CPSC 427a: Object-Oriented Programming

Michael J. Fischer

Lecture 13
October 18, 2011

Privacy Revisited (again)

Problem Set 2 Code Review

Privacy Revisited (again)

Visibility rules

Every class member has one of four **privacy attributes**: *public*, *protected*, *private*, or *hidden*.

These attributes determine the locations from which a class member can be seen.

- *public* members can be seen from any location.
- *protected* members can be seen from inside the class or its children.
- *private* members can only be seen from inside the class.
- *hidden* members cannot be seen at all.

Explicit privacy attributes

The privacy attributes for declared class members are given explicitly by the privacy keywords `public`, `protected`, and `private`.

There is no way to explicitly declare a hidden member.

Example:

```
class A {
private:   int x;
protected: int y;
public:    int z;
};
```

Implicit privacy attributes

Inherited class members are assigned implicit privacy attributes based on their attributes in the parent class and by the kind of derivation, whether `public`, `protected`, or `private`.

1. If the member is *public* in the parent class, then its attribute in the child class is given by the kind of derivation.
2. If the member is *protected* in the parent class, then its attribute in the child class is *protected* for public and protected derivation, and *private* for private derivation.
3. If the member is *private* or *hidden* in the parent class, then it is *hidden* in the child class.

Implicit privacy chart

Below is a revision of the chart presented in lecture 10.

Attribute in base class	Kind of Derivation: public	Kind of Derivation: protected	Kind of Derivation: private
public	public	protected	private
protected	protected	protected	private
private	hidden	hidden	hidden
hidden	hidden	hidden	hidden

Attribute in derived class.

Summary

1. All members of the base class are inherited by the derived class and appear in every instantiation of that class.
2. All inherited members receive implicitly defined privacy attributes.
3. Visibility of all data members is determined solely by their privacy attributes.
4. Public and protected base class variables are always visible within a derived class.
5. Private and hidden base class variables are never visible in the derived class.
6. The kind of derivation never affects the visibility of inherited members in the derived class; only their implicit attributes.

Problem Set 2 Code Review

A retrospective look at PS2

Class dependency structure.

- `class RandBit` is a biased random bit generator. It depends on nothing else and can be tested alone.
- `class Coin` is a coin flipper with dependency on the previous coin flip. It uses a `RandBit` object as its source of randomness to tell it whether to turn over the previous coin or leave it alone.
  It should not depend on any other class.
- `class Experiment` runs an experiment on a coin, where an experiment consists of some predetermined number of runs. It clearly needs a `Coin` but shouldn't know about the details of how the coin is built.
- `class Simulator` is in charge of controlling the experiments.

Question: Where should the parameter-processing go?

Testing

There should be unit tests for each of the classes. The code contains an incomplete set of unit tests.

(code demo)