Cairo University
Faculty of Computers and Information

CS251 – Software Engineering
Lecture 2: OOP Overview

Slides by

Mohammad El-Ramly, PhD

2015

http://www.acadox.com/join/75UDWT

Lecture Objectives

1. Learn the Characteristics of Object-Orientation
    1. Identity
    2. Classification
    3. Abstraction
    4. Encapsulation
    5. Aggregation
    6. Inheritance
    7. Polymorphism
    8. Genercity
2. Learn the OO Development Methodology
    - Steps of OOP Development
3. Examples of OO Modeling

Previous Lecture

1. Definition of Software Engineering
2. Software Body of Knowledge
   - Software Requirements
   - Software Design
   - Software Construction
   - Software Testing
   - Software Maintenance
   - Software Configuration Management
   - Software Engineering Management
   - Software Engineering Process
   - Software Engineering Tools and Methods
   - Software Quality
   - Software Engineering Professional Practice
   - Software Engineering Economics

أنت لا تعيش لكي تتنفس ، بدون التنفس ستموت طبعا ،
لكن التنفس ليس هدفك!
وأنت لا تعيش لأجل جمع المال ،
بدون المال يتكون حياتك صعبة طبعا،
لكن جمع المال ليس هدفك !
إذا ما هو هدفك ؟!

What is OO ?

- ***Object-orientation*** is a way of thinking about problems using models built from real-world concepts.
- The fundamental unit is the ***Object***
- An object has ***data*** and ***behavior***
- ***OO Software*** means we write our program in terms of objects, each tightly integrates data and operations on the data
- In ***Structured programming***, data and operations on the data were separated or loosely related.

I. OO Characteristics

1. Identity
2. Classification
3. Abstraction
4. Aggregation
5. Encapsulation
6. Inheritance
7. Polymorphism
8. Genercity

1.1 Identity

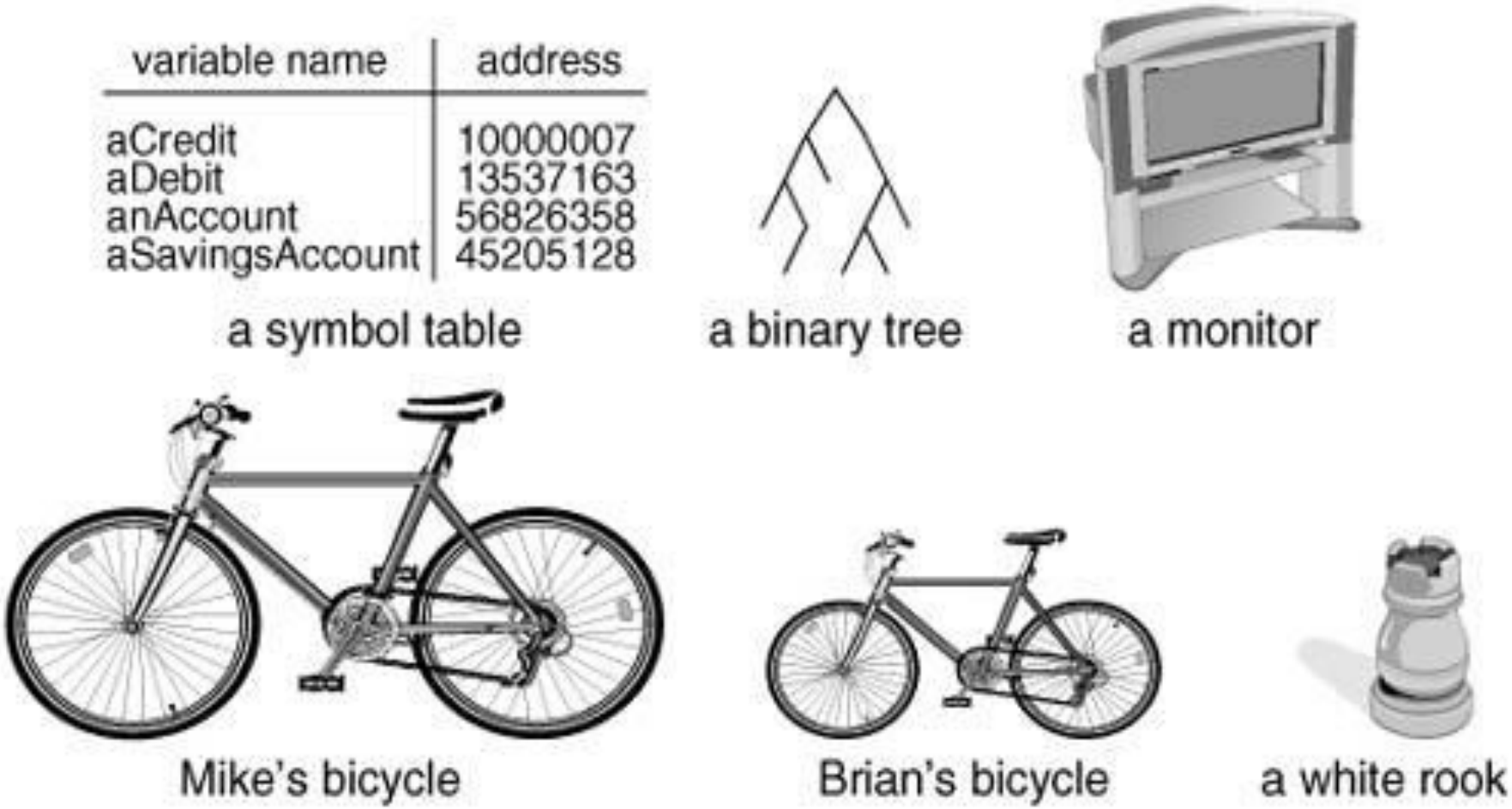


Figure 1.1 Objects. Objects lie at the heart of object-oriented technology.

Identity

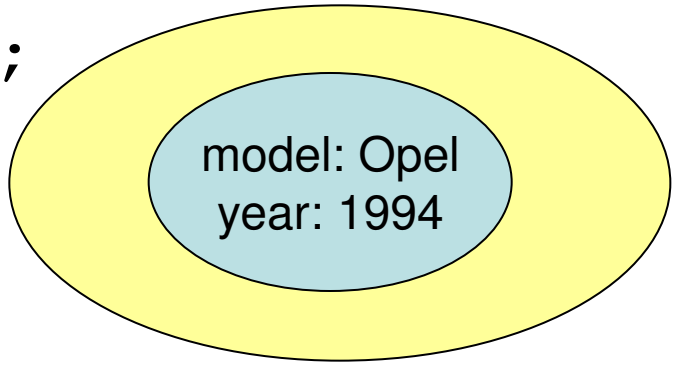
- ***Identity*** means that data is quantized in into discrete, distinguishable, entities called ***objects***
- An object can be ***concrete*** like a ***car***, a ***file***, …
- An object can be ***conceptual*** like a ***feeling***, a ***plan***,…
- Each object has its own ***identity*** even if two objects have exactly the same ***attributes***. They are still different separate objects.

Ali's car

Khaled's car

Identity

- ***Object identity*** is the property by which each object can be identified and treated as a distinct software entity
- Each object has something unique which distinguishes it from all its fellow objects. It is its memory address (or ***handle***) that is referred to by one or more ***object identifiers*** (OID)
- **`Car  myCar("Opel", 2005);`**
- The object handle is 0x00FDA610 is referenced by an object identifier **`myCar`**

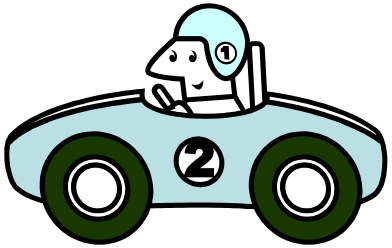


1.2 Classification

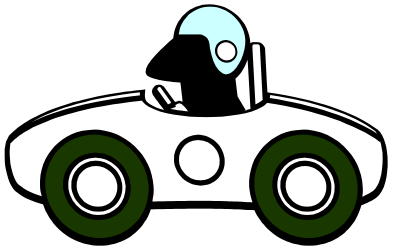
- ***Classification*** means that objects with the same data structure (***attributes)*** and behavior (***operations***) belong to the same ***class***
- A class is an ***abstraction*** that describes the properties important for a *specific* application
- The choice of classes is arbitrary and application-dependent.

Mina's car

Ali's car



Samir's car



A Car

Classification

- Each object is an ***instance*** of a class
- Each object has a reference to its class (knows which class it belongs to)

Figure 1.2 Objects and classes. Each class describes a possibly infinite set of individual objects.

1.3 Abstraction

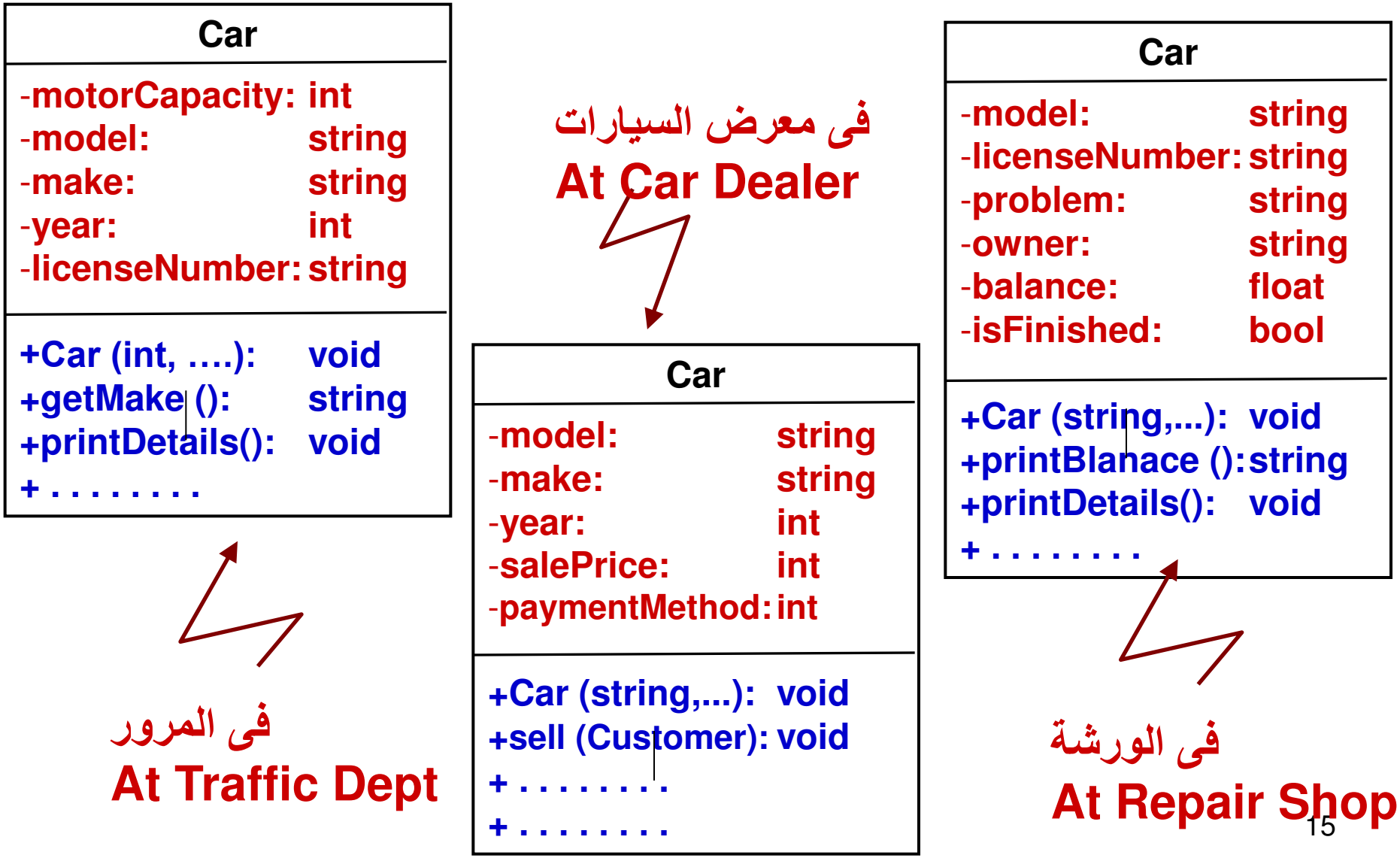
- ***Abstraction*** is the selective examination of certain aspects of a problem.
- Abstraction aims to isolate the aspects that are important for some purpose and suppress the unimportant aspects.
- The purpose of abstraction determines what is important and what is not.

Abstraction

- All abstractions are incomplete and inaccurate.
- In modeling, do not search for the truth but for adequacy for some purpose.
- There is *no single correct model* for a problem. Only adequate and inadequate ones.
- A ***good model*** captures the crucial aspects of a problem and omits the rest.
- A class abstracts a real concept *according to the needs of a specific application.*

Abstraction

Different abstractions for the concept of a car according to the application.

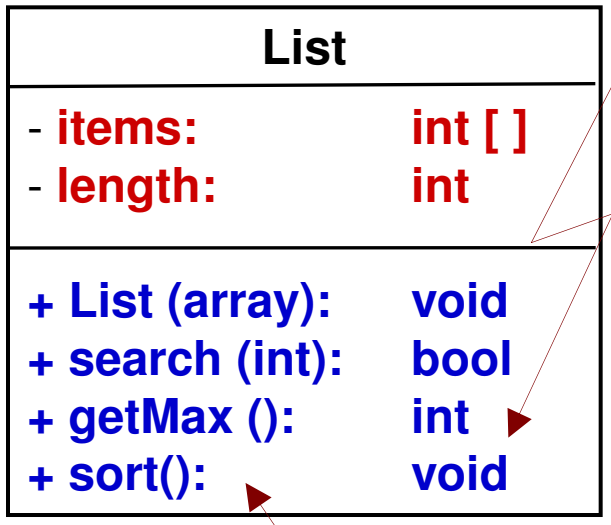


1.4 Encapsulation

- ***Encapsulation*** separates the external aspects of an object, that are accessible to other objects, from the internal implementation details that are hidden from other objects.
- Encapsulation reduces ***interdependency*** between different parts of the program.
- You can *change the implementation* of a class (to enhance performance, fix bugs, etc.) *without affecting* the applications that use objects of this class.

Encapsulation

It allows you to replace an algorithm with a faster one while keeping the class interface (public methods) the same.




```
void sort () {   // Bubble Sort
    int i, j;
    for (i = length - 1; i > 0; i-) {
        for (j = 0; j < i; j++) {
            if (items [j] > items [j + 1]) {
                int temp = items [j];
                items [j] = items [j + 1];
                items [j + 1] = temp;
            }
        }
    }
}`
```

```
void sort () {   // Quick Sort
    ……..
}`
```

Encapsulation

List	
- items:	int [ ]
- length:	int
+ List (array):	void
+ search (int):	bool
+ getMax ():	int
+ sort():	void

List	
items:	vector<int>
+ List (array):	void
+ search (int):	bool
+ getMax ():	int
+ sort():	void

It allows you to replace a data item with another one while keeping the class interface (public methods) the same.

```
void sort () {  // Bubble Sort
    int i, j;
    for (i = items.size()- 1; i > 0; i-) {
        for (j = 0; j < i; j++) {
            if (items [j] > items [j + 1]) {
                swap (items [j], items [j + 1]);
            }
        }
    }
}
```

Encapsulation

- ***Data hiding.*** Information from within the object cannot be seen outside the object.
- ***Implementation hiding.*** implementation details within the object cannot be seen from the outside.

1.5 Aggregation

- Aggregation is the ability to create new classes out of existing classes
  - Treating them as building blocks or components
- Aggregation allows reuse of existing code
- Aggregation describes a *"has a"* relationship. One object is a part of another object.

A car has wheels.

Aggregation

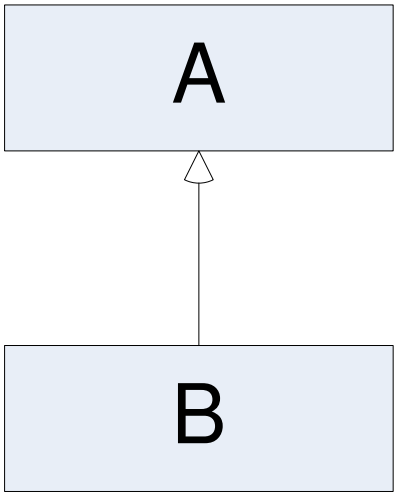
- Two forms of aggregation
  - Whole-Part relationships
    - Car is made of Engine, Chassis, Wheels
    - *composite* aggregation (the composite “owns” the part)
  - Containment relationships
    - A Shopping Cart contains several Products
    - A List contains several Items
    - *shared* aggregation (the part is shared by more than one composite).

1.6 Inheritance

- ***Inheritance*** is the sharing of ***features*** (attributes and operations) among classes based on a hierarchical relationship.
- A ***superclass*** (also ***parent*** or ***base***) has general features that ***sublcasses*** (***child*** or ***derived***) refine and elaborate.
- Each subclass ***inherits*** all the features of its superclass.
- Inheritance is one of the strongest features of OO technology.

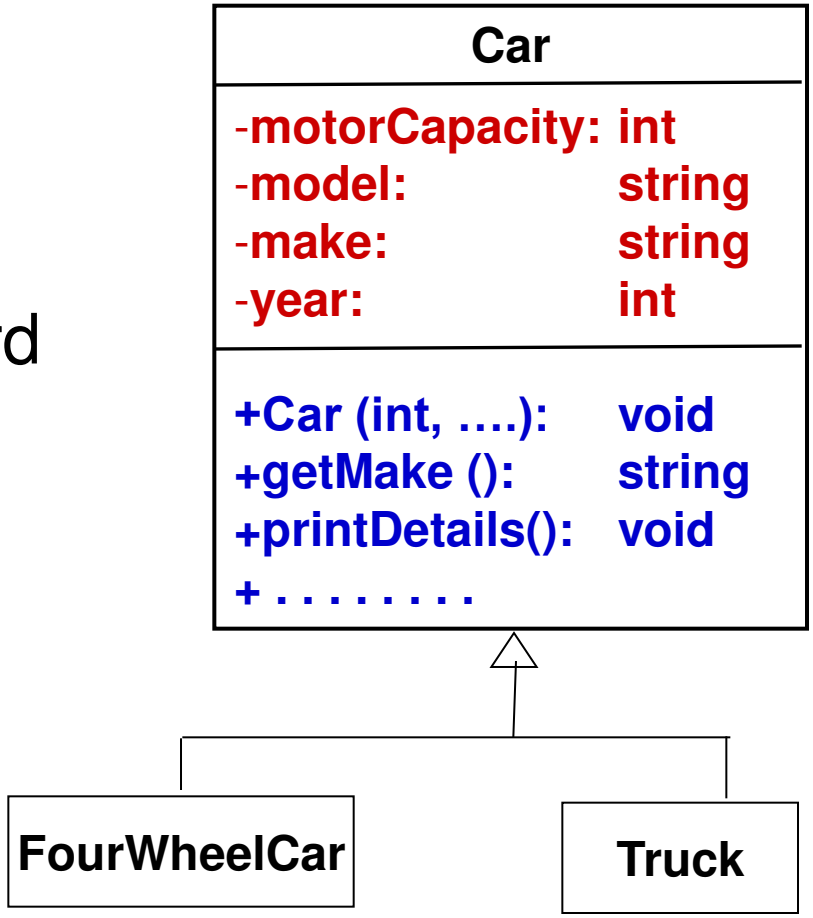
Inheritance

- Inheritance is the facility by which objects of a class (say B) may use the methods and variables that are defined only to objects of another class (say A), as if these methods and variables have been defined in class B
- Inheritance is represented as shown in ***UML*** notation.



How to use Inheritance?

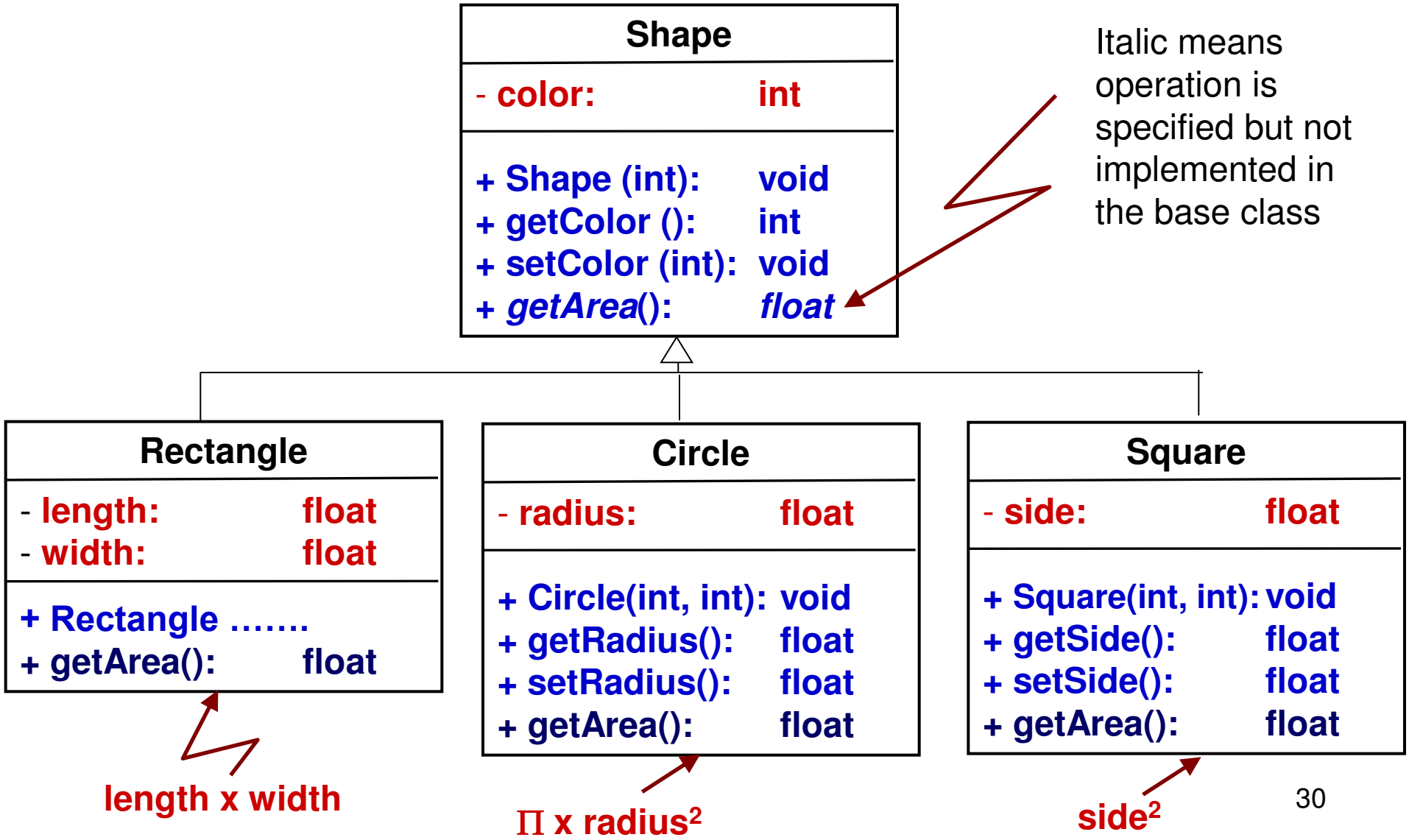
- Inheritance helps building software incrementally:
- *First*; build classes to cope with the most straightforward (or general) case,
- *Second*; build the special cases that inherit from the general base class. These new classes will have the same features of the base class plus their own.



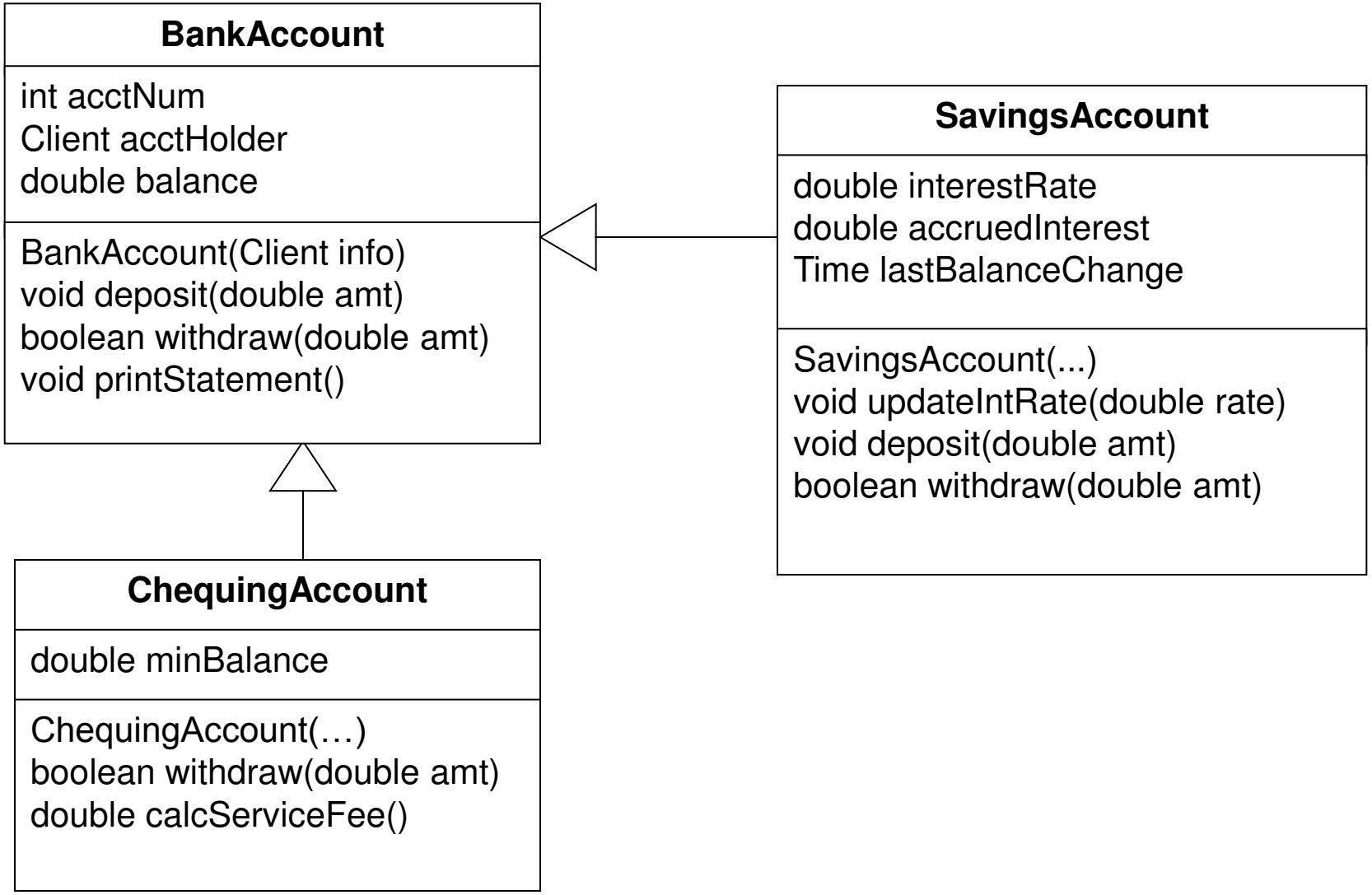
1.7 Polymorphism

- ***Polymorphism*** means that the same operation may behave differently for different classes.
- An ***operation*** is a procedure or a transformation that the object performs or is subject to.
- An implementation of an operation by a specific class is called a ***method***.
- Because an OO operation is polymorphic, it may have more than one method for implementing it, each for a different class.

Polymorphism

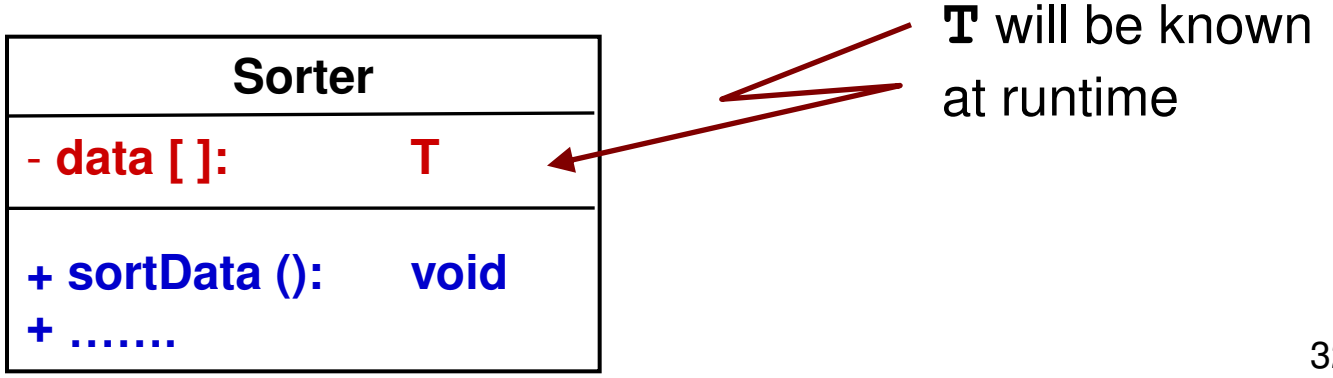


Inheritance, polymorphism and aggregation ?



1.8 Genericity

- ***Genericity*** is the construction of class ***A*** so that one or more of the classes it uses internally is supplied only at run-time (at the time that an object of class ***A*** is instantiated)
- This means that the class is ***parameterized***, i.e., the class gets a parameter which is the name of another type.



II. OO Development Methodology

- **Why building models?**

1. To test the system before building it
2. To communicate with the customer
3. To visualize your ideas
4. To reduce complexity

OO Development Stages

- System conception → **There is a need or idea**
- Analysis → **What will / will not be done**
- System design → **Overall architecture**
- Class design → **Detailed design**
- Implementation → **Programs**
- Testing → **Make sure it works well**
- Training → **Train the end user to use it**
- Deployment → **Install it**
- Maintenance → **Fix bugs & add functions to stay relevant**

http://www.computerweekly.com/news/2240105488/Olympic-software-engineers-enter-final-leg-of-marathon-IT-development-project

Class Model

- A ***class model*** captures the static structure of the system by characterizing
  - the *objects* in the system,
  - the *relationships* among the objects and
  - the *attributes* and *operations* for each class of objects
- Class model is the *most important* UML model
- **UML** is Unified Modeling Language for OO systems

Object and Class Concepts
Objects

- The class model describes the system's ***objects***
- Objects often appear as ***proper nouns*** in the problem description or discussion with the customer.
- Some object correspond to ***real world entities*** (Dr El-Ramly, Oxford University, the old turtle in the zoo)
- Some objects correspond to ***conceptual entities*** (the formula for solving an equation, operand checker, etc.)
- The choice of objects depends on the analyst's judgment and the problem in hand. There can be *more than one correct representation*.

Objects

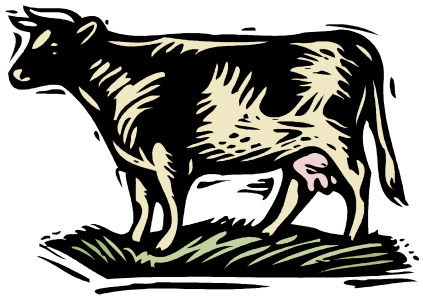
- Objects have ***identities*** and are distinguishable from each other
- Each object has a memory address that is referred to by one or more ***object identifiers*** (OID)

تفاحة كمان تفاحة و كمان تفاحة

Class

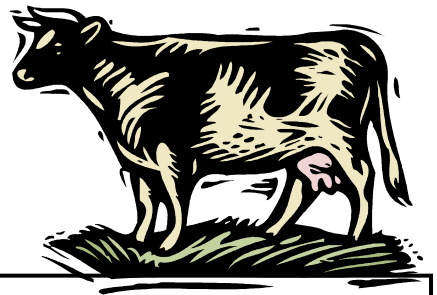
- An object is an ***instance of*** or ***occurrence*** of a class
- A class describes a group of objects with the same
  - Properties (attributes)
  - Behavior (operations)
  - Kinds of relationships and
  - Semantics
- *Person, Company and Window* are all classes
- Classes often appear as ***common nouns*** and ***noun phrases*** in problem description and discussion with customers or users

Class



- Objects in a class share a common ***semantic*** purpose in the system model
- Both car and cow have *price* and *age*
- If both were modeled as pure *financial assets*, they both can belong to the same class.
- If the application needs to consider that:
  - Cow eats and produces milk
  - Car has speed, make, manufacturer, etc.
- Then model them using separate classes.
- ***So the semantics depends on the application***

Class



Financial Asset
-type: int -age: float -currentValue: float -. . . -. . .
+getCurrentValue: int +printDetails(): void + ...

في معرض
السيارات

Car at Dealer

Car
-model: string -make: string -year: int -maxSpeed: int -paymentMethod: int
+Car (string,...): void +sell (Customer): void + ... + ...

Cow
-wight: float -age: string -spices : string -owner: string -gender: char -isPregnant: bool
+Cow (string,...): void +getOwner (): string +printDetails(): void + ...

البقرة و السيارة
فى نفس الكلاس

Cow & Car
Same Class

فى المزرعة

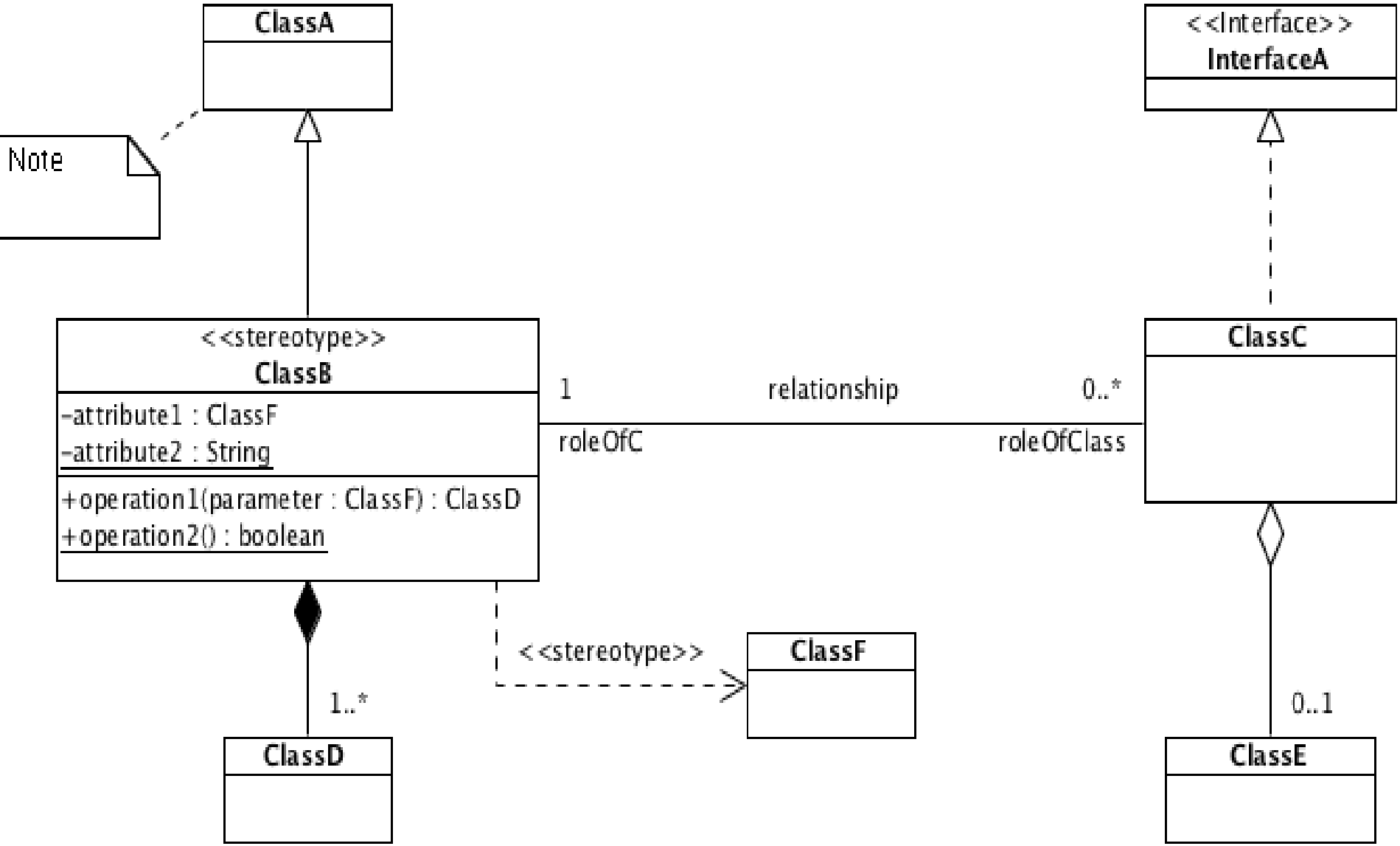
Cow in Farm

Problem Domain

- ***Problem domain*** is the set of objects, events and relations that define the environment of the problem.
- It is very very important to understand the problem domain before solving the problem.
- If needed get a *domain expert*.
- **An important task in OOP is identifying the classes that will represent the problem domain**
- This is not required for all classes but for the classes related to the problem solution.

UML Class Diagram Summary

- UML is the dominant OO modeling language today.



UML Class Diagram Links

- http://www.idt.mdh.se/kurser/cd5490/2004/lectures/UML%20Intro.pdf
- http://www.step-10.com/SoftwareDesign/UML/UMLClassDiagramsASummaryOfTheBasics.html
- http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/

III. Examples
Example 1: Car Service Quotes

- Ali's Mechanic Shop services expensive cars like Mercedes and BMW. When a customer brings a car to the shop, the manager gets the customer's name, address and phone number. Then the manager fills the car make, model and year. After inspection, the manager gives the customer a service quote. The quote shows the estimated parts charges, estimated labor charges and the total estimated charges.

Example 1: Car Service Quotes

1. Read the problem description very carefully
2. Identify all the nouns in the problem description
3. Refine the noun list
   - Remove nouns that represent the same concept
   - Remove nouns that represent concepts irrelevant to the problem solution
   - Remove nouns that represent objects or instances of classes
   - Remove nouns that can be data fields in classes represented by primitive data types
4. Identify the responsibilities of each class

Example 1: Car Service Quotes

- Ali's Mechanic Shop services expensive cars like Mercedes and BMW. When a customer brings a car to the shop, the manager gets the customer's name, address and phone number. Then the manager fills the car make, model and year. After inspecting the car, the manager gives the customer a service quote. The quote shows the estimated parts charges, estimated labor charges and the total estimated charges.

Example 1: Car Service Quotes

- Ali's Mechanic Shop services expensive cars like Mercedes and BMW. When a customer brings a car to the shop, the manager gets the customer's name, address and phone number. Then the manager fills the car make, model and year. After inspecting the car, the manager gives the customer a service quote. The quote shows the estimated parts charges, estimated labor charges and the total estimated charges.

Example 1: Car Service Quotes

- Ali's Mechanic Shop services expensive cars like Mercedes and BMW. When a customer brings a car to the shop, the manager gets the customer's name, address and phone number. Then the manager fills the car make, model and year. After inspecting the car, the manager gives the customer a service quote. The quote shows the estimated parts charges, estimated labor charges and the total estimated charges.

Example 1: Car Service Quotes

- Ali's Mechanic Shop services expensive cars like Mercedes and BMW. When a customer brings a car to the shop, the manager gets the customer's name, address and phone number. Then the manager fills the car make, model and year. After inspecting the car, the manager gives the customer a service quote. The quote shows the estimated parts charges, estimated labor charges and the total estimated charges.

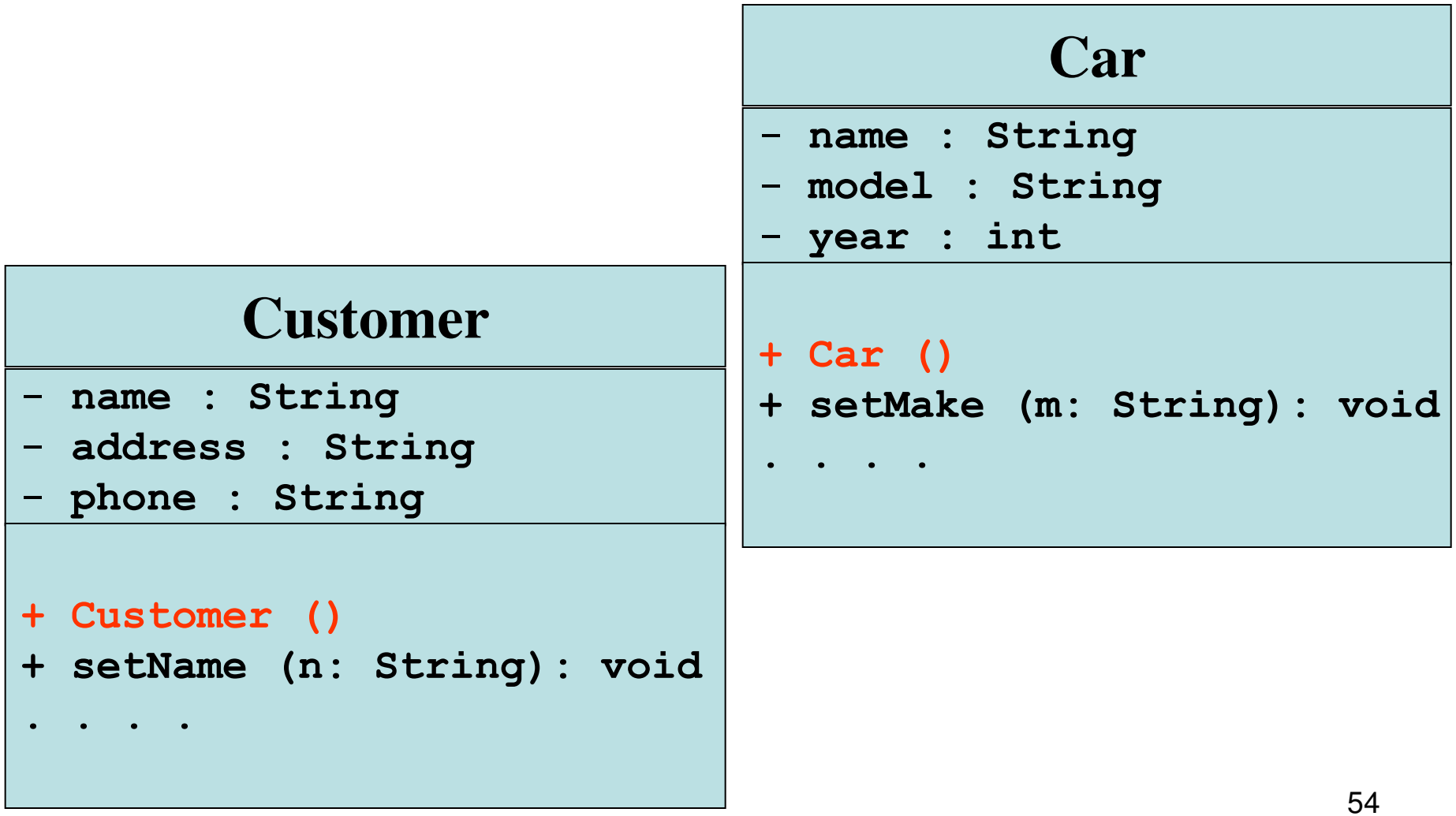
Example 1: Car Service Quotes

cars

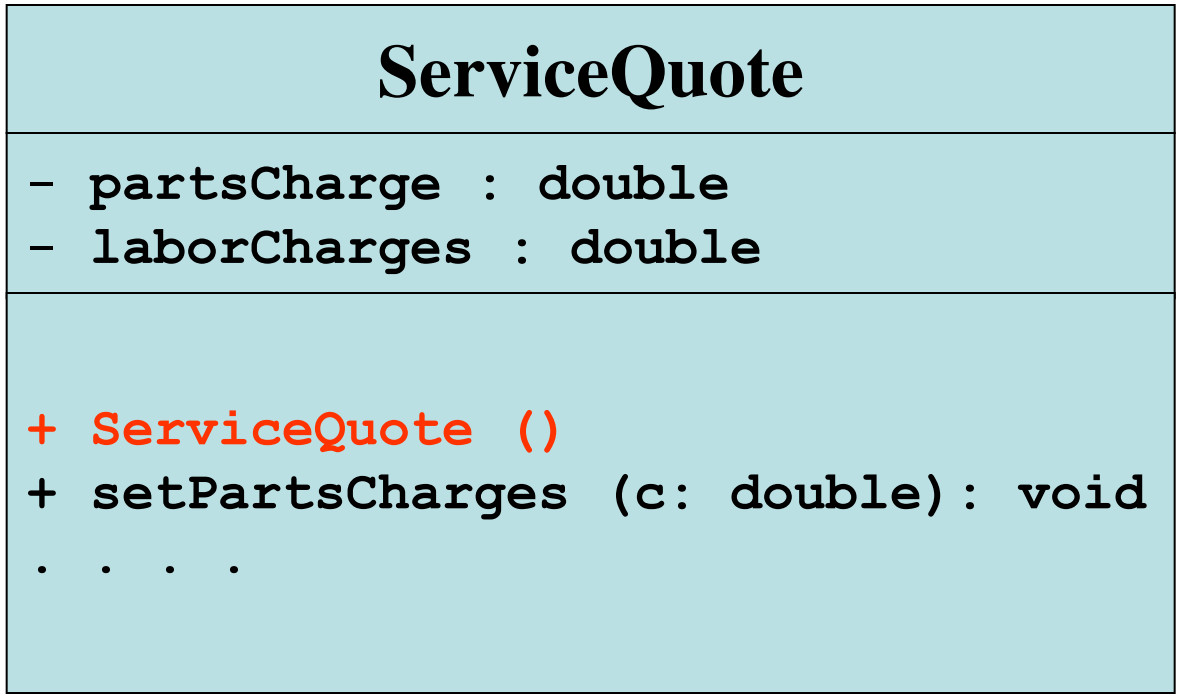
customer

service quote

Example 1: Car Service Quotes



Example 1: Car Service Quotes



Lecture Objectives

1. Learn the Characteristics of Object-Orientation
    1. Identity
    2. Classification
    3. Abstraction
    4. Encapsulation
    5. Aggregation
    6. Inheritance
    7. Polymorphism
    8. Genercity
2. Learn the OO Development Methodology
    - Steps of OOP Development
3. Examples of OO Modeling

Pearls of Wisdom

- Singapore has only people
- Skills is what you should get out of here not degree
  - Loving and caring about others
  - Sharing knowledge with others
  - Teamwork and cooperation
  - Self-discipline
  - Self-learning and self-motivation
  - Research skills
  - Computer and IT stuff

Next Lecture

1. Chronic Software Crisis
2. Arian 5
3. Therac 25
4. London Ambulance System
5. Vasa

To Learn OOP
Read, Watch and Solve

1. Review OOP videos and slides

2. Reading 1 – OOP in Java
    1. 7.1 to 7.8 && 7.12 to 7.16

For Next Time
Read, Watch and Solve

1. Watch Software Crisis videos

2. Reading 2 – Ariane 5
3. Reading 3 – London Ambulance System
4. 1nline till 23 Oct.

كيف تنجح فى هذا الكورس

1. جدد النية و استعن بالله و لا تعجز
2. نظم وقتك و خصص وقت و مكان مناسبين للدراسة وفق جدول ثابت يناسب ظروفك
3. شاهد فيديوهات المحاضرات و ذاكر للمحاضرة قبل أن تأتى
4. شارك فى تمارين المحاضرة و استفد من زملائك
5. اعمل تمارين المعمل سواءا فى الكلية أو البيت
6. شارك بقوة فى مشروع المقرر
7. **كن مستعدا للتغيير و تبني عقلية النجاح.**

- التغيير يحتاج لممارسة العادة الجديدة 30 يوماً

كن مستعدا للتغيير و تبني عقلية النجاح .....

- **التغيير الأول**
  - نام بكير فيق بكير شوف الصحة كيف بتصير
  - بورك لأمتى فى بكورها
  - فوائد النوم المبكر
    - GPA أعلى بدرجة كاملة
    - أكثر إيجابية
    - أكثر قدرة على مواجهة المشاكل
    - أكثر تفاؤلا
    - نوم أعمق و أفضل

http://www.forbes.com/pictures/gglg45gfd/benefits-of-early-risers/

كن مستعدا للتغيير و تبني عقلية النجاح .....

- **التغيير الثانى**
  - موعد مع الله
  - تخيل الرئيس أو الملك يقابلك كل يوم صباحا يعطيك عهده و أمانه و يسمع طلباتك.
  - قال رسول الله ص: من صلى الصبح فهو في ذمة الله ، فلا يطلبنكم الله من ذمته بشيء, فيدركه, فيكبه في نار جهنم.