association
    - Aggregation

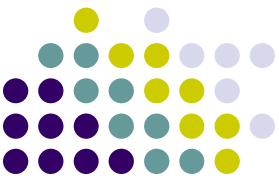
# Basic Class Diagram Symbols and Notations



- **Classes**
  - Illustrate classes with rectangles divided into compartments.
  - Place the name of the class in the first partition (centered, bolded, and capitalized)
  - list the attributes in the second partition
  - write operations into the third.



Note: Please only list the most important attributes and operations in hw

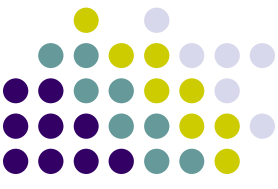


- **Visibility**

- Use visibility markers to signify who can access the information contained within a class.
- Private visibility hides information from anything outside the class partition.
- Public visibility allows all other classes to view the marked information.
- Protected visibility allows child classes to access information they inherited from a parent class.

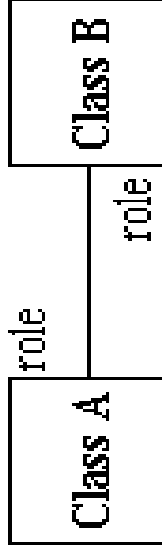
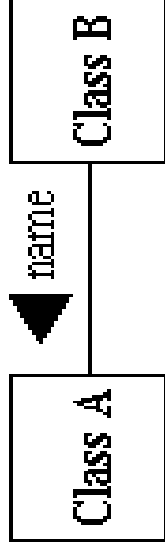
<b>Class Name</b>
- attribute
- attribute
+ operation
+ operation
+ operation

+ <i>public</i>
- <i>private</i>
# <i>protected</i>

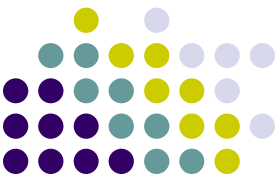


## ● Associations

- Associations represent static relationships between classes.
- Place association names above, on, or below the association line.
- Use a filled arrow to indicate the direction of the relationship.
- Place roles near the end of an association. Roles represent the way the two classes see each other.

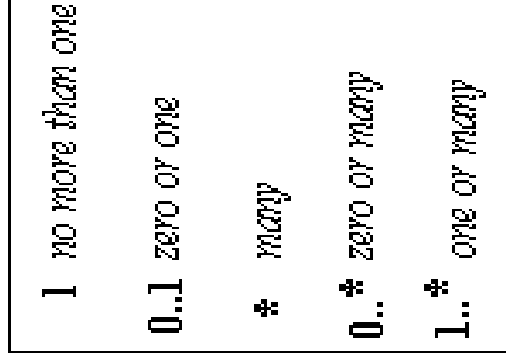
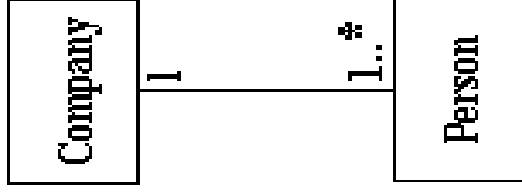


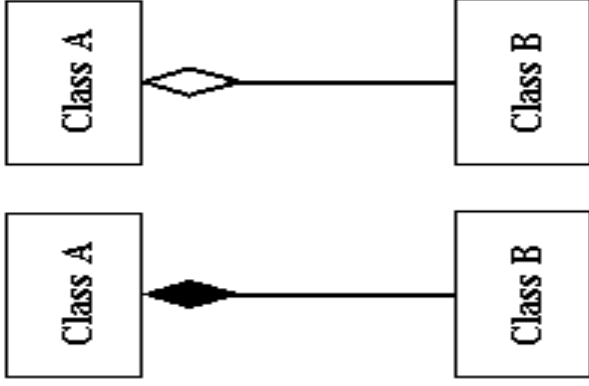
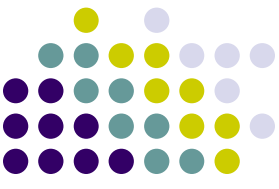
**Note:** It's uncommon to name both the association and the class roles.



## ● Multiplicity (Cardinality)

- Place multiplicity notations near the ends of an association.
- These symbols indicate the number of instances of one class linked to *one* instance of the other class.
- For example, one company will have one or more employees, but each employee works for one company only.



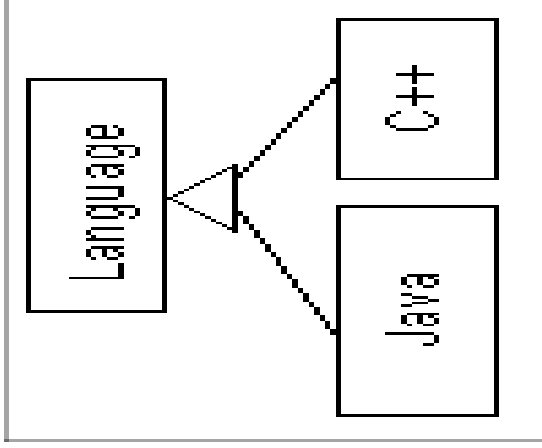


- **Aggregation and Composition**
  - Use a hollow diamond to represent a simple aggregation relationship, in which the "whole" class plays a more important role than the "part" class, but the two classes are not dependent on each other.
  - Composition is a stronger form of aggregation where the whole and parts have coincident lifetimes, and it is very common for the whole to manage the lifecycle of its parts. Illustrate composition with a filled diamond.
  - The diamond end in both a composition and aggregation relationship points toward the "whole" class or the aggregate.



## ● Generalization

- Generalization is another name for inheritance or an "is a" relationship. It refers to a relationship between two classes where one class is a specialized version of another.
- For example, Honda is a type of car. So the class Honda would have a generalization relationship with the class car.
- In real life coding examples, the difference between inheritance and aggregation can be confusing. If you have an aggregation relationship, the aggregate (the whole) can access only the PUBLIC functions of the part class. On the other hand, inheritance allows the inheriting class to access both the PUBLIC and PROTECTED functions of the superclass.

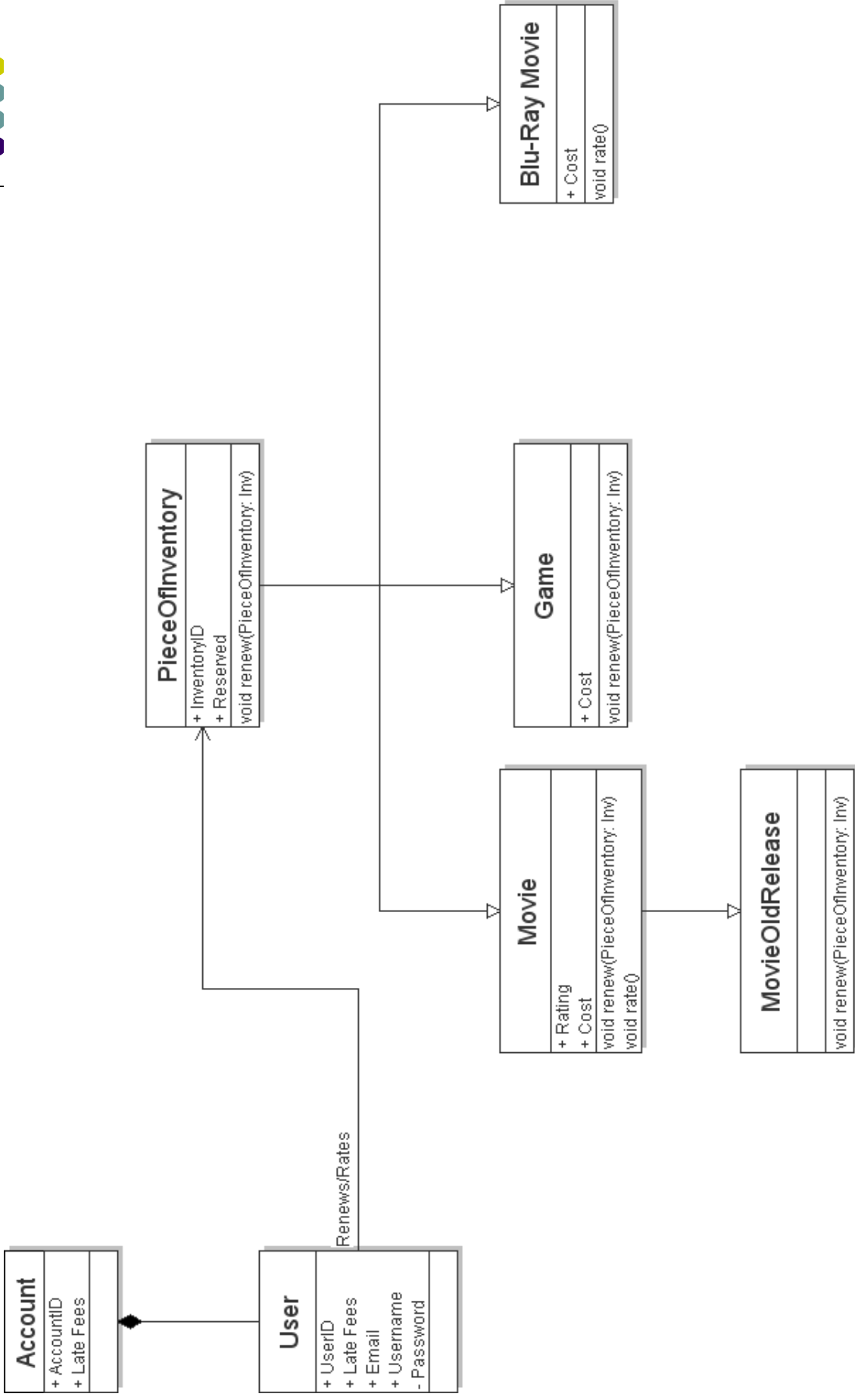


# Example: Online Component of VSS

---



- Review of Requirements:
  - Users login with their username and password
  - Can renew and rate movies
  - Users enter email
  - Can't have late fees, can't renew movie if it is already reserved





# ATM: Requirements

- Has a card reader, a customer console, an envelope acceptor, a cash dispenser, a receipt printer, and an operator panel. The ATM will communicate with the bank's computer over an appropriate communication link.
- A customer will be required to insert an ATM card and enter a personal identification number (PIN) - both of which will be sent to the bank for validation as part of each transaction. The customer will then be able to perform one or more transactions.
- The ATM must be able to provide the following services/transactions to the customer:
  - cash withdrawal with approval obtained from the bank.
  - Deposit with approval.
  - transfer of money between any two accounts linked to the card.
  - balance inquiry of any account linked to the card.
- The ATM will provide the customer with a printed receipt for each successful transaction
- The ATM will have a key-operated switch that will allow an operator to start and stop the servicing of customers.
- The ATM will also maintain an internal log of transactions
  - Entries will be made in the log
    - ATM is started up and shut down
    - for each message sent to the Bank (along with the response back, if one is expected),
    - for the dispensing of cash,
    - for the receiving of an envelope.



# ATM: Class Diagram

