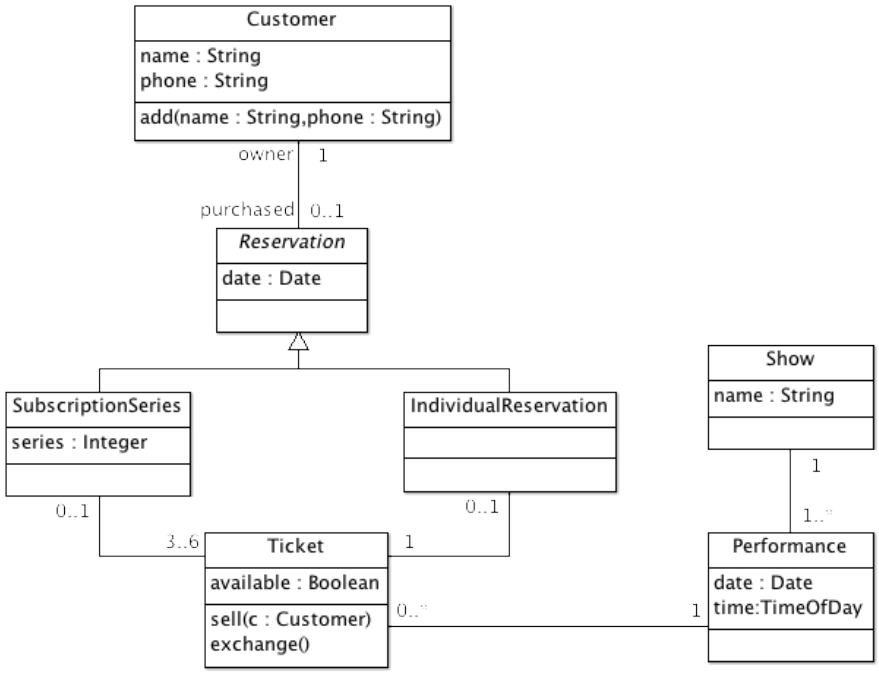
OO Design Practice Questions

Consider this class diagram, which models a theater ticketing system.



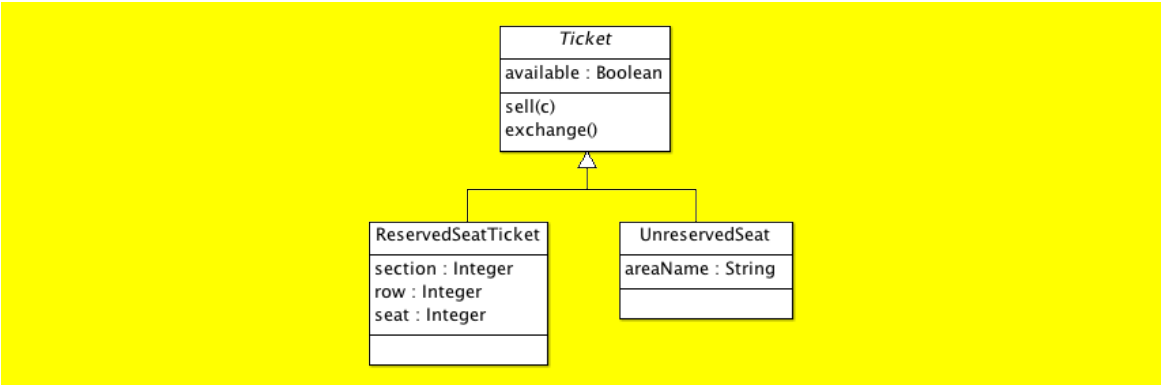
Question 1. How many owners may a reservation have? **1**

Question 2. How many tickets may a subscription series have? **3–6**

Question 3. Update the diagram so that there are two kinds of ticket:

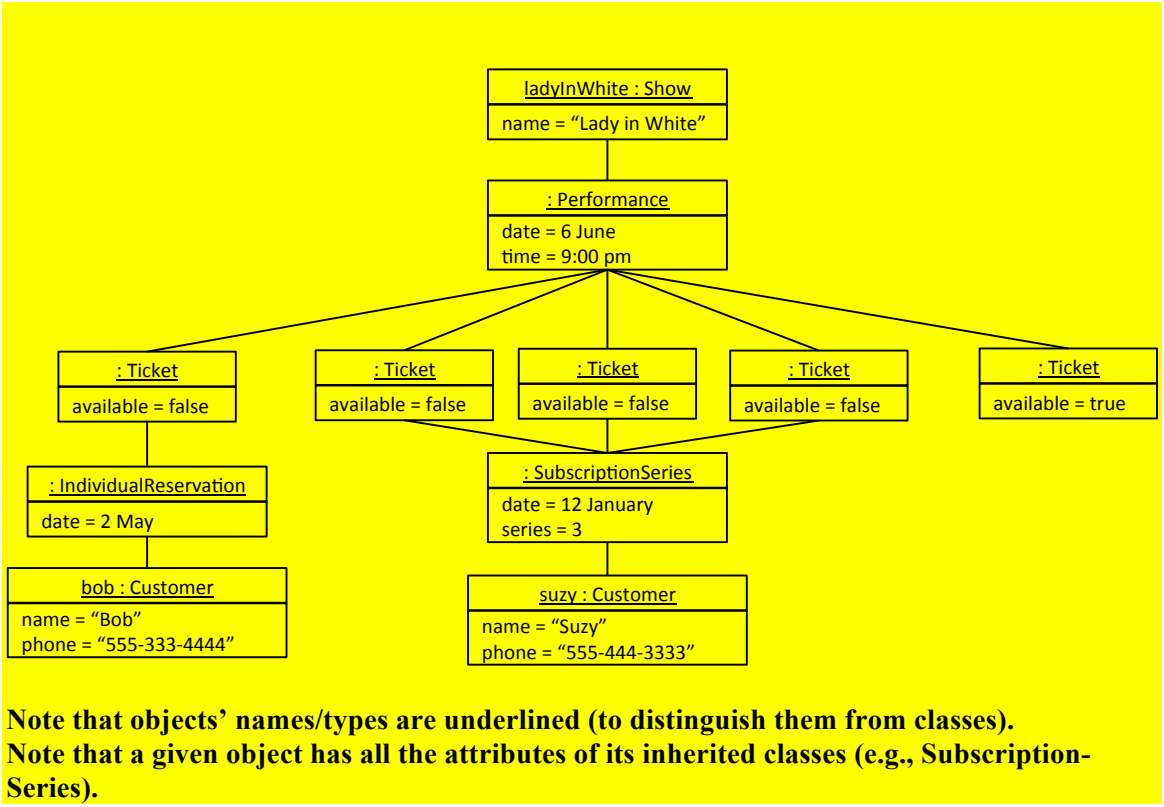
- Reserved-seating tickets that have section, row, and seat numbers
- Unreserved-seating tickets that have an area name

You may elide classes in the diagram that don't change (i.e., neither their contents nor their relationships change). **Note that Ticket is now an abstract class.**



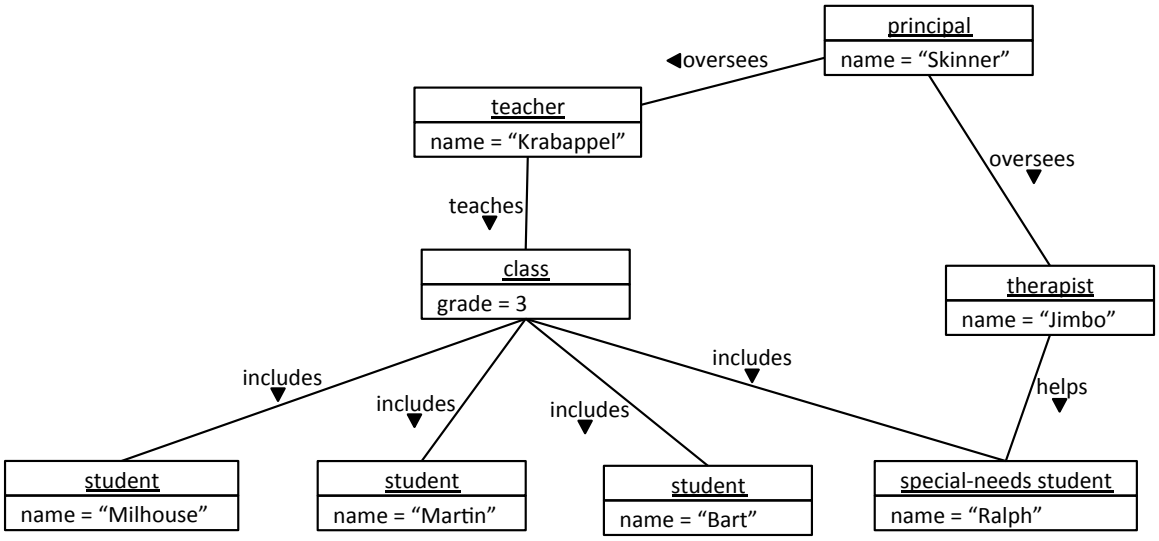
Question 4. Draw an object diagram that models an instance of the system such that:

- There is going to be a performance of the show *Lady in White* on 6 June at 9pm. There are only five tickets for the show (a very small venue) and four of them have been purchased (as follows).
- Customer Bob (555-333-4444) purchased an individual reservation on 2 May.
- Customer Suzy (555-444-3333) purchased subscription series (#3) on 12 January that includes three tickets.

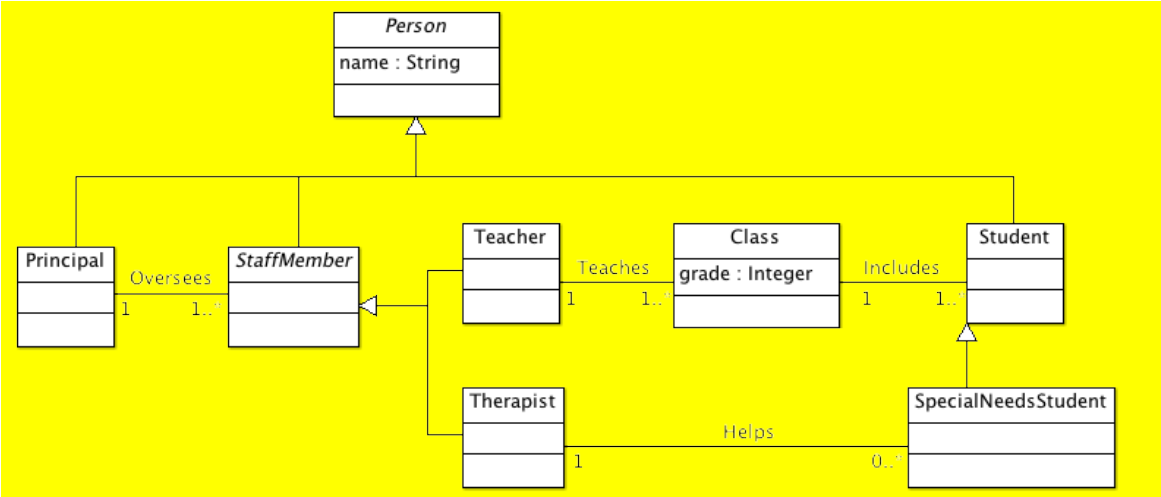


Note that objects' names/types are underlined (to distinguish them from classes).
Note that a given object has all the attributes of its inherited classes (e.g., SubscriptionSeries).

Consider this object diagram that models an instance of a school.



Question 5. Draw class diagram that models a general design for schools based on the object diagram. Think about what classes might be specializations of others and what classes might be abstract. (Note: you may have to invent classes not explicitly suggested in the object diagram.) Include sensible multiplicities and attribute types. You need not include operations.



Note the abstract classes.

Question 6. Define a contract (i.e., define preconditions and postconditions) for a bounded-size stack that can hold a maximum of *N* elements. Assume that from the client perspective, a stack has a *size*, which is the number of elements currently on the stack. Here are the class' operations:

- pop() : Element
- push(e : Element)
- isFull() : Boolean
- isEmpty() : Boolean
- size() : Integer

- pop() : Element
  - Precondition: size > 0
  - Postcondition: return value is the old top Element on the stack and the old top Element is removed from the stack; size = old size – 1
- push(e : Element)
  - Precondition: size < N
  - Postcondition: e is added to the top of the stack; size = old size + 1
- isFull() : Boolean
  - Precondition: true
  - Postcondition: return value is true if size = N; otherwise, return value is false
- isEmpty() : Boolean
  - Precondition: true
  - Postcondition: return value is true if size = 0; otherwise, return value is false
- size() : Integer
  - Precondition: true
  - Postcondition: return value is size

Question 7. Define a contract for the following 4-way traffic light (with directions N, S, E, and W). Make sure that cars won't crash and that a light always goes to yellow before it goes to red. (If it helps you think about the problem, assume that the lights all start out as red.)

- changeToGreen(light : Direction)
- changeToYellow(light : Direction)
- changeToRed(light : Direction)
- getLightColor(light : Direction) : Color

- changeToGreen(light : Direction)
  - Precondition: light is red and perpendicular lights are red
  - Postcondition: light is green
- changeToYellow(light : Direction)
  - Precondition: light is green and perpendicular lights are red
  - Postcondition: light is yellow
- changeToRed(light : Direction)
  - Precondition: light is yellow
  - Postcondition: light is red
- getLightColor(light : Direction) : Color
  - Precondition: true
  - Postcondition: return value is the color of light