Team 302
Unified Modeling Language (UML) – Class Diagrams

302 POWERED BY DRAGON FIRE

Interface vs.Class vs. Object

- Interface defines the methods available
- Class
- Object is the concrete instance of a class

Interface

- Abstract (Cannot be made into an object or instantiated)
- Defines methods Available
- No Attributes
- Useful for decoupling objects (methods work with an interface instead of a concrete class, so a new class can implement the interface and other classes don’t need to change).

Class

- Can implement an interface
- Can be made into an object
- Defines the blueprint of the object
- Defines Behavior (methods and attributes)

Object

- Concrete instance of a class
- The same class can be used to create multiple objects (e.g. there are 4 CANTalon Objects – one for each drive motor – but only 1 CANTalon class)
- Each object is accessed by pointer or reference

Object

CANTalon* m_leftDriveMotor = new CANTalon(1);

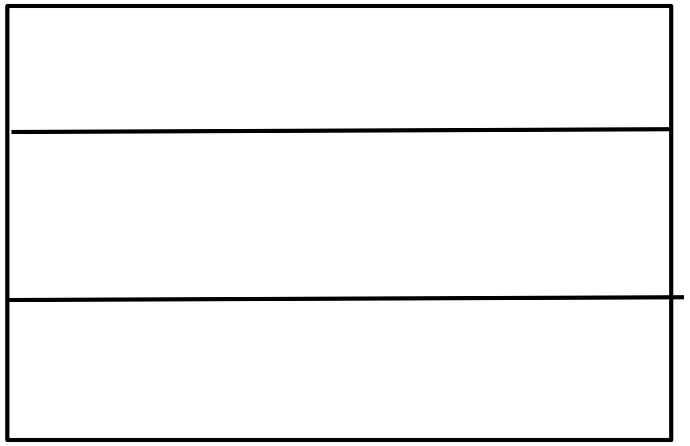
Is there an interface used? If so, what is it?

Is there a class? If so, what is it?

Is there an object defined? If so, what is it?

Class Diagrams

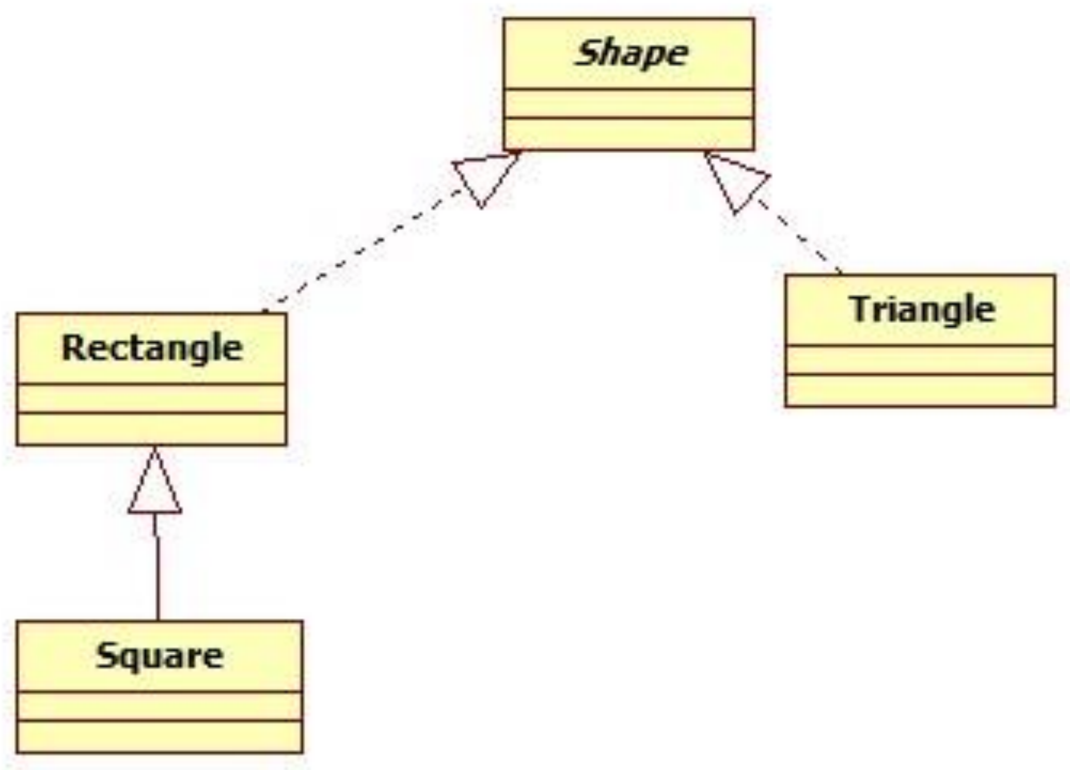
- Static Structure Diagram
- Shows Classes
  - Attributes
  - Methods (operations)
  - Visibility of Attributes/Methods
    - + Public
    - - Private
    - # Protected
- Interfaces
  - Pre-UML 2.0 - Class name in italics if interface
  - UML 2.0 - circle instead of rectangle



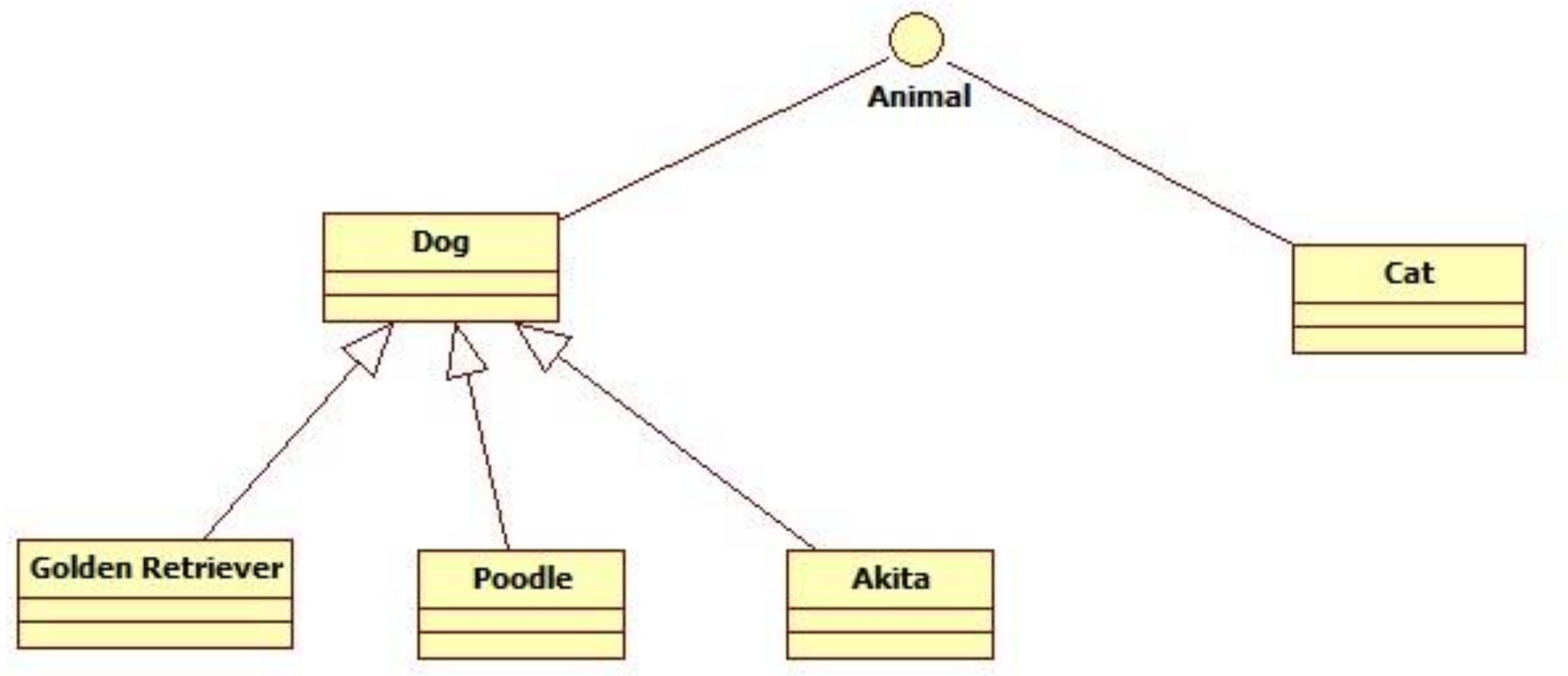
Class Diagrams – Hierarchy Relationships

- Realization (class implements an interface)
  - Interface-Class relationship
  - UML2.0 Interfaces
    - Shown with a Circle
    - Solid Line between Class and the Interface
  - Pre-UML 2.0 Interfaces
    - Shown just like a class except
      - name in italics (sometimes <<interface>> is shown above as well)
      - dashed line connects with a hollow triangle arrowhead
- Generalization (class extends another class)
  - Class-SubClass relationship
  - line connects with a hollow triangle arrowhead

Class Diagrams – Pre UML 2.0



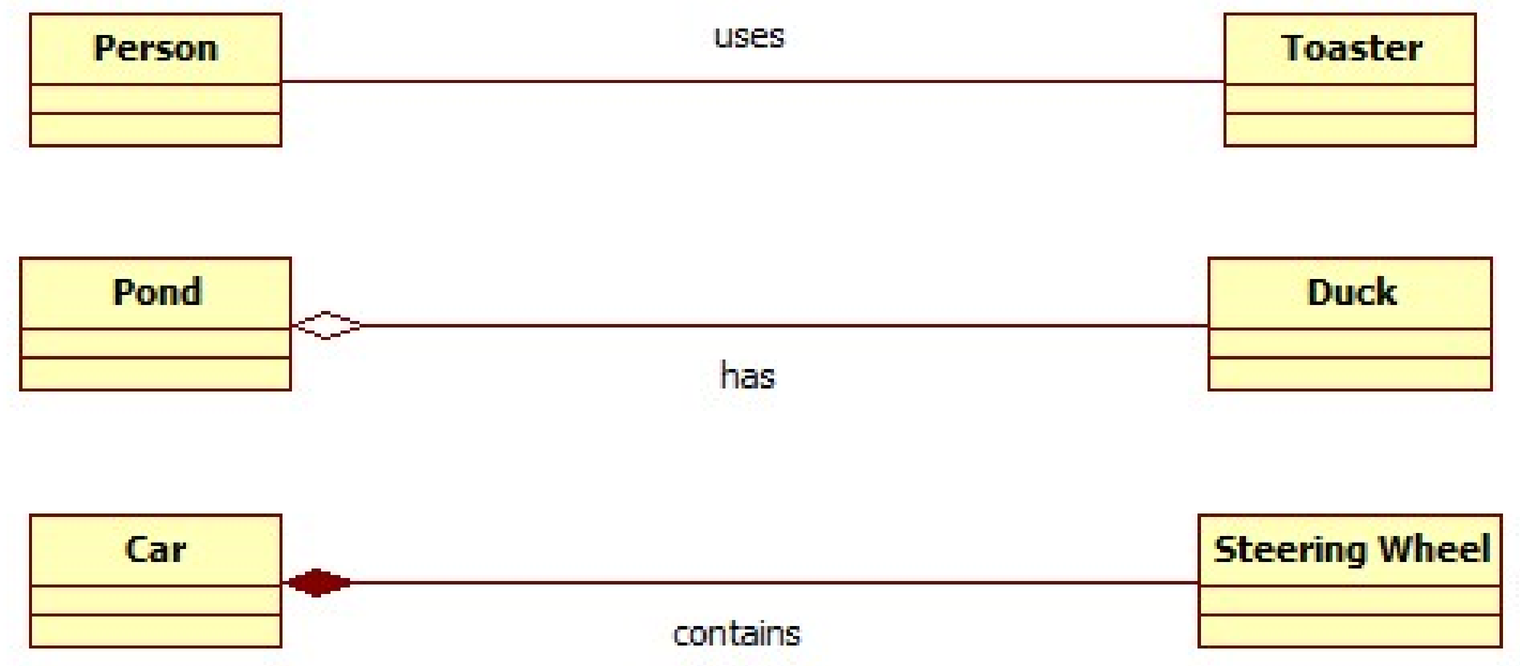
Class Diagrams – UML 2.0



Class Diagrams – Relationships

- Association
  - line connects
- Aggregation
  - line connects
  - diamond at the end where there is a “has a” relationship
- Composition
  - line connects
  - filled diamond at the end where there is a “contains” relationship
  - Stronger connection than aggregation (life cycle dependency)

Class Diagrams


Person
uses
Toaster
Pond
has
Duck
Car
contains
Steering Wheel