Encapsulation and Inheritance

Amer Diwan

Overview

• First of two papers to look at potential problems with inheritance
• Encapsulation and data abstraction
  – Hide internal representation
  – Provides a strict external interface
• Does inheritance break encapsulation?
  – Can I modify the “private” parts of a class without affecting its clients?
Clients of classes

- Two kinds
  - code that interacts with instances of the class
  - classes that inherit from the class
- Should not compromise encapsulation for both kinds of clients
  - Most people focus only on the first kind

Issues: Instance Variable Access

Option 1: Child accesses all

- Example:
  ```java
class T { int i; int j; }
class S : T { m() { i = i + j; } }
```
- Advantages:
- Disadvantages:
Issues: Instance Variable Access
Option 2: Accessed through methods

• Example:
  
  ```cpp
  class T {
    private: int i; int j;
    protected: int get_i(); int get_j(); }
  ```

  • Advantages:
  • Disadvantages:

Inheritance for subtypes

• Advantages: Simple (single mechanism), simpler implementation
• Disadvantages: exposes inheritance structure
• Example language: C++, M3, Java
Problem: inheritance for subtypes

- class T { void f() {...} }
  class S: T { void g() {...} }
  void aProc(T *t) {...}
aProc(new S)
- class T { void f() {...} }
  class S { void f() {...}; void g() {...} }
  void aProc(T *t) {...}
aProc(new S)

Inheritance for reuse

- Advantages: maximal reuse, finer granularity
- Disadvantages: 2 mechanisms so more complex
- Example language:
Multiple inheritance

• Conflicts
  – Two methods have the same name and signature
• How to deal with conflicts?
  – The mechanism used may expose the “hidden” inheritance structure

Conflict resolution-1

• Disallow unless from same base class
Conflict resolution-2

Redefine locally

Further resolution techniques

- Select “first” definition
- Linearize class hierarchy
- Convert to tree
  - When combined with “redefine locally” alleviates situation when an operation is performed multiple times
Discussion

• Are the points in the paper significant?
• What can we do about them?
• Is multiple inheritance of classes a good idea?

Next lecture

• Type inference
• Reading: Mitchell book sections