CS 213 – Software Methodology
Spring 2023

Sesh Venugopal

Mar 1

UML Class Diagram - I

UML Diagrams

- UML stands for Unified Modeling Language, which is a (mainly) graphical notation used to express object-oriented design

- There are three kinds of UML diagrams that are used in practice:

  - **Class diagram**, used to show classes and the relationships between them

  - **Sequence diagram**, used to show sequences of activity when methods are invoked on classes

  - **State diagram**, used to represent state-based designs

Class Diagram

- A class diagram shows classes and the relationships between them
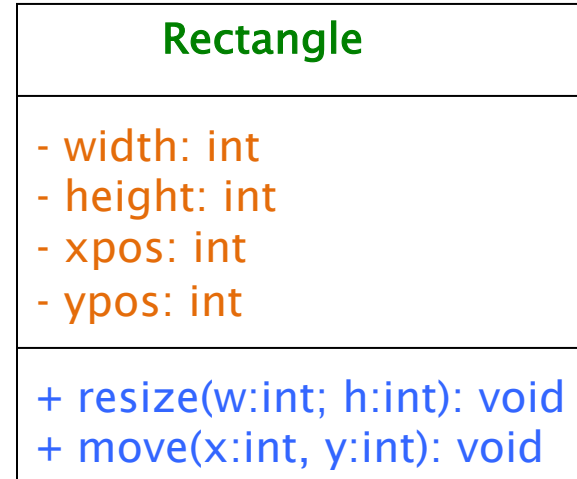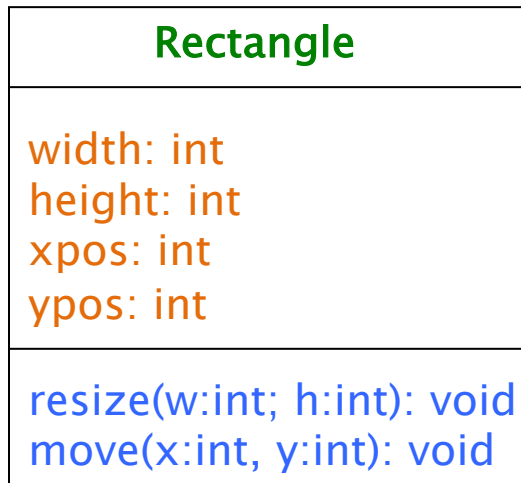
- The simplest way to draw a class is like this:

Rectangle

- Details may be added to the class like this:

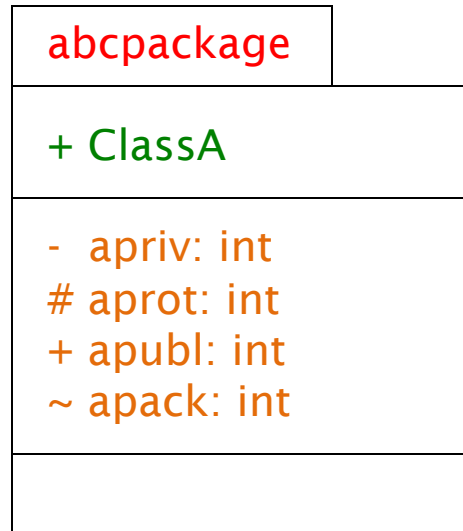| Rectangle |
|-----------|
| width
height
xpos
ypos |
| resize( )
move( ) |
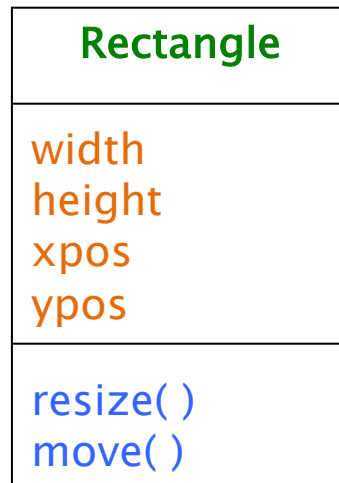
Attributes appear below the class name, and operations (methods) appear below the attributes
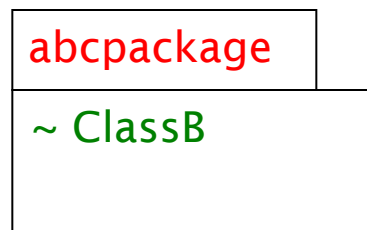
Class Diagram with Attributes and Methods

- An even greater level of detail can be added by specifying the type of each attribute, as well as type of each parameter and return type for each method:

- And the access level (private, public, etc.) for each member:

| Rectangle |
|---|
| width: int
height: int
xpos: int
ypos: int |
| resize(w:int; h:int): void
move(x:int, y:int): void |

Rectangle
- width: int - height: int - xpos: int - ypos: int
+ resize(w:int; h:int): void + move(x:int, y:int): void

UML Notation for Access Levels

abcpackage

+ ClassA

- apriv: int
aprot: int
+ apubl: int
~ apack: int

+ : public

: protected

~ : package

− : private

abcpackage

~ ClassB

Class Diagram: Relationships

- Relationships between classes are essentially of three kinds: *generalization/specialization* (super/sub), *association*, and *dependency*
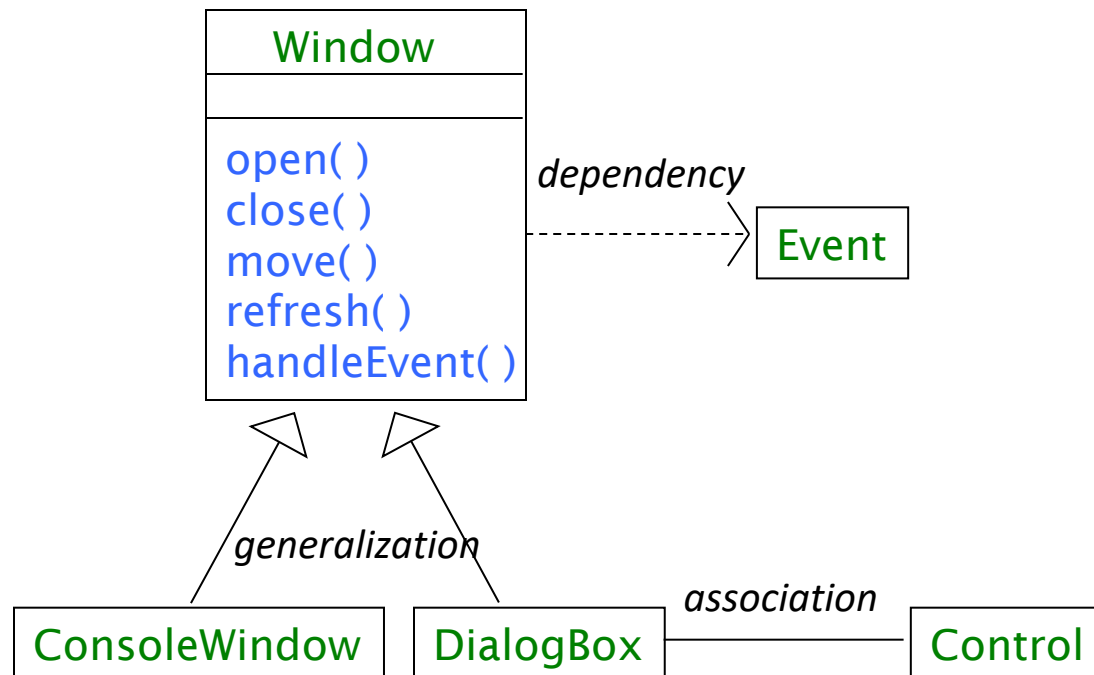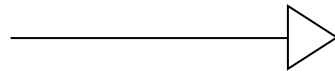

This example from "The Unified Modeling Language User's Guide" by Booch, Rumbaugh, Jacobson
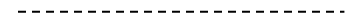
Class Diagram: Relationships

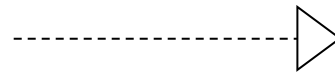- Relationships between classes are represented by various kinds of lines
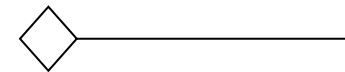
Inheritance

Association Class
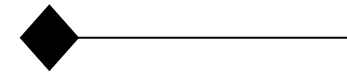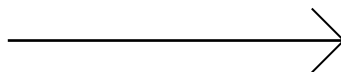
Interface Implementation
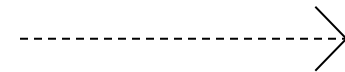
Aggregation

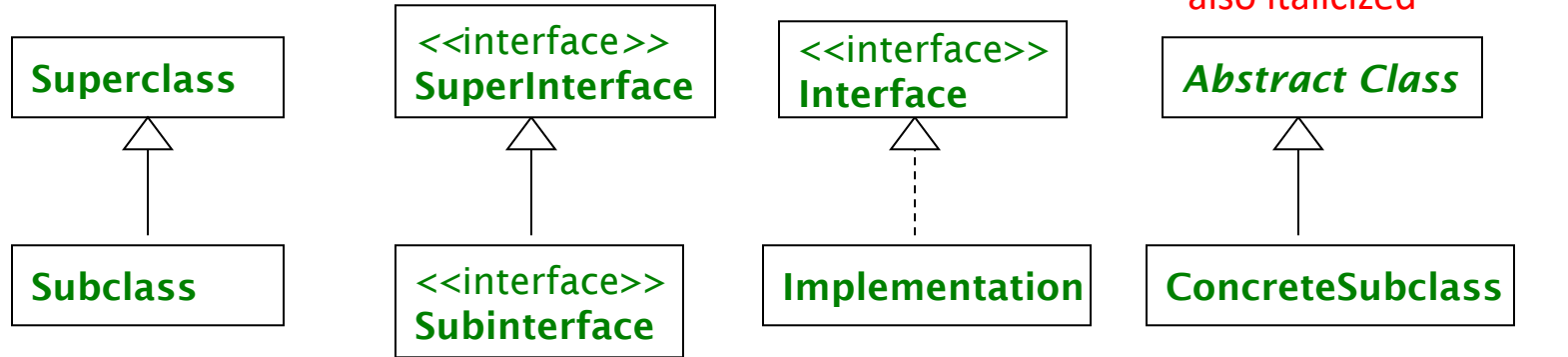Bi-directional Association

Composition

Uni-directional Association

Dependency

Generalization (and Interface Implementation)

- Notation

abstract methods are also italicized

Superclass

△

Subclass

<<interface>> SuperInterface

△

<<interface>> Subinterface

<<interface>> Interface

△ (dashed)

Implementation

Abstract Class

△

ConcreteSubclass

- Examples

PrintBook

- - ▷

<<interface>> Book

△

PDFBook

- - ▷

<<interface>> eBook

Shape

△

Polygon

△   △

Rectangle

Triangle