CS 617 Object Oriented Systems
Lecture 2
Abstractions, Interfaces, Implementations
3:30-5:00pm Monday, Jan 7

Rushikesh K Joshi

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

Outline

Outline

What's abstraction?

Is it for the first time that we see it,
through object orientation?

What's abstraction?

No, we knew it even before.
The process of abstraction and its outcomes- the abstractions themselves, are very much part of our day to day thinking, and practice.

Some Example Abstractions

- 12 Notes in Music
- A Raaga
- Mathematical models representing natural phenomena
- Course description in terms of course content and lecture breakup
- Menu card in a hotel
- A Manual describing a processor's behavior in terms of signals to and from its pins
- Regions in a Country
- Classification of Species
- Classification of students in IIT
- Any other examples?

Outline

Some Properties Associated with Abstractions I

- Existence of a more detailed, more complex system

$\rightarrow$ In actuality the system can be quite complex. The actual system is referred to as **Implementation**

Some Properties Associated with Abstractions II

- Existence of a simpler view of the actual

$\rightarrow$ The actual system is perceived through this view. The user of this view can handle the system without having to handle the internal details of the system. Using the external view is thus simpler than having to use the actual system. The view is referred to as **Interface**

Some Properties Associated with Abstractions III

- One can understand, refer to, or use the system by abstracting it out

$\longrightarrow$ Abstraction is useful in understanding complex systems as well as in manipulating them.

Some Properties Associated with Abstractions IV

- Abstractions can form hierarchies

$\rightarrow$ A class of abstractions can be further abstracted out. A more abstract view is called a **Generalization** of its lesser abstract more detailed entity. (Is-A Hierarchy)

Some Properties Associated with Abstractions V

- A system may be composed of many subsystems

$\rightarrow$ Each of the subsystems may be represented by their respective abstractions, while the whole has an abstraction of its own (Part-Whole Hierarchies)

Outline

Abstractions in Programming

Examples?

Abstractions in Programming

- Types
- Structures
- Functions, Procedures, Processes
- Data Structures
- Conditional Branching
- Loops
- Files, ..

We have Control Abstractions and Data Abstractions

Abstractions that you see in OOPLs

- A conceptual layer (of abstractions) is introduced
- In this layer, we don't separate data from control
- Instead, a system is decomposed in terms of objects that respond to external interactions
- Focus in given on external interactions with objects
- At the lower levels of detailing, the separation of data from control can be seen

A new way of modularizing software artifacts

Outline

Defining Abstractions I

- Define an entity type in terms of the behavior (operations) applicable to all its possible values, examples or instances
- Do not worry about the implementation
- Assume that no other operations that will break the meaning of the entity being defined are possible
- Examples:
  - Vending machine: drop a coin, choose an item
  - Fan in your room: switch on, switch off, set speed
  - Window: resize, move, enlarge, shrink, hide, show, terminate
  - File: open, close, read, write

Defining Abstractions II

- File System: create subdirectory, create file, delete file/dir, secure file/del..
- Online Registration System (Student): select course, confirm, unselect course,..
- Online Registration (Facad): approve, disapprove,..
- Online System (ASC): create course, assign slot,..

Outline

Another Property Associated with Abstractions

An entity may be perceived simultaneously through different abstractions $\longrightarrow$ Multiple Interfaces $\longrightarrow$ Role Modeling

Outline

Expressing Interfaces

- In terms of collections of function prototypes
- Merely syntactic representations
- In actuality, there are meanings associated with each operation which is part of the abstraction represented by the interface
- Can there be semantically richer specifications of interfaces?

An Example Interface: A Simple Account

```
interface Account {
    boolean deposit (int amount);
    boolean withdraw (int amount);
    int balance();
}
```

An Example Interface: A Fork

```
interface Fork {
    void pickup ();
    void putdown ();
}
```

An Example Interface: A Server

```
interface Server {
    ResultPacket request(ReqNumber t, ParamPacket p);
}
```

An Example Interface: A Name Server

```
interface NameServer {
    boolean addName (Name n, Location l);
    Location getLocation (Name n);
}
```

An Example Interface: An Event Server

```
interface EventServer {
    complete this!
}
```

An Example Interface: Suppliers and Consumers in an Event based System

```
interface PushSupplier {
    ..
}
interface PushConsumer {
    ..
}
interface PullSupplier {
    ..
}
interface PullConsumer {
    ..
}
```

Many Abstractions in a System of Many Components/Objects

- Many entities (objects)
- They interact
- The interactions occur through the interfaces
- Each entity perceives the other through an abstraction of the other

Outline

Readings

- OMG, Event Service Specification, Oct. 2004, http://www.omg.org