MULTI-CLASS PROGRAMS AND THE UML

Object-Oriented programs consist of connected objects

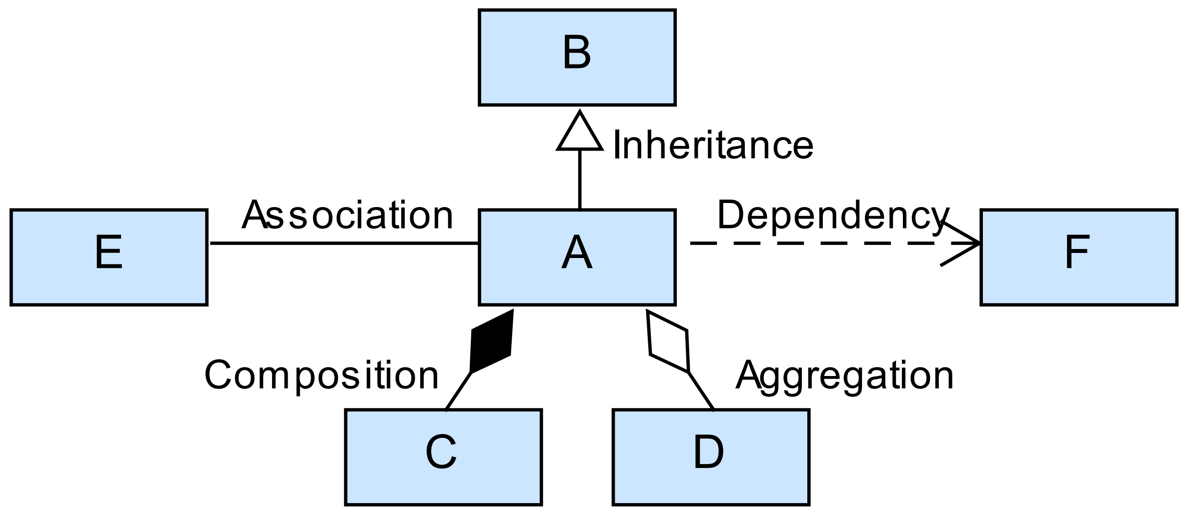
Delroy A. Brinkerhoff

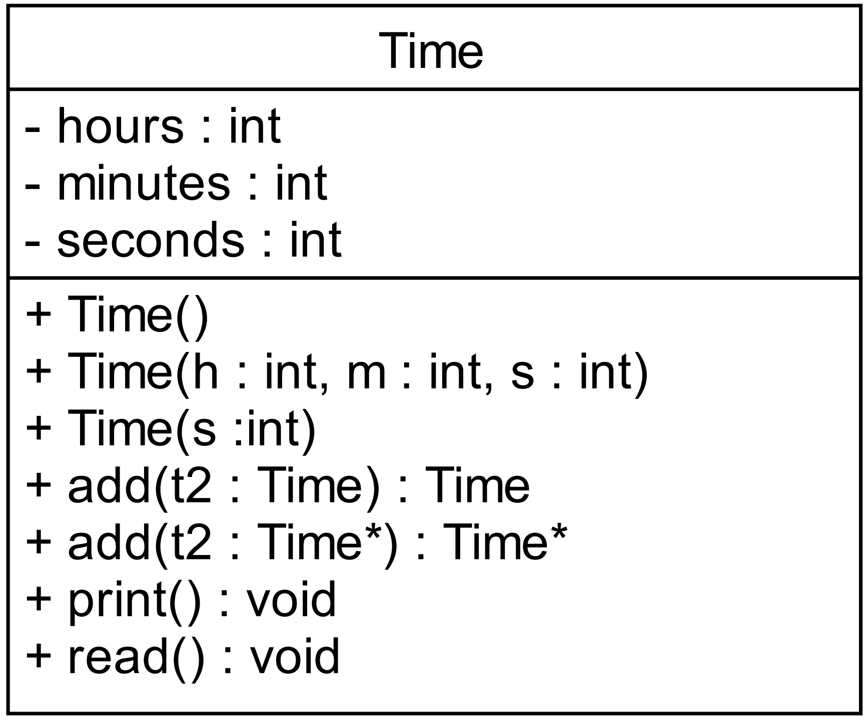
CONNECTING OBJECTS

- Classes are connected by class relationships
- Objects are instantiated from classes
- The connections between objects are derived from the relationships between the classes
- The connections between objects
  - Bind the objects together
  - Allow the objects to work together
  - Provide communication pathways along which the object can send messages

CLASS RELATIONSHIPS

- Inheritance
  - Second characteristic of the object-oriented paradigm
  - Required for polymorphism
- Constructive relationships
  - Aggregation
  - Composition
  - Association
- Dependency





SHARING CLASS RESPONSIBILITIES

- Classes are responsible for managing their member data and providing and for providing functions to operate on it
- Classes share responsibilities through their relationships
- One class can "ask" another for help by sending it a message
- Class relationships form message pathways

MESSAGE SENDING: A WHOLE / PART EXAMPLE

PART

```
class Engine
{
    private:

    public:
        void start();
};
```

WHOLE

```
class Car
{
    private:
        Engine motor;

    public:
        void function()
            { motor.start(); }
};
```

OBJECT-ORIENTED PROGRAMS

- Program classes must match the entities appearing in the original problem
  - Car: how many doors, what color, etc.
  - Engine: size, how to measure RPM's, oil pressure, etc.
- Class relationships must match the way entities relate to each other
  - Whole/part: "A Car has an Engine" or "An Engine is part of a Car"
- The meaning of the class relationship matches the way that entities relate to each other in the original problem

CATEGORIZING CLASS RELATIONSHIPS

- Semantics or meaning
- Directionality or navigability
- Lifetime
- Sharing

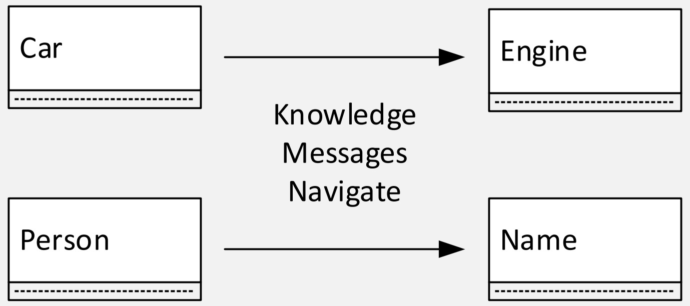
- Binding strength
  - Strong/tight implies
    - coincident lifetimes
    - exclusive ownership (no sharing)
  - Weak/loose implies
    - independent lifetimes
    - sharing is allowed

SEMANTICS / MEANING PROPERTY

- Inheritance
  - “is a”
  - A Student is a Person
- Aggregation and composition
  - “has a” or “is a part of”
  - A Car has an Engine, or an Engine is part of a Car
- Association
  - “has a” in both directions
  - A Contractor has a Project, and a Project has a Contractor

DIRECTIONALITY / NAVIGABILITY PROPERTY

- Every relationship is between two objects and is directional
  - Unidirectional or one direction (most class relationships)
  - Bidirectional or in both directions (only association)
- Ways of thinking about directionality
  - The direction messages travel
  - Which object "knows about" the other
  - How a program can navigate or move from one object to the other



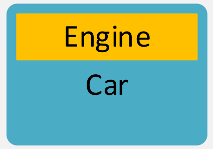
BINDING STRENGTH: LIFETIME AND SHARING PROPERTIES

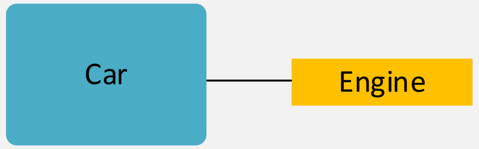
LIFETIME

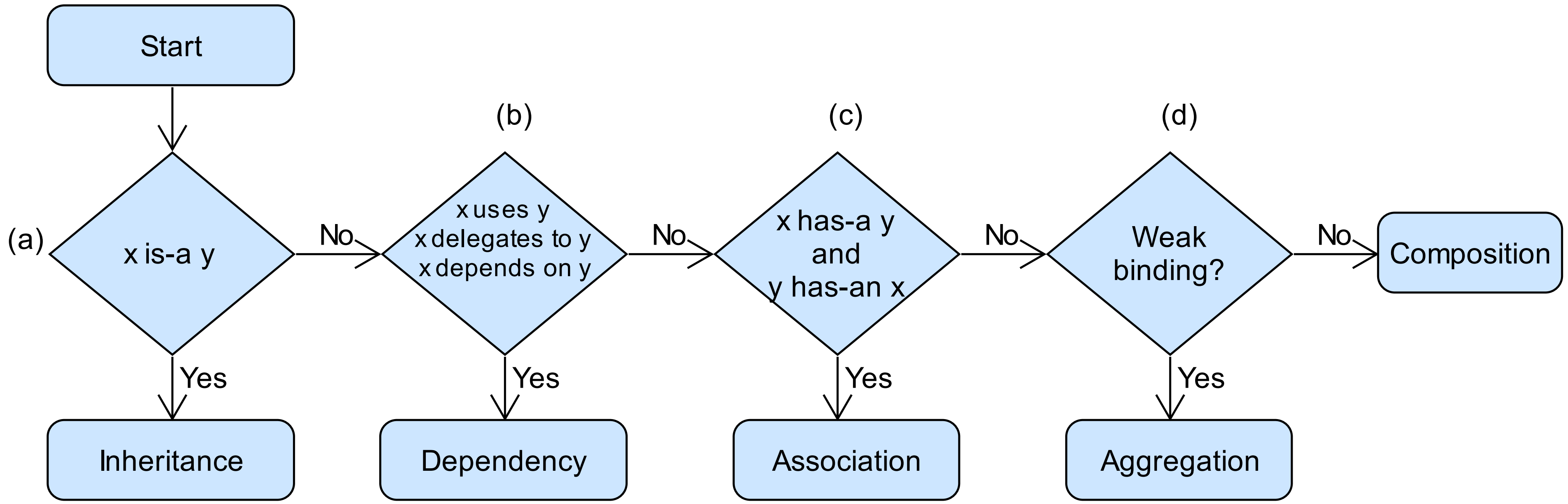
- Coincident or same
  - Both objects and the relationship are created and destroyed at the same time
- Independent
  - The objects and the relationship may be created and destroyed at different times

SHARING

- Exclusive
  - A whole object does not share it part object with any other object
- Sharable
  - A whole object may share its part object with other objects in the program







IDENTIFICATION WITH A DICHOTOMOUS KEY