Object Oriented Programming Concepts

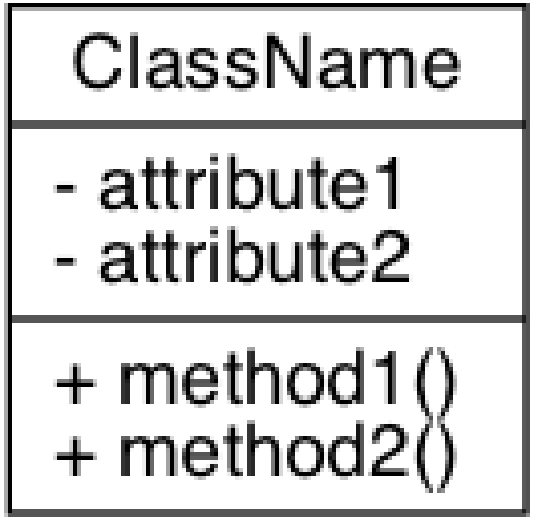
CSC 243 - Java Programming

Object Oriented Programming

- The main task of a programmer is to solve problems
- Object Oriented Programming (OOP) is *one* way to model and solve problems
- OOP models problems using objects (nouns) containing data and methods (verbs) that perform operations on the data

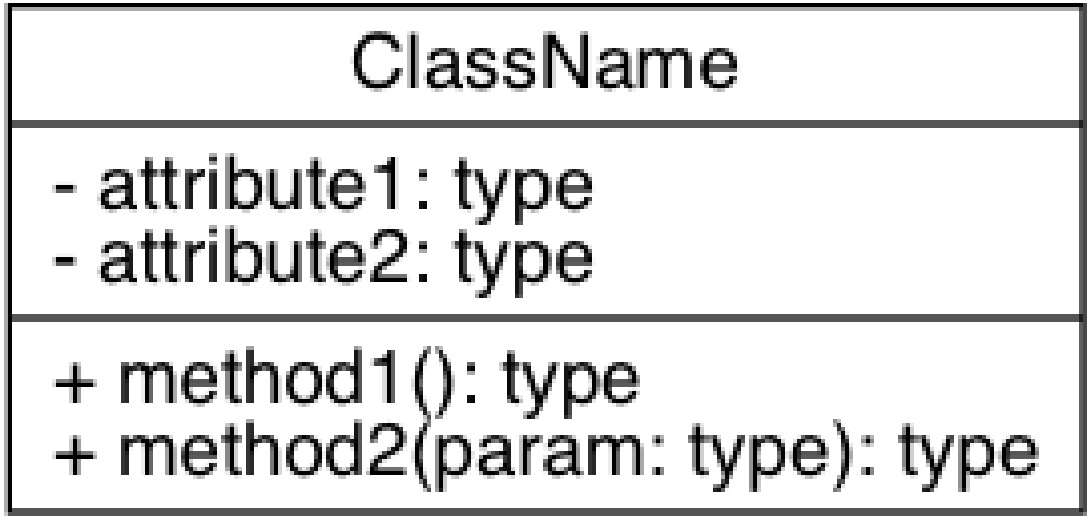
Introduction to UML Class Diagrams

- A Unified Modeling Language (UML) Class diagram describes implementation information about a class
- The first compartment contains the class name
- The center (optional) compartment contains data attributes
- The lower (optional) compartment contains methods
- The $(-)$ denotes a private modifier and $(+)$ denotes a public modifier



Introduction to UML Class Diagrams

- Type information is described using colons:



Class Abstraction and Encapsulation

- Class abstraction separates class implementation from class usage
- A class's contract is the collection of public methods and attributes and a description of the expected behavior of the public members
- Class encapsulation is hiding the implementation details of public methods from the client code that uses these methods

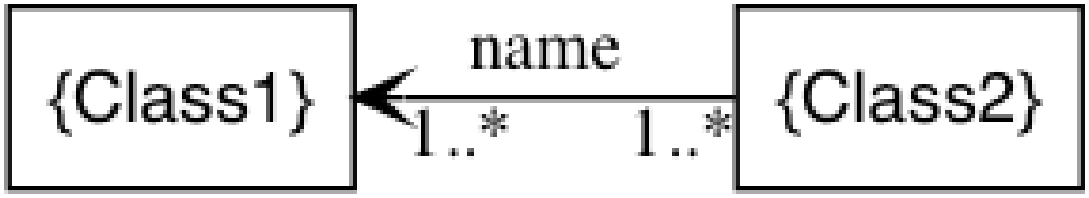
Advantages of Encapsulation

- Client code does not need to know the implementation details of a class in order to use it
- The implementation of the class methods can change without breaking existing client code
- Compiled class code can be reused in many applications

Class Relationships

- Association: describes an activity between two classes
- Aggregation: an association relationship that represents ownership
  - The owner object is called an *aggregating object*
  - The owner class is called an *aggregating class*
  - The owned object is called an *aggregated object*
  - The owned class is called an *aggregated class*
- Composition: A special case of the aggregation relationship where the aggregating object exclusively owns the aggregated object
- Inheritance

UML Association Relationship

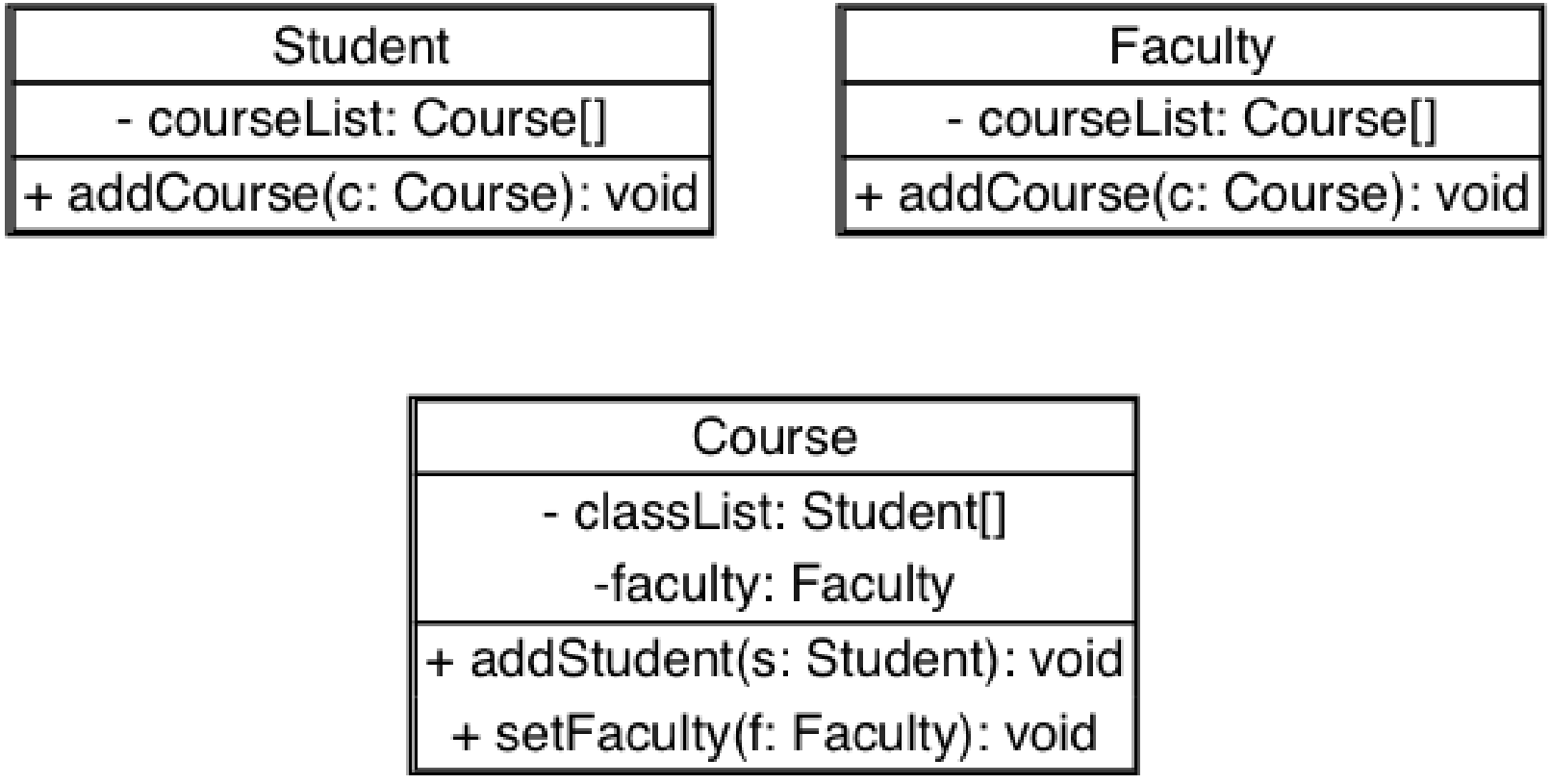


- The label of the edge denotes the name of the relationship
- The annotations on the head and tail of an edge denote multiplicity
- Multiplicity specifies how many `objects` are involved in the relationship
- A multiplicity of * denotes an unlimited number of objects

UML Association Relationship Example

- A student may take any number of courses
- Each course must have at least 10 students and at most 60 students
- Each course is taught by one faculty member
- A faculty member may teach between 0 and 4 courses per semester

Modeling the Example Relationship



UML Aggregation and Composition Example

- Aggregation is denoted by a open diamond attached to the aggregating class
- Composition is denoted by a filled diamond attached to the aggregating class