CSPP 511-01:
Introduction to Object-Oriented Programming

Harri Hakula

Ryerson 256, tel. 773-702-8584
hhakula@cs.uchicago.edu

August 2, 2000
Outline

- More on Classes
- Inheritance vs Interface
- Project: Display
More on Classes

Every class designer wants to address the following questions:

- what can and cannot be extended,
- what part of the implementation is necessary and sufficient.

There are three more keywords each of which address the issues raised above.

- abstract
- final
- protected
Keyword: abstract

A class which is declared to be abstract cannot be instanciated. Consider the following classic example:

```java
abstract class Benchmark {
    abstract void benchmark();
    public final long repeat(int count) {
        long start = System.currentTimeMillis();
        for(int i = 0; i < count; ++i)
            benchmark();
        return (System.currentTimeMillis() - start);
    }
}
```
Keyword: abstract Cntd.

Benchmark cannot be instanciated, since there is no implementation of the method `void benchmark()`. This is a class that is designed to be a base class. For instance:

```java
public class MethodBenchmark extends Benchmark {
    void benchmark() { }
    public static void main(String[] args) {
        int count = 10000;
        long time =
            new MethodBenchmark().repeat(count);
        System.out.println(count + " : " +
            time + "ms.");
    }
}
```
Keyword: final

Benchmark has a method:

```java
public final long repeat(int count)
```

The attribute `final` means that the derived classes cannot overload this method. This implementation is final.
Keyword: protected

Private and public are easy to understand. However, there is a third class of visibility: protected.

When something in a class is declared protected, it is visible to every class which is extending the class within which it was originally defined.

Whether something should be protected or not is a subject of heated debate.
Inheritance vs Interface

What is the difference between inheritance and interfaces?

Inheritance (and polymorphism) fixes the type, whereas interface fixes the *functionality*.

Macintoshes and empires are apples, jaffas are oranges. If one is counting apples, one should not allow jaffas. If one wants to print the labels of fruits, both jaffas and empires should be allowed.
Project: Display