Class diagrams

• Used to describe the internal structure of the system.
• Also used to describe the application domain.
• They describe the system in terms of
  ✦ Classes, an abstract representation of a set of objects
  ✦ Attributes, properties of the objects in a class
  ✦ Operations that can be performed on objects in a class
  ✦ Associations that can occur between objects in various classes

Class Diagrams: details

• Can represent classes OR objects
• Classes (and objects) are boxes composed of three compartments:
  ✦ Top compartment: name
  ✦ Center compartment: attributes (may be omitted for simplicity/delay)
  ✦ Bottom compartment: operations (may be omitted for simplicity/delay)
• Conventions:
  ✦ Object names (when used) are underlined to indicate they are instances
  ✦ Class names start with uppercase letter
  ✦ Named objects start with lowercase

Class diagram for a simple watch

• Boxes are classes
• Lines show associations (between objects)
• Numbers show how many objects must be associated
**UML class diagram:** classes that participate in the ReportEmergency use case.

**UML object diagram:** objects that participate in the warehouseOnFire scenario.

**Associations and links**

- A link represents a connection between two objects.
- Associations are relationships between classes and represent the fact that links may (or do) exist between object instances.
  - Links and associations are noted with a line between the boxes.
- Associations can be symmetrical (bidirectional) or asymmetrical (unidirectional).
  - Unidirectional association is indicated by using a line with an arrow
  - The arrow indicated in which direction navigation is supported.
  - If the line has no arrows, it's assumed to be bidirectional.

**Example of a unidirectional association**

This system supports navigation from the Polygon to the Point, but not vice versa. Given a specific Polygon, it is possible to query all Points that make up the Polygon. But a given Point does not know which Polygon(s) it belongs to.

*Note: the diagram in the book is wrong*
Association class

- **Association class**: an association with attributes and/or operations
- Depicted by a class symbol that contains the attributes and operations and is connected to the association symbol with a dashed line.

Roles

- Each end of an association can be labeled by a role.
- Allows us to distinguish among the multiple associations originating from a class.
  - An employee can belong to a department and be the head of the department.
- Roles clarify the purpose of the association.

Examples of multiplicity

- Multiplicity: a set of integers labeling one end of an association
- Indicates how many links can originate from an instance of the class at the other end of the association.
- * is shorthand for 0..n, called “many”

- Most associations belong to one of these three types:
  - A **one-to-one** association has a multiplicity 1 on each end.
  - A **one-to-many** association has a multiplicity 1 on one end and 0..n (*) or 1..n on the other.
  - A **many-to-many** association has a multiplicity 0..n or 1..n on both ends.

Examples of multiplicity:

- **PoliceOfficer** → **1**: **BadgeNumber**
- **CourseSection** → ***: Student**
- **FieldOfficer** → **1**: **author**: **report**: **IncidentReport**

A CourseSection contains many Students
A Student is enrolled in many CourseSections

How many reports can a FieldOfficer write? How many authors of a report can there be?
Aggregation and Composition

• Aggregation is a kind of association that specifies a whole/part relationship between the aggregate (whole) and component part.
  ✦ Useful to denote hierarchical relationships (directory contains files)
  ✦ Specified with an open diamond on the aggregate (whole) side.
• Composition is a special case of aggregation where the composite object has sole responsibility for the life cycle of the component parts.
  ✦ The composite is responsible for the creation and destruction of the component parts.
  ✦ An object may be part of only one composite.
  ✦ Specified with a closed diamond on the composite (whole) side.

Examples of aggregations

Could any of these be composites?
• can a County belong to more than one State?
• can a County exist without a State?

Inheritance

• Inheritance is a relationship between a base class and a more refined class.
  ✦ The refined class has attributes and operations of its own, as well as the attributes and operations of the base class (it inherits them).

Inheritance example

Note the operations
Overridden operations are not shown
Example of a hierarchical file system

When and how to use Class Diagrams

- All the time.
- Try to keep them simple, don’t use unnecessary notation.
  - Especially if you are using them to model the application domain.
  - If you want to specify the implementation very specifically, you will use more of the notation.
- Don’t draw models for every part of the program (at least not all in great detail)
- Focus first on concepts, then add detail as the design process continues.

Class diagram with an interface

Sequence Diagrams

- Represent the dynamic behavior of the system
- Describe patterns of communication among a set of interacting objects.
- An object interacts with another object by sending messages.
  - The message must be an operation of the receiving object.
- Arguments may be passed along with a message
  - they correspond to the parameters of the receiver's operation.

A Sequence diagram should be consistent with any existing class diagrams (this is where the messages (operations) are specified).
Sequence diagram for a simple watch

- Actor and objects (not classes) across the top
- Vertical lines are time lines of the objects
- Labeled arrows are messages sent to another object

Sequence Diagrams

- Objects across the top (horizontal axis)
- Time goes down (vertical axis)
- Diagram components:
  - Solid horizontal arrows: messages
  - Labels on arrows: message names (arguments optional)
  - Return arrows (at end of operation, going back) are optional
  - Vertical rectangle: an activation (lifetime) of an operation
  - Interactions involving actors may not be operations.

Sequence diagrams: setting the time on 2Bwatch.

Sequence Diagrams: iteration and branching

- Notation for iteration (loops)
  - Repeated message has an asterisk (see next slide: *op3()*)
  - Optional: indicate basis of iteration in brackets: *[for all order lines] op3()*
- Notation for conditions (alternatives)
  - Conditional message is marked with a guard (see next slide) OR
  - Alternative message(s) in a partitioned box labeled "alt"
  - each partition has a guarded message (see next next slide)
  - May be easier to draw a separate diagram for each alternative
- If you really want to model control flow, you should use an activity diagram instead.
Conditions and iterators in sequence diagrams.

Arrows with common start point are mutually exclusive alternatives.

Using alt box to show branching.

Sequence Diagrams: additional notations

- Self-call (operation calls itself)
  - Message arrow back to original activation:

- Creating new instances:
  - "new Class()" message points to object's box:

25

26

27

28
When and how to use Sequence Diagrams

- When you want to look at the behavior of several objects within a single use case.
- When the order of the method calls in the code seems confusing.
- When you are trying to determine which class should contain a given method.
  - ✦ to uncover the responsibilities of the classes in the class diagrams
  - ✦ to discover even new classes
- During Object-Oriented Design, sequence diagrams and the class diagram are often developed in tandem.