MIAMI UNIVERSITY

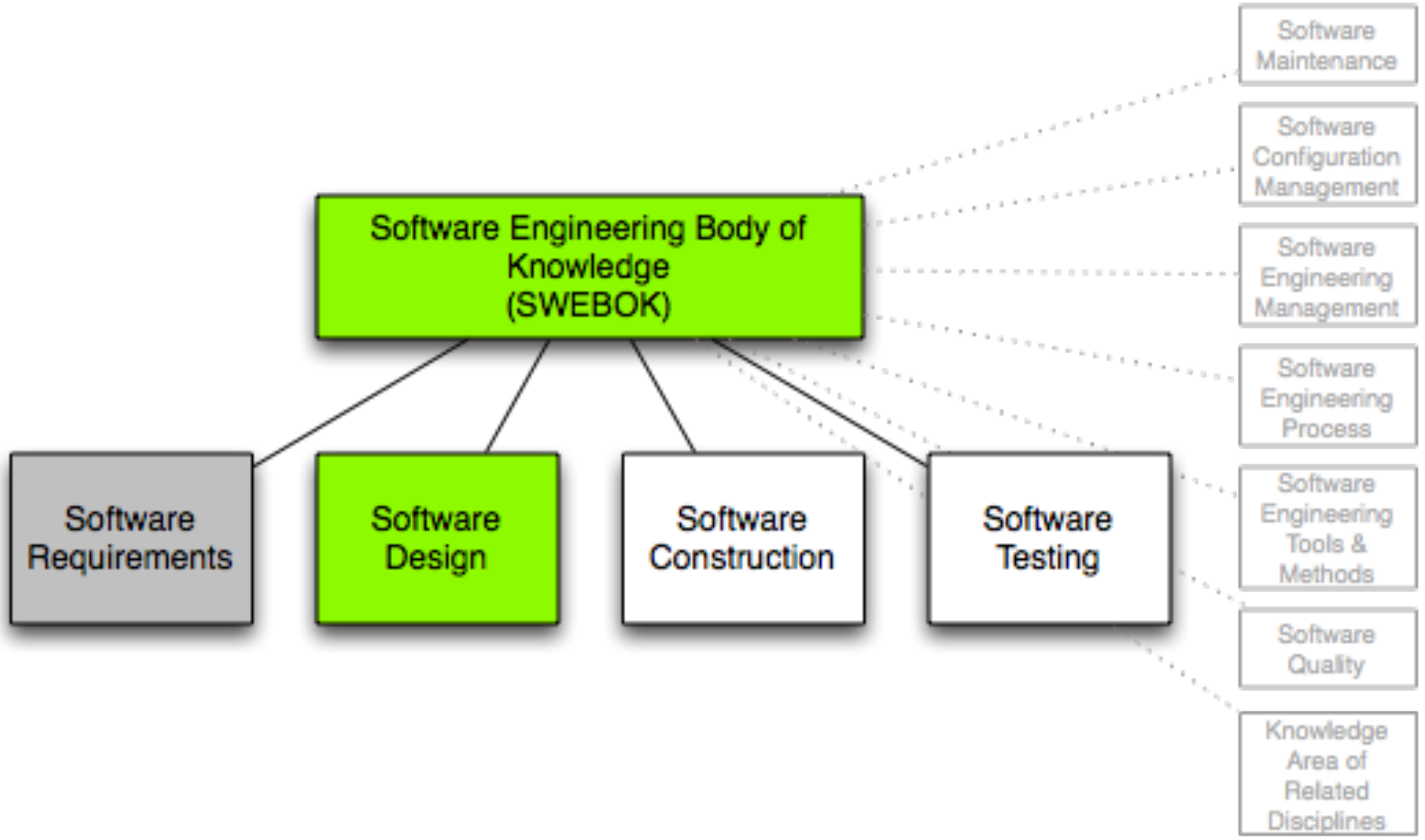
SOFTWARE DESIGN - II

Content in part from Chapter 5 of "Head First Software Development", Pilone et al.

Miami University Software Technology & Analysis Group (MUSTANG)
Computer Science & Software Engineering
Miami University, Oxford, Ohio, USA

AGENDA

- Review
- Object-Oriented Languages
- UML Class Diagrams
  - Attributes
  - Operations
- UML Class Diagram Activity

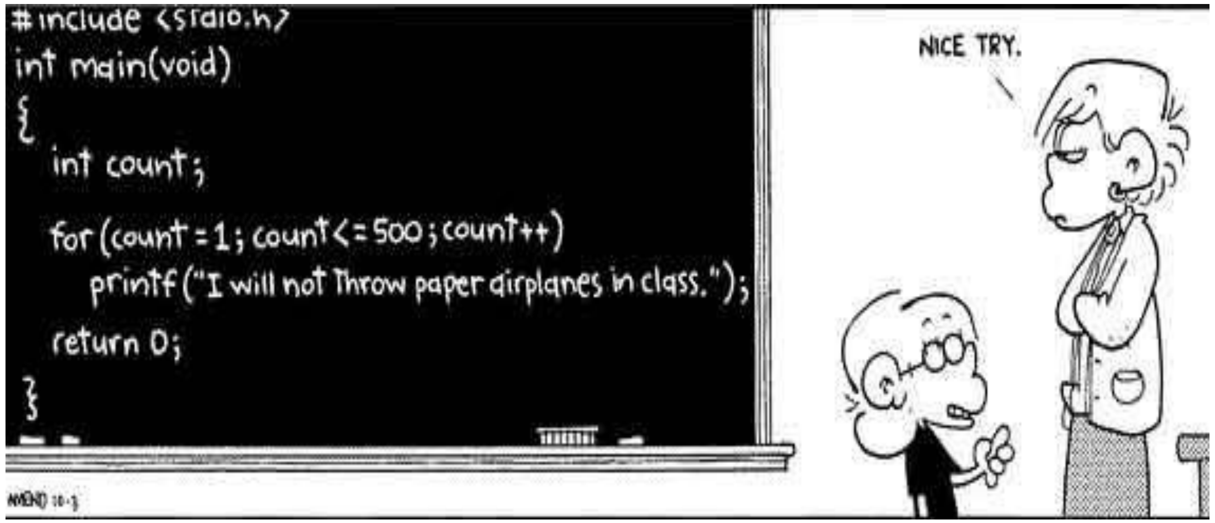

Software Engineering Body of Knowledge (SWEBOK)
Software Requirements
Software Design
Software Construction
Software Testing
Software Maintenance
Software Configuration Management
Software Engineering Management
Software Engineering Process
Software Engineering Tools & Methods
Software Quality
Knowledge Area of Related Disciplines

AGENDA

- Review
- Object-Oriented Languages
- UML Class Diagrams
  - Attributes
  - Operations
- UML Class Diagram Activity

Language Paradigms {Arch. Design Choices}

- Imperative (procedural): C, Python?, Perl?
- Functional
- Object-oriented (OO): C++, Java, C#, Ruby
- Logic


```
#include <stdio.h>
int main(void)
{
  int count;

  for (count = 1; count <= 500; count++)
    printf("I will not Throw paper airplanes in class.");

  return 0;
}
```

Object-Oriented Languages

1. SQL
2. **Java**
3. HTML
4. JavaScript
5. **C++**
6. C#
7. XML
8. C
9. Perl (may be OO)
10. Python (may be OO)

MODELING? (DETAILED DESIGN)

- Modeling languages
  - Syntax
  - Semantics
- High-level abstraction
  - From a specific perspective
  - For a specific purpose
- UML?

UML

- Unified Modeling Language
  - Used to communicate just the important details about code and structure
  - Modeling always involves abstraction
    - Captures important details that developers or customers need.
- Better than looking at 200 lines of code
  - Focus on big picture.

Objectives of UML

- UML is a general purpose notation that is used to
  - visualize,
  - specify,
  - construct, and
  - document

  the artifacts of a software systems.

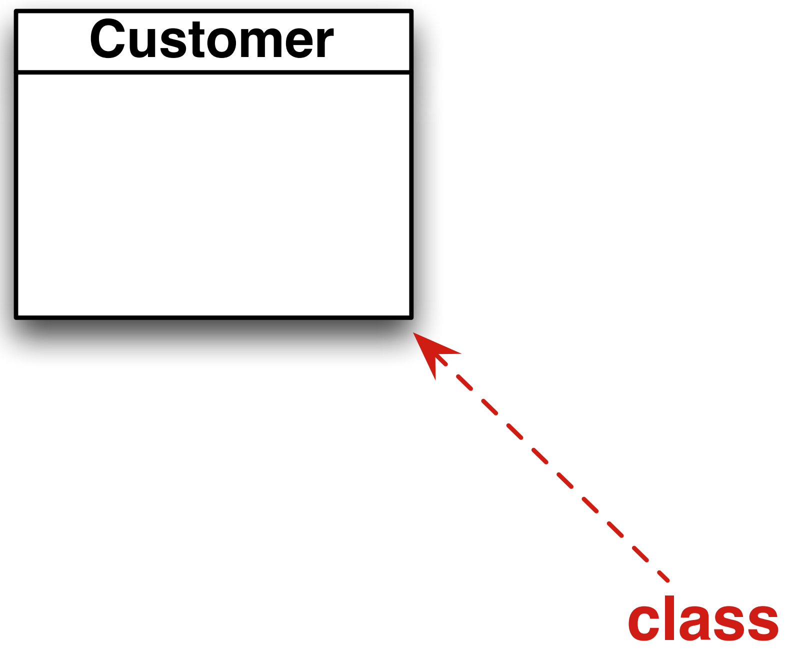
BOOK: UML DISTILLED

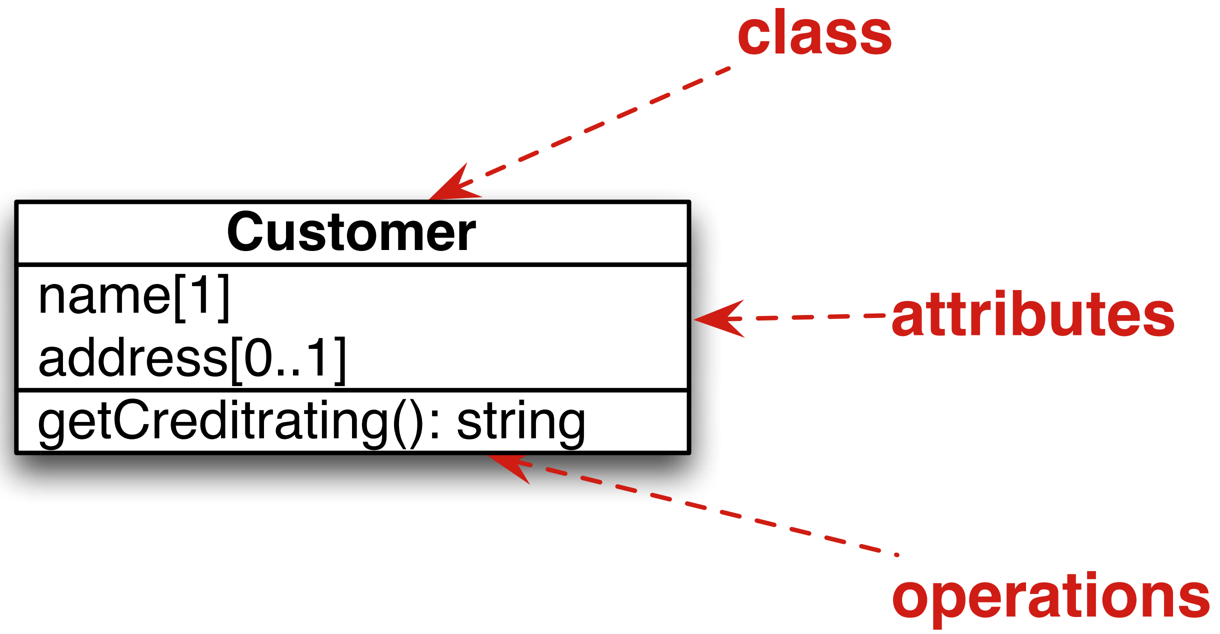
- Free (for you) Book about UML:
  - UML Distilled: A Brief Guide to the Standard Object Modeling Language (Third Edition)
  - By: Martin Fowler
- [UML-DISTILLED]
  - https://learning.oreilly.com/library/view/uml-distilled-a/0321193687/

AGENDA

- Review
- Object-Oriented Languages
- UML Class Diagrams
  - Attributes
  - Operations
- UML Class Diagram Activity

UML Class Diagrams



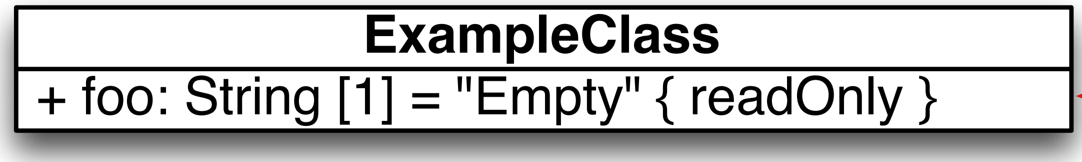

class
Customer
name[1]
address[0..1]
getCreditrating(): string
attributes
operations

AGENDA

- Review
- Object-Oriented Languages
- UML Class Diagrams
  - Attributes
  - Operations
- UML Class Diagram Activity

ATTRIBUTES

```
<visibility> <name> : <type> <multiplicity>    = <default>   { <property-string> }

+              foo  : String [1]               = "Empty"     { readOnly          }
```



ATTRIBUTES

```
<visibility> <name> : <type> <multiplicity>    = <default>   { <property-string> }

+             foo   : String [1]               = "Empty"     { readOnly          }
```

- <visibility>
  - + Public
  - - Private
  - # Protected

ATTRIBUTES

```
<visibility> <name> : <type> <multiplicity>    = <default>   { <property-string> }

+             foo   : String [1]               = "Empty"     { readOnly          }
```

- <name>
  - Name of field

ATTRIBUTES

```
<visibility> <name> : <type> <multiplicity>    = <default>   { <property-string> }

+             foo   : String [1]               = "Empty"     { readOnly          }
```

- <type>
  - Kind of object

ATTRIBUTES

```
<visibility> <name> : <type> <multiplicity>    = <default>   { <property-string> }

+             foo   : String [1]               = "Empty"     { readOnly          }
```

- <multiplicity>
  - 1 (must have exactly one)
  - 0.. n (lower to upper bound)
  - * (zero or more)

ATTRIBUTES

```
<visibility> <name> : <type> <multiplicity>    = <default>   { <property-string> }

+             foo   : String [1]               = "Empty"     { readOnly          }
```

- <default>
  - If attribute is not initialized, what is the default value

ATTRIBUTES

```
<visibility> <name> : <type> <multiplicity>    = <default>   { <property-string> }

+             foo   : String [1]               = "Empty"     { readOnly          }
```

- <property-string>
  - Additional properties

AGENDA

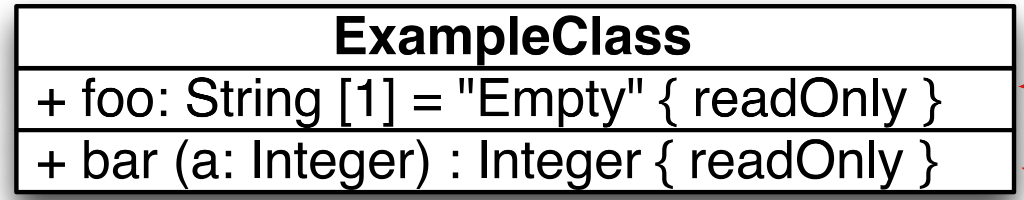
- Review
- Object-Oriented Languages
- UML Class Diagrams
  - Attributes
  - Operations
- UML Class Diagram Activity

OPERATIONS

- Represent the methods for your class
- Typically don't show operations that just manipulate attributes

ATTRIBUTES

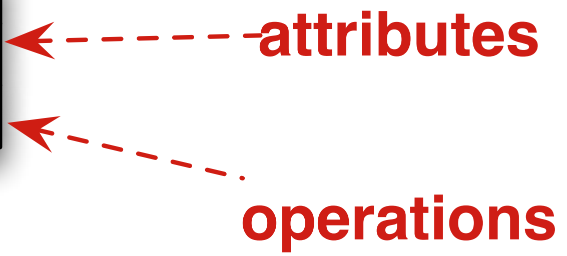
```
<visibility> <name> (<parameter-list>)   : <return-type>      { <property-string> }

+               foo    (a: Integer)         : Integer             { readOnly           }
```



ExampleClass

+ foo: String [1] = "Empty" { readOnly }

+ bar (a: Integer) : Integer { readOnly }



ATTRIBUTES

```
<visibility> <name> (<parameter-list>)   : <return-type>      { <property-string> }

+                foo  (a:  Integer)      : Integer            { readOnly          }
```

- <parameter-list>
  - Parameters to method
  - Similar format as attributes

ATTRIBUTES

```
<visibility> <name> (<parameter-list>)  : <return-type>    { <property-string> }

+            foo    (a: Integer)         : Integer          { readOnly          }
```

- <return-type>
  - Return type of the method

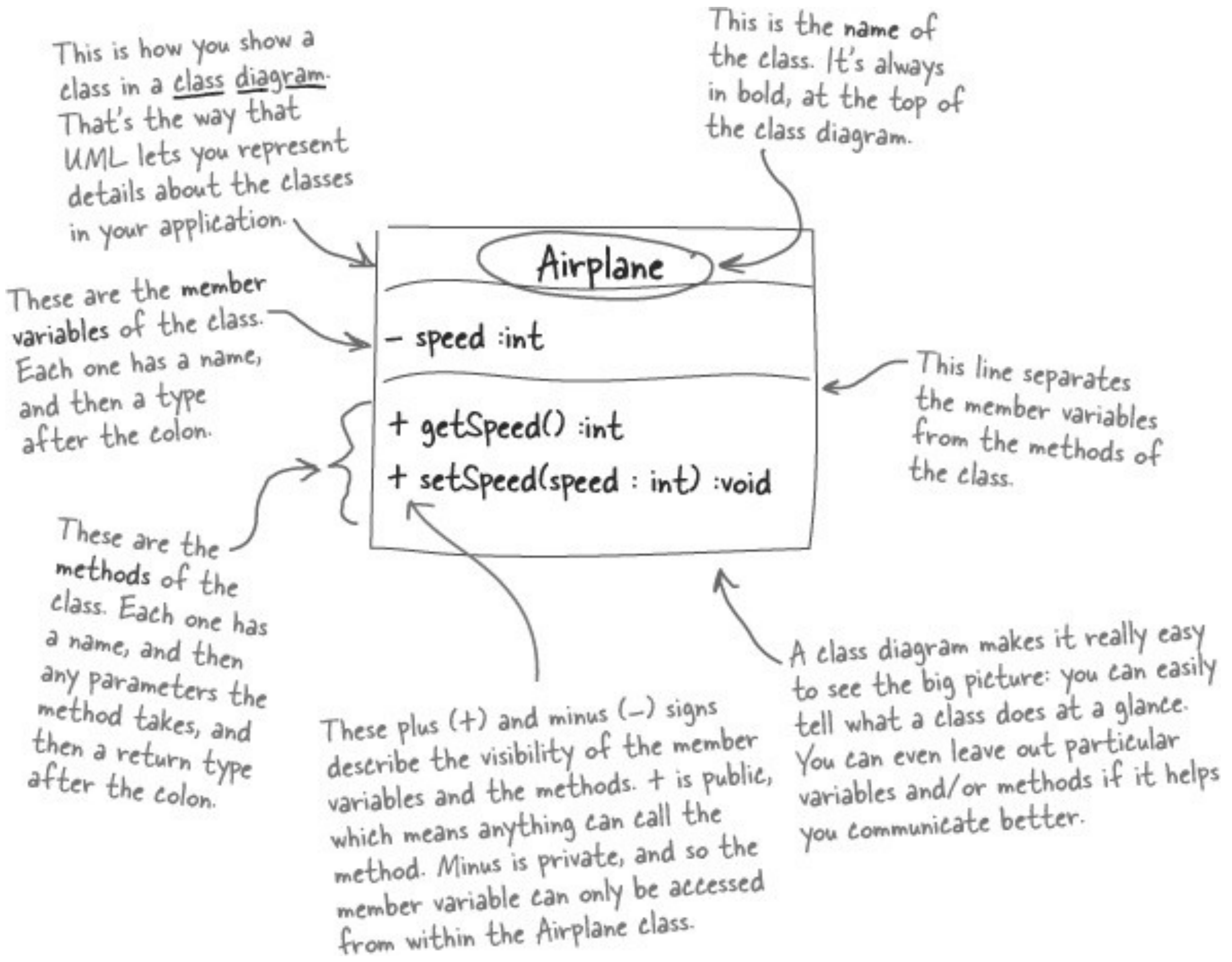
AGENDA

- Review
- Object-Oriented Languages
- UML Class Diagrams
  - Attributes
  - Operations
- UML Class Diagram Activity

UML Activity

- The university wishes to create a new course registration system. An administrator should be able to add/remove/modify student records and a course records. Each course has a minimum and maximum student enrollment number as well as the a designated number of credits. Each student has a minimum and maximum number of credit hours that they can register for in a given semester. Administrators, students, and professors should be able to login to the system. One professor is assigned to each course. Multiple students are assigned to each course. Students may register and unregister themselves for a course. Professors may register/unregister students for a course.

UML Class Diagram Summary – From Book


This is how you show a class in a class diagram. That's the way that UML lets you represent details about the classes in your application.
This is the name of the class. It's always in bold, at the top of the class diagram.
Airplane
- speed :int
+ getSpeed() :int
+ setSpeed(speed : int) :void
These are the member variables of the class. Each one has a name, and then a type after the colon.
This line separates the member variables from the methods of the class.
These are the methods of the class. Each one has a name, and then any parameters the method takes, and then a return type after the colon.
These plus (+) and minus (–) signs describe the visibility of the member variables and the methods. + is public, which means anything can call the method. Minus is private, and so the member variable can only be accessed from within the Airplane class.
A class diagram makes it really easy to see the big picture: you can easily tell what a class does at a glance. You can even leave out particular variables and/or methods if it helps you communicate better.

Retrospective Questions

What is (the) UML? Have you used it in your other classes

Additional Thoughts on the Activity?

I Posted a Class Diagram "Exercises" On Canvas

- Canvas > Modules > Week-05 > **Class Diagram Exercises**

Next Class

- Project Introduction
- Advanced UML Class Diagrams