

CS 617 Object Oriented Systems

Lecture 1

Jan 1, 2008

3:30-5:00 pm

Rushikesh K Joshi

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

Outline

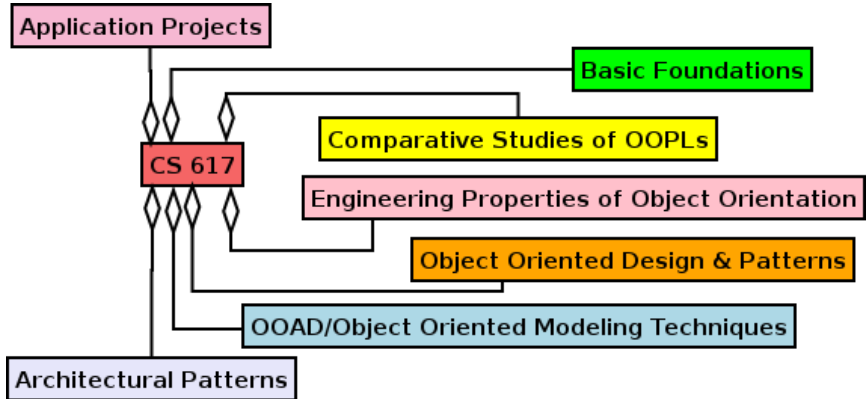
- 1 About the Course
- 2 The Origins
- 3 Some Insights into the Non-OO Imperative World

Outline

- 1 About the Course
- 2 The Origins
- 3 Some Insights into the Non-OO Imperative World

The Content of Course

- A PG level Advanced Course on Object Oriented Systems
- Not a First Course



The TAs

- Akankshi Ratadiya
- Namrata Jain

Course Planning I

- 1 Course Overview, Introduction (1.5)
- 2 Abstractions, Encapsulation, Interfaces and Implementations (1.5)
- 3 Abstract Data Types (1.5)
- 4 Design by Contract (1.5)
- 5 Classes, Implementation sharing, 'this' (1.5)
- 6 Single inheritance, Dynamic Binding, Polymorphism (1.5)
- 7 Subtyping in Inheritance (1.5)
- 8 'self or this' again, and 'super'; Multiple Inheritance (1.5)
- 9 Dynamic Dispatch Implementations (1.5)
- 10 Inheritance Variations and Comparative Studies (1.5)
- 11 Metaclasses, Comparative Studies (1.5)

Course Planning II

- 12 Engineering Properties: building abstractions and hierarchies, Metapatterns, Making frameworks (1.5)
- 13 Object Oriented Metrics and Design Properties (1.5)
- 14 Design Reuse, Creational Patterns with Example Applications (3)
- 15 Structural Patterns with Example Applications (4.5)
- 16 Behavioral Patterns with Example Applications (3)
- 17 Lifecycle Models and Impact of Object Orientation (1.5)
- 18 Architectural Patterns (3)
- 19 Static Modeling with Object Orientation (3)
- 20 Dynamic Modeling with Object Orientation (3)
- 21 UML, Tools (1.5)
- 22 Discussions on Projects (from time to time)

Evaluation Pattern

- Quiz I: 10
- Midsem: 20
- Class Participation, Interactions, Impact, Attendance: 5
- Endsem: 50
- Term Project (Implementation Projects, can be done in groups of 2): 15

Course Website

<http://www.cse.iitb.ac.in/rkj/cs617-08>

past offerings of cs 686 are also available through my home page.

see esp: <http://www.cse.iitb.ac.in/rkj/cs686-2006.html>

References

- Selected Papers from Journals, Conferences
- Technical Reports
- Books on Design and Modeling
- Reference Books for Languages
- Class slides!

The reading details will be announced from time to time.

Outline

- 1 About the Course
- 2 The Origins
- 3 Some Insights into the Non-OO Imperative World

Why was Object Orientation Proposed?

- For Managing the complexity in program organization
- It's about how you structure your programs
- Abstractions have impact on Programming Language Design
- Programming Paradigm has impact on Software Development Methodology
- Object orientation has applications beyond programming and software development
- So what were the techniques for program structuring prior to Object Orientation?

Some Program Structuring Methods

- Structures
- Procedures & Functions
- Files
- Structured Programming

Some OOPLs

- Simula I, Simula 67 (1962-1967)
- Lisp based languages, Smalltalk (1970s)
- C++ (starting early 80s)
- Eiffel (1980s)
- Java (mid 90s)
- C sharp (around 2000)
- Scripting Languages: javascript, python, ruby

Impact of OOPs on Software Engineering

- Coad and Yourdon: OOA
- Beck, Cunningham: CRC
- Booch: OOD
- Rumbaugh et al.: OMT
- Jacobson: Use Case Driven OOSE
- Unified Modeling Language, later into OMG

Outline

- 1 About the Course
- 2 The Origins
- 3 Some Insights into the Non-OO Imperative World

A program with C Globals I

```
#include <stdio.h>
// another instance cannot be handled by these func
// they use globals

int accountNumber;
int balance;

void initialize() {balance=0;} // inlined member f
void deposit (int amount) { balance+=amount; }
void withdraw(int amount) {balance-=amount;}
void getbalance() {printf("%d\n",balance); }

int main () {
```

A program with C Globals II

```
initialize();  
initialize();  
deposit(500);  
deposit(1500);  
withdraw(100);  
withdraw(100);  
getbalance();  
getbalance();  
  
}
```

A Program with static state per function I

```
#include <stdio.h>
// static members keep local state
// but functions cannot share state

void deposit (int amount) { static balance=0;
    balance+=amount;
    printf("%d\n",balance); }
void withdraw(int amount) {static balance=0;
    balance-=amount;
    printf("%d\n",balance); }

int main () {
```

A Program with static state per function II

```
deposit(500);  
deposit(1500);  
withdraw(100);  
withdraw(100);  
}
```

A Program with static state shared, just one function I

```
#include <stdio.h>
// static members keep local state
// but functions cannot share state

void account (int func_no, int amount) {

    static balance=0;

    switch (func_no) {
    case 0: balance+=amount; break;
    case 1: balance-=amount; break;
    }

    printf("%d\n",balance);
```

A Program with static state shared, just one function II

```
}
```

```
int main () {
```

```
    account (0, 500);
```

```
    account (0, 1500);
```

```
    account (1, 100);
```

```
    account (1, 100);
```

```
}
```

A program with C structures-functions shared I

```
#include <stdio.h>
```

```
struct Account {
```

```
int accountNumber;
```

```
int balance;
```

```
};
```

```
void initialize(struct Account *acc) {acc->balance=
```

```
void deposit (struct Account *acc, int amount) { ac
```

```
void withdraw(struct Account *acc, int amount) {acc
```

```
void getbalance(struct Account *acc) {printf("%d\n"
```

A program with C structures-functions shared II

```
int main () {  
  
    struct Account acc1, acc2;  
    initialize(&acc1);  
    initialize(&acc2);  
    deposit(&acc1, 500);  
    deposit(&acc2, 1500);  
    withdraw(&acc1, 100);  
    withdraw(&acc2, 100);  
    getbalance(&acc1);  
    getbalance(&acc2);  
  
}
```

A program in C - objects introduced I

```
#include <stdio.h>
// compiles in g++
// structures hold functions and behave as object d
// structures are classes except that default visib
// in classes, default visibility is private

struct Account {

void deposit(int amount);
void withdraw(int amount) ;
void getbalance();
void initialize() {balance=0;} // inlined member f
```

A program in C - objects introduced II

```
private:
```

```
int accountNumber;
```

```
int balance;
```

```
};
```

```
void Account::deposit (int amount) { balance+=amount;
```

```
void Account::withdraw(int amount) {balance-=amount;
```

```
void Account::getbalance() {printf("%d\n",balance);
```

```
int main () {
```

```
Account acc1, acc2;
```

```
acc1.initialize();
```

A program in C - objects introduced III

```
acc2.initialize();  
acc1.deposit(500);  
acc2.deposit(1500);  
acc1.withdraw(100);  
acc2.withdraw(100);  
acc1.getbalance();  
acc2.getbalance();  
  
}
```