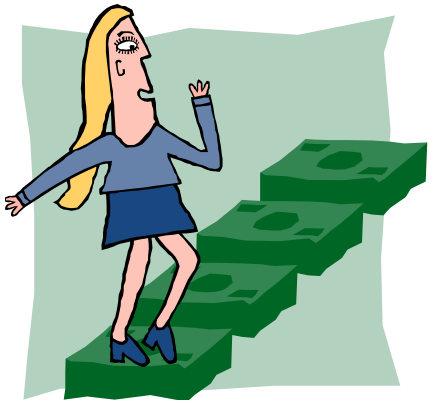
School of
Information Systems

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Object Oriented Programming

Class Diagrams



Q: What did the Java code say to the C code?
A: You've got no class.

Overview

- Objective
  - To learn about Object Orientation Concepts
- Content
  - Classes versus Objects
  - Principles of Object Orientation
  - Why do Modeling?
  - Class Diagram
- After this module, you should be able to
  - Explain the basic principles of Object Orientation
  - Explain the importance of modeling
  - Read a class diagram and translate it to code

The Object Technology

- OO is a way of looking at a software system as a collection of interactive objects

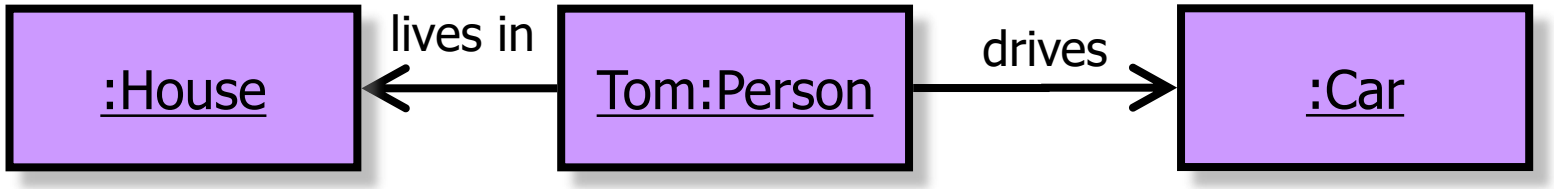
Reality

House

Tom

Car

Model



What is an Object?

- Informally, an object represents an entity, either physical, conceptual, or software.
  - Physical entity

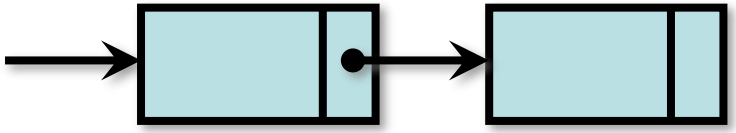
Car

  - Conceptual entity

US$100,000,000,000

Bill Gate's bank account

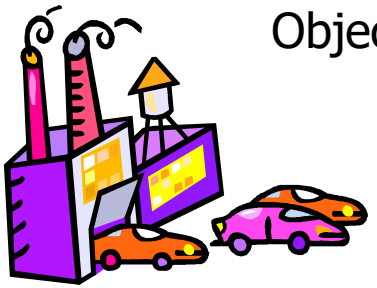
  - Software entity



Linked List

Classes

- A class is the blueprint from which individual objects are created.
- An object is an instance of a class.



Object factory

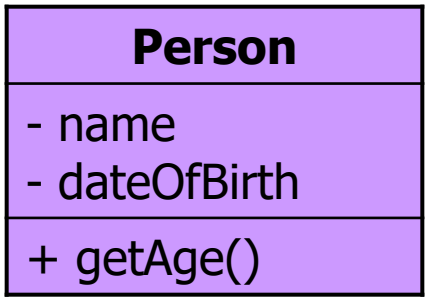
Cookie Cutter

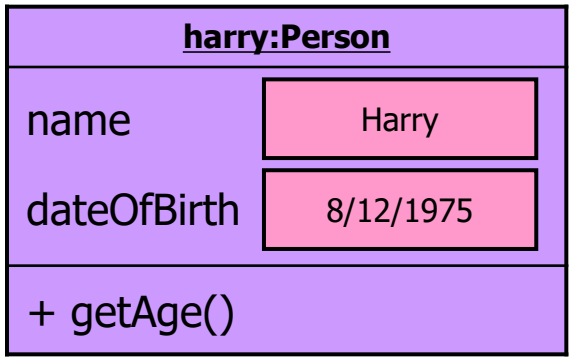
```
public class StudentTest {
  public static void main(String[] args) {
    Student s1 = new Student();
    Student s2 = new Student();
  }
}
```

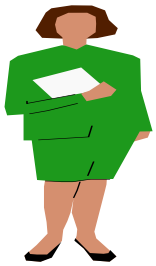
```
- class -

public class Student {
  private String name;
  // …
}
```

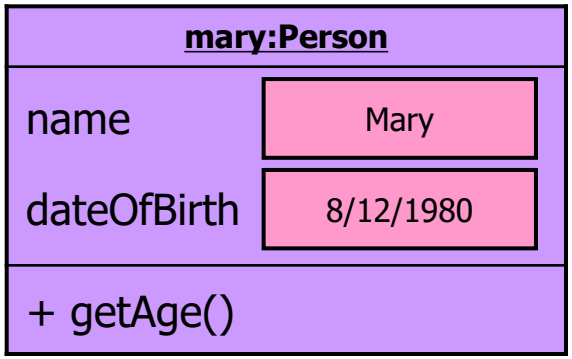
Classes & Objects

- All the objects share the same attribute names and methods with other objects of the same class
- Each object has its own value for each of the attribute

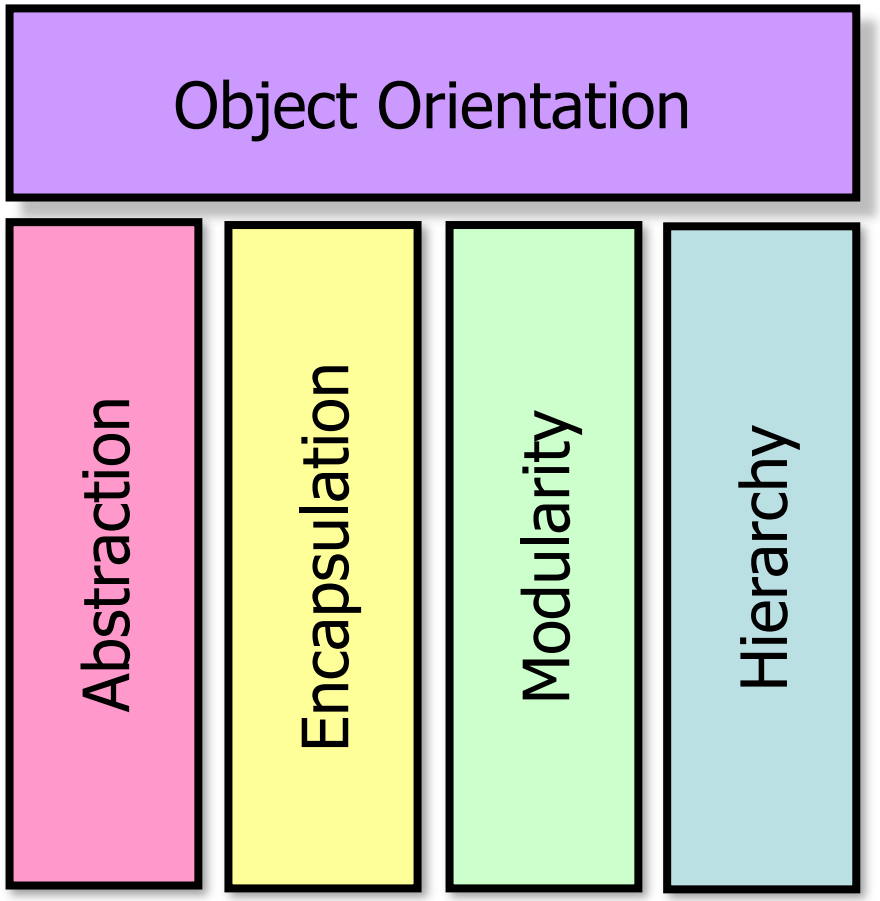








Basic Principles of Object Orientation



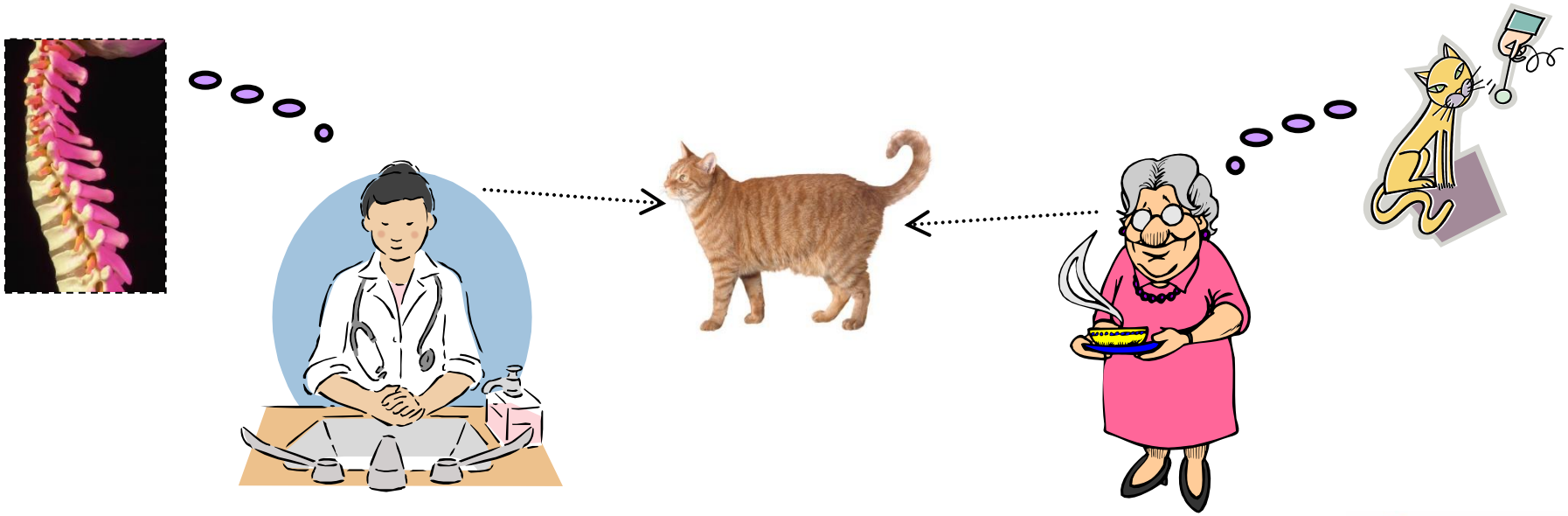
Abstraction

- Determine the relevant properties and features while ignoring non-essential details

Cat
- bloodType - numberOfBones - lastVisitDate

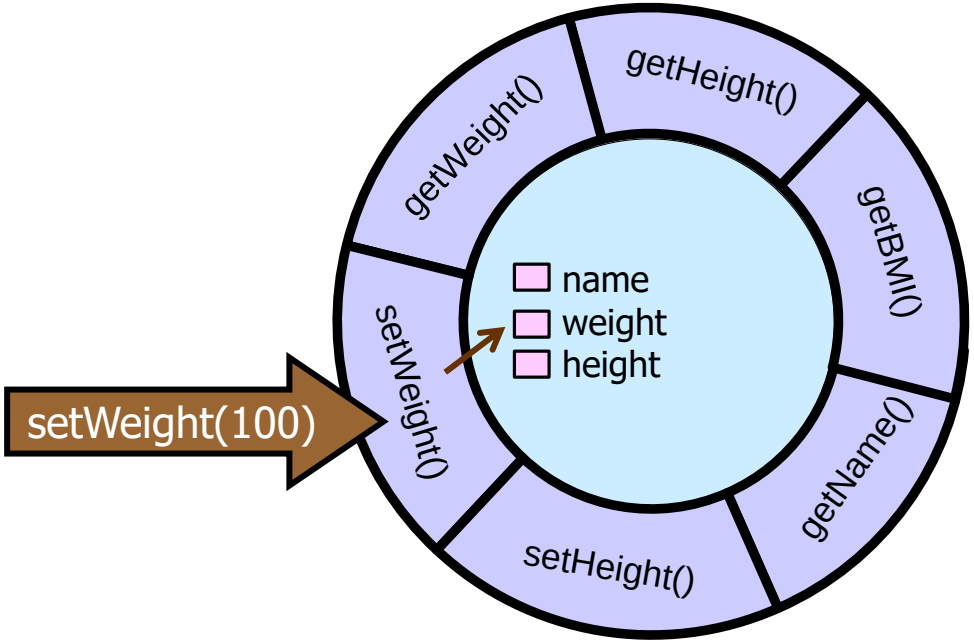
abstraction

Cat
- birthday - favouriteFood - favouriteToy

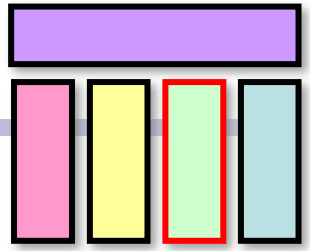


Encapsulation

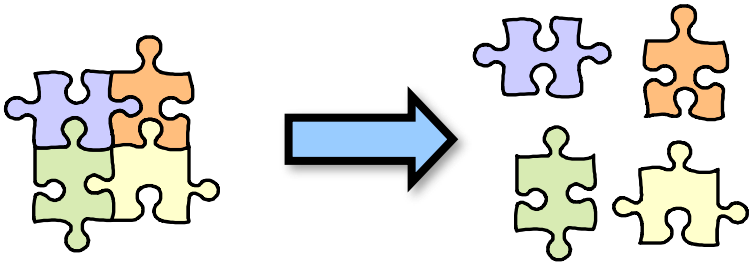
- Separate components into external and internal aspects

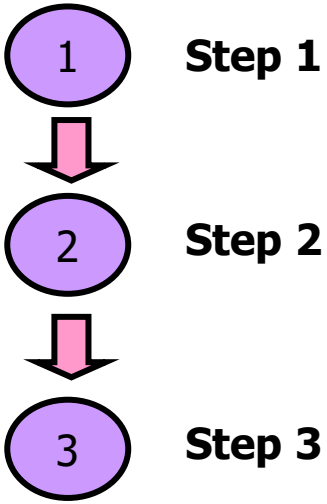


Modularity

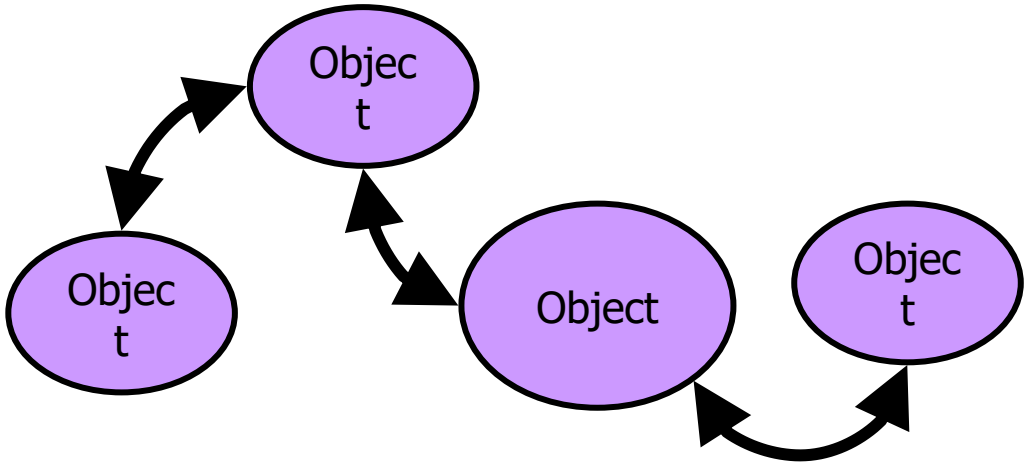


- Break something complex into manageable pieces
  - Functional Decomposition
  - Object Decomposition





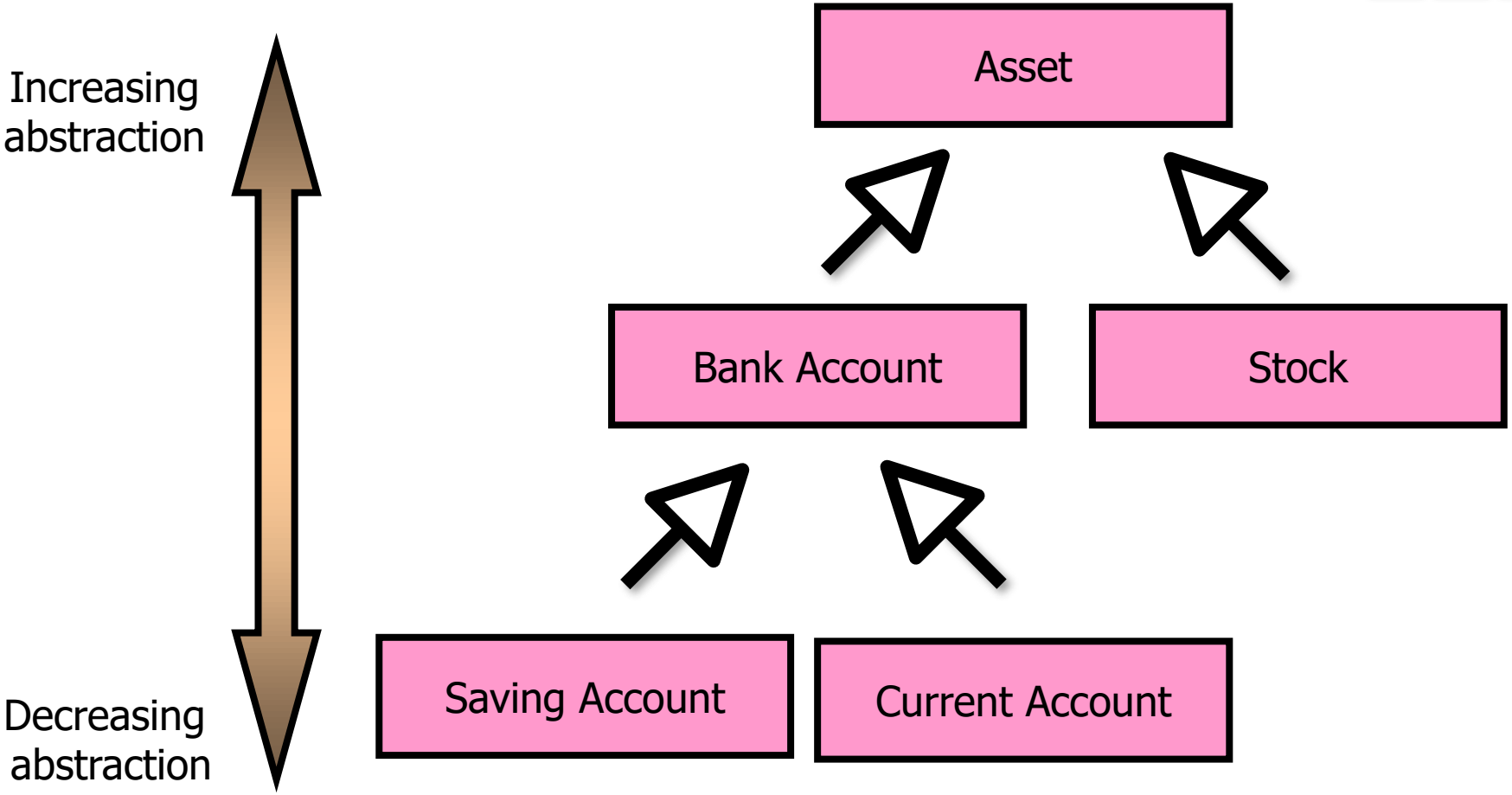
Functional Decomposition



Object Decomposition

Hierarchy

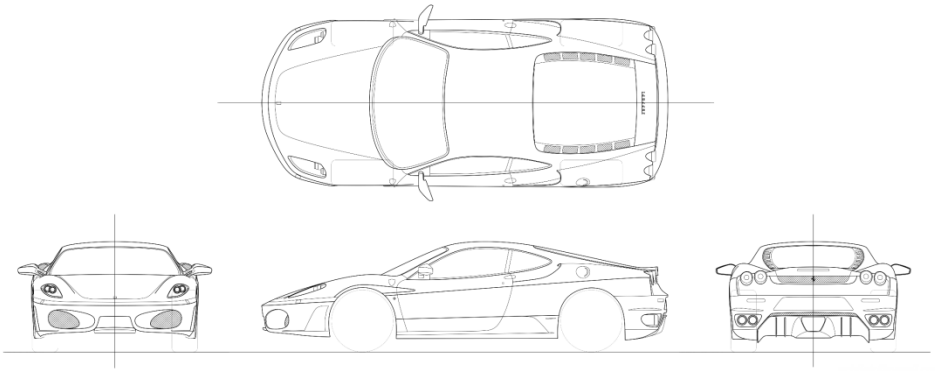
- Ranking or ordering of objects



What is a model?

https://www-computer-org.libproxy.smu.edu.sg/csdl/mags/so/2002/01/s1008.pdf

- A simplification of reality.



Why do we model?

http://www.ibm.com/developerworks/rational/library/6007.html

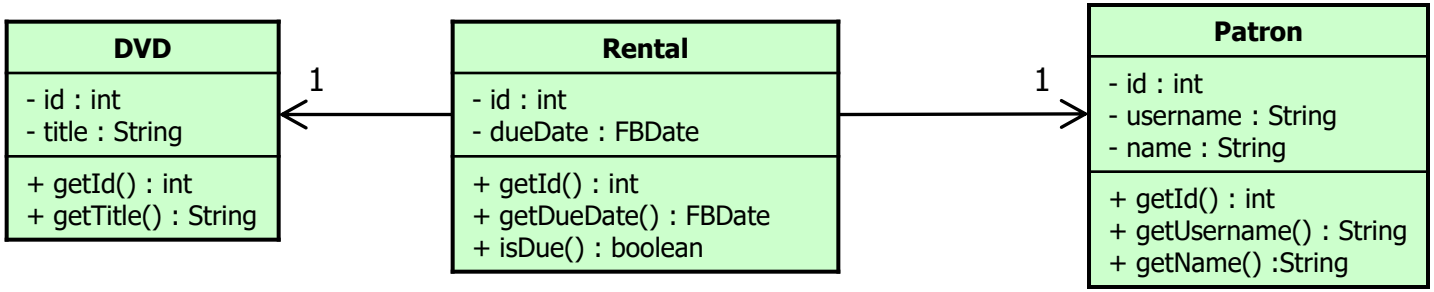
- Modeling achieves four aims:
  - Helps you to visualize a system as you want it to be.
  - Permits you to specify the structure or behavior of a system.
  - Gives you a template that guides you in constructing a system.
  - Documents the decisions you have made.

Unified Modeling Language

- A standard language for specifying, visualizing, constructing and documenting the artifacts of software systems
- Advantages
  - Facilitate communication
  - manage the complexity of systems as they increase in scope and scale

What is a Class Diagram?

- Shows the static view of a system
  - What are the classes?
  - What are their relationships?



Graphical Representation

- Class
  - We use a rectangle to represent a class with its name appearing inside the rectangle.

<Class Name>

- Example

Person

Account

- Object
  - We use a rectangle to represent an object and place the underlined name of the object inside the rectangle.

<u><Object Name>:<Class Name></u>

- Example

<u>Tom:Person</u>

Class Diagrams

- A class without attributes & methods

UML	Implementation
Person	`public class Person {` `}`

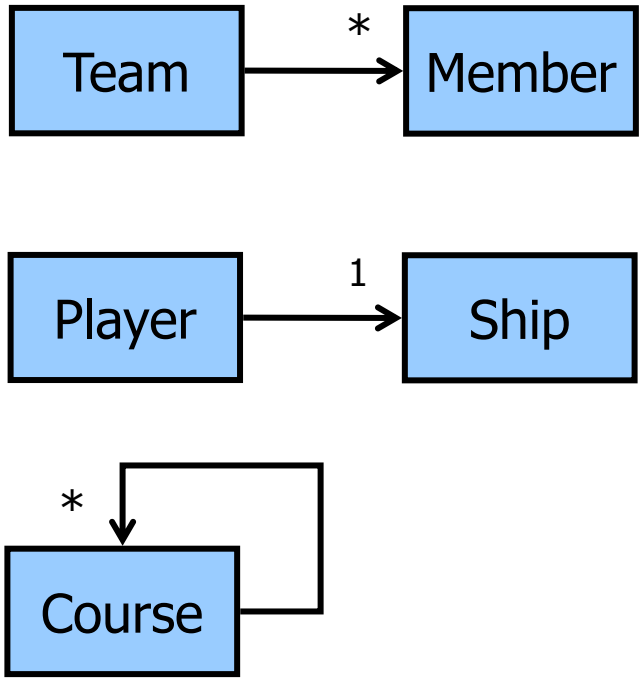
- A class with attributes & methods

UML	Implementation

Person
- name : String
- age : int
+ Person(name : String)
+ calculateBMI() : double
+ isOlderThan(another : Person) : boolean
	`public class Person {`
	`  private String name;`
	`  private int age;`
	
	`  public Person(String name) { ...}`
	
	`  public double calculateBMI() { ... }`
	
	`  public boolean isOlderThan(Person another) {...}`
	`}`

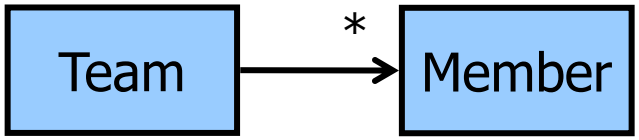
Association

- A structural relationship specifying that objects of one thing are connected to objects of another thing.

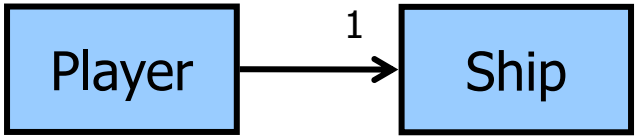


Association

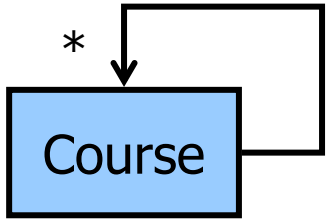
- Associations between classes most often represent instance variables that hold references to other objects.




```
public class Team {
  private ArrayList<Member> members;
}
```



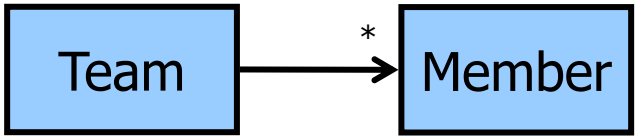

```
public class Player {
  private Ship ship;
}
```



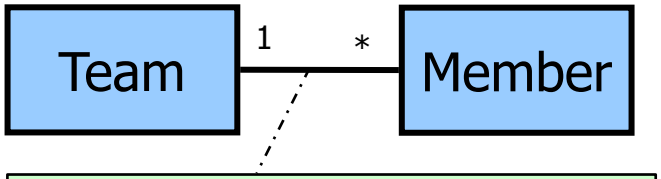

```
public class Course {
  private ArrayList<Course> preRequisites;
}
```

Navigability

- Indicates whether it is possible to navigate from a associating class to the target class using the relationship (association).
- Examples




```
public class Team {
  private ArrayList<Member> members;
}
```



The association is assumed to be navigable in both directions when no arrowheads are shown.

```
public class Team {
  private ArrayList<Member> members;
}
```

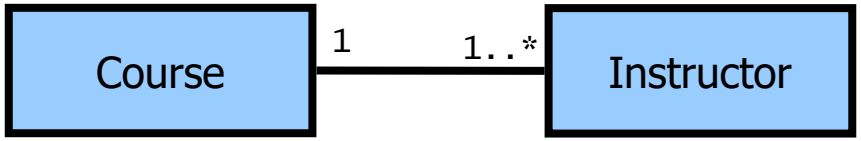
```
public class Member {
  private Team team;
}
```

Multiplicity

- Multiplicity is the number of instances of one class that can or must be related to ONE instance of another class.

- For each association, there are two multiplicity decisions to make, one for each end of the association.

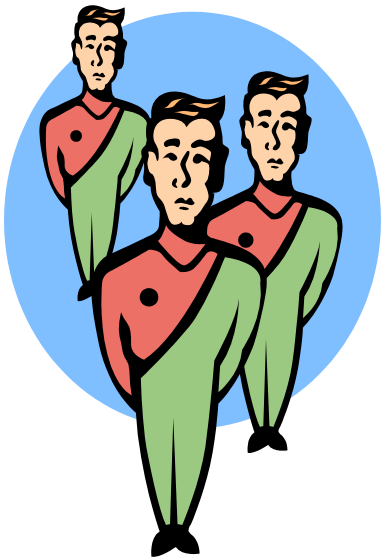
- Example
  - For ONE Course, it is taught by ONE or MORE instructors.
  - For ONE instructor, he will teach only ONE course



Multiplicity

- Multiplicity indicators

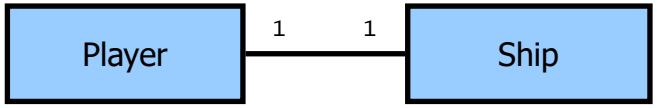
Multiplicity	Description
	Unspecified
1	One
0..1	Optional One
0..*	Zero or more
*	Zero or more
1..*	One or more



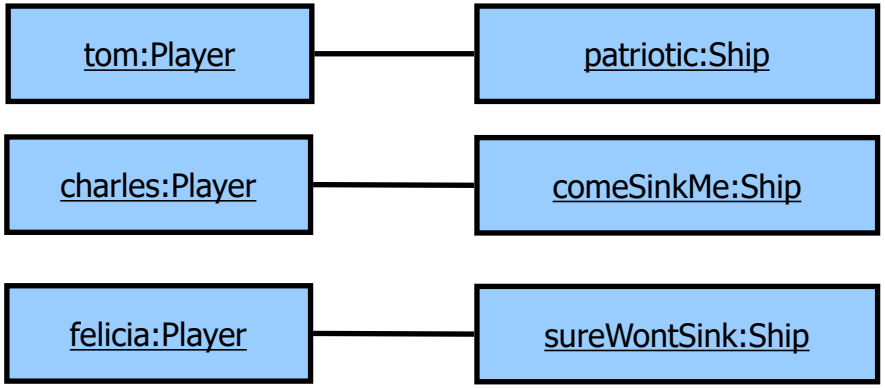
- Three basic "flavors" of multiplicity
  - One-to-One
  - One-to-Many
  - Many-to-Many

Multiplicity: One-to-One

- Example
  - A Player has a Ship.
  - A Ship belongs to a Player.



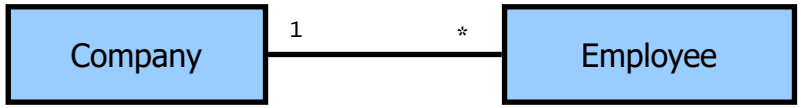
Class Diagram



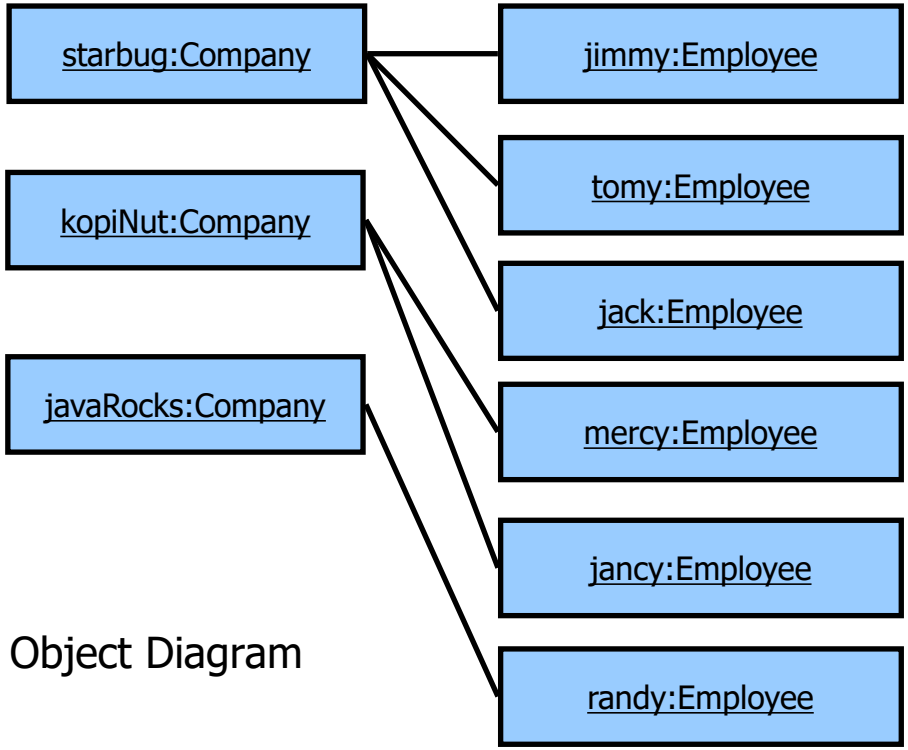
Object Diagram

Multiplicity: One-to-Many

- Example
  - A company hires many employees.
  - An employee works for only one company.



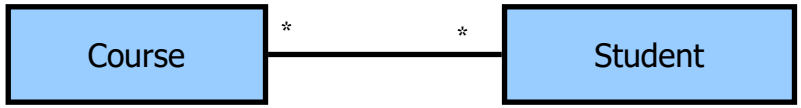
Class Diagram



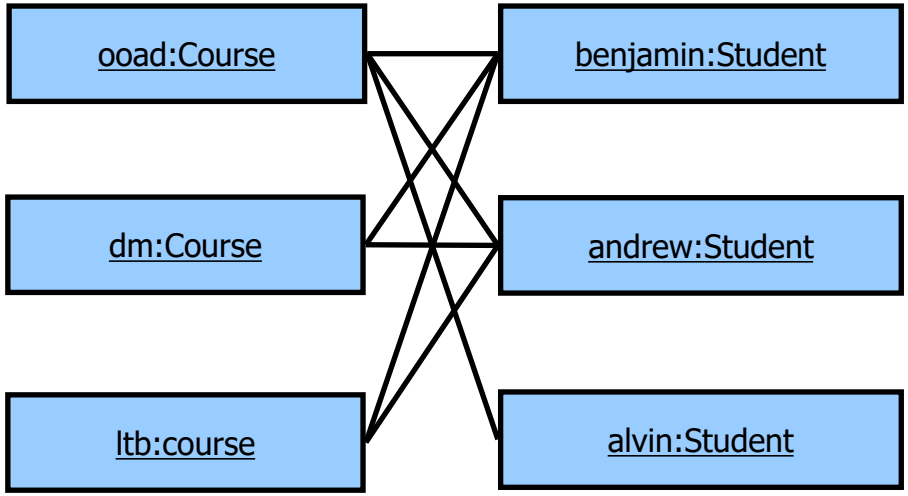
Object Diagram

Multiplicity: Many-to-Many

- Example
  - A course has many students enrolled in it.
  - A student enrolls in many courses.



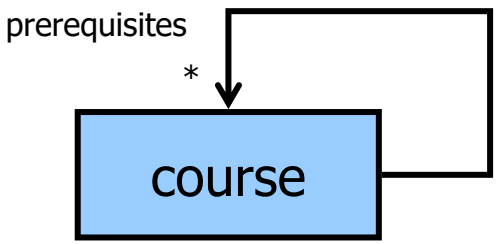
Class Diagram



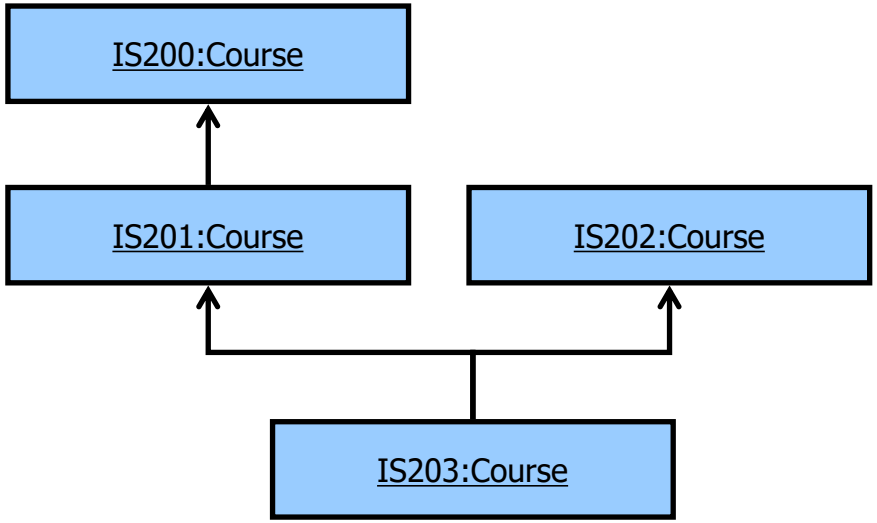
Object Diagram

Reflexive Association

- One instance of the class has associations to other instances of the same class



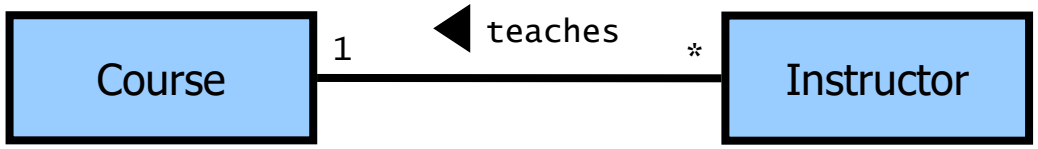
Class Diagram



Object Diagram

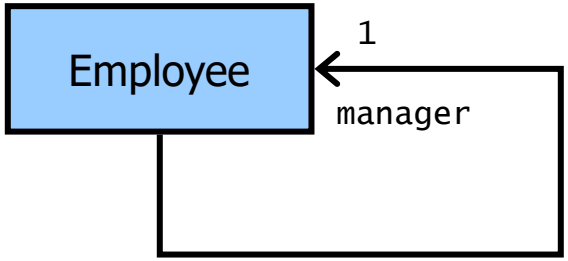
Verb Phrase

- Used to label an association so that the relationship can be used in a sentence.
- The direction of the arrow has no meaning in terms of the model.
- It is only an aid to the reader of the diagram.
  - The instructor teaches a Course
- Its use is optional.



Association Roles

- Alternative to using a verb phrase.
- Roles on the end of the association can add clarity.




```
public class Employee {
  private Employee manager;
}
```

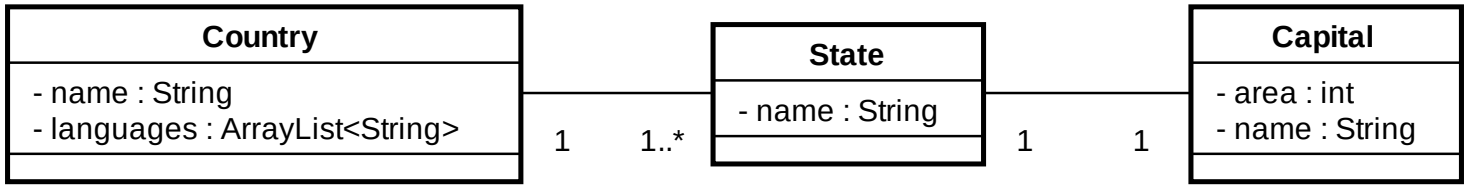
Exercise

- Draw the class diagram for the following scenario:
  - An author writes one or more books.
  - Each book is written by one or more authors.

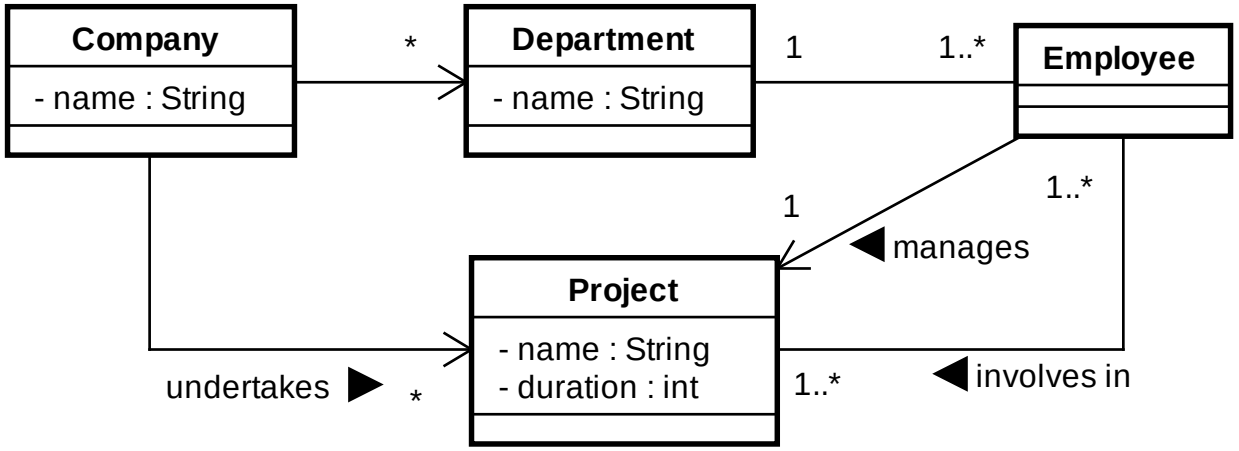
Exercise

- Translate the following class diagram to Java code.

A.



B.



Object Relationships: Composition

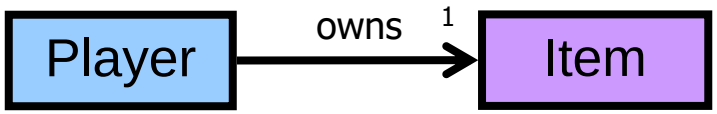
- Composition
  - "is-a-part-of" Relationship
  - Models the notion of one object "owning" another and thus being responsible for the creation and destruction of another object.

Object Relationships: Association

- "has-a" relationship

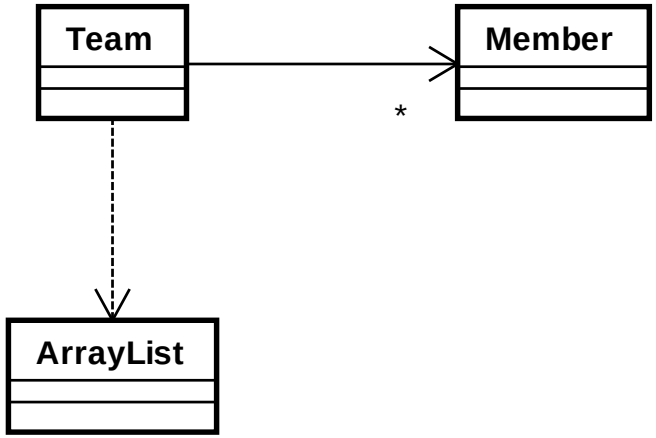
```
public class Player {
  private Item item;

  // ...
}
```

```
public class Item {
  // ...
}
```



Object Relationships: Dependency

- "uses" relationship
- A change to the definition of one class may ca changes to the other class.
  - One class sends a message to another
  - One class has another as part of its data
  - One class mentions another as a parameter




```
import java.util.*;

public class Team{
  private ArrayList<Member> members;


}
```

Summary

- Class versus Object
  - A class is the blueprint from which individual objects are created.
  - An object is an instance of a class.
- Principles of Object Orientation
  - Abstraction
    - determines the relevant properties and features while ignoring non-essential details.
  - Encapsulation
    - Separate components into external and internal aspects.
  - Modularity
    - Break something complex into manageable pieces.
  - Hierarchy
    - Ranking or ordering of objects
- A model is a simplification of reality.
- UML is a standard language used for modeling software
  - Facilitates communication
  - Manage complexity
- Class diagrams shows the static view of a system
  - An association is represented by a reference variable in the Java code.

References

- UML Distilled (http://blue.smu.edu.sg/distilled)
  - Chapter 3. Class Diagrams: The Essentials

- UML basics: The class diagram
  - http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/