パッケージ，アクセス修飾子

2008年4月27日

海谷 治彦

パッケージ

- クラス，インタフェースをグルーピングするためのからくり.
- 加えて，特定のクラス(インタフェース)を同じパッケージに属するクラスからのみ利用できるように情報隠蔽できる.
  - ま，これはないとあんまりパッケージの意味がない.
- 異なるパッケージに属する同じ名前のクラスを作る(使う)ことができる. (いわゆる名前空間)

アクセス制御のルール

	public	protected	無し	private
同一パッケージ	○	○	○	×
サブクラス	○	○	×	×
無関係なクラス	○	×	×	×

個々のメソッドだけでなく,
クラス全体，およびコンストラクタにも指定が必要.

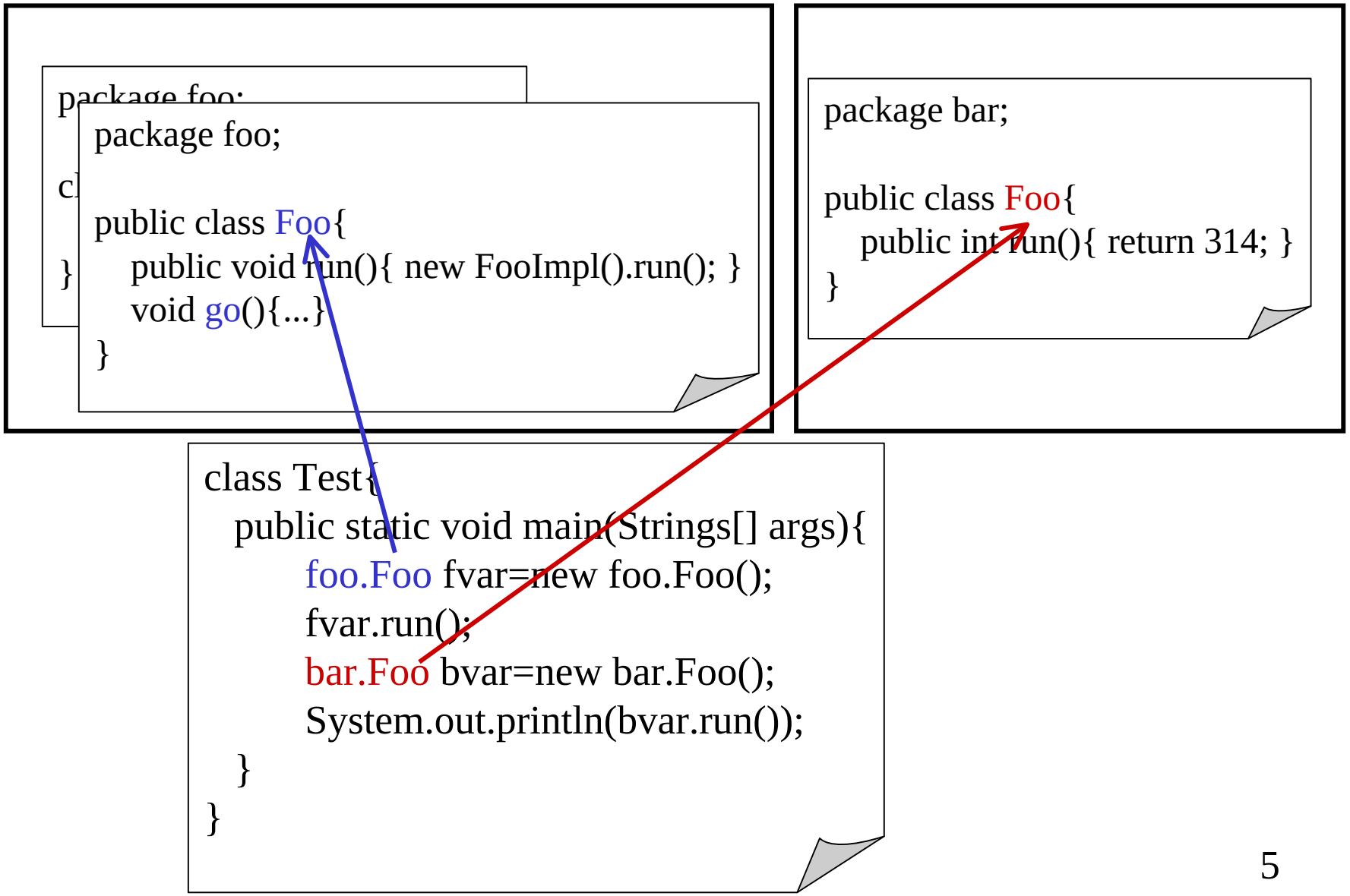
例

```
package foo;

class FooImpl{
    void run(){ print(“Hello”); }
}
```

```
package foo;

public class Foo{
    public void run(){ new FooImpl().run(); }
    void go(){...}
}
```

```
class Test{
   public static void main(Strings[] args){
         foo.Foo fvar=new foo.Foo();
         fvar.run();
         fvar.go(); // × Foo は public だが，go()はちがう
         foo.FooImpl fi=new FooImpl(); // ×そもそも FooImplが非public
   }
}
```

例


package foo;
public class Foo{
public void run(){ new FooImpl().run(); }
void go(){...}
}
package bar;
public class Foo{
public int run(){ return 314; }
}
class Test{
public static void main(Strings[] args){
foo.Foo fvar=new foo.Foo();
fvar.run();
bar.Foo bvar=new bar.Foo();
System.out.println(bvar.run());
}
}

実際的なパッケージ

- これまたJavaの標準APIは，それぞれ機能・特性別にパッケージ分けされている.
- どんなクラスなどう隠蔽されているかは定かでない. (ソースを見ればわかるが・・・)

import

- 他のパッケージ内の公開クラスを，パッケージ名無しで利用するための仕組み.
- 利用する場合，名前が短くなるので便利.
- しかし，名前の衝突(同じ名前のクラスを別途利用してしまった)の危険度は増す.

例

```
class Test{
  void run(){
    java.util.Vector v=new java.util.Vector();
  }
}
```

```
import java.util.Vector;

class Vector{
  void run(){
    Vector v=new Vector();
  }
}
```

```
import java.util.Vector;

class Test{
  void run(){
    Vector v=new Vector();
  }
}
```

```
import java.util.*;

class Test{
  void run(){
    Vector v=new Vector();
  }
}
```