Lecture 23

Introduction to Reuse through Inheritance

(you may encounter this concept during your project implementations this week)

April, Fri April 8th, 2011

Prof. R K Joshi
Computer Science and Engineering
IIT Bombay
Email: rkj@cse.iitb.ac.in
Revision

- function overloading
  - multiple definitions of a function
- operators are also functions
  - operators: functions which are named in terms of symbols
- operator overloading:
  - making an operator work on a different combination of types
- When there are may implementation, which one is chosen?
  - the one that exactly matches the input types
  - e.g.
    - cout << x;
    - array << x;
- cascading: need to return itself
- note the use of reference types in some cases: when a change happens to the object inside a member function/operator and the change needs to be visible in the caller outside.
Reusing an old class

We can extend an earlier class to add new behaviour

- **Student** extended as TA
  - TA has all the properties of Student
  - TA has some additional properties

- Simple vending machine extended to Smart Vending Machine, with an addition of overflow and underflow alarms
  - flash a blinking light when the coin-box overflows, or when the machine runs out of stocks
Example 1

- class Student { ... }
- class TA : public Student { .... }

- In class TA, we can redefine old member functions to give specialized richer meanings
  - Student::responsibility()
  - TA::responsibility()

- over to demo
Example 2

- class Student { … }
- class TA : public Student { ….}

  - in class TA, we can also add new member functions extending the old class
    - TA::monetaryReturns()
      - not present in Student

  - over to demo
Example 3

- class Person { … }
- class Student : public Person { … }
- class TA : public Student { …. }
- class Businessman : public Person { …. }
  - All are persons
  - They can all benefit from what's there in person
  - They add their own specialized member functions

- over to demo