ICS211 Section 3, Fall Semester 2013

Class invariants

In computer programming, specifically object-oriented programming, a **class invariant** is an invariant used to constrain objects of a class. Methods of the class should preserve the invariant. The class invariant constrains the state stored in the object.

Class invariants are established during construction and constantly maintained between calls to public methods. Temporary breaking of class invariance between private method calls is possible, although not encouraged.

An object invariant, or representation invariant, is a computer programming construct consisting of a set of invariant properties that remain uncompromised regardless of the state of the object. This ensures that the object will always meet predefined conditions, and that methods may, therefore, always reference the object without the risk of making inaccurate presumptions. Defining class invariants can help programmers and testers to catch more bugs during software testing.

Class invariants and inheritance

The useful effect of class invariants in object-oriented software is enhanced in the presence of inheritance. Class invariants are inherited, that is, "the invariants of all the parents of a class apply to the class itself."[1]

Inheritance can allow descendant classes to alter implementation data of parent classes, so it would be possible for a descendant class to change the state of instances in a way that made them invalid from the viewpoint of the parent class. The concern for this type of misbehaving descendant is one reason object-oriented software designers give for favoring composition over inheritance (i.e., inheritance breaks encapsulation).[2]

However, because class invariants are inherited, the class invariant for any particular class consists of any invariant assertions coded immediately on that class, logically "and-ed" with all the invariant clauses inherited from the class's parents. This means that even though descendant classes may have access to the implementation data of the their parents, the class invariant can prevent them from manipulating those data in any way that produces an invalid instance at runtime.

Assertions

Common programming languages like C++ and Java support assertions by default, which can be used to define class invariants. A common pattern to implement invariants in classes is for the constructor of the class to throw an exception, if the invariant is not satisfied. Since methods preserve the invariants, they can assume the validity of the invariant and need not explicitly check for it.

Java

This is an example of a class invariant in the Java programming language with Java Modeling Language. The invariant must hold to be true after the constructor is finished and at the entry and exit of all public member functions. Public member functions should define precondition and postcondition to help ensure the class invariant.

```
public class Date {
    int /*@spec_public@*/ day;
    int /*@spec_public@*/ hour;

    /*@invariant 1 <= day && day <= 31; @*/ //class invariant
    /*@invariant 0 <= hour && hour < 24; @*/ //class invariant

    /*@
    @requires 1 <= d && d <= 31;
    @requires 0 <= h && h < 24;
    @*/
    public Date(int d, int h) { // constructor
        day = d;
        hour = h;
    }

    /*@
    @requires 1 <= d && d <= 31;
    @ensures day == d;
    @*/
    public void setDay(int d) {
        day = d;
    }

    /*@
    @requires 0 <= h && h < 24;
    @ensures hour == h;
    @*/
    public void setHour(int h) {
        hour = h;
    }
}
```

References

1. ^ Meyer, Bertrand. *Object-Oriented Software Construction, second edition,* Prentice Hall, 1997, p. 570.
2. ^ E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, Massachusetts, 1995., p. 20.