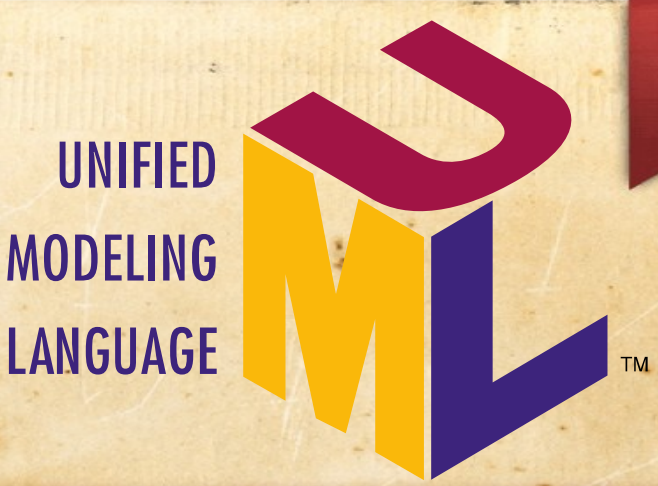
A-Level Information Technology



Topic 9: JavaScript

Unified Modeling Language (UML)

Class Diagrams

Lecture Contents

- References
- UML Basics
- UML Class Diagrams
  - Attributes
  - Operations
  - Visibility
  - Multiplicity
  - Relationships

References

- The most recent UML Specification from the *Object Management Group* (OMG)

https://www.omg.org/spec/UML/

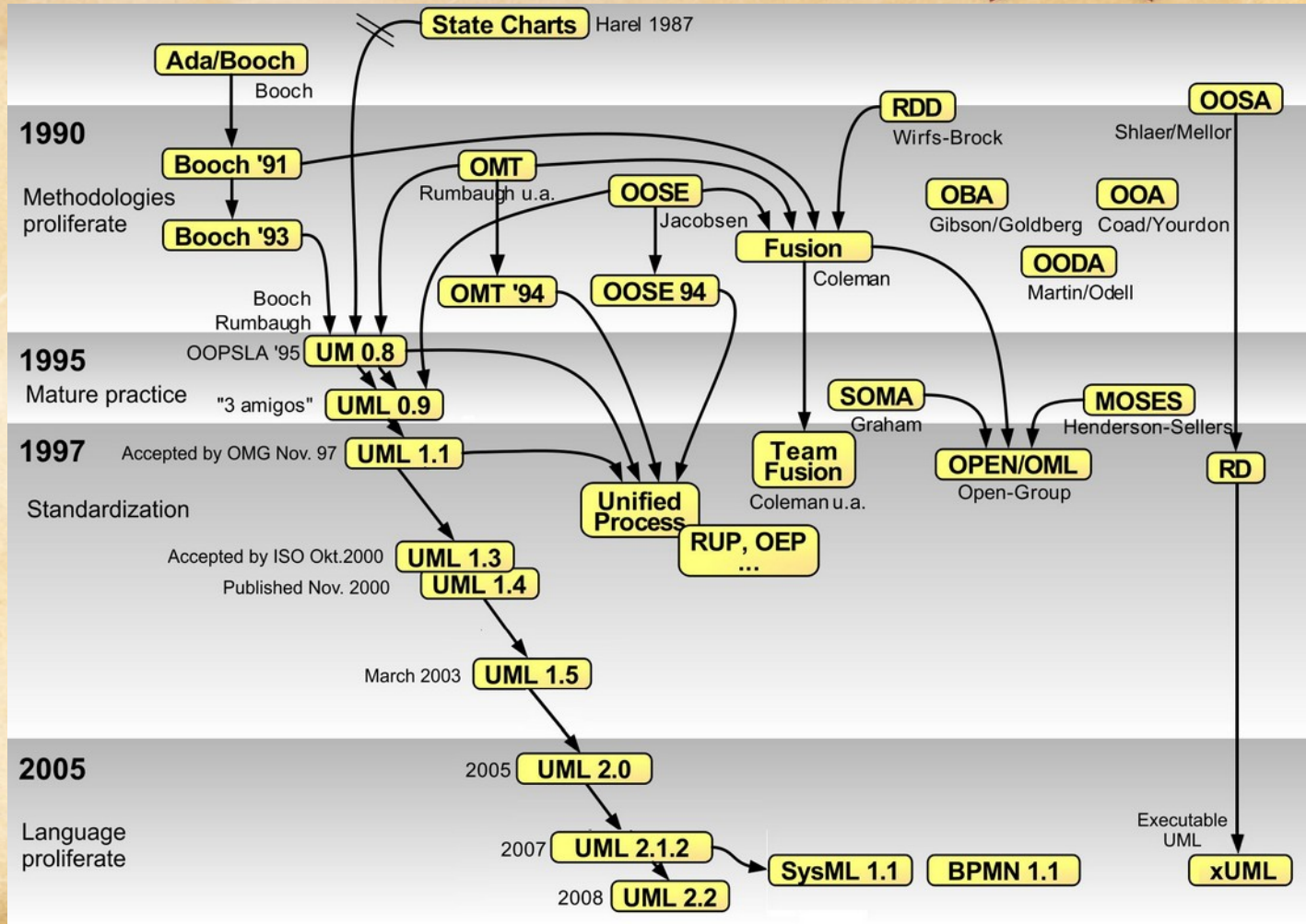
Unified Modeling Language

- By the Object Management Group (OMG)
  - Version 1.0 in January 1997

- A graphical way of describing software systems
  - Easy to read and understand the system prior to coding
  - Independent of programming language
  - Facilitates communication between developers

Unified Modeling Language

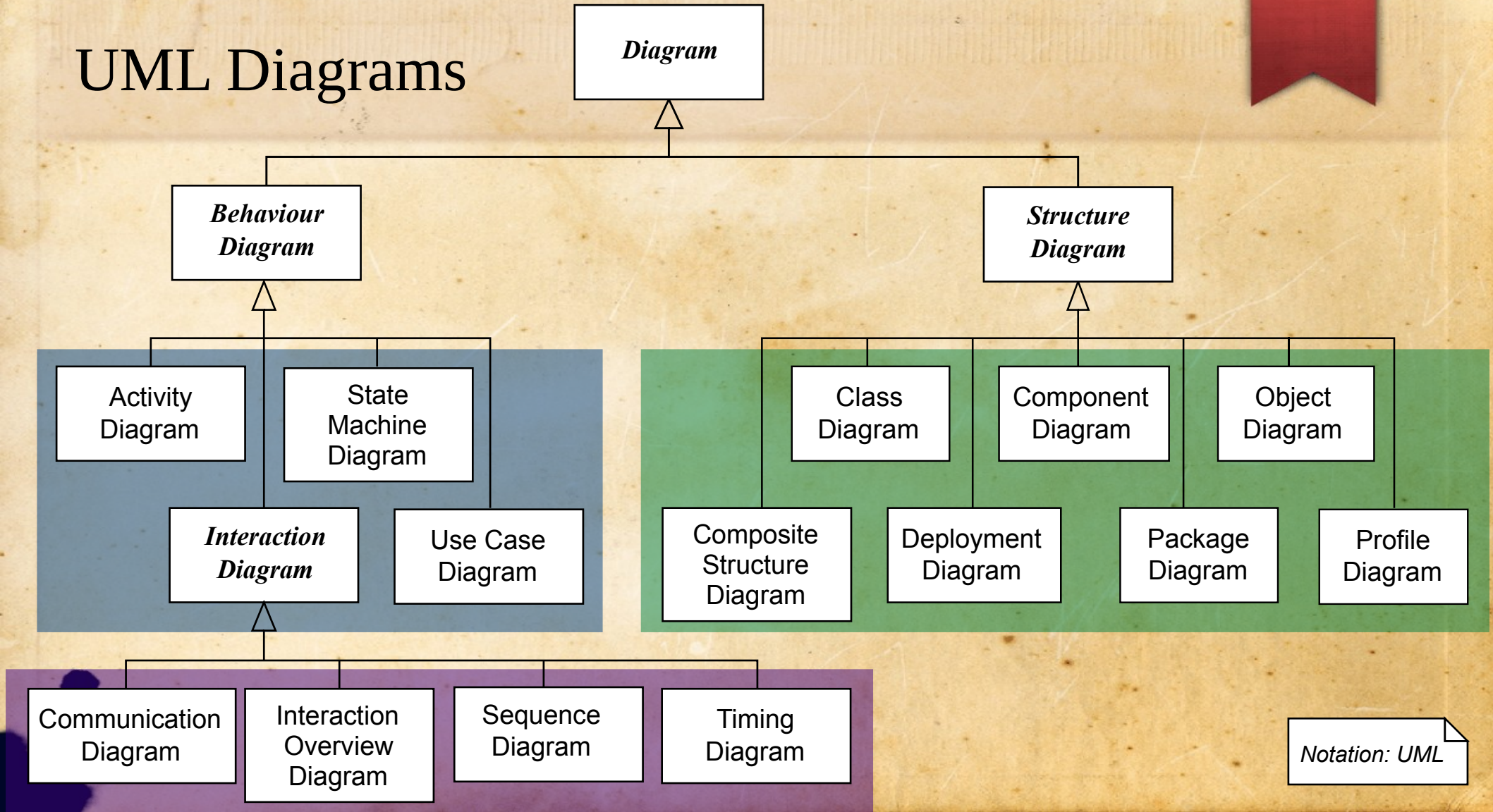
- Evolution



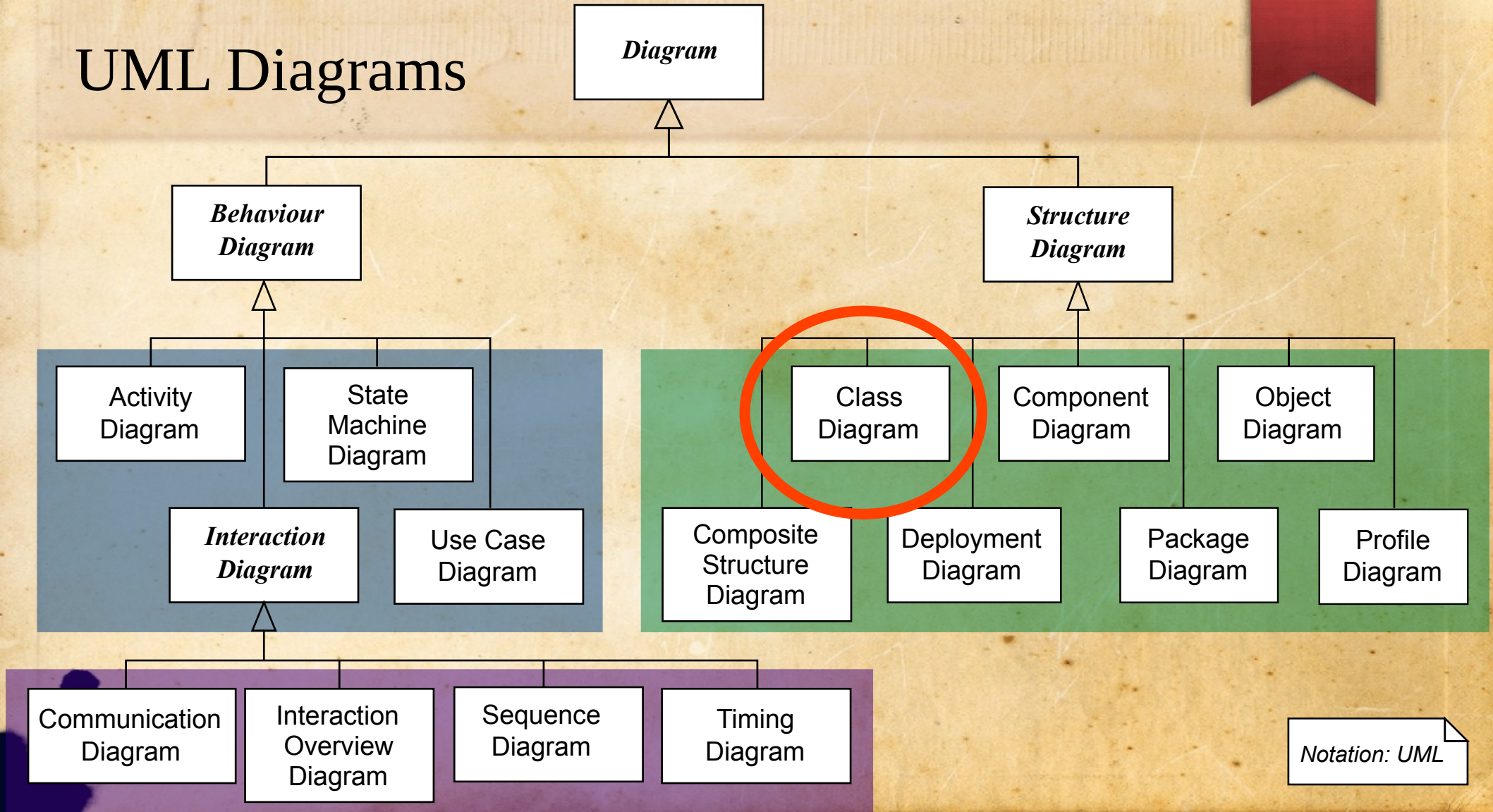
Unified Modeling Language

- UML Sketching
  - Communicate or rough out ideas; look at alternative approaches
- UML Blueprint
  - Detailed, including design decisions
- Model-Driven Architecture
  - Platform independent OR platform specific

UML Diagrams

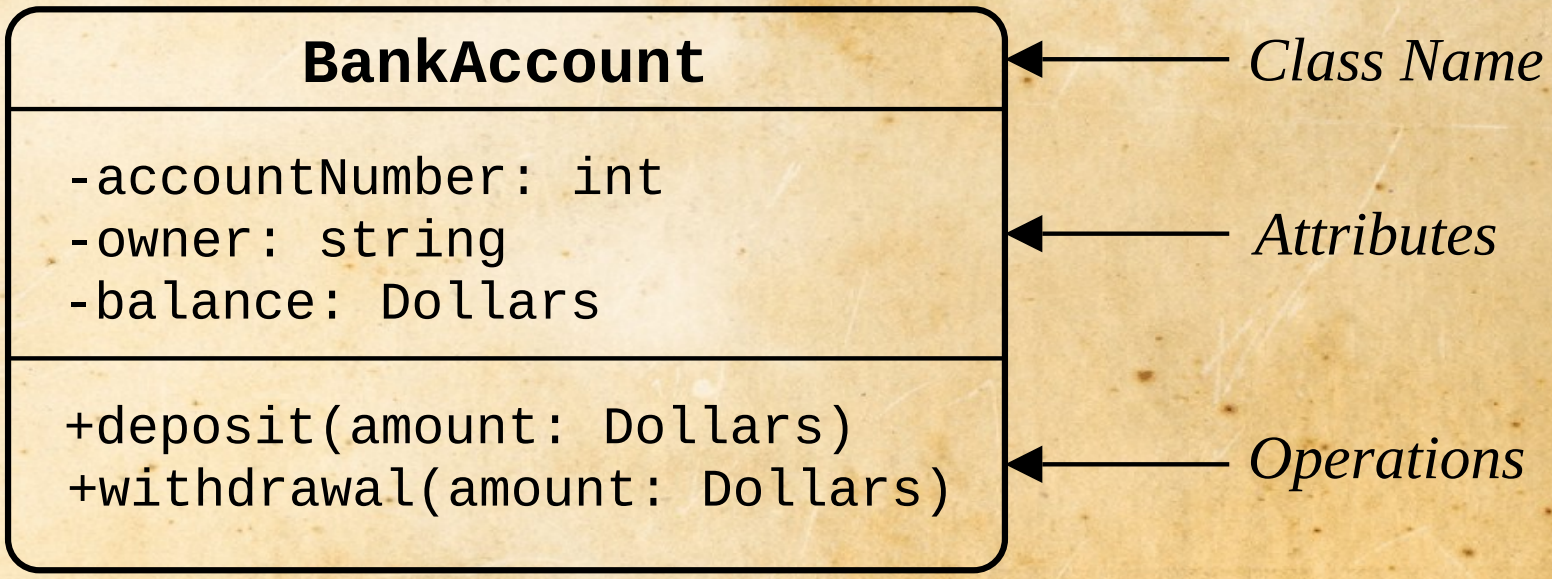


UML Diagrams



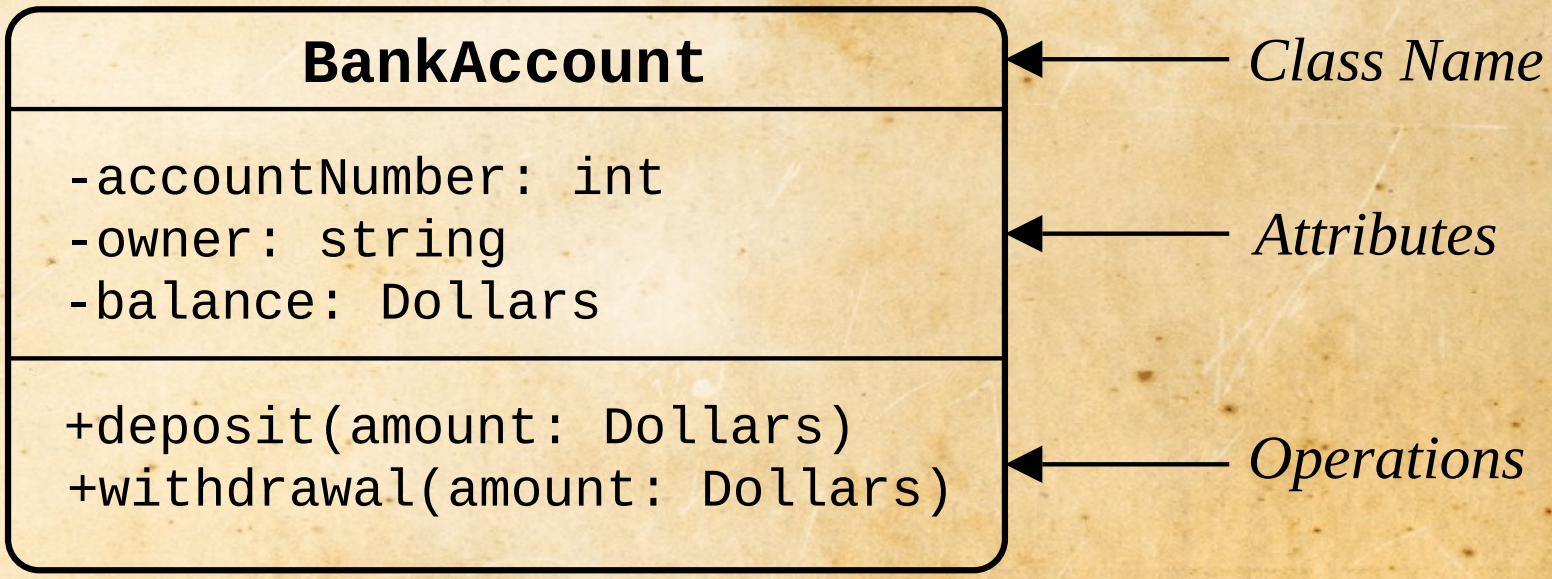
UML Class Diagram

- A static structure diagram showing the systems ***classes*** and their ***relationships***
- Classes are represented with boxes that have three compartments:



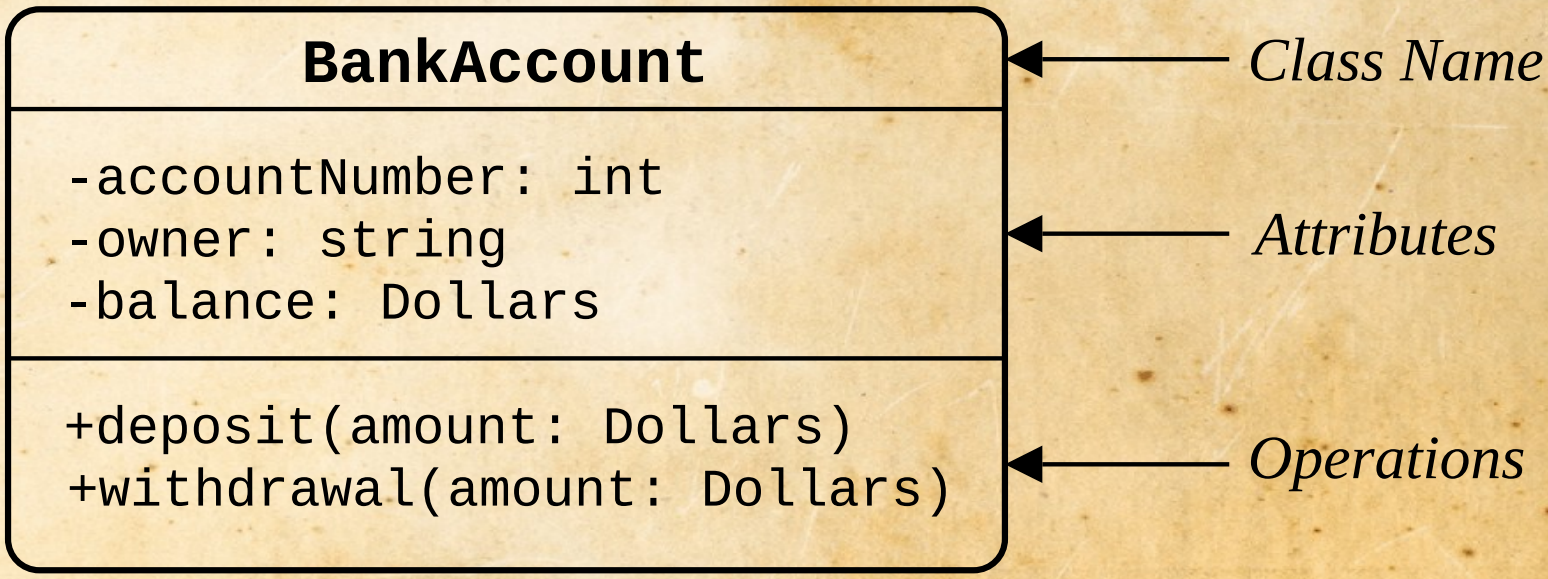
UML Class Diagram

- The class name is bold and centered; the first letter is capitalized
- Attributes are left-aligned; the first case is lower case
- Operations are left-aligned; the first letter is lower case



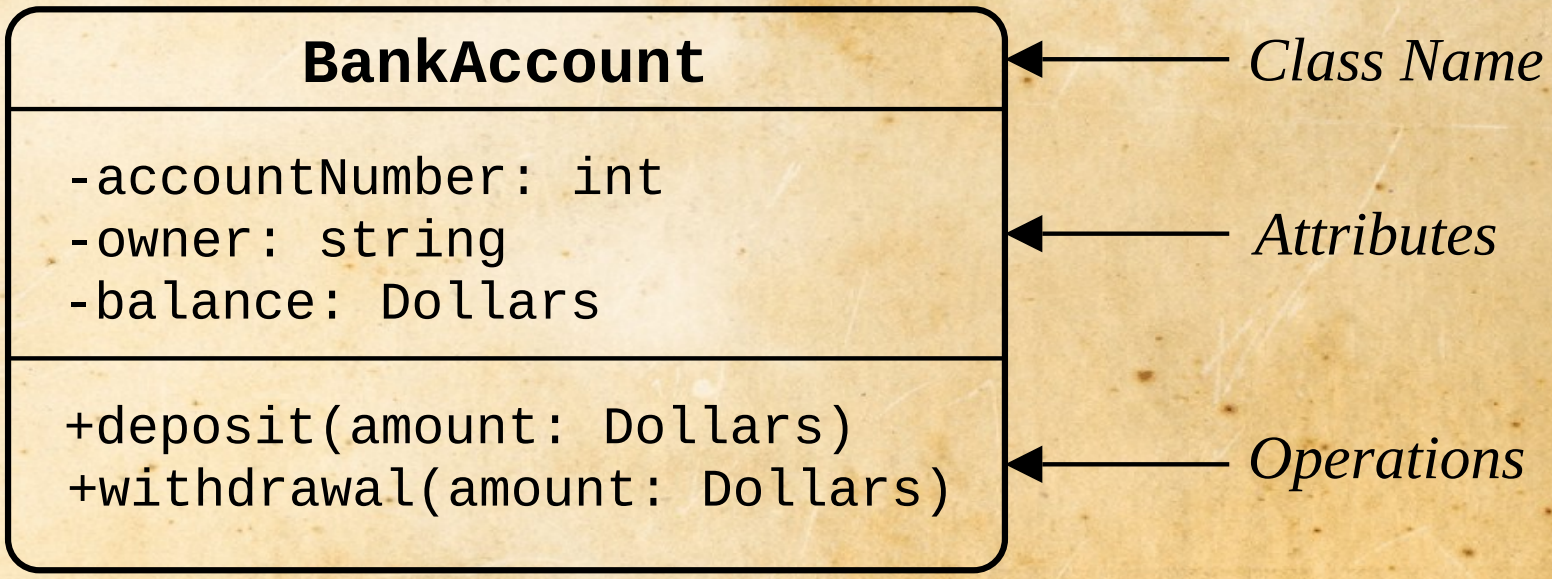
UML Class Diagram

- Classes are represented with boxes that have three compartments
  - ***Attributes*** in object oriented programming are called ***fields*** (basically *variables*)
  - ***Operations*** in object oriented programming are called ***methods***



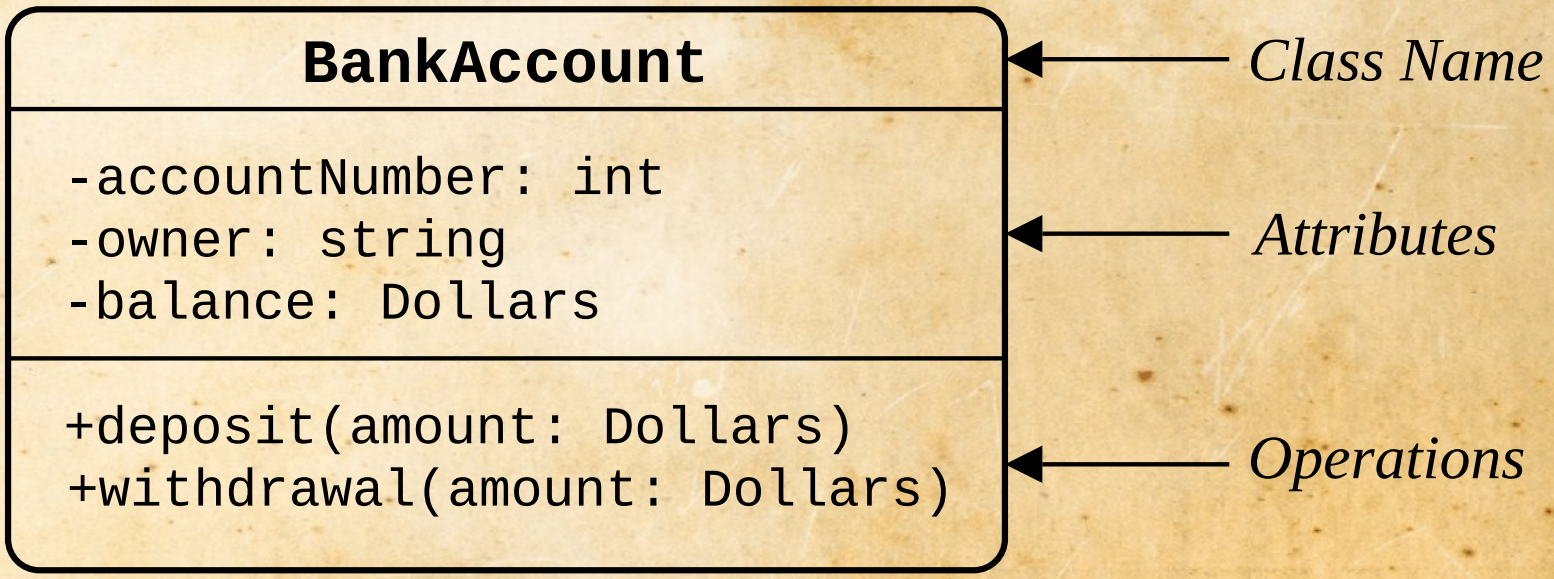
Attributes (*Fields*)

- Significant piece of data containing values that describe each instance of that class.
- Also known as: ***variables***, ***states***, or ***properties***



Operations (*Methods*)

- Specify behavioral features of a class.
  - What an object can do, or what can be done to it
- Also known as: ***behaviors*** or ***functions***



Visibility

- Sets the accessibility for ***field*** or ***method***.
  - \+ ***public*** – accessible to all
  - ~ ***package*** (***default***) – accessible by classes within the same package
  - # ***protected*** – accessible by the class and subclasses
  - – ***private*** – only accessible within the class

- ***Attributes*** (***fields***) generally should be ***private*** or ***protected***

Multiplicity of *Attributes* (*Fields*)

- Rules for attributes that will represent a group of objects
  - For a list with the number of elements in a range:

```
–parents: Person[0..2]   // between zero and two parents
```

  - For a list with an unknown number of elements:

```
–friends: Person[*]      // zero or more friends
–friends: Person[1..*]   // one or more friends
```

Multiplicity of *Attributes* (*Fields*)

- Rules for attributes that will represent a group of objects
  - For a set where elements are unique (a mathematical set)

```
–friends: Person[*] {unique}  //alternatively {notUnique}
```

  - If the list should be ordered

```
–siblings: Person[*] {ordered}
```

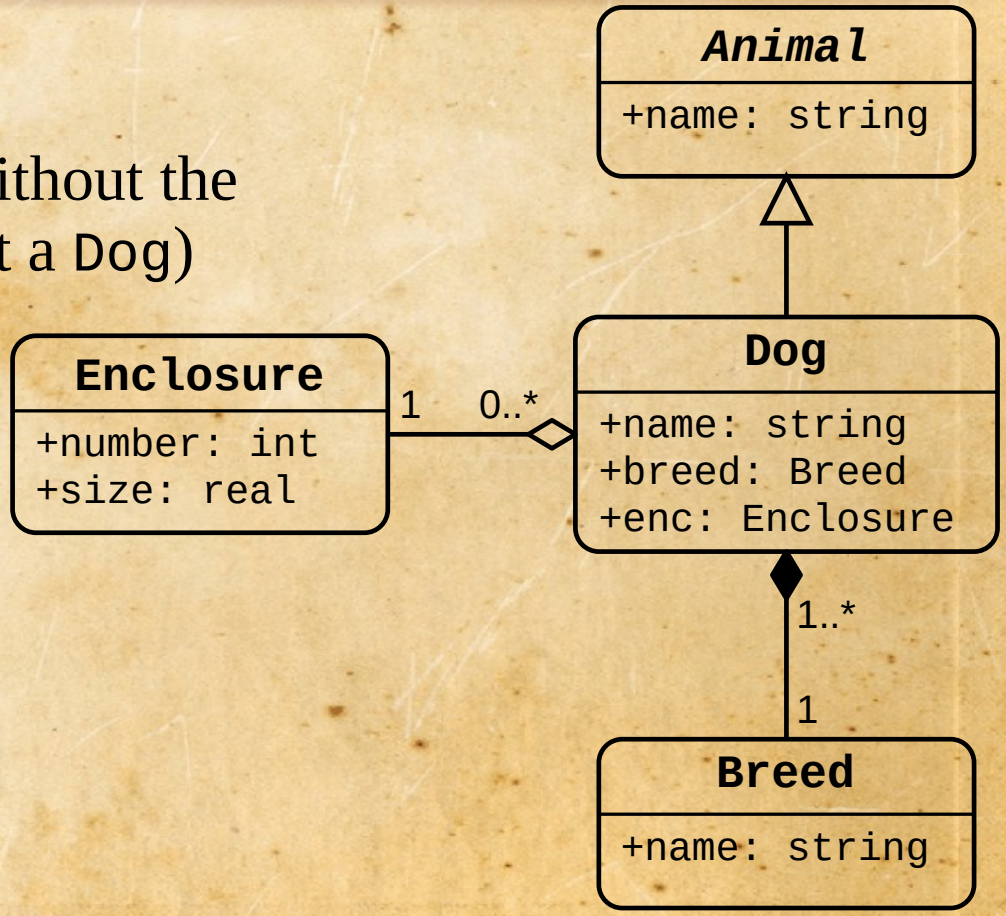
Scope of *Attributes* and *Operations*

- Two types of ***scope*** for members:
  - ***Class members***, represented by underlined names
    - One *attribute* is shared by all instances
    - *Operations* cannot affect the state of instance *attributes*
  - ***Instance members***, not underlined
    - *Attributes* may vary between instances
    - *Operations* may affect that instance's state (change the *attributes*)

- *Class members* are typically referred to as ***static*** in object-oriented programming languages.

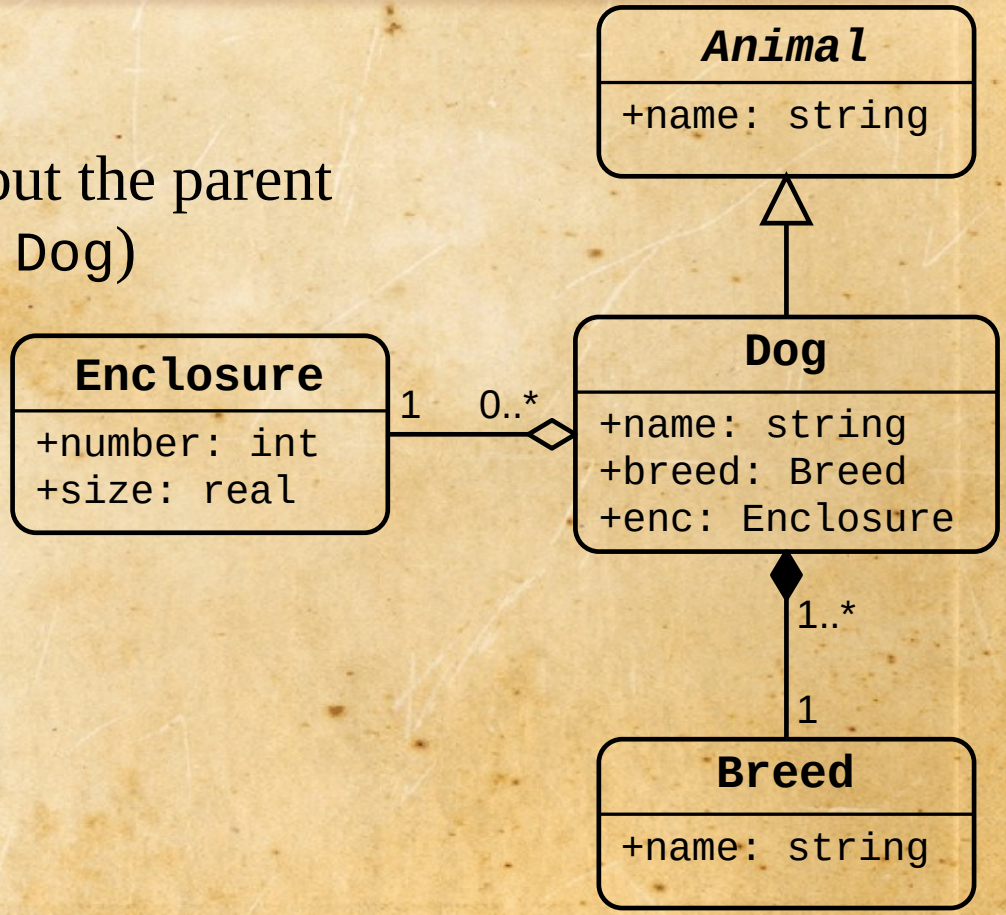
Relationship: ***Composition***

- When a class contains an object
- The contained class cannot exist without the parent (example: no `Breed` without a `Dog`)
- Shown with a connection with a closed diamond, ♦, on the containing class



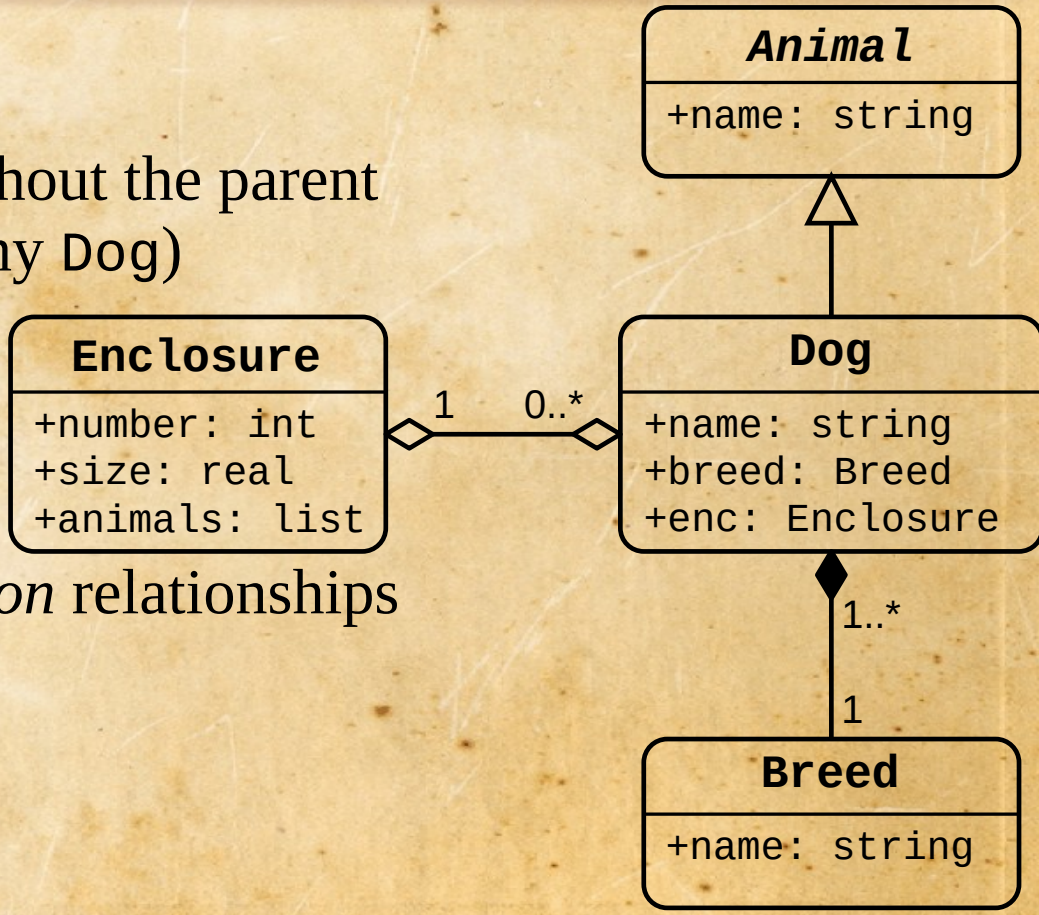
Relationship: ***Aggregation***

- When a class contains an object
- The contained class can exist without the parent (example: `Enclosure` without any `Dog`)
- Shown as a connection with an open diamond, ◊, on the containing class



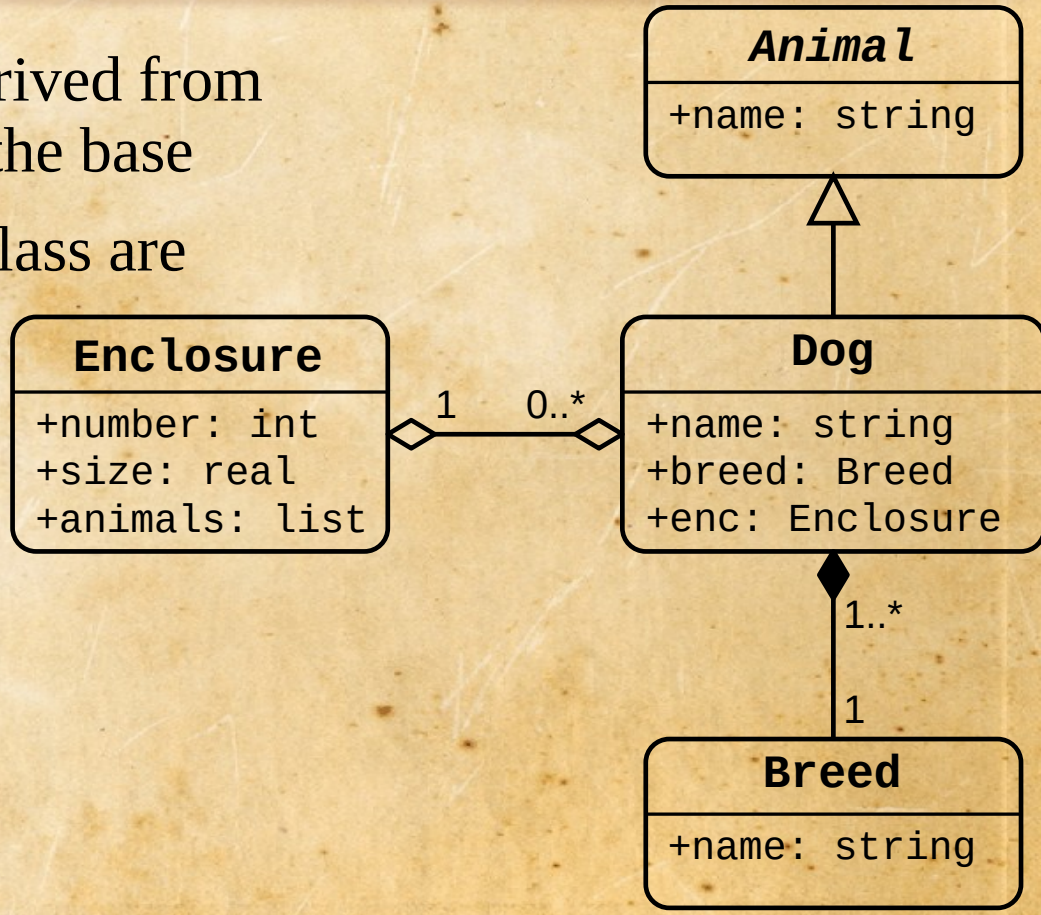
Relationship: ***Aggregation***

- When a class contains an object
- The contained class can exist without the parent (example: `Enclosure` without any `Dog`)
- Shown as a connection with an open diamond, ◊, on the containing class
- Note: *composition* and *aggregation* relationships may be bidirectional



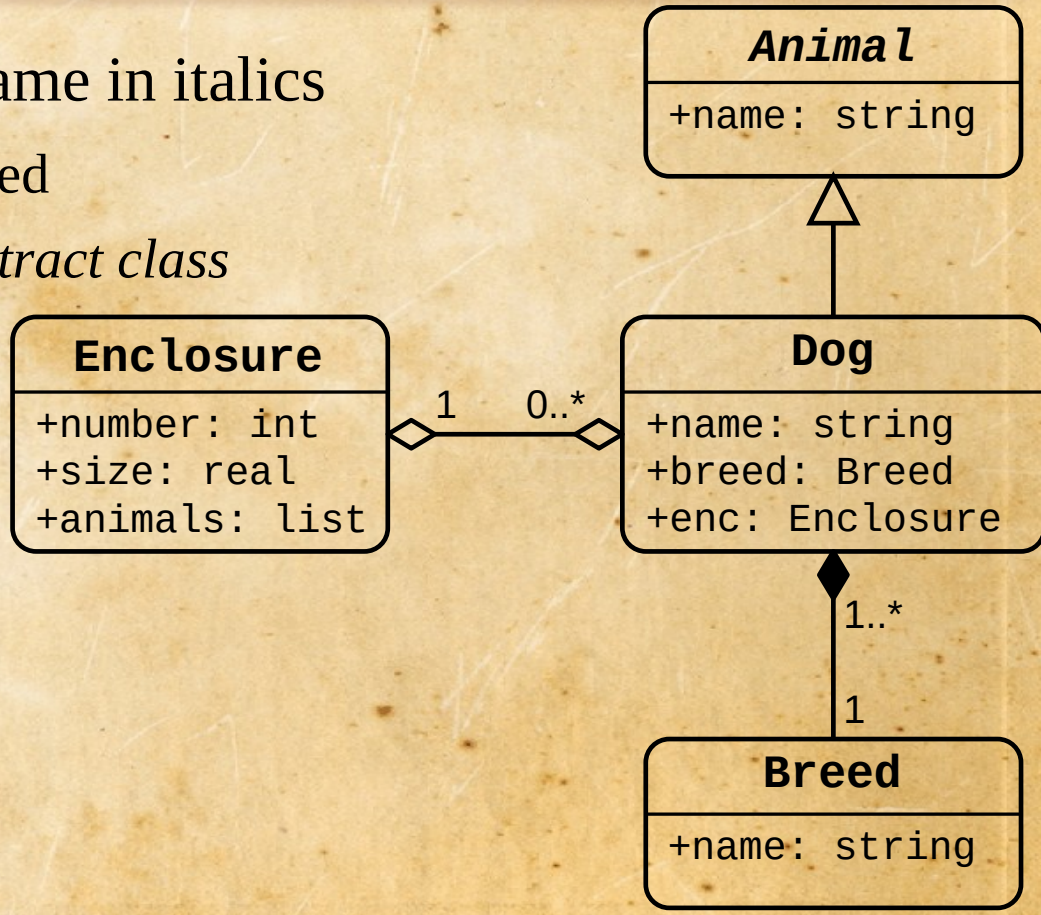
Relationship: ***Inheritance***

- When a class (the subclass) is derived from another class (the superclass) as the base
- Fields and methods of the superclass are inherited by the subclass, if *public*, *protected*, or *package*/*default*.
- Show as a connection with an arrow with an open arrow head ⊳



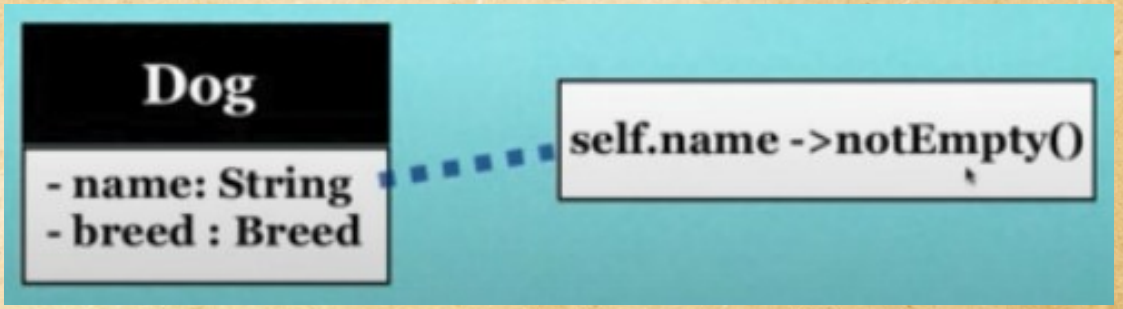
Abstract Classes

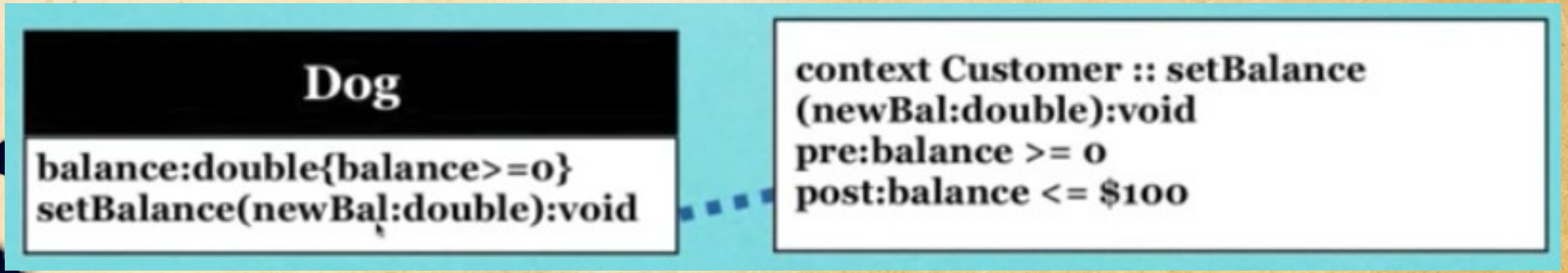
- An *abstract class* has the class name in italics
  - These classes cannot be instantiated
  - Here, the *`Animal`* class is an *abstract class*



Constraints

- Define rules for parts of classes.
  - Linked with a dotted line



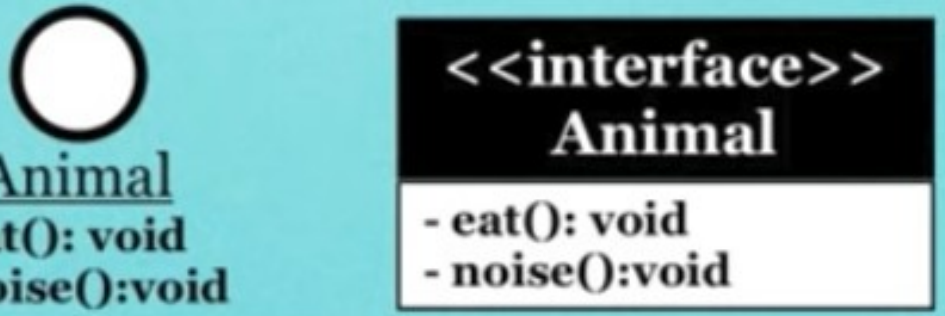


Object Constraint Language (OCL)

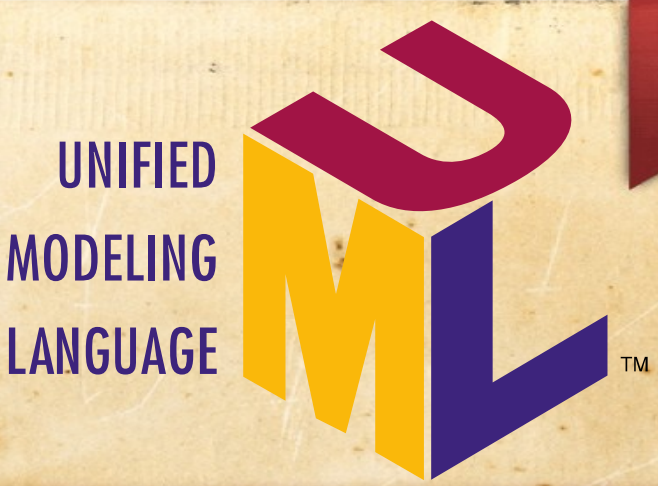
- Standard way of creating constraints
- Data Types: Boolean, Integer, Real, String
- Arithmetic: +, -, *, /, a->mod(b), abs(), min(), max()
- Comparison: <, >, <=, >=, =, <>
- Boolean: and, or, xor, not

Interface Class Diagrams

- Interfaces contain only abstract methods. Attributes are either static or constants
- Use either ball notation or the stereotype



A-Level Information Technology



Topic 9: JavaScript

Unified Modeling Language (UML)

Class Diagrams