UML Diagrams

Concepts, diagram types, class diagram notations/terms, sequence diagram notations/terms, usage

What is UML?

UML. Unified Modeling Language.

- A software blueprint language for object-oriented programming.
- UML consists of a collection of types of *diagrams* for computing projects.
- UML diagrams are used for requirements, organizational information, sequence and action information, state information.

OMG - Object Modeling Group - is responsible for overseeing UML and its changes.

See: www.omg.org

UML diagram types

14 UML diagram types: 7 structural. 7 operational.

Our HTML lecture notes show 8 traditional types:

- **Use case diagram**
- **Activity diagram**
- **Class diagram**
- **Sequence diagram**
- **Collaboration diagram**
- **Statechart diagram**
- **Component diagram**
- **Deployment diagram**

These lectures cover traditional types only.

Structural diagram types

Three traditional types:

- **Class diagram**: static snapshots of the classes in a system and the relationships between them.
- **Component diagram**: dependencies among the software components in a system.
- **Deployment diagram**: the physical arrangement of software and hardware on which the final system is deployed (where the system resides)

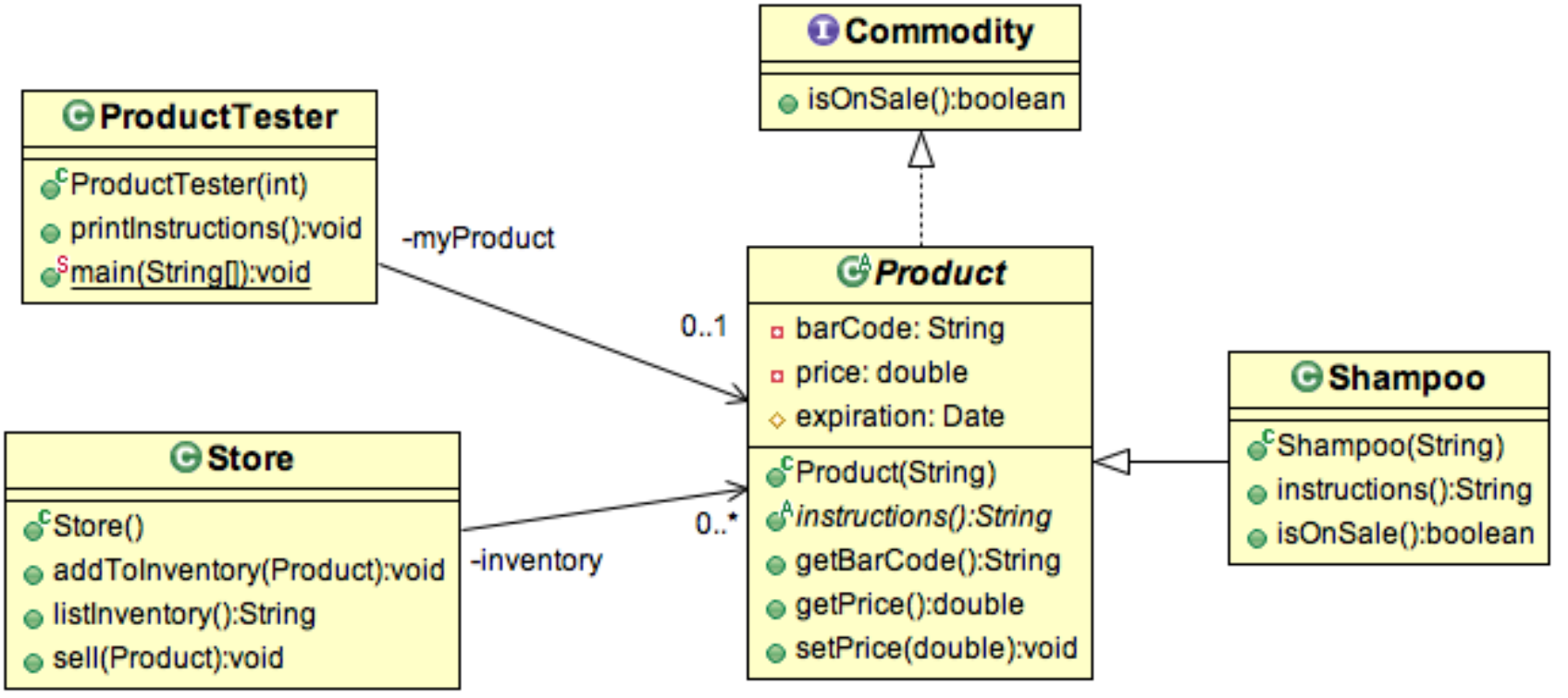
Operational diagram types

Five traditional types

- **Use Case diagram**: how the system provides services to a user.
- **Activity diagram**: flow of actions in accomplishing a task.
- **Sequence diagram**: the sequence of actions (statements) that result from a message (a method call).
- **Collaboration diagram**: how objects collaborate to perform a task (to complete a message call).
- **Statechart diagram**: how objects change state in performing tasks.

Simple class diagram

Classes and the relationships among them.

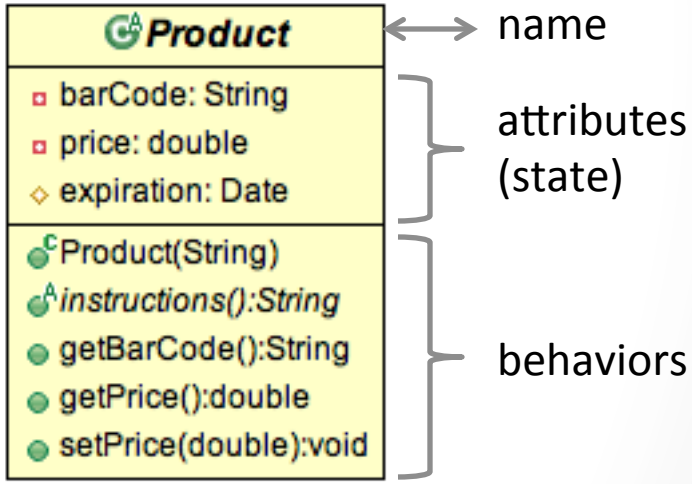


Class diagram notation: Classes

Classes are rectangles.

Class members - standard notation:

- + means public
- - means private
- Underlined means static
- Italics means abstract or interface



Software that constructs class diagrams from code may use non-standard notation (symbols and colors). In this image:

- Red = private
- Green = public
- Gold = protected

Class diagram notation: Relationships

Relationships are links. Four kinds:

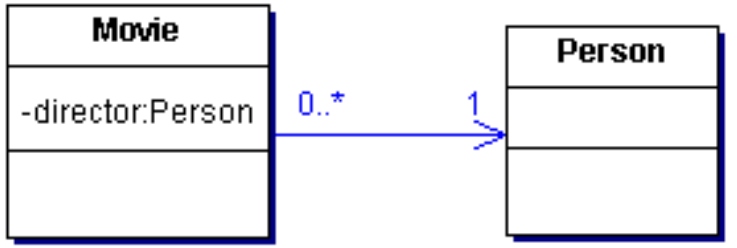
- **Association** - *has-a* relationship
  - Arrow points to the contained object type/class.
  - Arrow may be labeled with the name of the instance variable of the contained class type. In that case, the instance variable will not show in the attributes of the class rectangle.
  - Can mark with **multiplicity** to indicate possible number of instances contained (or containing).
  - Can further refine into aggregation or composition.
- **Generalization** - *is-a* relationship
- **Realizations** - implements relationship
- **Dependency** – (informal) dependency relationship (not used in this course)

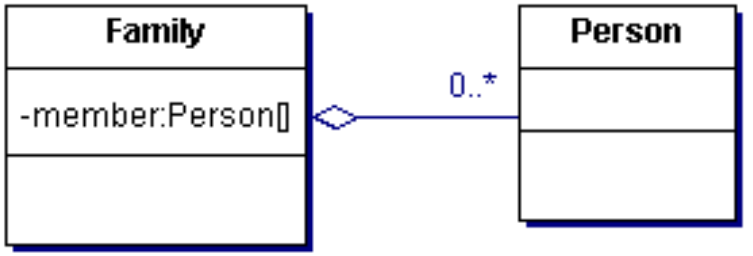
Class diagram notation: Associations

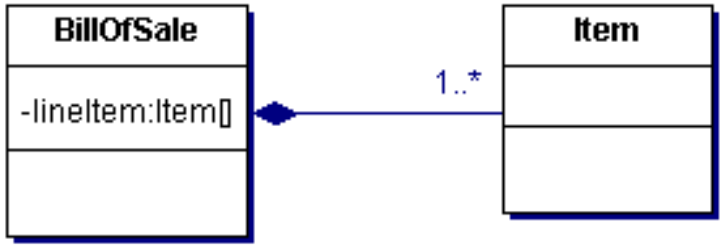
Association. *Has-a* relationship. Arrowheads are optional. Refinements:

- **Aggregation**. The contained object can live apart from the containing one. Example: people in a house.
- **Composition**. The contained object cannot live apart from the containing one. Example: room in a house.

There is endless confusion over aggregation vs. composition. The difference is not an issue for this course!



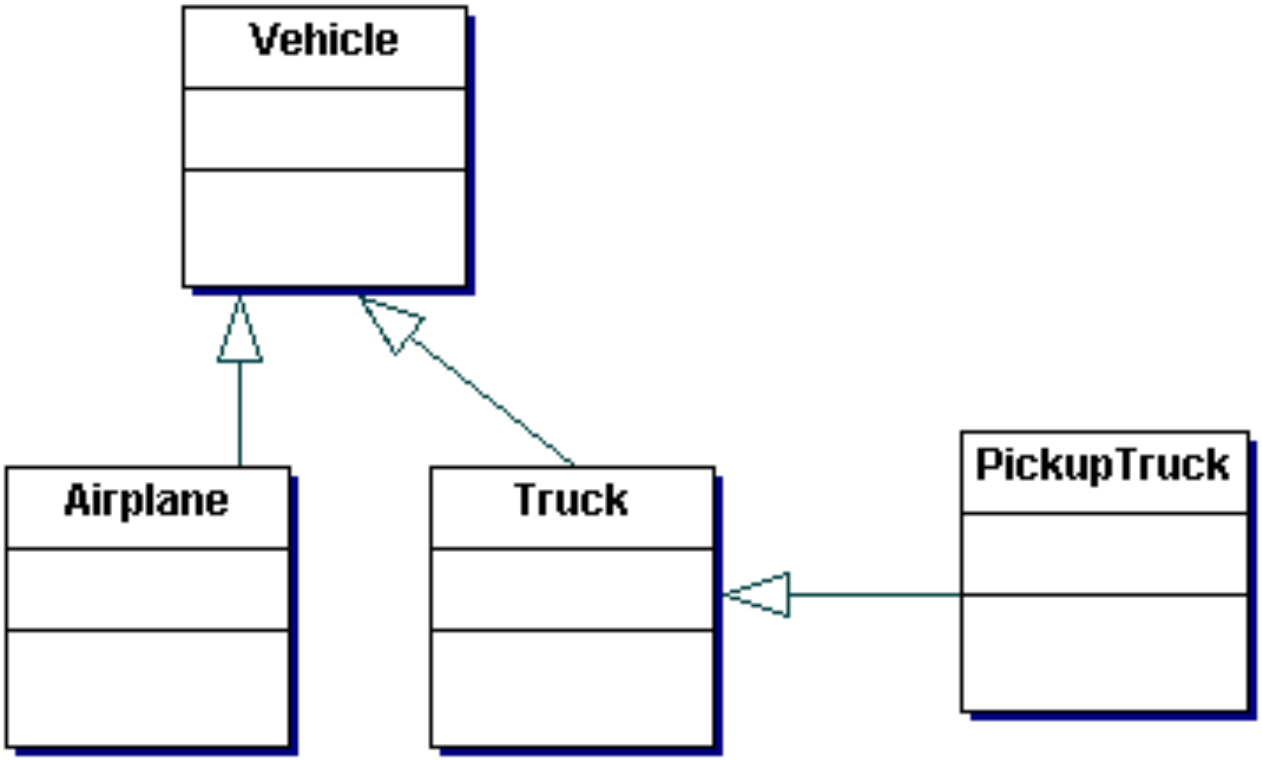




Class diagram notation: Generalization

Generalization = inheritance

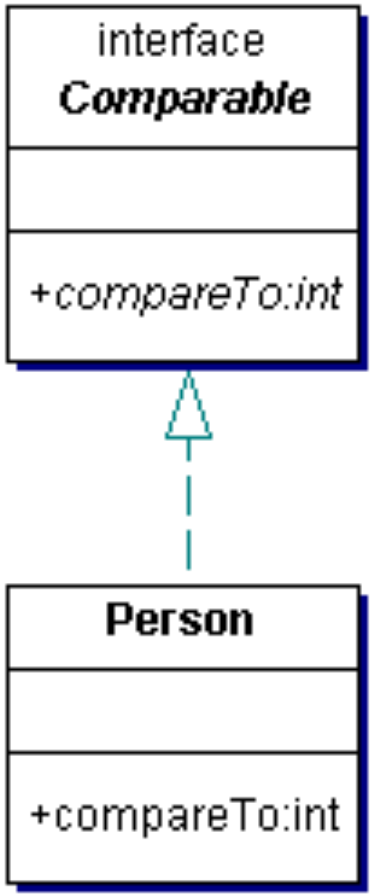
Solid line from child to parent. The arrowhead is a triangle.



Class diagram notation: Realization

Realization = implements

Dotted line from class to interface. The arrowhead is a triangle.



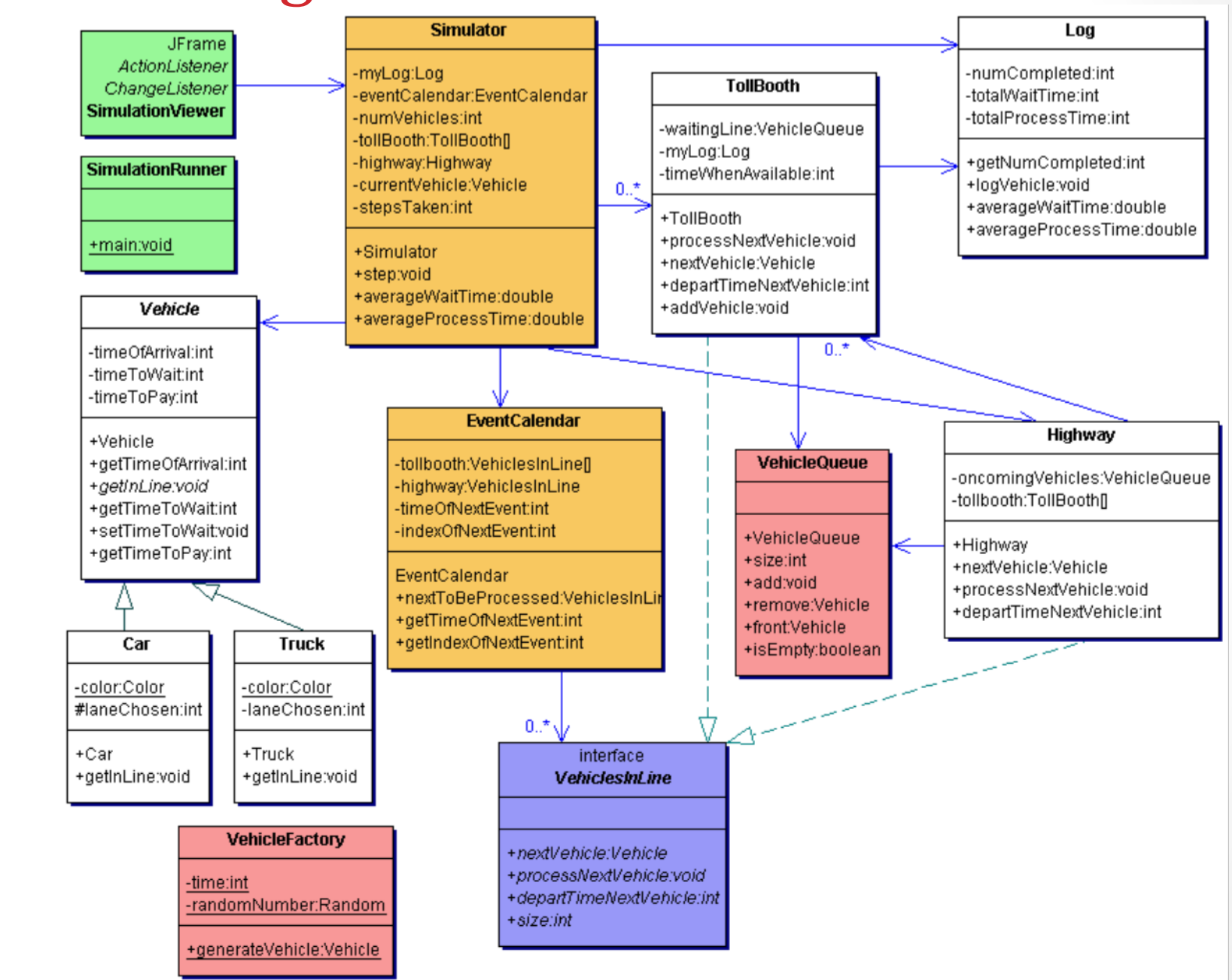
An old example

Toll booth/highway simulation

"In order to plan a toll highway, the highway department must estimate the wait times for cars and trucks that line up at toll booths. The highway department wants to simulate the activities around vehicles, the highway, and the toll booths. The simulation will keep track of the time each vehicle spends in line to pay its toll and how much it pays, logging this information as the vehicle leaves the toll booth line. The simulation log contains the running statistical results as they accumulate during the simulation."

The diagram on the next slide was generated from code by an old tool.

Class diagram



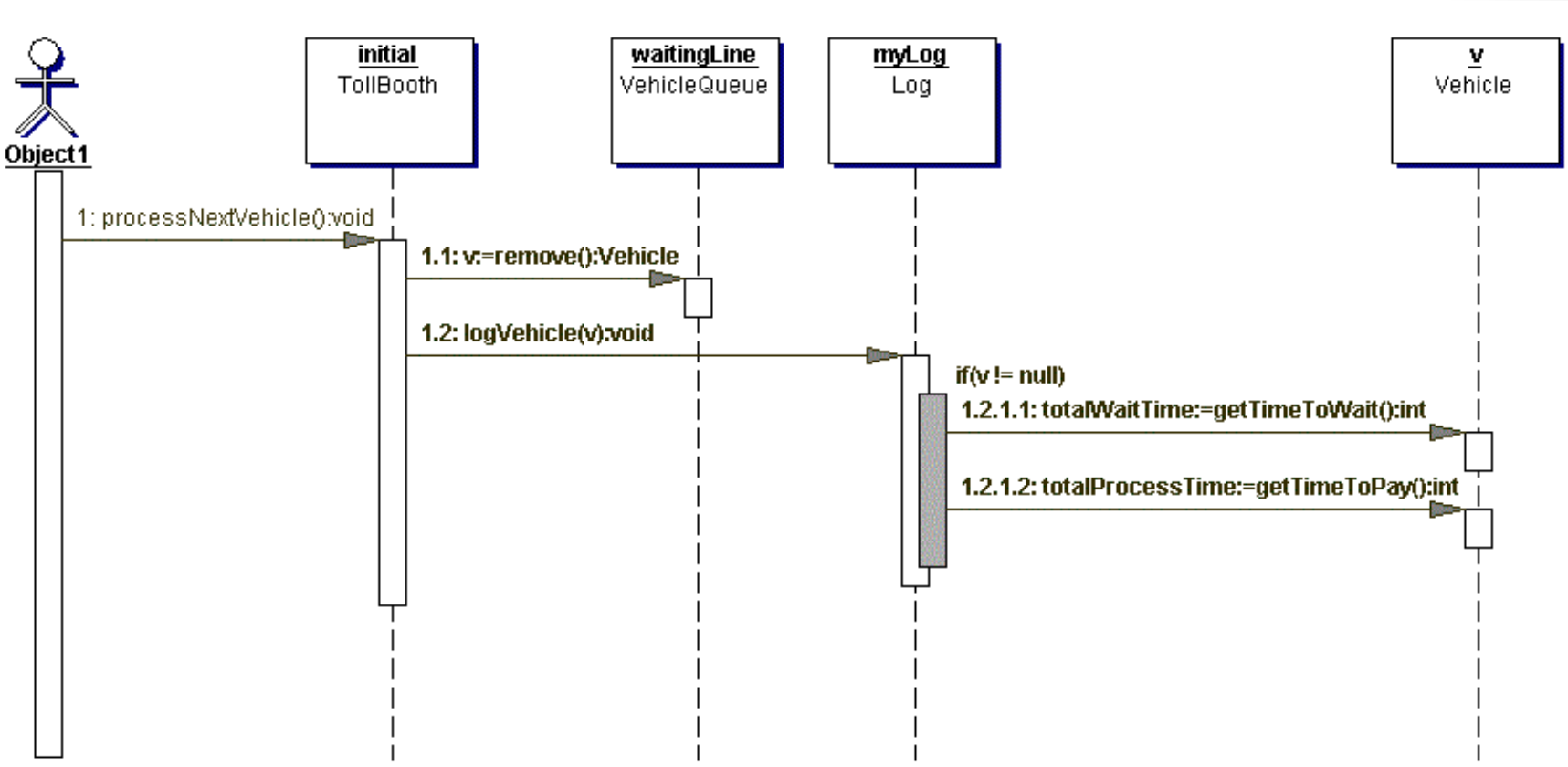
Sequence diagrams

Sequence diagrams show what happens when messages are passed (methods are called).

- They are dynamic (as opposed to static class diagrams).
- Message call always starts with an “actor” – some object that makes the call. Each message is passed to another object.
- Top of the diagram has objects receiving the message.
- Below each object is its **lifeline**.
- The rectangular bar on each lifeline corresponds to the **activation record** for the method call.

Sequence diagram example

TollBooth.processNextVehicle()

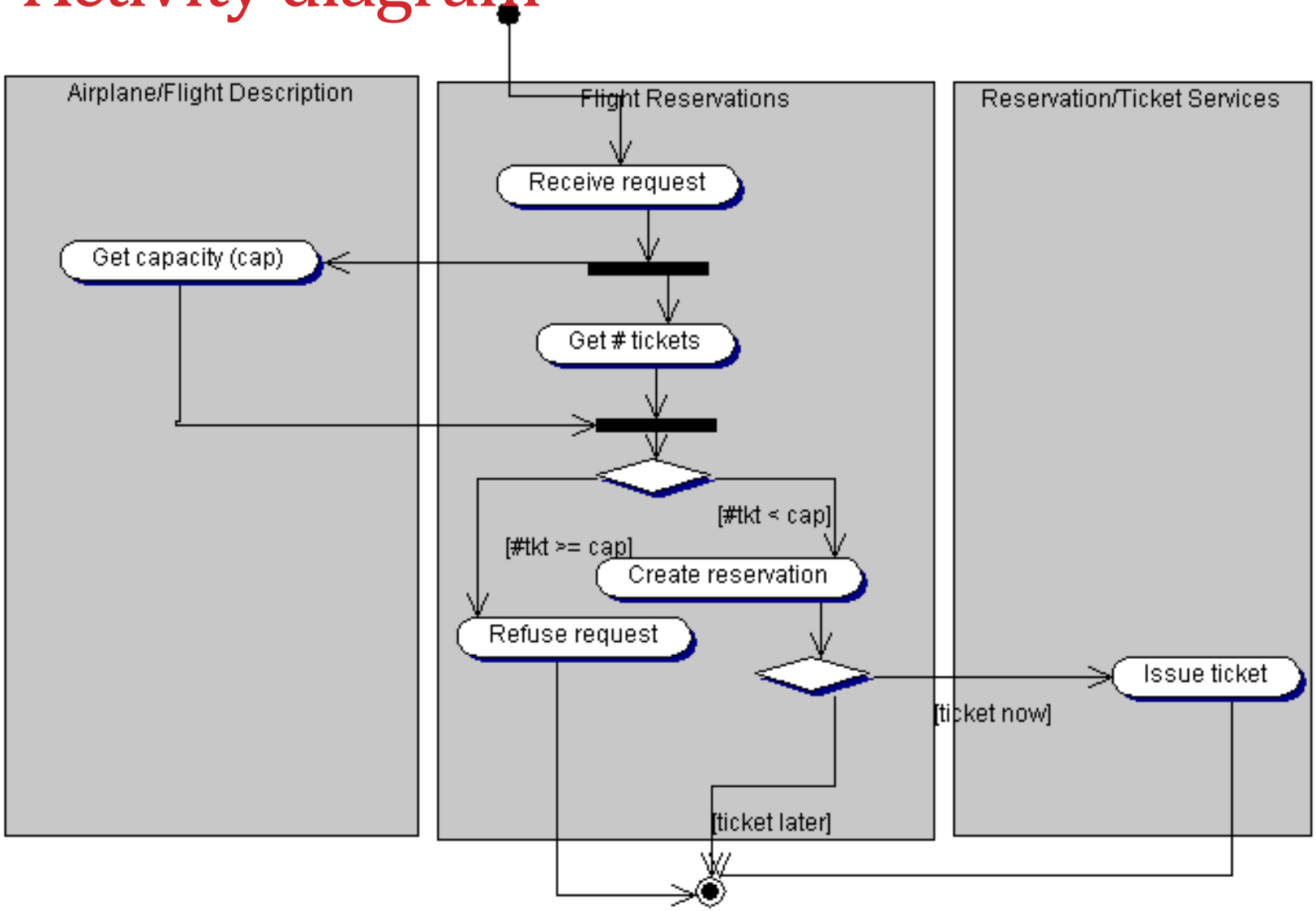


Activity diagrams

Activity diagrams show how a particular task is performed. They are like classic flow diagrams.

- Subtasks are represented as roundtangles.
- Decisions are diamonds.
- Subtasks can be divided into **swimlanes** to indicate who is responsible for fulfilling each subtask.
- Activity diagrams are useful outside the domain of software programming. Can be used for process descriptions for example.

Activity diagram



Statechart diagrams

Statechart diagrams show the change in state of an object during the execution of a procedure. (Statechart diagrams are analogous to finite state machines.)