1B Lecture #13: Packages, Access

Package Mechanics

espond to things being modeled (represented) in one's

collections of "related" classes and other packages.

ndard libraries and packages in package `java` and `javax`.

class resides in the *anonymous package*.

ewhere, use a `package` declaration at start of file, as in

```
atabase;     or     package ucb.util;
```

ac uses convention that class `C` in package `P1.P2` goes in
`P1/P2` of any other directory in the *class path*.

e:

```
LASSPATH=.:$HOME/java-utils:$MASTERDIR/lib/classes/junit.jar
it.textui.TestRunner MyTests
```

TestRunner.class in ./junit/textui, ~/java-utils/junit/textui
ooks for junit/textui/TestRunner.class in the junit.jar
a single file that is a special compressed archive of an
ory of files).

Access Modifiers

fiers (**private**, **public**, **protected**) do not add anything
of Java.

w a programmer to declare which classes are supposed
ccess ("know about") what declarations.

also part of security—prevent programmers from accessing
would "break" the runtime system.

always determined by static types.

ine correctness of writing `x.f()`, look at the definition
static type of `x`.

static type? Because the rules are supposed to be
by the compiler, which only knows static types of things
es don't depend on what happens at execution time).

The Access Rules: Public

of a member depends on (1) how the member's declaration
nd (2) where it is being accessed.

d C4 are distinct classes.

either class C2 itself or a subtype of C2.

```
C1 ... {
ethod, field,...
M ...
)
 ... } // OK.
```

```
package P2;
class C2 extends C3 {
  void f(P1.C1 x) {... x.M ...} // OK
  void g(C2a y) {... y.M ...  } // OK
}
```

```
C4 ... {
)
 ... } // OK.
```

Public members are available everywhere.

The Access Rules: Private

4 are distinct classes.

either class C2 itself or a subtype of C2.

```
C1 ... {
ethod, field,...
 M ...
)
 ... } // OK.
```

```
package P2;
class C2 extends C1 {
  void f(P1.C1 x) {... x.M ...} // ERROR
  void g(C2a y) {... y.M ...  } // ERROR
}
```

```
C4 ... {
)
 ... } // ERROR.
```

Private members are available only within the text of the same class, even for subtypes.

rivate to a Class, Not an Object

mple list implementation using linking:

```
ss LinkedList {
te Node _head;
te static class Node {
bject _head;
ode _tail;
```

```
wap the contents of this LinkedList and OTHER. */
t void swap(LinkedList other) {
inkedList tmp = other._head; other._head = _head; _head = tmp;
```

vious and correct implementation of `swap`, *even though* it
ivate member of a `LinkedList` object *other than* `this`.

ly OK to do so by the rules, because `swap` is in the text
st and therefore has access to all private fields.

acy" does *not* apply to individual objects, only to classes.

he Access Rules: Package Private

4 are distinct classes.

either class C2 itself or a subtype of C2.

```
C1 ... {
ethod, field,...
nt M ...
)
 ... } // OK.
```

```
C4 ... {
)
 ... } // OK.
```

```
package P2;
class C2 extends C1 {
  void f(P1.C1 x) {... x.M ...} // ERROR
  void g(C2a y) {... y.M ...  } // ERROR
}
```

Package Private members are available only within the same package (even for subtypes).

What May be Controlled

nterfaces that are not nested may be public or package

ields, methods, constructors, and (later) nested types—may he four access levels.

e a method only with one that has *at least* as permissive el. Reason: avoid inconsistency:

```
;
ss C1 {
nt f() { ... }


ss C2 extends C1 {
lly a compiler error; pretend
ot and see what happens
{ ... }
```

```
package P2;
class C3 {
  void g(C2 y2) {
    C1 y1 = y2
    y2.f(); // Bad???
    y1.f(); // OK??!!?
  }
}
```

e's no point in restricting C2.f, because access control tatic types, and C1.f is public.

The Access Rules: Protected

4 are distinct classes.

either class C2 itself or a subtype of C2.

```
C1 ... {
ethod, field,...
nt M ...
)
 ... } // OK.
```

```
C4 ... {
)
 ... } // OK.
```

```
package P2;
class C2 extends C1 {
  void f(P1.C1 x) {... x.M ...} // ERROR
       // (x's type is not subtype of C2.)
  void g(C2a y) {... y.M ...  } // OK
  void g2()     {... M ... } // OK (this.M)
}
```

Protected members of C1 are available within P1, as for package private. Outside P1, they are available within subtypes of C1 such as C2, but only if accessed from expressions whose static types are subtypes of C2.

Intentions of this Design

rations represent *specifications*—what clients of a package l to rely on.

vate declarations are part of the *implementation* of a ust be known to other classes that assist in the implementation.

eclarations are part of the implementation that subtypes t that clients of the subtypes generally won't.

larations are part of the implementation of a class that ss needs.

Quick Quiz

```
;
{

 OK

y1;
;
```

```
// Anonymous package

class A2 {
  void g(SomePack.A1 x) {
    x.f1();  // OK?
    x.y1 = 3; // OK?
  }
}

class B2 extends SomePack.A1 {
  void h(SomePack.A1 x) {
    x.f1();  // OK?
    x.y1 = 3; // OK?
    f1();     // OK?
    y1 = 3;   // OK?
    x1 = 3;   // OK?
  }
}
```

hree lines of `h` have implicit **this.**'s in front. Static type

Quick Quiz

```
;
{

 OK?

y1;
;
```

```
// Anonymous package

class A2 {
  void g(SomePack.A1 x) {
    x.f1();  // OK?
    x.y1 = 3; // OK?
  }
}

class B2 extends SomePack.A1 {
  void h(SomePack.A1 x) {
    x.f1();  // OK?
    x.y1 = 3; // OK?
    f1();     // OK?
    y1 = 3;   // OK?
    x1 = 3;   // OK?
  }
}
```

hree lines of `h` have implicit **this.**'s in front. Static type

Quick Quiz

```
;                   // Anonymous package
{
                    class A2 {
                      void g(SomePack.A1 x) {
 OK                     x.f1();  // ERROR
                        x.y1 = 3; // ERROR
y1;                   }
;                   }

                    class B2 extends SomePack.A1 {
                      void h(SomePack.A1 x) {
                        x.f1();  // OK?
                        x.y1 = 3; // OK?
                        f1();     // OK?
                        y1 = 3;   // OK?
                        x1 = 3;   // OK?
                      }
                    }
```

hree lines of h have implicit **this**.'s in front. Static type

Quick Quiz

```
;                   // Anonymous package
{
                    class A2 {
                      void g(SomePack.A1 x) {
 OK                     x.f1();  // ERROR
                        x.y1 = 3; // ERROR
y1;                   }
;                   }

                    class B2 extends SomePack.A1 {
                      void h(SomePack.A1 x) {
                        x.f1();  // ERROR
                        x.y1 = 3; // OK?
                        f1();     // ERROR
                        y1 = 3;   // OK?
                        x1 = 3;   // OK?
                      }
                    }
```

hree lines of h have implicit **this**.'s in front. Static type

Quick Quiz

```
;                   // Anonymous package
{
                    class A2 {
                      void g(SomePack.A1 x) {
 OK                     x.f1();  // ERROR
                        x.y1 = 3; // ERROR
y1;                   }
;                   }

                    class B2 extends SomePack.A1 {
                      void h(SomePack.A1 x) {
                        x.f1();  // ERROR
                        x.y1 = 3; // OK?
                        f1();     // ERROR
                        y1 = 3;   // OK
                        x1 = 3;   // ERROR
                      }
                    }
```

hree lines of h have implicit **this**.'s in front. Static type

Quick Quiz

```
;                   // Anonymous package
{
                    class A2 {
                      void g(SomePack.A1 x) {
 OK                     x.f1();  // ERROR
                        x.y1 = 3; // OK?
y1;                   }
;                   }

                    class B2 extends SomePack.A1 {
                      void h(SomePack.A1 x) {
                        x.f1();  // OK?
                        x.y1 = 3; // OK?
                        f1();     // OK?
                        y1 = 3;   // OK?
                        x1 = 3;   // OK?
                      }
                    }
```

hree lines of h have implicit **this**.'s in front. Static type

Quick Quiz

```
;                   // Anonymous package
{
                    class A2 {
                      void g(SomePack.A1 x) {
 OK                     x.f1();  // ERROR
                        x.y1 = 3; // ERROR
y1;                   }
;                   }

                    class B2 extends SomePack.A1 {
                      void h(SomePack.A1 x) {
                        x.f1();  // ERROR
                        x.y1 = 3; // OK?
                        f1();     // OK?
                        y1 = 3;   // OK?
                        x1 = 3;   // OK?
                      }
                    }
```

hree lines of h have implicit **this**.'s in front. Static type

Quick Quiz

```
;                   // Anonymous package
{
                    class A2 {
                      void g(SomePack.A1 x) {
 OK                     x.f1();  // ERROR
                        x.y1 = 3; // ERROR
y1;                   }
;                   }

                    class B2 extends SomePack.A1 {
                      void h(SomePack.A1 x) {
                        x.f1();  // ERROR
                        x.y1 = 3; // OK?
                        f1();     // ERROR
                        y1 = 3;   // OK
                        x1 = 3;   // OK?
                      }
                    }
```

hree lines of h have implicit **this**.'s in front. Static type

Access Control Static Only

vate" don't apply to dynamic types; it is possible to call
cts of types you can't name:

```
                                | package mystuff;
hings. */                       |
ce Collector {                  | class User {
ect x);                         |    utils.Collector c =
                                |      utils.Utils.concat();
---------------                 |
                                |    c.add("foo");  // OK
tils {                          |    ... c.value(); // ERROR
c Collector concat() {          |    ((utils.Concatenator) c).value()
 Concatenator();                |                     // ERROR
                                |
                                ----------------------------------


 class that collects strings. */
ater implements Collector {
 stuff = new StringBuffer();

add(Object x) { stuff.append(x); n += 1; }
t value() { return stuff.toString(); }
```


Quick Quiz

```
;                       // Anonymous package
{
                        class A2 {
                          void g(SomePack.A1 x) {
 OK                         x.f1();  // ERROR
                            x.y1 = 3; // ERROR
y1;                       }
;                       }

                        class B2 extends SomePack.A1 {
                          void h(SomePack.A1 x) {
                            x.f1();  // ERROR
                            x.y1 = 3; // ERROR
                            f1();     // ERROR
                            y1 = 3;   // OK
                            x1 = 3;   // ERROR
                          }
                        }
```

hree lines of `h` have implicit **this.**'s in front. Static type