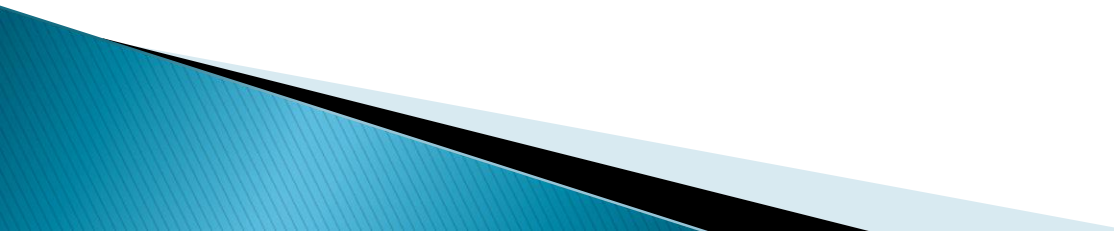


Class Diagrams

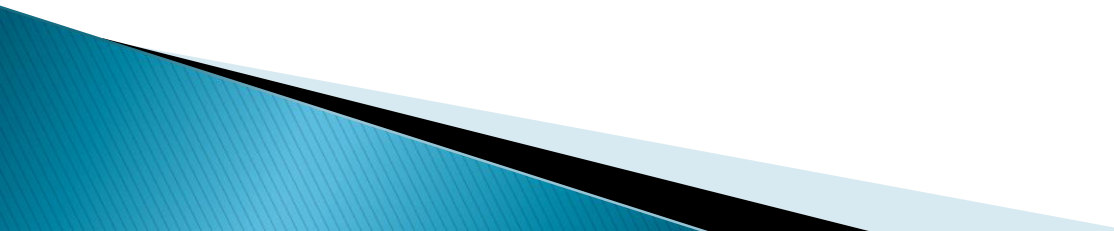
Purpose

- ▶ The purpose of the class diagram is to show the types being modeled within the system. In most UML models these types include:
 - a class
 - an interface
 - a data type
 - a component

Usage

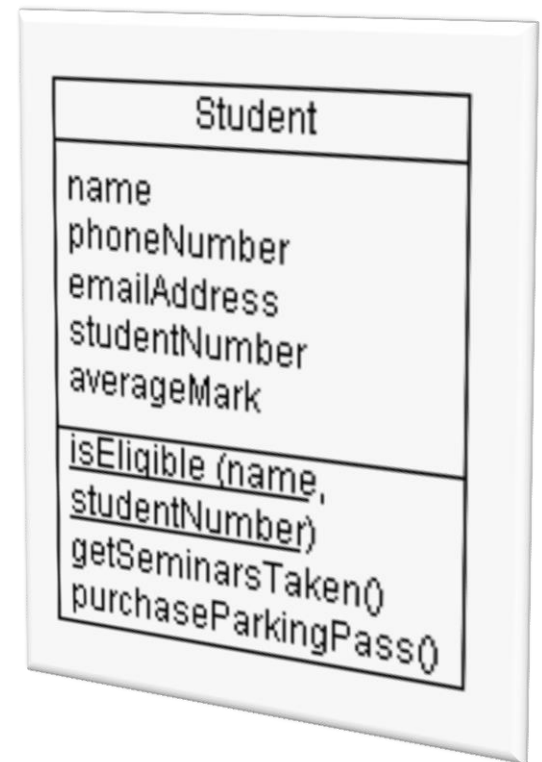
- ▶ Class diagrams are used to:
 - Analyze program objects in a graphical model
 - Analyze requirements in the form of a standardized model
 - Show a detailed design of object-oriented software
- 

```
class Student
{
    string name;
    string phoneNumber;
    string emailAddress;
    int studentNumber;
    float averageMark;
    bool isEligible(string name, int studentnumber);
    int getSemistarsTaken();
    void purchaseParkingPass();
};
```



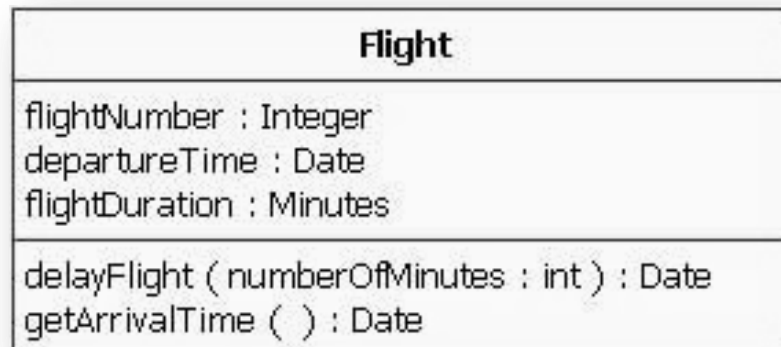
Class Representation

- ▶ Classes are shown as boxes with three sections:
- ▶ top for the name of the class
- ▶ middle for the attributes
- ▶ bottom for the operations



Attribute Detail

- ▶ The attribute types are units that make sense to the likely readers of the diagram (i.e., minutes, dollars, etc.). A class diagram also includes attribute types provided by the programming language.



Visibility Symbols

Mark	Visibility type
+	Public
#	Protected
-	Private
~	Package

Access Specifiers

- ▶ To display visibility on the class diagram, place the visibility mark in front of the attribute's or operation's name.



Functions

- ▶ When an operation has parameters, they are put inside the operation's parentheses; each parameter uses the format "parameter name : parameter type".

Flight
flightNumber : Integer departureTime : Date flightDuration : Minutes
delayFlight (in numberOfMinutes : Minutes) getArrivalTime () : Date

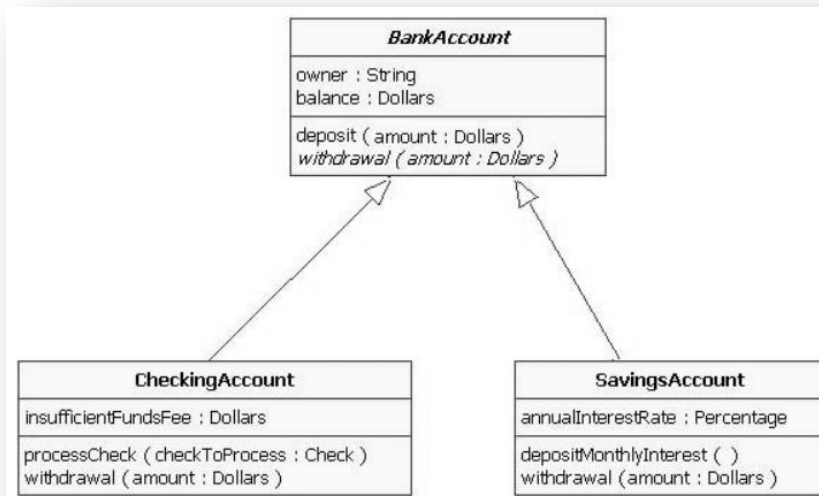
Parameter Indicators

- ▶ When documenting an operation's parameters, an optional indicator can be used to show whether or not the parameter is input to, or output from, the operation. This optional indicator appears as an "in" or "out".

Flight	
flightNumber : Integer	
departureTime : Date	
flightDuration : Minutes	
delayFlight (in numberOfMinutes : Minutes)	
getArrivalTime () : Date	

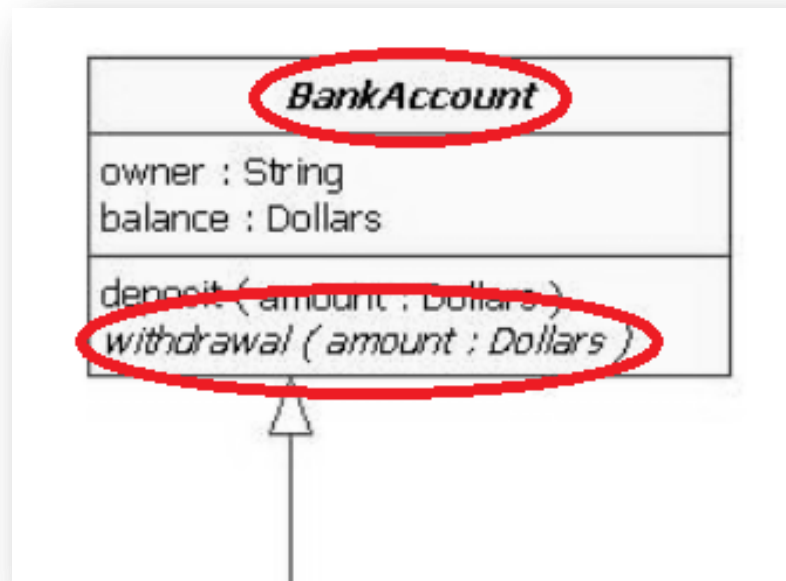
Inheritance

- ▶ To model inheritance on a class diagram, a solid line is drawn from the child class with a closed, unfilled arrowhead (or triangle) pointing to the super class.



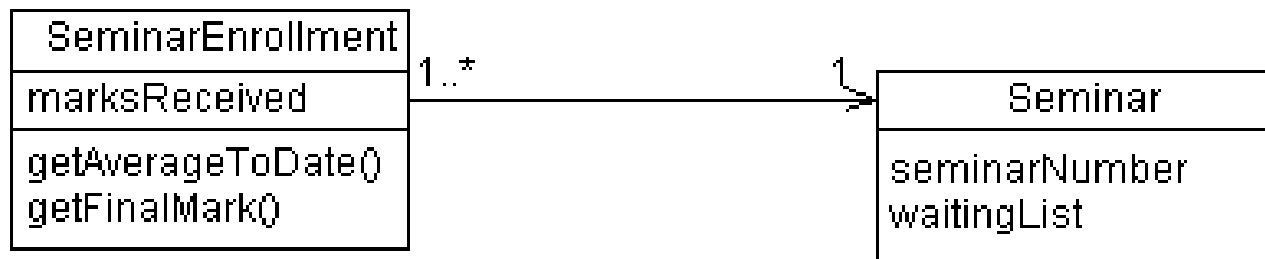
Abstract Classes

- ▶ Use italicized text for a class that is abstract class and the withdrawal method is an abstract operation.



Associations

- Associations between classes are depicted as lines between classes. Associations can include multiplicity indicators at each end.

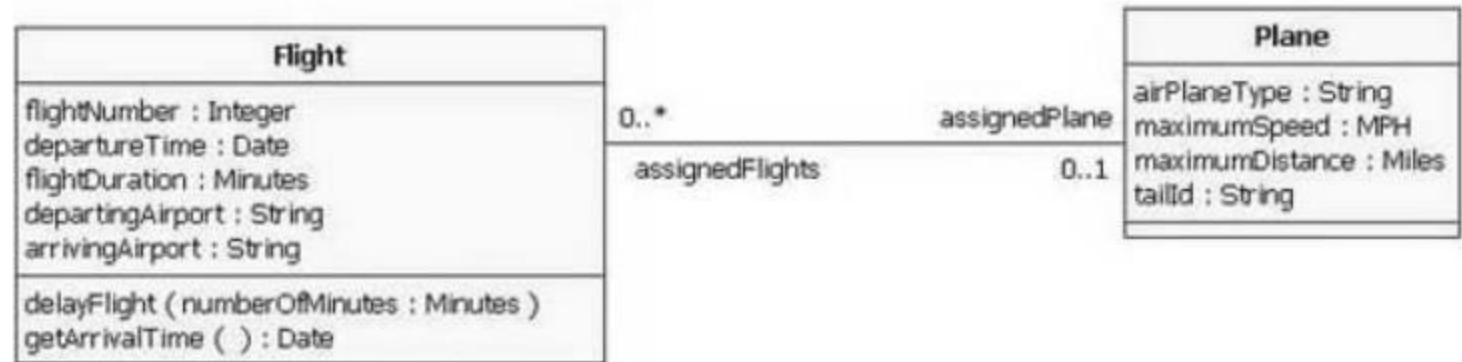


Multiplicity

Potential Multiplicity Values	
Indicator	Meaning
0..1	Zero or one
1	One only
0..*	Zero or more
*	Zero or more
1..*	One or more
3	Three only
0..5	Zero to Five
5..15	Five to Fifteen

Bi-Directional Association

- ▶ An association is a linkage between two classes. Associations are always assumed to be bi-directional; this means that both classes are aware of each other and their relationship.



UniDirectional Association

- ▶ A unidirectional association is drawn as a solid line with an open arrowhead pointing to the known class. Like standard associations, the uni-directional association includes a role name and a multiplicity value.

