# SYD366 Week 2

AN INTRODUCTION TO CLASSES

# Agenda

- Housekeeping
- What is a "Class"?
- Associations
- Examples

# Housekeeping

- Remember to familiarize yourself with the course schedule!
  - Being unaware of a due date is not an excuse for missing it
- As I explained there are two different weekly deliverables:
  - The lab activity for which you must attend to get a mark
  - The lab assignment, due on Sundays
- Read the course document on UML
  - Can be found under "Course Notes" on Blackboard

# Classes

# What is a class?

- As you are aware from your OOP course, a class is a uniquely defined object with its down member variables, functions and characteristics
- In OOP, we treat all data as a class
  - In contrast to functional programming where most things are treated like functions
  - That said, functional programming still relies on clearly defined data in your database

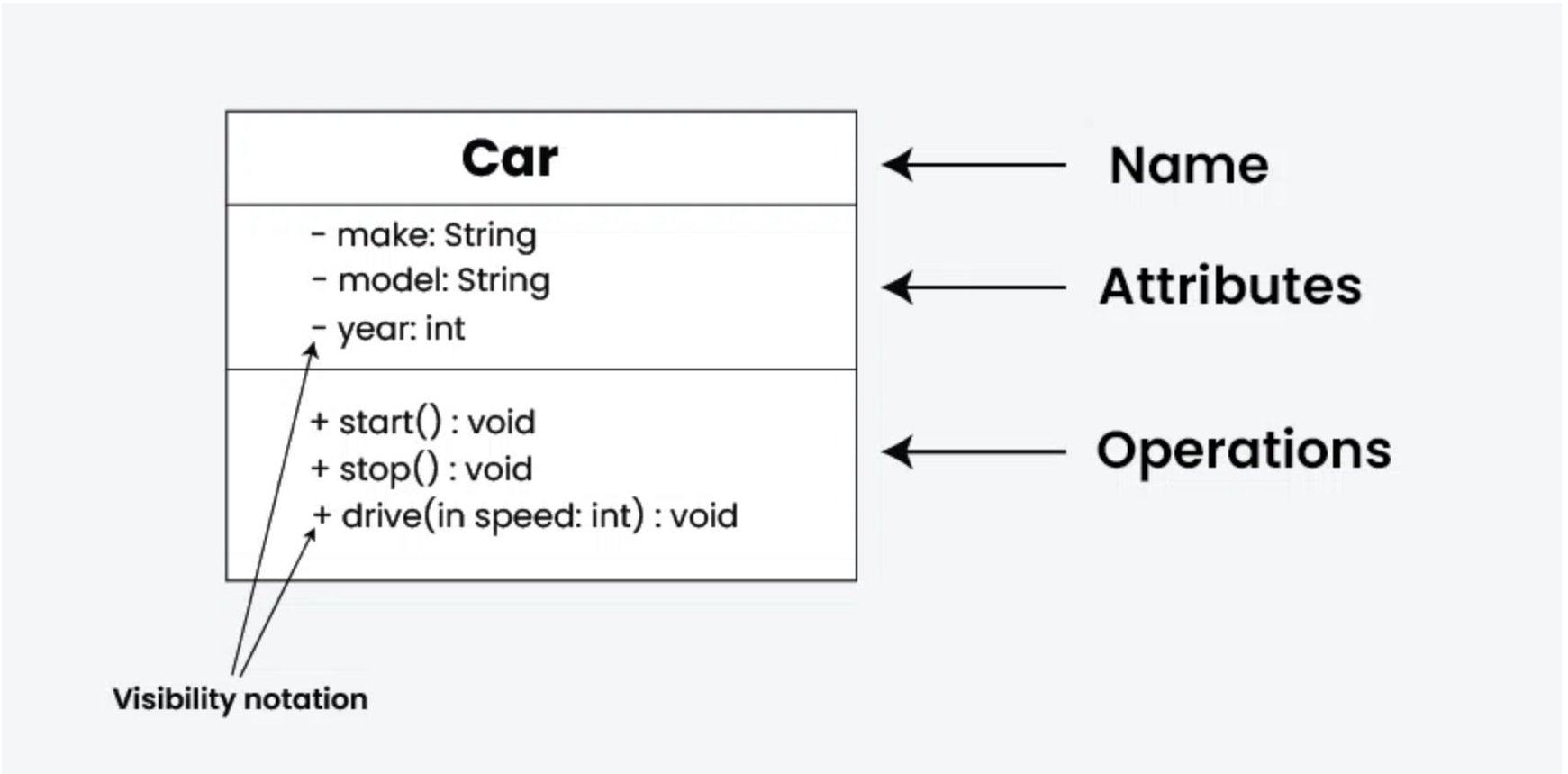
# Class Diagrams

- UML class diagrams are a foundational element in object-oriented modeling
- They the static structure of a system
- They capture the blueprint of an application by detailing:
  - The system's classes
  - Their attributes
  - Methods
  - Relationships

# Class Diagrams - Purpose

- Help in visualizing the system's structure, understanding the system's requirement and guiding implementation
- They are useful in various stages of software development:
  - Analysis
  - Design
  - Documentation

# Class Structure



# Class Structure

- A class is a rectangle divided into three sub-rectangles
- The top rectangle contains the class name
- The middle one lists its attributes
- The bottom one shows its operations (methods)
  - These outline the capabilities and responsibilities of a class outside basic OOP functions

# A Note on Controller Classes

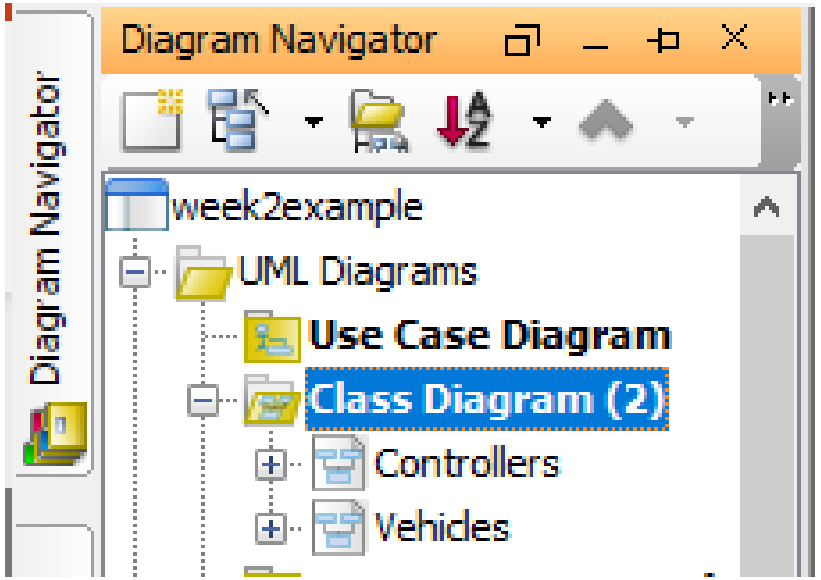
- These classes won't make a lot of sense until we cover sequence diagrams next week
- Right now, when you create a class diagram, you are also expected to create controller classes
- These are special classes that **do not** get member attributes/variables/etc.
  - They will eventually get functions

# A Note on Controller Classes

- Controller classes go in a separate class diagram in the diagrams folder from your other classes
- The diagram will look like this for now:

# A Note on Controller Classes

- Your folder structure in Visual Paradigm should look like this:



# Associations

# What is an Association?

- Your programs will always end up with multiple classes
- How do we relate the classes together in a meaningful way?
  - We create associations!
- Associations are lines which connect classes together and can explain their relations at first glance
  - These will become more complex as you take more systems courses, but for now they're just lines with numbers

# Relationships

- UML class diagrams depict various types of relationships between classes, including associations, generalizations and dependencies
- We will primarily cover associations (the most basic form of relationships) but will also get into generalizations (such as inheritance) later

# Types of Associations

Classes/objects participate in a variety of relationships with other classes/objects including the following:

- **Simple Associations**:
  - Objects from associated classes know about each other and can pass messages and invoke functions
  - Can be uni-directional or bi-directional
    - Should only be uni-directional though
    - Use bi-directional sparingly

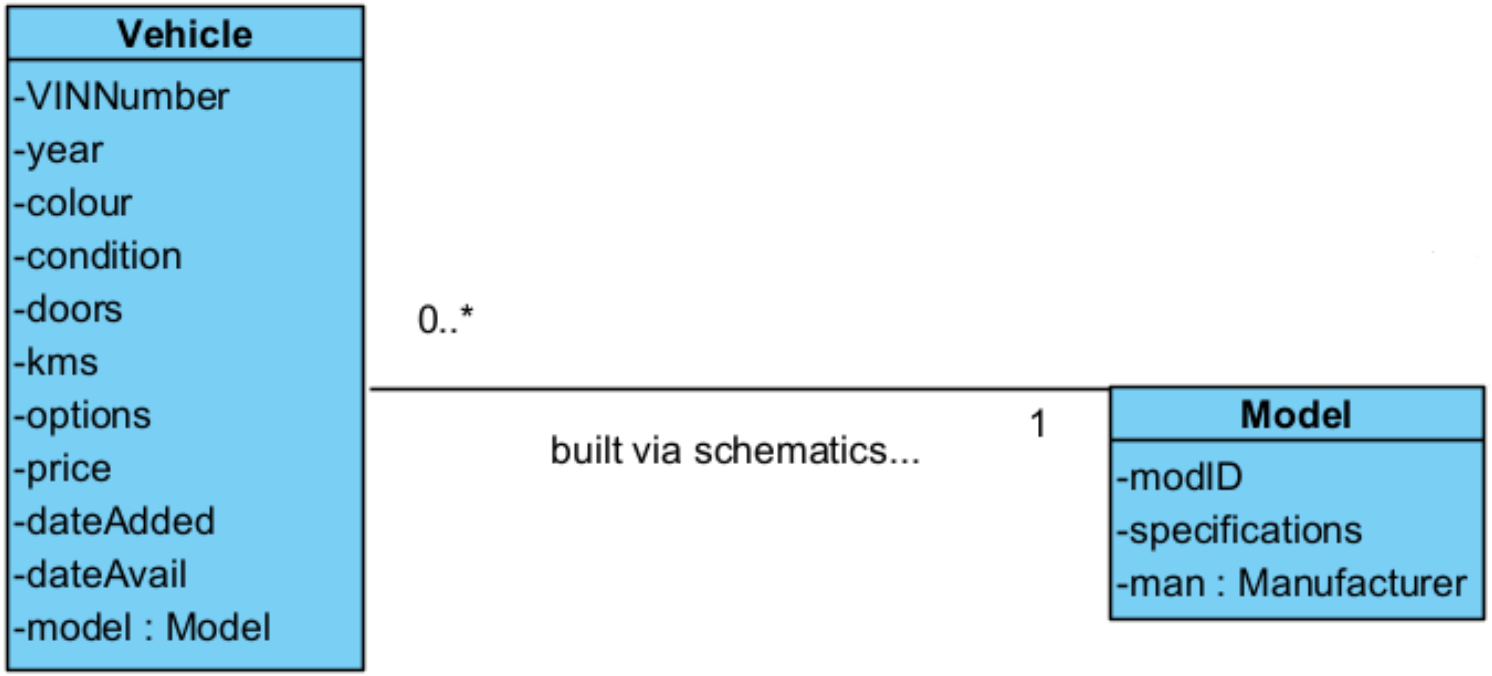
# Types of Associations

- **Compositions** (containment):
  - A container class "contains" other classes. (next semester)
- **Generalizations:**
  - A relationship in which specific "child" or sub classes are derived from a generalized "parent" or super class. (next semester)

# When to use Associations

- We can only define and show associations successfully when we have enough information from our requirements or our business knowledge. For example:
  - Does the system need to remember that a specific clerk made a sale or is clerk not related to sale in the system?
  - Does the system need to remember the model of a bicycle or simply the manufacturer?
- Class associations typically indicate relationships which need to be remembered.

# Drawing Associations



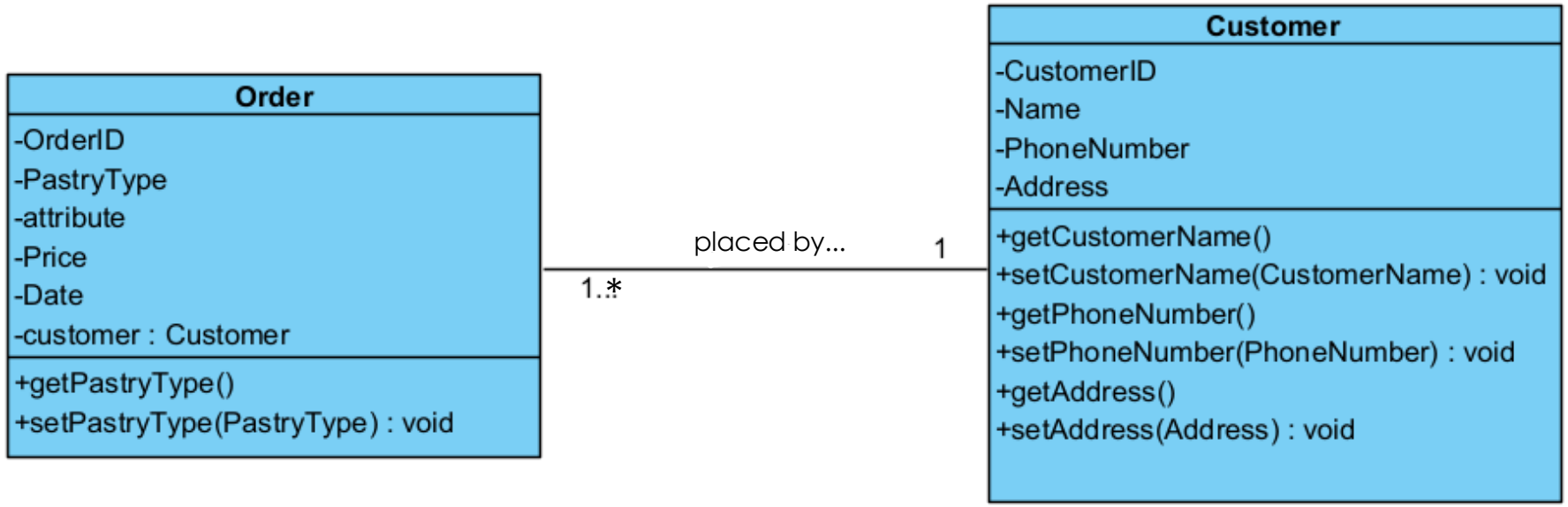
# Drawing Associations

- You'll notice on the last slide that the connection has not only numbers, but a verb!
- The numbers indicate how many of the class are in a relationship
  - The "..*" indicate more than 1
  - You can also have "0..*" Which means 0 or greater
- The verb is important
  - It indicates to your reader or system admin how the classes are related to one another
  - Don't leave that blank!

# Associations Denoting Reference Attributes

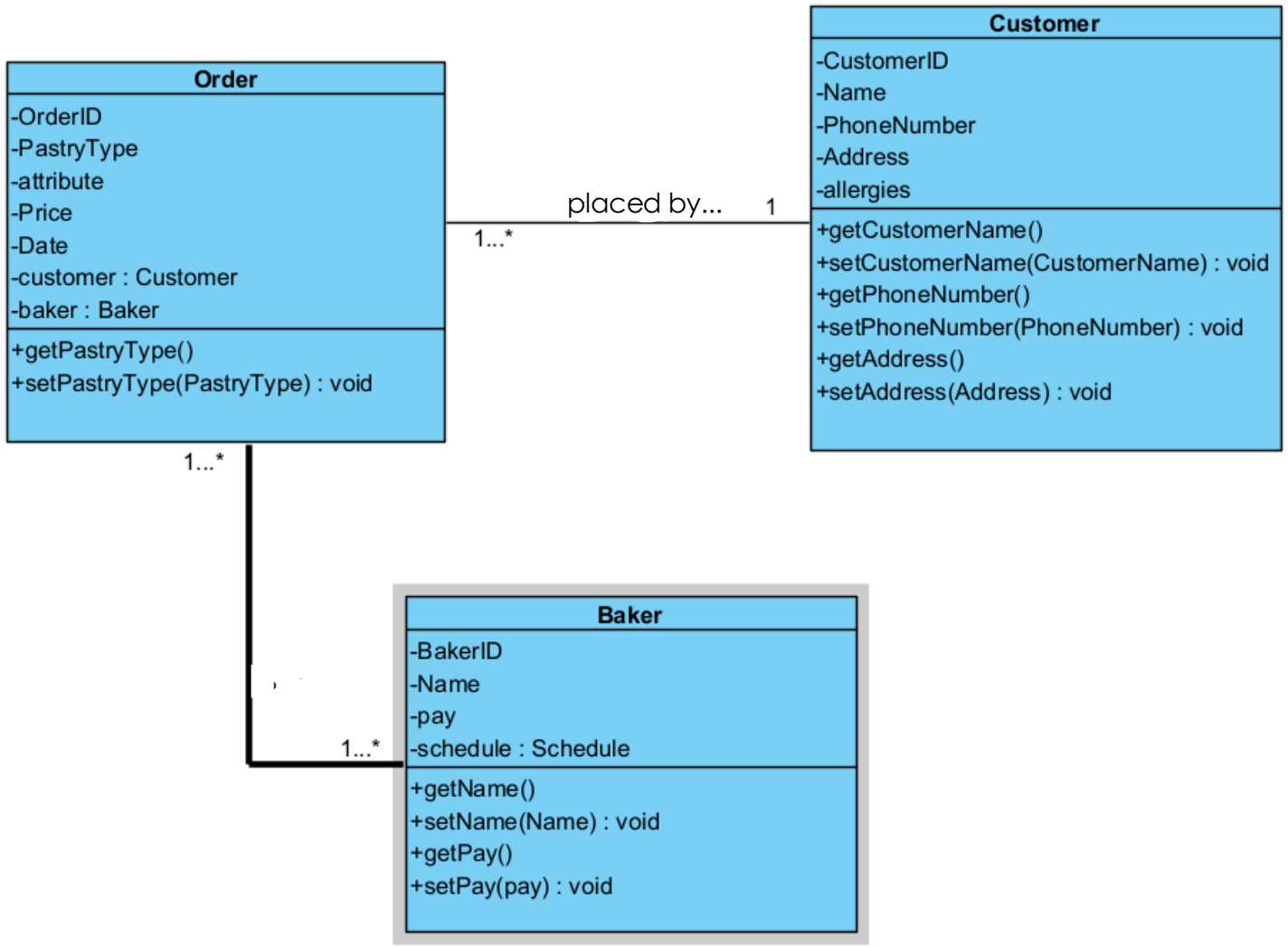
- Consider the scenario:
  - Cassandra wants to check the bakery orders
  - She clicks an order and she wants to call the customer to inform them that their order is ready!
  - Clicking the customer will give her access to all the customer information.
    - Including its getters and setters*

# Associations Denoting Reference Attributes



# Associations Denoting Reference Attributes

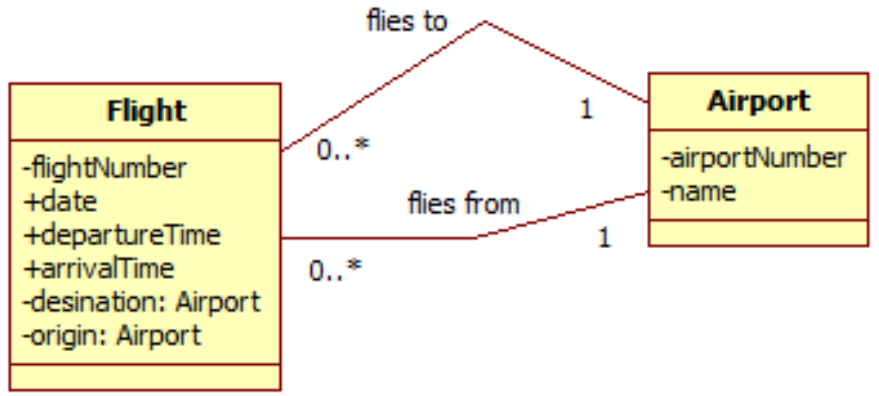
- The attribute "customer" is defined as an instance of the class Customer.
- In UML we can see this in the attribute properties—the type is defined as Customer
- In the model on the previous slide, we see that we can access the attributes (and operations) of Customer through customer.
- The reference attribute defines how we navigate this association.




# Reference Attributes As Sets

# Multiple Associations

- Sometimes you will have classes which rely on each other
  - Generally, I advise against this
  - Circular associations and relationships are bad news!
- Should you find yourself requiring one, this is how you would do it.



# Utility of Class Diagrams

- Class diagrams are particularly useful for designing and communicating complex systems
  - This ensures all stakeholders have a clear understanding of the system's architecture and functionalities
- They are also used to create blueprints for code generation in object-oriented programming languages.

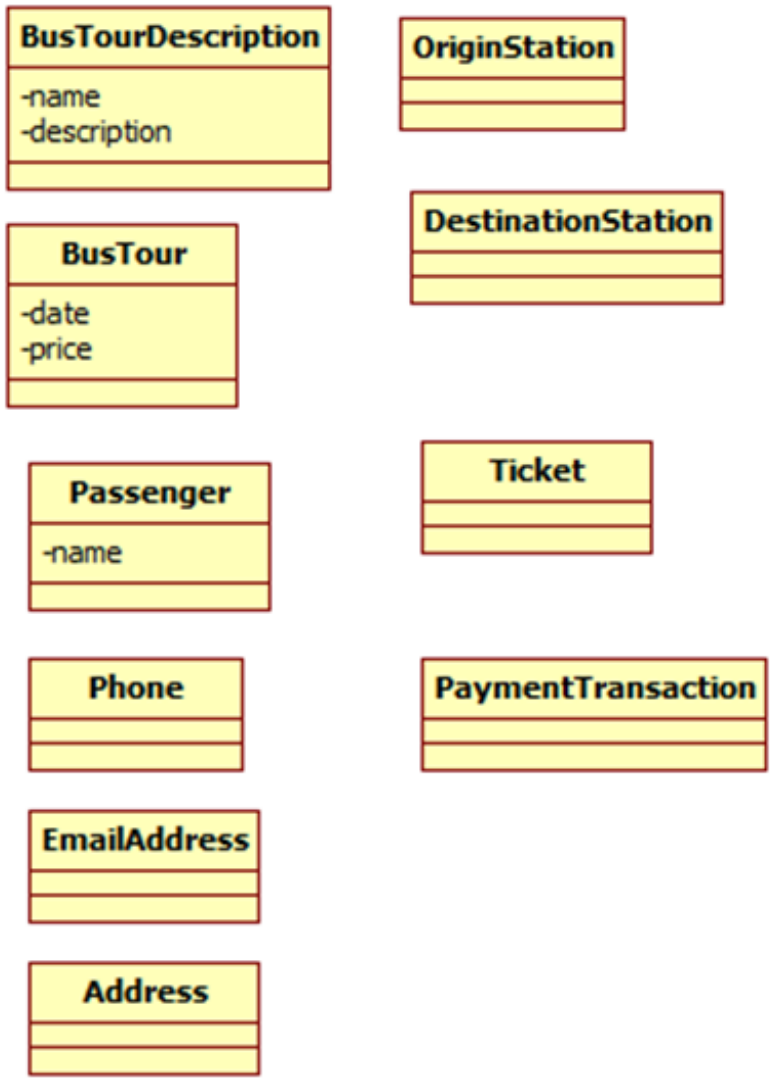
# Summary

- Classes are objects which you have defined which contain:
  - Their own data
  - Their own functions
- Associations are the relationships between these classes:
  - They indicate relationships, such as 1 to 1 and 1 to many, etc.
  - They show directional flow of data

# Questions?

# Exercise #1

| Actor (Tournament Coordinator) | System |
|---|---|
| Enters date, name of tournament and maximum number of golfers. | Creates the tournament and displays an entry area for 10 golfers with spaces for name, contact information—email address and/or phone number, handicap. |
| Enters golfer information and requests to add. | Checks that maximum number of golfers has not been exceeded and adds the golfers to the tournament.<br>Displays an entry area for more golfers. |
| Repeats step 2 until done | Displays and prints a list of golfers registered for the tournament. |




| Actor (online customer) | System |
|---|---|
| Chooses a bus tour from a list of tours | Displays tour name, originating station and destination station and tour description. Also displays a list of dates on which the tour is offered.<br>Displays tour price for each of the dates (summer tours are more expensive than spring and fall tours). |
| Selects one of the dates and requests to book a ticket for the tour | Displays an entry form for name, address, phone and email. |
| Enters name, address, phone, email. | Displays total price and all tour information for confirmation. |
| Confirms | Transfers to paypal and completes the payment transaction.<br>Emails a ticket to the traveller. |

# Exercise #2