Object-Oriented Design

Lecturer: Raman Ramsin

Lecture 6: Analysis Workflow

Analysis Workflow

- The aim of the analysis workflow is to produce the *Analysis Model*.
- The Analysis Model focuses on *what* the system needs to do, but leaves the details of *how* it will do it to the design workflow.
- The Analysis Model defines and models:
  - Analysis classes - which model key concepts in the problem domain.
  - Use case realizations - which illustrate how instances of analysis classes can interact to realize system behavior specified by a use case.

Analysis Workflow: Phases and Activities

- Most of the work in the analysis workflow occurs toward the end of the Inception phase and throughout the Elaboration phase.

- The analysis workflow consists of the following activities:
  - Architectural analysis
  - Analyze a use case
  - Analyze a class
  - Analyze a package

Analysis Modeling

- Rules of thumb:
  - expect about 50 to 100 analysis classes in the analysis model of an average system
  - only include classes that model the vocabulary of the problem domain
  - do *not* make implementation decisions
  - focus on classes and associations - minimize coupling
  - use inheritance where there is a natural hierarchy of abstractions
  - keep it simple

Objects

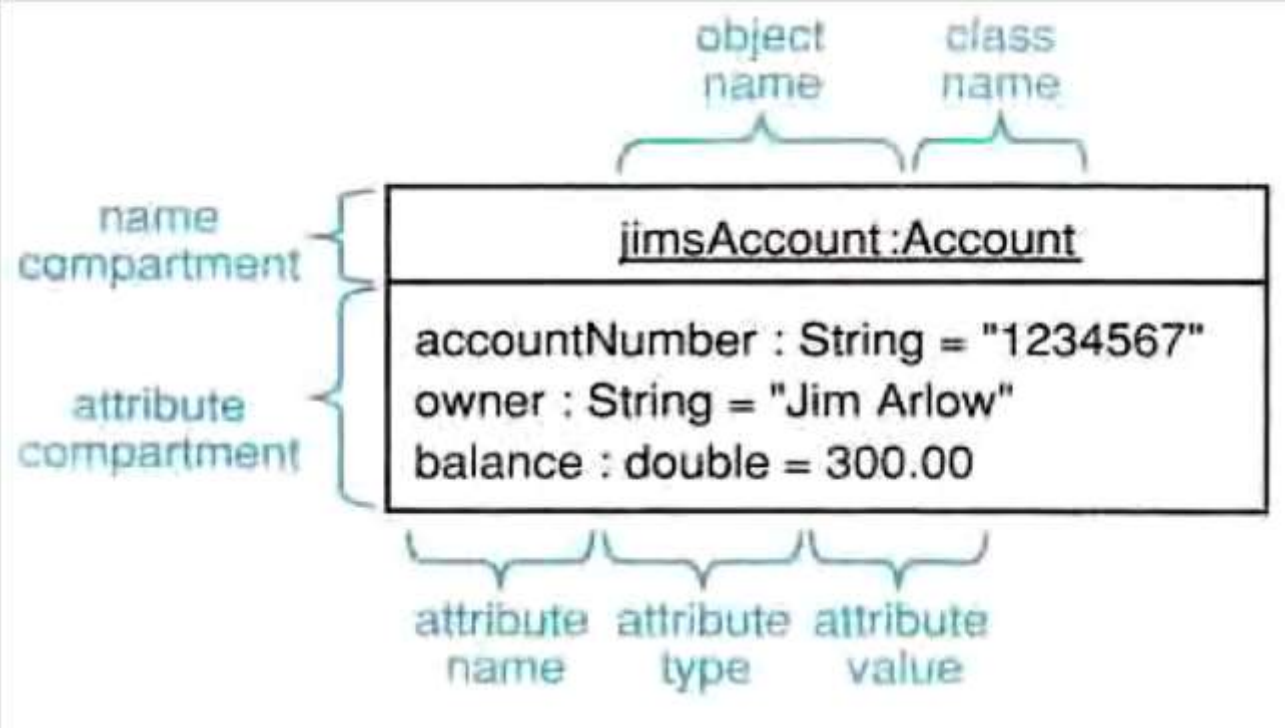
- Object: "A discrete entity with a well-defined boundary that encapsulates state and behavior; an instance of a class."
- Objects are cohesive units that combine data and function.
- Encapsulation - the data inside an object is hidden and can only be manipulated by invoking one of the object's functions.
  - *operations* are specifications for object functions created in analysis
  - *methods* are implementations for object functions created in implementation

Objects: Features

- Every object has the following features:
  - *Identity* - its unique existence - you use object references to uniquely refer to specific objects.
  - *State* - a meaningful set of attribute values and relationships for the object at a point in time.
    - Only those sets of attribute values and relationships that constitute a semantically important distinction from other possible sets constitute a state. For example, BankAccount object - balance< 0, state = Overdrawn; balance>0, state = InCredit.
    - State transition - the movement of an object from one meaningful state to another.
  - *Behavior* - services that the object offers to other objects:
    - modeled as a set of operations;
    - invoking operations *may* generate a state transition.

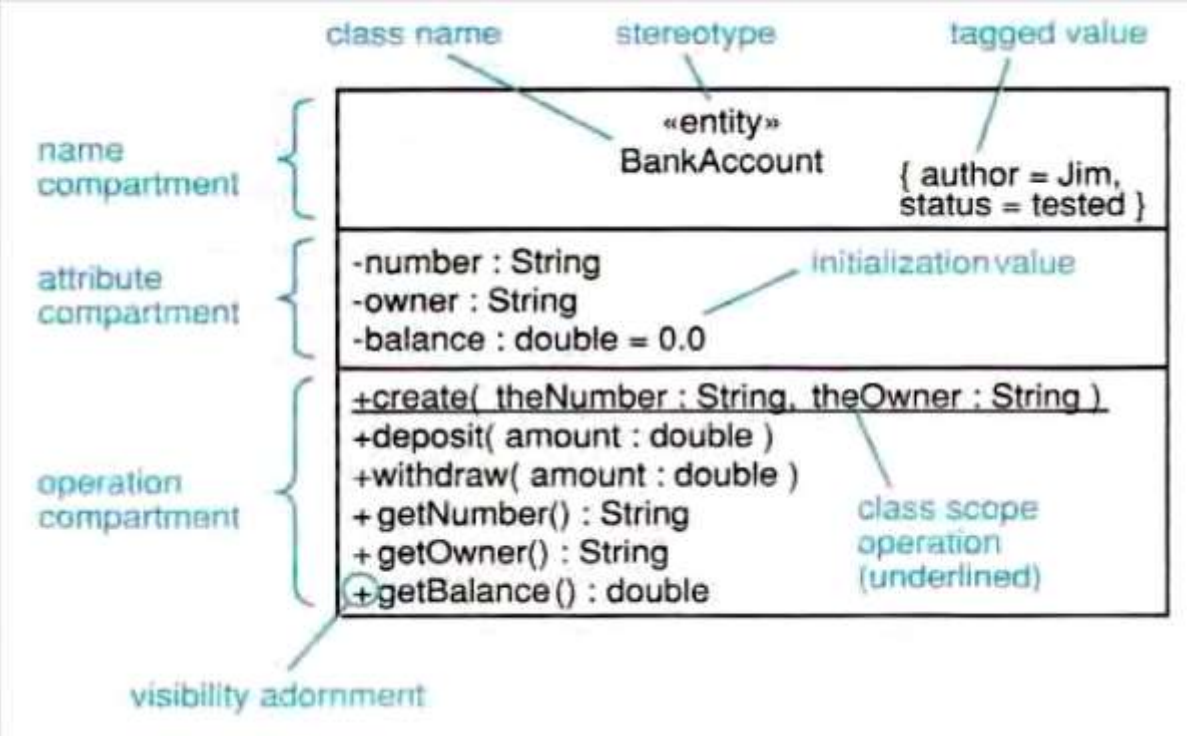
UML Object Notation

- No special symbols, punctuation marks, or abbreviations in *object/class* names.



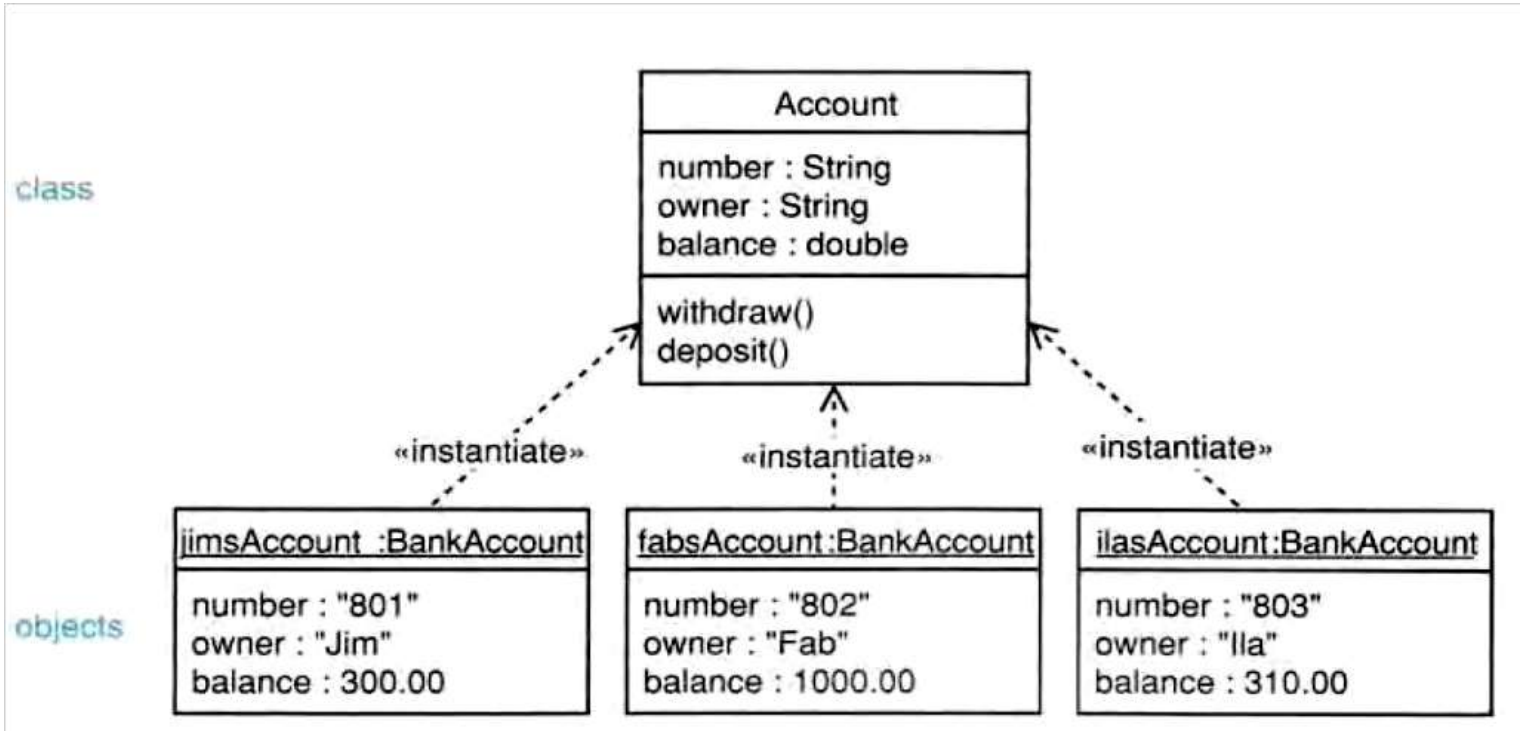
Classes and UML Class Notation

- Class: "The descriptor for a set of objects that share the same attributes, operations, methods, relationships, and behavior."



Instantiate Relationship

- You can show the instantiate relationship between a class and one of its objects by using a dependency stereotyped as «instantiate»:
  - a *dependency* relationship indicates that a change to the supplier affects the client.



Visibility

Adornment	Visibility name	Semantics
+	Public visibility	Any element that can access the class can access any of its features with public visibility
–	Private visibility	Only operations within the class can access features with private visibility
#	Protected visibility	Only operations within the class, or within children of the class, can access features with protected visibility
~	Package visibility	Any element that is in the same package as the class, or in a nested subpackage, can access any of its features with package visibility

Type

	Primitive type	Semantics
UML	Integer	A whole number
	UnlimitedNatural	A whole number >= 0 Infinity is shown as *
	Boolean	Can take the value true or false
	String	A sequence of characters String literals are quoted, e.g., "Jim"
OCL	Real	A floating point number

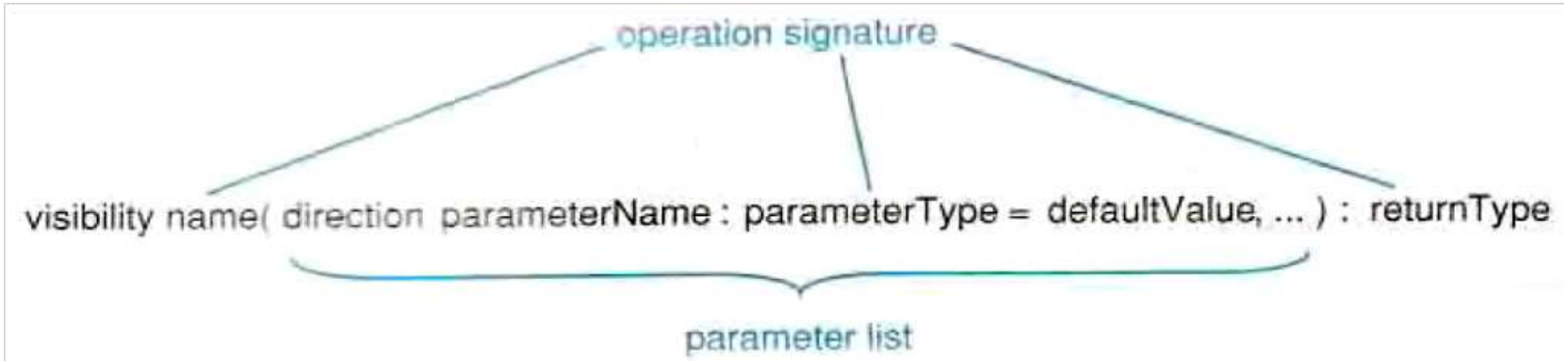
- The Object Constraint Language (OCL) is a formal language for expressing constraints in UML models.
- OCL defines standard operations for the UML primitive types (except UnlimitedNatural) and adds a new type called Real.

Attributes

visibility name : type [multiplicity] = initialValue

mandatory

Operations: Signatures

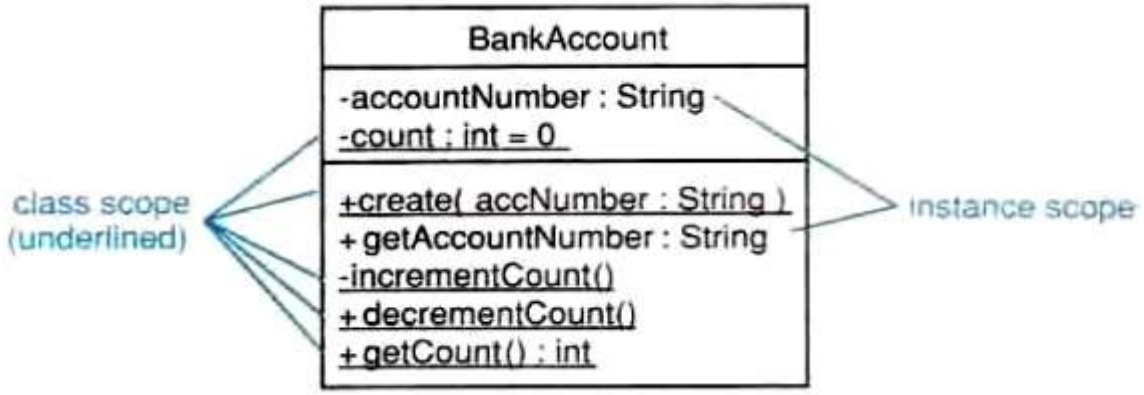


Operation (in p1:Integer, inout p2:Integer, out P3:Integer, return P4:Integer, return P5:Integer)

Canvas
drawCircle(origin: Point = Point(0, 0), radius : Integer) drawSquare(origin: Point = Point(0, 0), size : Dimension)

Scope

- Instance scope attributes and operations belong to or operate on specific objects:
  - instance scope operations can access instance-scope and class-scope operations/attributes;
- Class scope attributes and operations belong to or operate on the whole class of objects:
  - class scope operations can only access other class scope operations.



Reference

- Arlow, J., Neustadt, I., *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*, 2nd Ed. Addison-Wesley, 2005.