Programming Paradigms

Data Abstraction and Object-Orientation (Part 2)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Summer 2020

Overview

- Encapsulation and Information Hiding
- Inheritance ←
- Initialization and Finalization
- Dynamic Method Binding
- Mix-in and Multiple Inheritance

Inheritance

- **Code reuse** **by defining a new abstraction as** **extension or refinement of an existing abstraction**
- **Subclass** **inherits members of** **superclass**
  - Can add members
  - Can modify members

Subclasses vs. Subtypes

Are subclasses a subtype of the superclass?

- In principle, no
  - Subclassing is about reusing code inside a class
  - Subtyping enables code reuse in clients of a class
    - Client written for supertype works with any subtype
- In practice, most PLs merge both concepts

Liskov's Substitutability Principle

- **Each subtype should behave like the supertype when being used through the supertype**
- **Let `B` be a subtype of `A`**
  - Any object of type `A` may be replaced by an object of type `B`
  - Clients programming against `A` will also work with objects of type `B`

"A behavioral notion of subtyping" by B. Liskov and J. Wing, ACM T Progr Lang Sys, 1994

Demo

Liskov.java

Modifying Inherited Members

- **Can a subclass modify inherited members?**
- **Answer depends on the PL**
  - Java: Any method can be overridden
  - C++: Only methods declared as `virtual` by the base class can be overridden

Demo

Virtual.cpp

Modifying Inherited Members (2)

- Can a subclass hide inherited members?
  - Again, answer depends on the PL
- Java and C#: Subclass can neither increase nor decrease the visibility of members
- Eiffel: Subclass can both restrict and increase visibility

Modifying Inherited Members (3)

- **Public/protected/private inheritance in C++**
  - Makes all inherited members at most public/protected/private
  - E.g., all members (incl. public members) that are privately inherited are private in the subclass
  - Private inheritance does not imply a subtype relationship

Demo

Inheritance.cpp

Modifying Inherited Members (4)

- **More C++ rules**
  - Subclass can decrease visibility of superclass members, but never increase it
  - Subclass can hide superclass methods by deleting them

Alternatives to Inheritance

- **Inheritance: Is-a relation**
- **Instead, sometimes a Has-a relation is sufficient for code reuse**
  - Field with class to reuse
  - Forward calls to object stored in this field
  - E.g., reuse class `List` in class `Registrations`
    - Could inherit from `List` (store all registrations)
    - Instead: Field of type `List` in `Registrations`

Quiz: Inheritance

Where is the compilation error (and why)?

```
1   class A {
2       protected:
3       int f = 23;
4       void foo() {}
5
6       public:
7       void bar() {}
8   };
9   class B : protected A {
10      public:
11      void baz() {
12          this->foo();
13      }
14  };
15  int main() {
16      B b;
17      b.bar();
18  }
```

Please vote via Ilias.

Quiz: Inheritance

Where is the compilation error (and why)?

Error: `bar` is not visible

- B inherits A as `protected` class, hence, all members are at most `protected`
- Clients cannot call `protected` methods

```
class A {
    protected:
    int f = 23;
    void foo() {}

    public:
    void bar() {}
};
class B : protected A {
    public:
    void baz() {
        this->foo();
    }
};
int main() {
    B b;
    b.bar();
}
```

Please vote via Ilias.