Welcome to CS680!

Tue Thu 5:30pm - 6:45pm

Y02-2120

Who am I?

- Academics
  - Associate Professor, UMass Boston (2010–)
  - Assistant Professor, UMass Boston (2004–2010)
    - Distributed systems, software engineering and AI
    - www.cs.umb.edu/~jxs/; dssg.cs.umb.edu
  - Post-doctoral Research Fellow, UC Irvine, CA (2000–2004)
  - Ph.D. in Comp Sci from Keio University, Japan (2001)
- Industrial
  - Consultant, cloud computing platform vendor, supply chain mgt. company
  - Tech Director, Object Management Group Japan
  - Co-founder and CTO, TechAtlas Comm Corp, Austin, TX
  - Programmer Analyst, Goldman Sachs Japan
- Professional
  - Member, ISO SC7/WG 19
  - Specification co-lead, OMG Super Dist. Objects SIG

Course Topics

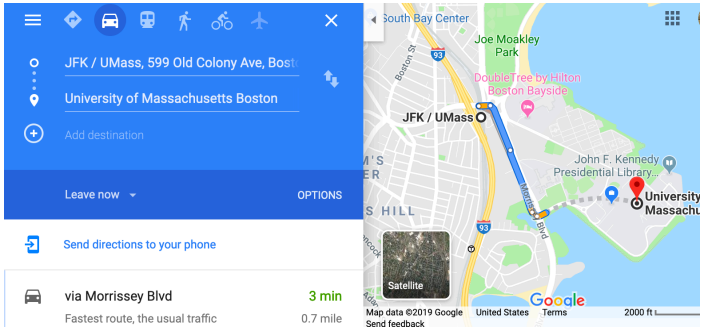
- Object-oriented design (OOD)
  - Unified Modeling Language (UML)
  - Refactoring
  - Design patterns
  - Object-oriented programming (OOP) with Java
- Continuous testing
  - Automated build of programs
  - Unit testing, static code inspection, etc.
  - Versioning (maybe)
- Basics in functional programming (with Java)
  - Lambda expressions in Java
  - Integration of functional programming with OOP

Please Understand…

- You are assumed to be familiar with basics in OOP.
  - Classes, methods, interfaces, inheritance, collections, etc.
- Objective in CS680
  - Learn ***design*** and ***organization*** of object-oriented programs
- Analogy in carpentry:
  - *Objective*: Learn how to design (i.e., draw reasonable blueprints for) things you build (e.g., stairs, deck, kitchen, house).
    - You need to be familiar with basic materials and tools such as screws, nuts, nails, screwdrivers, nail guns, levels, etc.
    - Your focus is to reasonably design a deck, for example,
      - considering proper footings and solid framing to meet given requirements (e.g., structural stability).

An Example Scenario

- Your team is expected to develop a navigation app like G Maps.
  - For users to drive and walk (two navigation features)



- How can two sub-groups of the team develop these two features *independently* (i.e. *in parallel*)?
  - How can those 2 features be implemented in a *loosely-coupled* manner?
    - NOT in a tightly-coupled manner.
    - To maximize productivity (development efficiency)
  - How can they be *integrated* in the end of the project in a cost-effective manner?
- How can *something common* be implemented in between the 2 features?
  - Basic data structures and algorithms (e.g. maps, landmarks and shortest-path algorithms)

- Goal in CS680
  - Answer this kind of questions by learning about ***design*** and ***organization*** of (object-oriented) programs.
- It is easy to say “separating 2 features in a loosely-coupled manner and integrating them later”
- However, unfortunately, it is not always that easy to DO it actually.

An Extended Scenario

- Your team is asked to implement extra navigation features.
  - e.g., with public transportation, with a bike, etc.
- How can those extra features be implemented with no/minimum impacts on existing features?
  - How to keep individual features loosely-coupled,
    - so extra features can be introduced in a maintainable and cost-effective manner?

- Goal in CS680
  - Think about productivity and maintainability by learning about ***design*** and ***organization*** of (object-oriented) programs.

Textbooks

- No official textbooks.
- Recommended textbooks
  - *Object-Oriented Analysis and Design with Applications (3rd edition)*
    - by Grady Booch et al. (Addison Wesley)
    - General intro to OOAD.
  - *Refactoring: Improving the Design of Existing Code*
    - by Martin Fowler
    - Addison-Wesley
  - *Head First Design Patterns*
    - by Elizabeth Freeman et al.
    - O'Reilly
  - *Effective Java (3rd Edition)*
    - by Joshua Bloch
    - Addison-Wesley

- The most authoritative and "Bible-like" book on design patterns:
  - *Design Patterns: Elements of Reusable Object-Oriented Software*
    - By Eric Gamma et al.
    - Addison-Wesley

Course Work

- Lectures
- Homework
  - Reading
  - Coding (in Java)
- Grading factors
  - Homework (80%)
  - Quizzes (20%)
    - Occasionally, in lectures
- No midterm and final exams.

- HWs have submission deadlines.
  - Do your best to meet the deadlines.
    - If you regularly meet them, you will get some extra points.
  - You can miss the deadlines. You DO NOT have to notify me that you will miss or have missed a deadline.
    - Up to a week late or so: No problem. I favor better code than timeliness. Focus on your work, not making excuses to me.
    - Beyond that: Depends.
- In principle, you cannot replace your HW solution after you submit it.

My Email Addresses

- Questions → **jxs@cs.umb.edu**
  - I regularly check this account.
- HW solutions → **TBD/A**
  - May use GitHub rather than email.

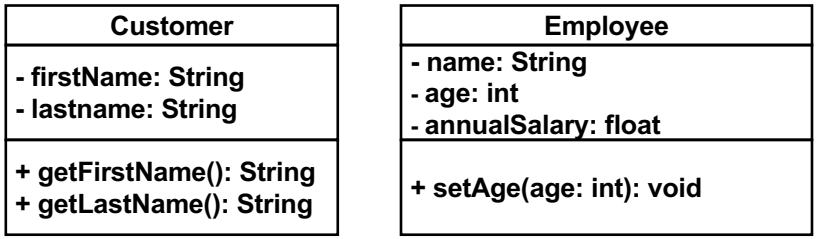
Your Email Address

- Send your (preferred) email address to jxs@cs.umb.edu ASAP.
  - I will use that address to email you lecture notes, announcements, etc.
    - Please stay tuned at your email address.
  - You will use that address to submit your HW solutions.

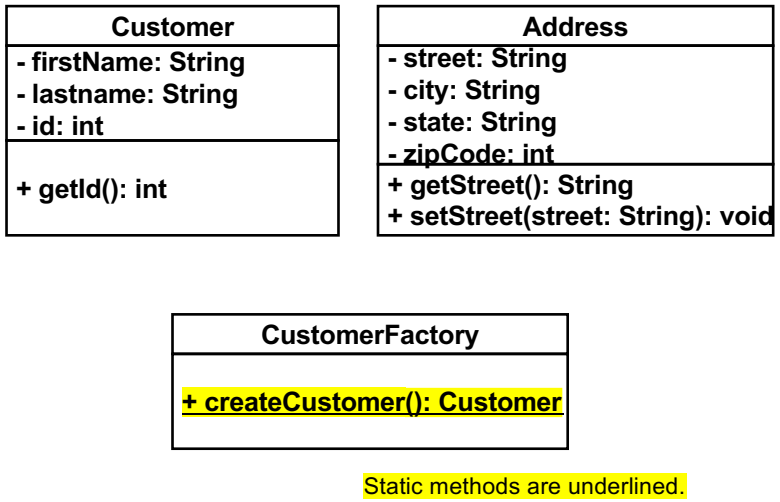
Preliminaries: Unified Modeling Language (UML)

Unified Modeling Language (UML)

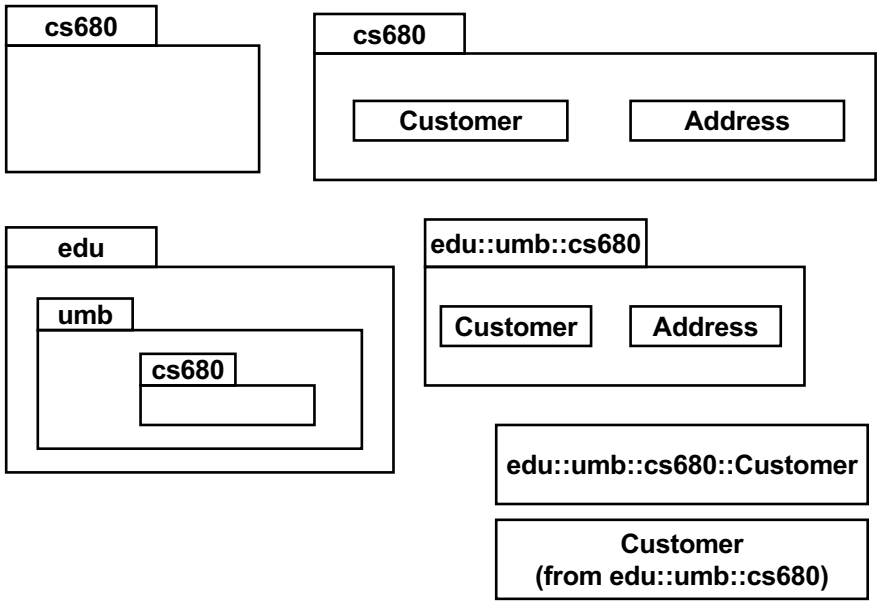
- A language to visually ***model*** (or specify) software
  - Intuitively, it is a set of icons, symbols and diagrams to visualize particular elements in software designs.



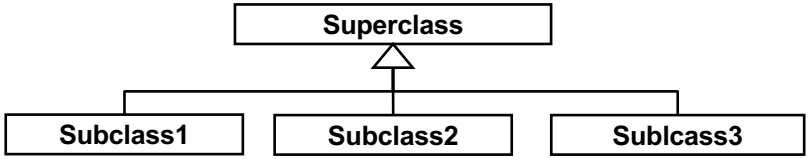
Classes in UML



Packages in UML



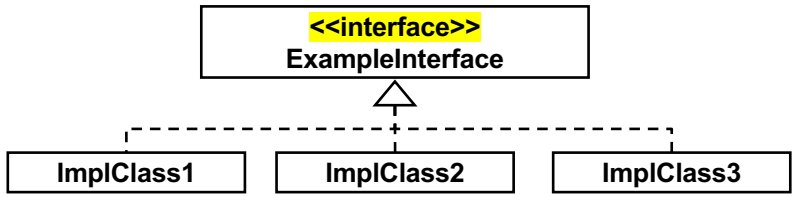
Class Inheritance




```
class Superclass{ ... }

class Subclass1 extends Superclass { ... }
class Subclass2 extends Superclass { ... }
class Subclass3 extends Superclass { ... }
```

Interface-Class Relationship

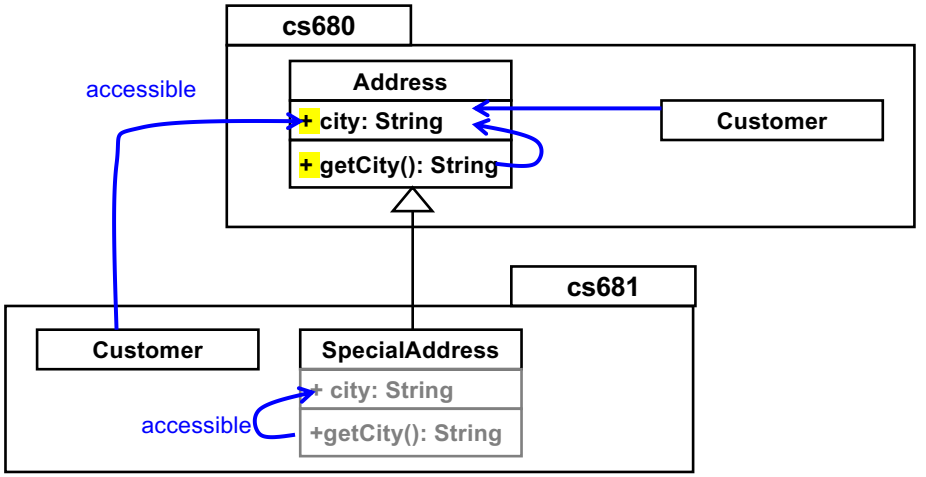


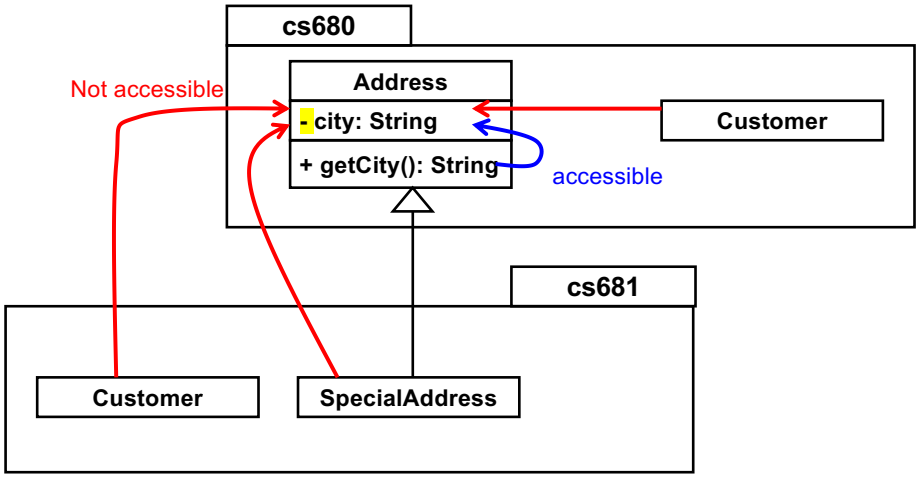

```
interface ExampleInterface{ ... }

class ImplClass1 implements ExampleInterface { ... }
class ImplClass2 implements ExampleInterface { ... }
class ImplClass3 implements ExampleInterface { ... }
```

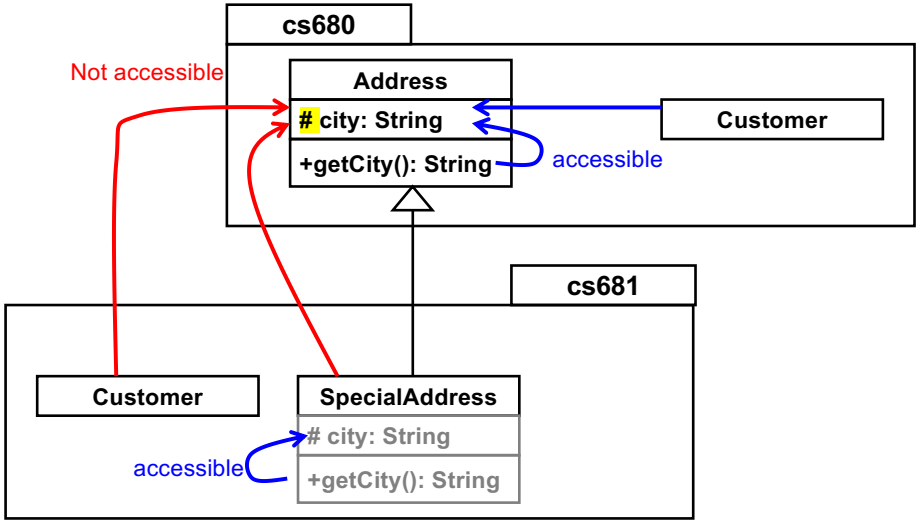

Java's Visibility in UML

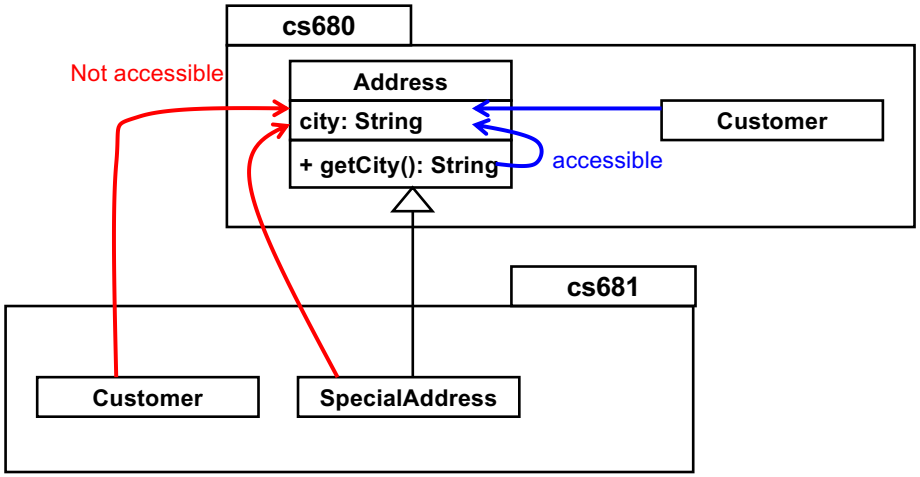
- Defines who can access a data field or a method
  - Public (+), private (-) or protected (#)





Encapsulation principle: Use private/protected visibility as often as possible to encapsulate/hide the internal data fields and methods of a class.

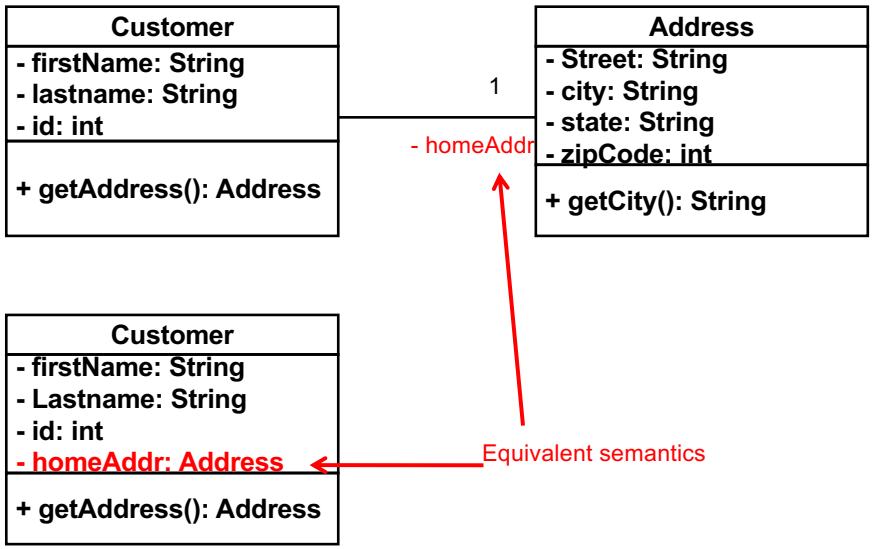


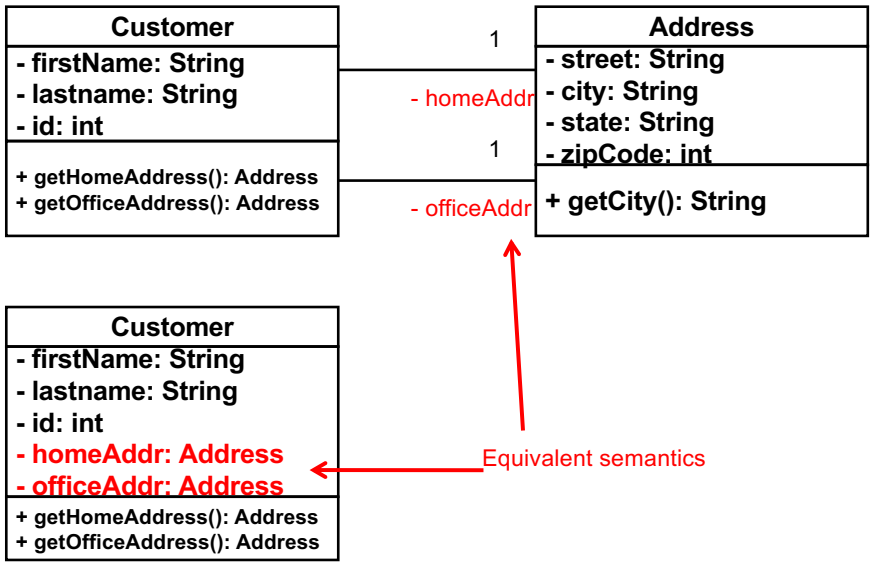


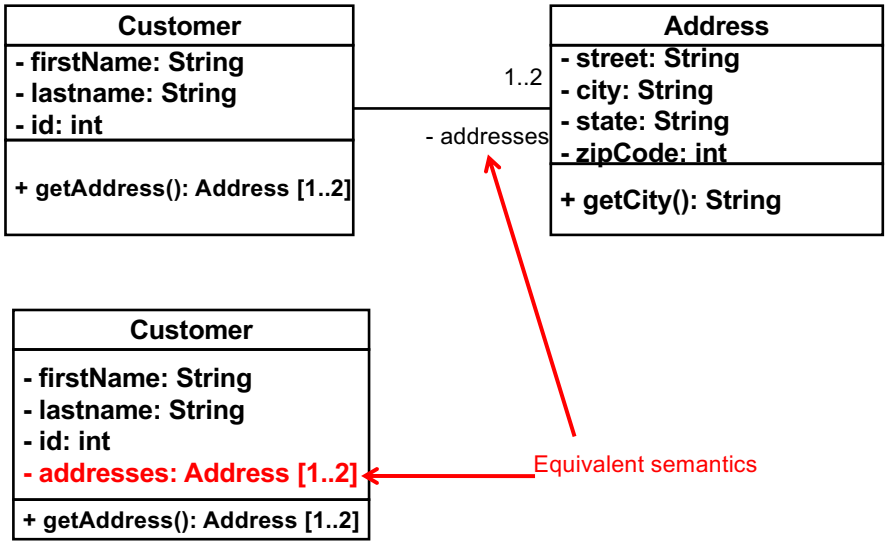
Default visibility (package private) to be used when no modifier is specified.

- Specify the modifier for every data field and every method.
- Do not to skip specifying it. (Do not use package-private.)
- It is always important to be aware of the visibility of each data field and method.

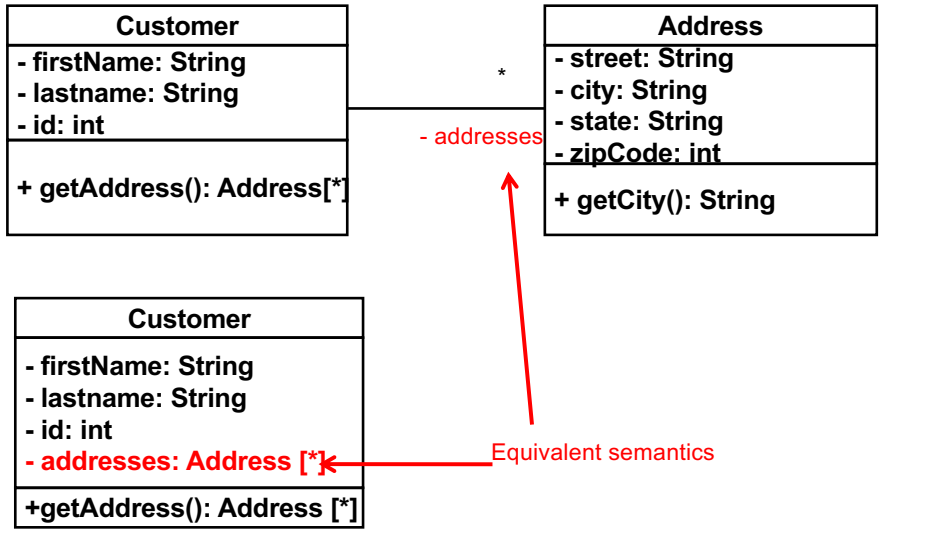
Association



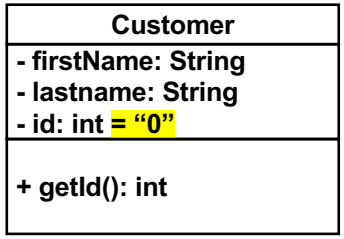



Customer
- firstName: String
- lastname: String
- id: int
+ getAddress(): Address [1..2]
1..2
- addresses
Address
- street: String
- city: String
- state: String
- zipCode: int
+ getCity(): String
Customer
- firstName: String
- lastname: String
- id: int
- addresses: Address [1..2]
+ getAddress(): Address [1..2]
Equivalent semantics

31


Customer
- firstName: String
- lastname: String
- id: int
+ getAddress(): Address[*]
*
- addresses
Address
- street: String
- city: String
- state: String
- zipCode: int
+ getCity(): String
Customer
- firstName: String
- lastname: String
- id: int
- addresses: Address [*]
+getAddress(): Address [*]
Equivalent semantics

32


Customer
- firstName: String
- lastname: String
- id: int = “0”
+ getId(): int

33