Data Abstraction and Object-Oriented Languages

Prof. Evan Chang
Meeting 27, CSCI 3155, Fall 2009

Announcements
- HW10 out, practice only
- Review for final on Thu. Come with questions.
- Final preparation
  - Midterm 1 and 2
  - Homeworks and Project
  - Skills List
  - Midterms/finals from prior semesters
- TA FCQs in recitation today

One-Slide Summary
- An abstract data type (ADT) is a type definition with a set of operations manipulating values of that type
- The representation or implementation type is hidden from clients. This is known as information hiding.
- An ADT allow developers to impose a representation invariant—a property of the data structure preserved across operations
- Objects give rise to a related but different form of data abstraction

User-Defined ADTs
- Example: A stack of values
  - an empty value
  - push a new value
  - pop a value

ADTs in SML
- Signatures and Structures
Data Abstraction in OO Languages

• Java-like languages permit information hiding through access control

Why is Information Hiding Useful?

• user doesn’t do more than it says
• can’t violate any invariant you have
• can change representation (hidden fields) without affecting client code

What is different about Abstract Data Types and Objects with public interfaces?

• i.e., SML-style data abstraction and Java-style data abstraction

One-Slide Summary

• An abstract data type (ADT) is a type definition with a set of operations manipulating values of that type
• The representation or implementation type is hidden from clients. This is known as information hiding.
• An ADT allow developers to impose a representation invariant—a property of the data structure preserved across operations
• Objects give rise to a related but different form of data abstraction
About Your Classmates

• Favorite class is Philosophy of Science
• Lived in China
• Works on cars (used to have a Mustang)
• Enjoys poking old people softly to see whether or not they are dead

Class-Based Object-Oriented Languages

One-Slide Summary

• An object is a record of data fields and operations.
• Operations on objects are called methods (or messages). Methods are self-referential: they can refer to the object through a “self” reference (this in Java)
• Inheritance is a way to reuse code and is a defining feature of class-based object-oriented languages.
• A class that is defined through inheritance is a derived class or a subclass (inherits from a parent class or a superclass).
• Dynamic dispatch (or dynamic binding of messages, virtual method dispatch) with overridden methods is another defining feature

Inheritance

• Want to define a class B that is just slightly different from class A. What are things you might want to change?
  ✔ add fields (toys, ...)
  ✔ change type of a field
  ✔ add methods
  ✔/x change method implementations
  ✗ delete field
  ✗ delete method

Does subclassing imply subtyping?

• Adding methods? ✔ Yes
Does subclassing imply subtyping?

- Dropping methods? No
  ```
  f, g 3 \leq x + 3
  ```

Does subclassing imply subtyping?

- Overriding methods?
  ```
  class C {
  void p() { print "I'm C"; 3 
  }

  class D extends C {
  void p() { print "I'm D"; 3 }

  
  C c = new D();
  c.p();
  ```

Dynamic dispatch

```
class C extends void p() { print "I'm C"; 3

C c = new D();

C.
p();
```