

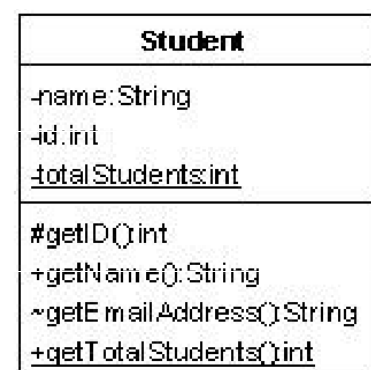
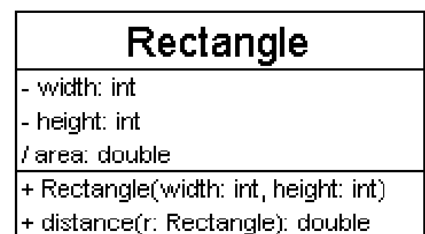
UML class diagrams

- What is a UML class diagram?
 - **UML class diagram**: a picture of
 - the classes in an OO system
 - their fields and methods
 - connections between the classes
 - that interact or inherit from each other
- What are some things that are not represented in a UML class diagram?
 - details of how the classes interact with each other
 - algorithmic details; how a particular behavior is implemented

9

Diagram of one class

- class name in top of box
 - write <<interface>> on top of interfaces' names
 - use *italics* for an *abstract class* name
- attributes (optional)
 - should include all fields of the object
- operations / methods (optional)
 - may omit trivial (get/set) methods
 - but don't omit any methods from an interface!
 - should not include inherited methods



10

Class attributes

- attributes (fields, instance variables)
 - *visibility name : type [count] = default_value*
 - visibility:
 - + public
 - # protected
 - private
 - ~ package (default)
 - / derived
 - underline static attributes
 - **derived attribute**: not stored, but can be computed from other attribute values
 - attribute example:
 - balance : double = 0.00

Rectangle
- width: int
- height: int
/ area: double
+ Rectangle(width: int, height: int)
+ distance(r: Rectangle): double

Student
-name:String
-id:int
<u>-totalStudents:int</u>
#getID():int
+getName():String
~getEmailAddress():String
<u>+getTotalStudents():int</u>

11

Class operations / methods

- operations / methods
 - *visibility name (parameters) : return_type*
 - visibility:
 - + public
 - # protected
 - private
 - ~ package (default)
 - underline static methods
 - parameter types listed as (name: type)
 - omit *return_type* on constructors and when return type is void
 - method example:
 - + distance(p1: Point, p2: Point): double

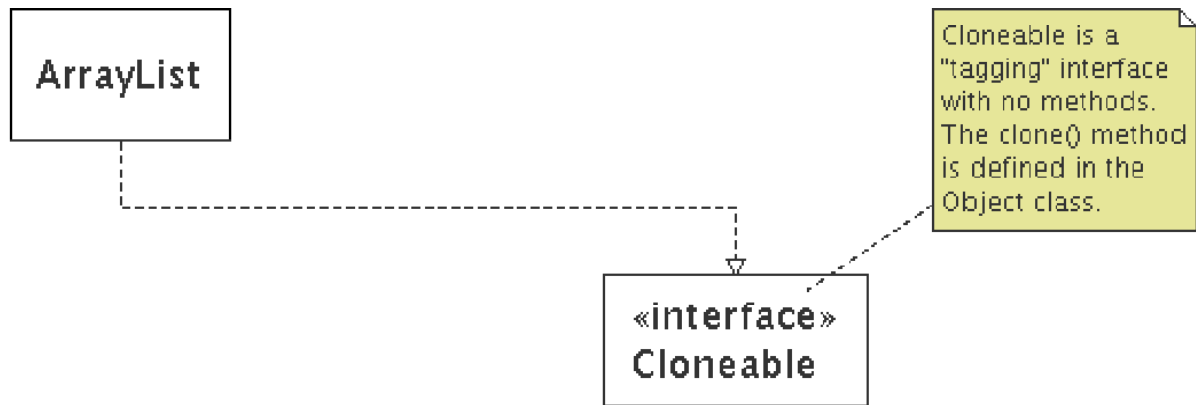
Rectangle
- width: int
- height: int
/ area: double
+ Rectangle(width: int, height: int)
+ distance(r: Rectangle): double

Student
-name:String
-id:int
<u>-totalStudents:int</u>
#getID():int
+getName():String
~getEmailAddress():String
<u>+getTotalStudents():int</u>

12

Comments

- represented as a folded note, attached to the appropriate class/method/etc by a dashed line



13

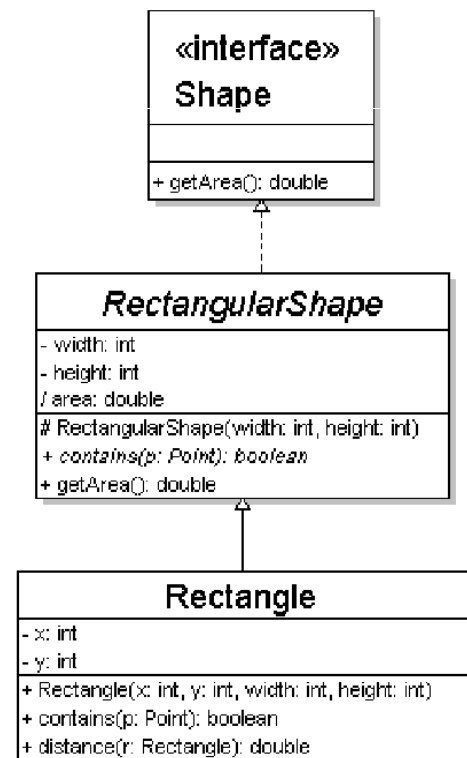
Relationships btwn. classes

- **generalization**: an inheritance relationship
 - inheritance between classes
 - interface implementation
- **association**: a usage relationship
 - dependency
 - aggregation
 - composition

14

Generalization relationships

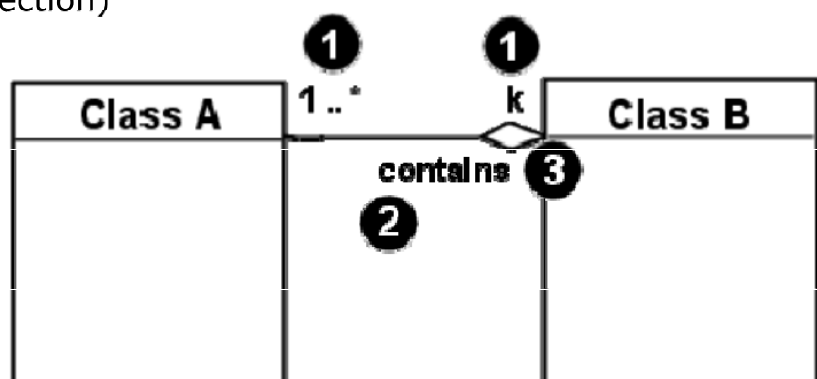
- generalization (inheritance) relationships
 - hierarchies drawn top-down with arrows pointing upward to parent
 - line/arrow styles differ, based on whether parent is a(n):
 - class: solid line, black arrow
 - abstract class: solid line, white arrow
 - interface: dashed line, white arrow
 - we often don't draw trivial / obvious generalization relationships, such as drawing the Object class as a parent



15

Associational relationships

- associational (usage) relationships
 1. multiplicity (how many are used)
 - * \Rightarrow 0, 1, or more
 - 1 \Rightarrow 1 exactly
 - 2..4 \Rightarrow between 2 and 4, inclusive
 - 3..* \Rightarrow 3 or more
 2. name (what relationship the objects have)
 3. navigability (direction)

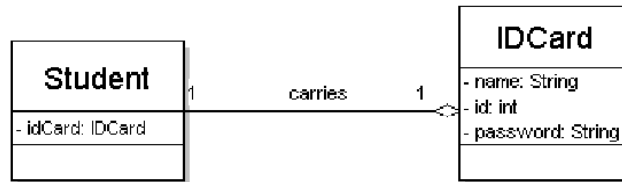


16

Multiplicity of associations

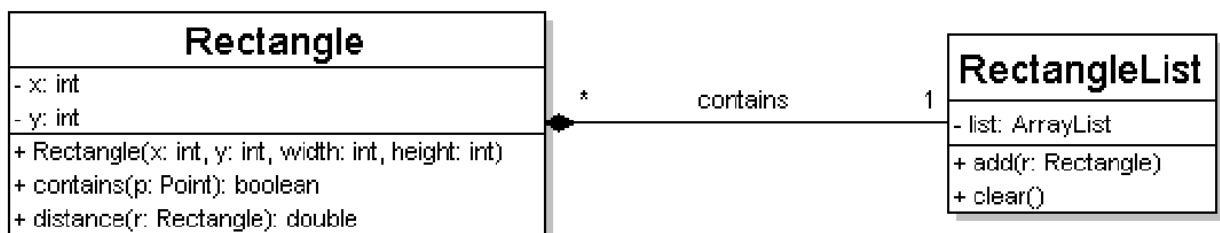
- one-to-one

- each student must carry exactly one ID card



- one-to-many

- one rectangle list can contain many rectangles

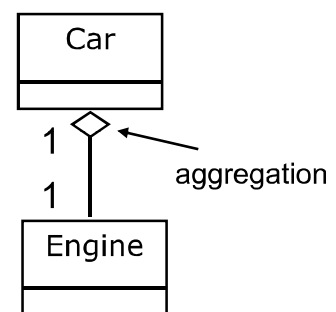


17

Association types

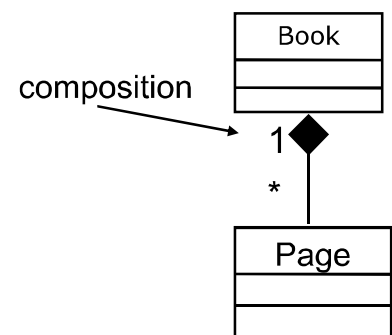
- **aggregation:** "is part of"

- symbolized by a clear white diamond



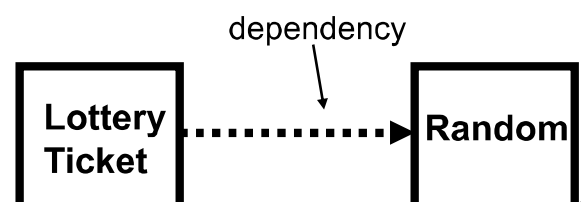
- **composition:** "is entirely made of"

- stronger version of aggregation
 - the parts live and die with the whole
 - symbolized by a black diamond



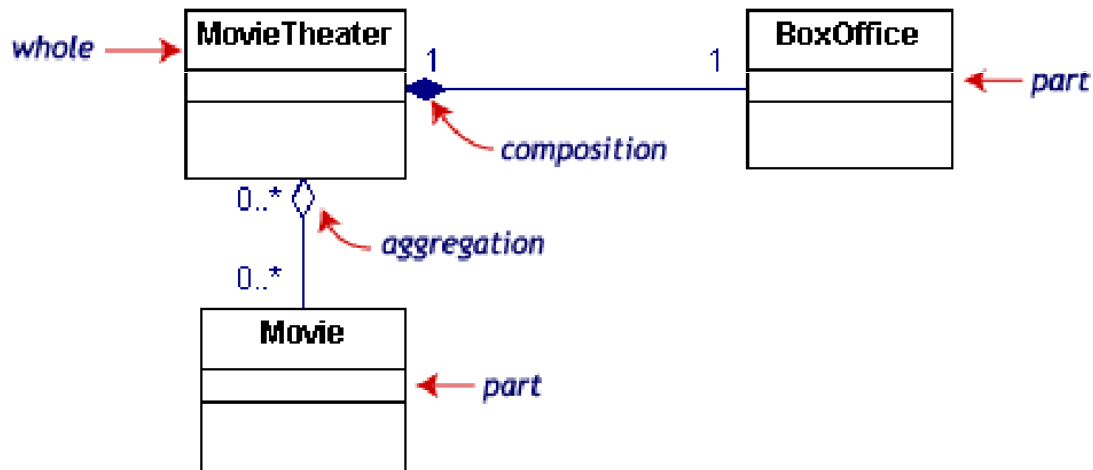
- **dependency:** "uses temporarily"

- symbolized by dotted line
 - often is an implementation detail, not an intrinsic part of that object's state



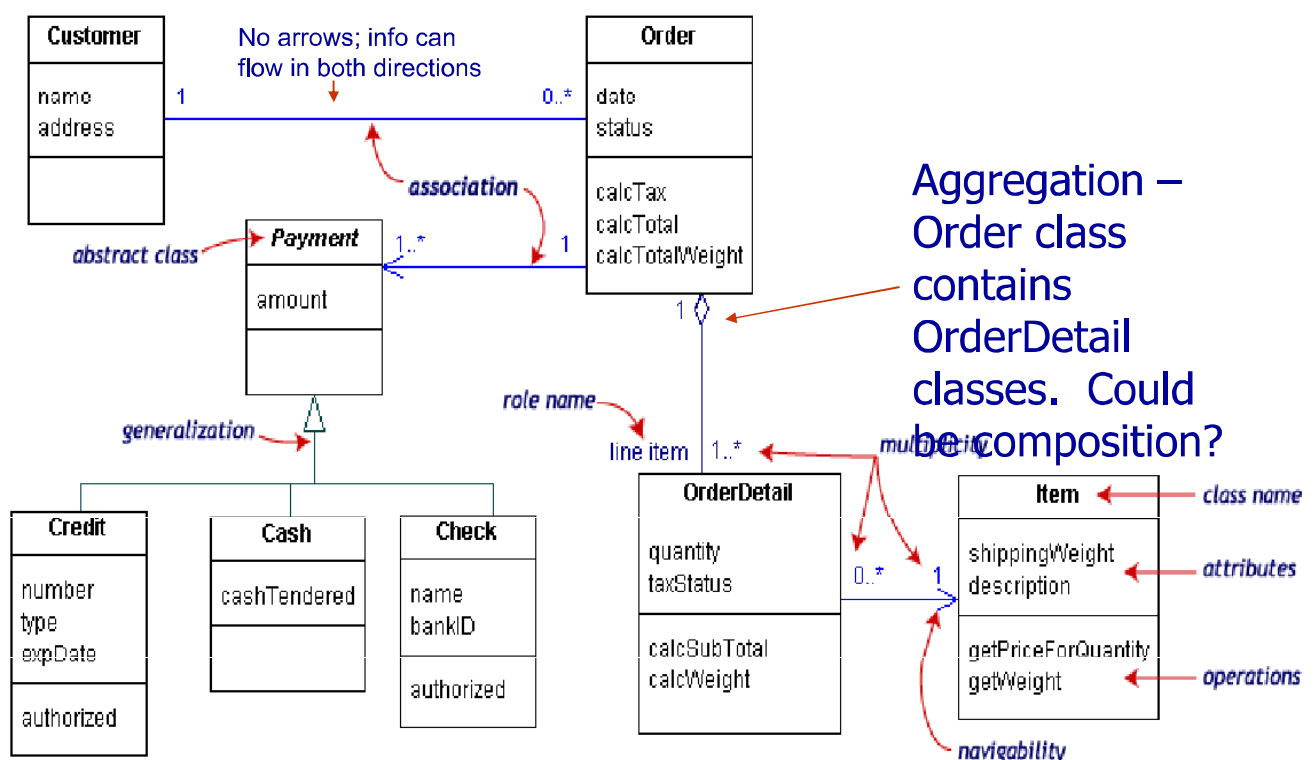
18

Composition/aggregation example

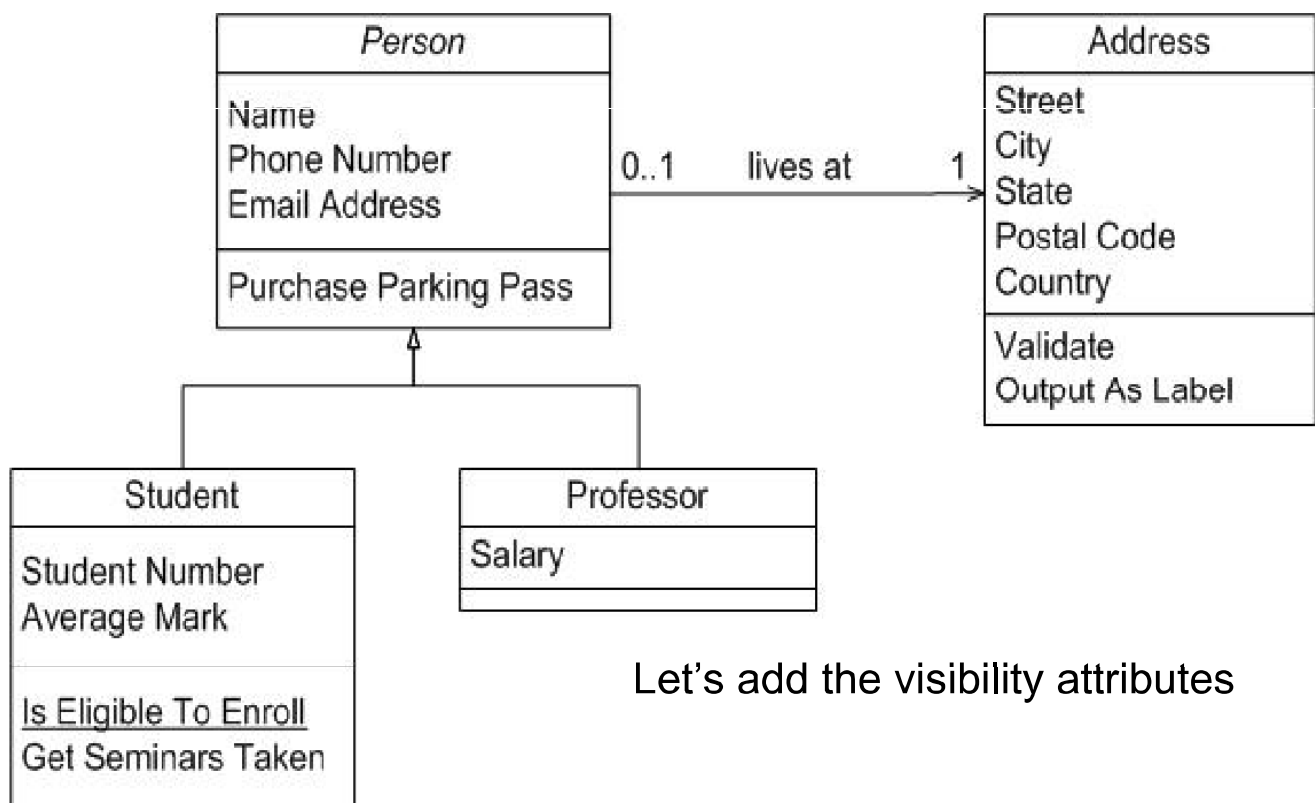


If the movie theatre goes away
so does the box office => composition
but movies may still exist => aggregation

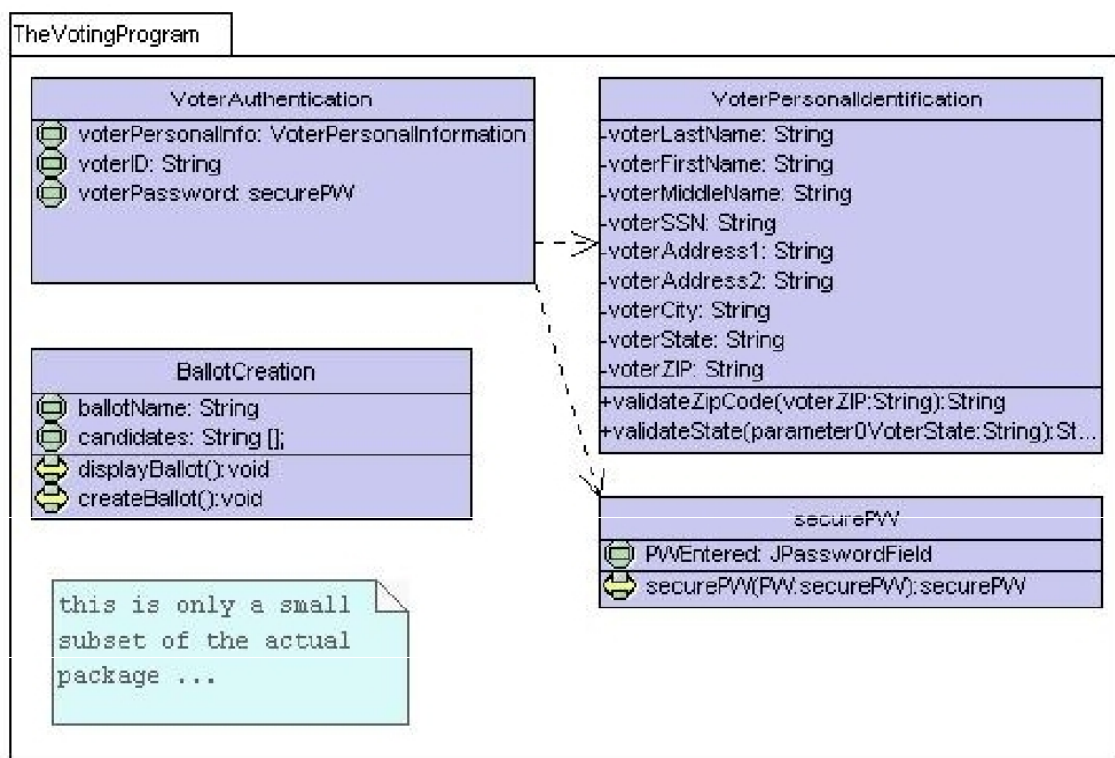
Class diagram example



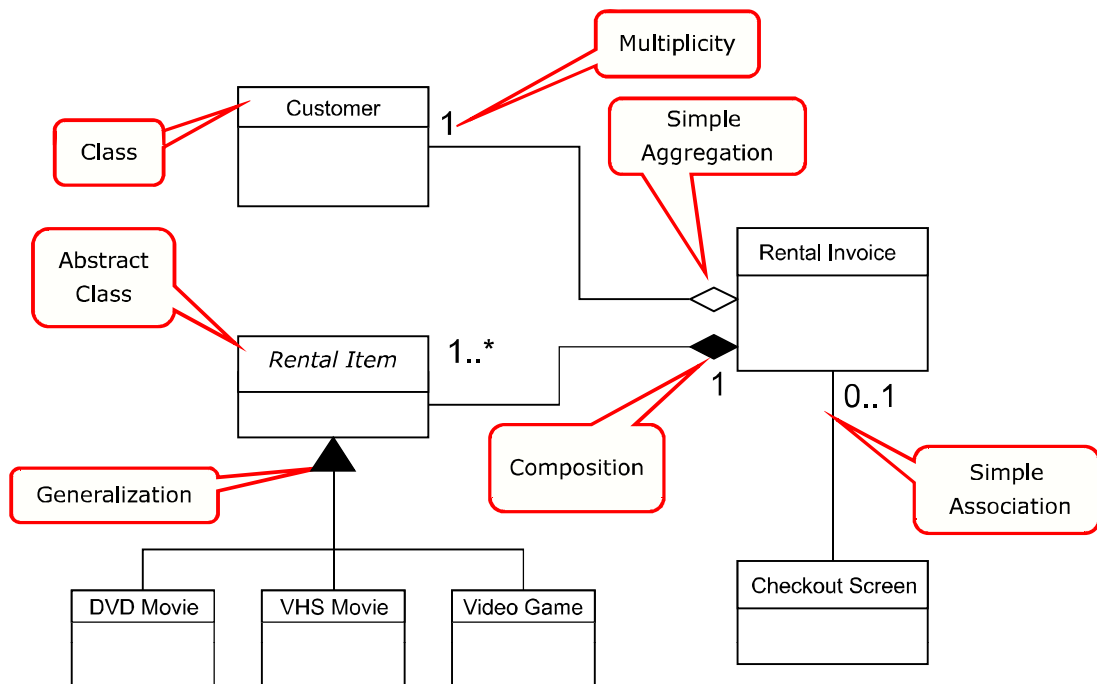
UML example: people



Class diagram: voters

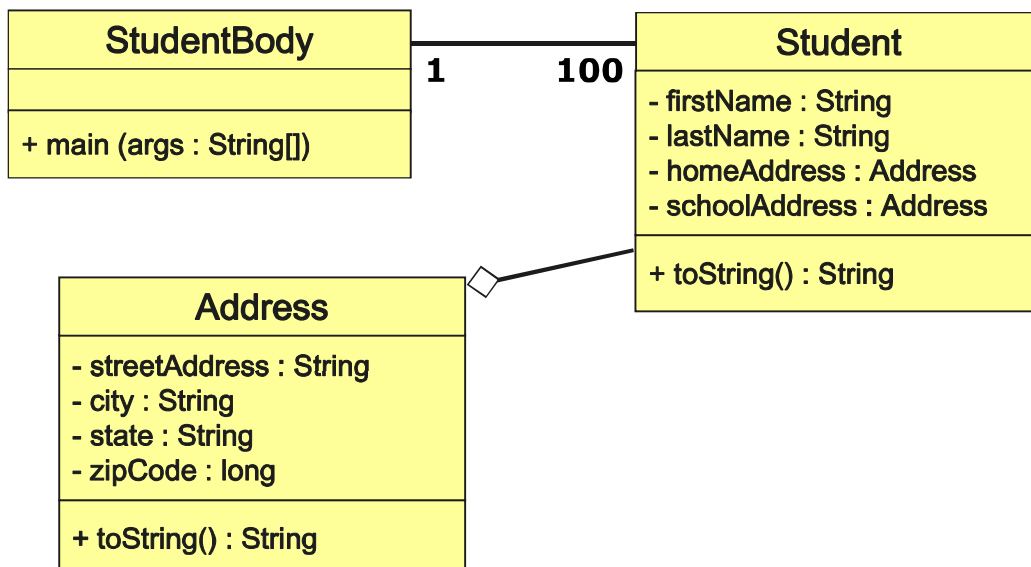


Class diagram example: video store



23

Class diagram example: student



24