

---

## Lecture 5: Designing Classes

Dragan Mirkovic  
Department of Computer Science  
University of Houston

---

D. Mirkovic, C++ Programming, Spring 2005

## Announcements

---

- Today:
  - Designing classes in OOL
  - Ch 5. In Irvine.
  - Midterm exam
    - Thursday 3/10/2005

---

D. Mirkovic, C++ Programming, Spring 2005

# Introduction

---

- So far we have discussed the syntax and semantics of C++
- Here we discuss the basic ideas on the design of classes and components
- Object model
- How to build components

---

D. Mirkovic, C++ Programming, Spring 2005

# Analysis, Design and Programming

---

- OO programming concentrates on the physical entities that make up an application
- OO design: method that uses oo decomposition and abstraction to implement a model of a system
- OO decomposition:
  - Using class and object abstractions to logically structure a system
- OO analysis:
  - Examination of application requirements in terms of classes and objects

---

D. Mirkovic, C++ Programming, Spring 2005

# The Object Model

---

- Definition:
  - Programming approach that uses principles of abstraction, modularity, hierarchy and typing
- Very efficient in designing complex applications
  - Greater demands on software and simplified approach for the programmers
  - Better productivity
  - Improved maintenance

---

D. Mirkovic, C++ Programming, Spring 2005

# Abstraction

---

- Simplified view of an object
  - Only the essential characteristics are modeled
- The choices depend on the application and the environment
- Examples:
  - Programming language
    - Shields the programmer of the hardware complexities
  - Variables, class interfaces,

---

D. Mirkovic, C++ Programming, Spring 2005

## Abstraction types

---

- Two general types of abstractions:
  - Control (action) type:
    - Operations (insert, move, resize, find, ...)
  - Entity type
    - Specific static elements in the problem domain
    - Easy to recognize (correspond to real objects)
    - Examples:
      - Window, display, cursor, screen, ...
      - Employee, Student, Course, ...

---

D. Mirkovic, C++ Programming, Spring 2005

## Encapsulation

---

- Enclosing of both variables and functions inside a class with a well defined interface
- Achieved through information hiding
  - Implementation details are hidden
  - User interface is public and well defined
- Leads to standardization of interfaces
  - Great improvement in software development

---

D. Mirkovic, C++ Programming, Spring 2005

# Modularity

---

- One of the oldest and best concepts in programming
  - Compilation units (sharing and hiding of information)
  - Subroutines (common subtasks used in different parts of the application)
  - Modules
- Procedural languages achieve modularity by using procedural abstractions
  - Defining problem in terms of its operations

---

D. Mirkovic, C++ Programming, Spring 2005

# Hierarchy

---

- Hierarchy (inheritance)
  - Parent-child relationship between classes
  - Derived classes:
    - Inherit operations and data from their base classes
    - Only additional attributes need to be defined
- Example:
  - Point 

x	y
---	---
  - Point3D 

x	y	z
---	---	---

---

D. Mirkovic, C++ Programming, Spring 2005

# Typing

---

- Mechanism for checking of compatibility between objects
- Strongly typed languages have strict rules on the possible assignments
- Many conversions are performed automatically
- In C++ we can define conversion constructors
- Example:

```
Student S;  
Employee E = S; // Error
```

– Unless

```
class Employee {  
public:  
    Employee( const Student & S );  
    ...  
}
```

---

D. Mirkovic, C++ Programming, Spring 2005

# Basic principles of OO design

---

- Efficient design of complex applications is very difficult
  - They cannot be coded directly
- Fast pace in hardware development has created a bottleneck in software design
- It is very important to organize the design process in an efficient way
  - Better productivity
  - Easier maintenance
- OO approach has been recognized as one of the most efficient ways for designing complex applications

---

D. Mirkovic, C++ Programming, Spring 2005

# Software design goals

---

- **Simplicity**
  - Well designed and meaningful objects
  - User friendly class interfaces
- **Flexibility**
  - Easy accommodation of changes
- **Extensibility**
  - Additions are easy to implement without destroying initial design
- **Portability**
  - Hardware independence
- **Reusability**
  - Well designed software can be used in large number of applications

---

D. Mirkovic, C++ Programming, Spring 2005

# Designing OO Programs

---

- OO program should be a faithful model of reality
  - It is important to identify key concepts in an application
  - Construct corresponding classes
  - Iterative process
- Traditional approach:
  - Top-down procedural design
  - Dependencies are hierarchical
  - Changes may be difficult to implement

---

D. Mirkovic, C++ Programming, Spring 2005

# Components

---

- Component = group of related classes that work together
  - Well defined interfaces
- Design steps:
  - Find the classes
  - Specify class operations
  - Specify class dependencies
  - Specify class interfaces

---

D. Mirkovic, C++ Programming, Spring 2005

# Find the Classes

---

- This is the most obvious step in the design
  - If the class is not obvious it shouldn't be there!
- Discovery of inherent structure of the application
- It is important to understand the application well

---

D. Mirkovic, C++ Programming, Spring 2005

## Specify class operations

---

- Basic set of operations appear naturally in C++
  - Constructors, initializers, destructors, modifiers, etc.
- More difficult task
  - Need to anticipate the interaction between classes
- C++ advantage:
  - If the initial design is not sufficient it can be easily refined

---

D. Mirkovic, C++ Programming, Spring 2005

## Class dependencies

---

- Relation between different classes
- Three types of dependencies:
  - Inheritance
    - Derived classes
    - Example: `GraduateStudent` is a specialized type of the `Student` object.
  - Composition
    - One object contains a copy of another object
    - Example: `student` object contains `Address` object
  - Link
    - Independent objects communicating with each other
    - Example: `student` object passes messages to `CourseCatalog` object.

---

D. Mirkovic, C++ Programming, Spring 2005

# Class interfaces

---

- Function prototypes for member functions define the class interface
- The interfaces can be shared with other programmers
- Efficient design

---

D. Mirkovic, C++ Programming, Spring 2005

# Example

---

- Doctor's Office Scheduling
- 1<sup>st</sup> Step: Specifications
  - Automatically schedule appointments for patients
  - Multiple patients and doctors
  - 15 min time slots between 8:00 am and 6:00 pm.
  - Print out a separate daily schedule for each doctor
  - Interactive program with screen i/o and file output.

---

D. Mirkovic, C++ Programming, Spring 2005

## 2<sup>nd</sup> Step: Analysis

---

- Find the classes:
  - What are the objects in the application
    - `Doctor`
    - `Patient`
    - `DailySchedule`
    - `Appointment`
    - `Scheduler`
- What would be a typical scenario for the appointment scheduling process:
  - A `Patient` enters her/his name into the `Scheduler`
  - `Patient` chooses a `Doctor`
  - `Scheduler` adds the appointment to the doctor's schedule and to the patients records
  - `Scheduler` confirms the appointment

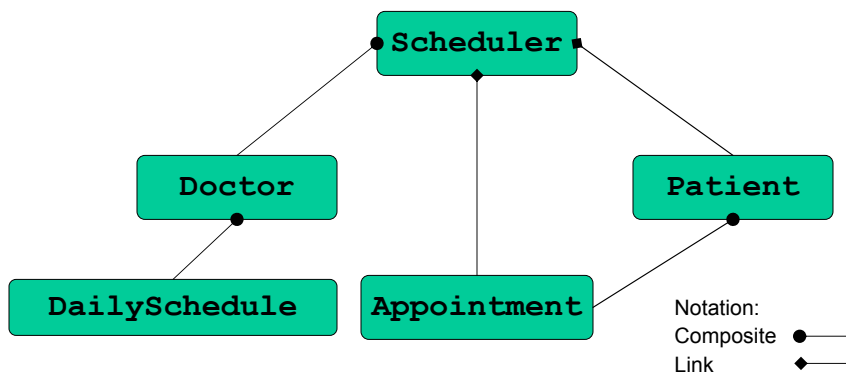
---

D. Mirkovic, C++ Programming, Spring 2005

## Class Dependencies

---

- Interactions between objects



---

D. Mirkovic, C++ Programming, Spring 2005

# Operations

---

- **Doctor** should support the following operations:
  - **AddToSchedule**
  - **ShowAppointments**
- **Patient** should support the following operations:
  - **InputName**
  - **ChooseDoctor**
  - **ChooseTimeSlot**
  - **SetAppointment**
- **DailySchedule** should support the following operations:
  - **SetAppointment**
  - **IsTimeSlotFree**
  - **ShowAppointments**

---

D. Mirkovic, C++ Programming, Spring 2005

# Operations (cont.)

---

- **Appointment** should support the following operations:
  - **Constructor**: Create an appointment object from a time slot, doctor number and patient name
  - **IsScheduled**
- **Scheduler** should support the following operations:
  - **PrintAllAppointments**: Produce a formatted display of all appointments currently in the schedule.
  - **ScheduleOneAppointment** Add one appointment to the schedule. Interactive input from the user
  - **ScheduleAllAppointments** Repeatedly prompt the user for appointments to be added to the schedule.

---

D. Mirkovic, C++ Programming, Spring 2005

# Additional Classes

---

- We will need a time slot class
  - Translation and formatting of appointment times (enter/display time in hh:mm format, store as integer)

```
class TimeSlot {
public:
    TimeSlot( const unsigned n = 0 );
    unsigned AsInteger() const; // Returns the integer value of the time.
    friend istream & operator >>(istream & inp, TimeSlot & T);
    friend ostream & operator <<(ostream & os, const TimeSlot & T);
    // ----- Static member functions -----
    static unsigned GetStartHour();
    static unsigned GetApptLen();
    static void SetStartHour( unsigned n );
    static void SetApptLen( unsigned n );
private:
    static unsigned StartHour;
    static unsigned ApptLen;
    unsigned intValue;
    void SetIntValue( unsigned n );
};
```

---

D. Mirkovic, C++ Programming, Spring 2005

# 3<sup>rd</sup> Step: Design

---

- Once all classes have been identified we can specify their interfaces
- Result of the design phase is the header file with all classes and their interfaces defined
- Classes:
  - TimeSlot
  - Doctor
  - Patient
  - DailySchedule
  - Appointment
  - Scheduler

---

D. Mirkovic, C++ Programming, Spring 2005

## 4<sup>th</sup> Step: Main program

---

- The main program creates an instance of the Scheduler

```
#include "doctors.h"
#include "schedlr.h"

static Doctor doctorArray[NumDoctors];

int main()
{
    cout << "Doctors Office Scheduling Program\n\n";
    Scheduler officeSchedule( doctorArray );
    officeSchedule.ScheduleAllAppointments();
    officeSchedule.PrintAllAppointments("appts.txt");
    return 0;
}
```

---

D. Mirkovic, C++ Programming, Spring 2005

## 5<sup>th</sup> Step: Class Implementations

---

- See the code in the text.

---

D. Mirkovic, C++ Programming, Spring 2005

## Summary

---

- Object oriented analysis, design and programming
- Object model
- Abstraction
- Encapsulation
- Inheritance
- Typing and type checking
- Design principles
- Example

---

D. Mirkovic, C++ Programming, Spring 2005

## Homework

---

- Exercises:
  - 5.7.2 pp. 174
  - 5.7.3 pp. 174. Project #2.

---

D. Mirkovic, C++ Programming, Spring 2005