CS61B Lecture #10

Reminders:

- Extra handouts in 283 Soda (and online).
- Please use `bug-submit` for submitting any programming problems you have with homework and projects.

Readings: Chapters 4 and 5 of the Blue Reader.

Today's Topics:

- Modularization facilities in Java.

Package Mechanics

- Classes correspond to things being modeled (represented) in one's program.
- Packages are collections of "related" classes and other packages.
- Java puts standard libraries and packages in package `java` and `javax`.
- By default, a class resides in the *anonymous package*.
- To put it elsewhere, use a `package` declaration at start of file, as in

  `package database;` *or* `package ucb.util;`
- Sun's `javac` uses convention that class `C` in package `P1.P2` goes in subdirectory `P1/P2` of current directory ...
- ...or of any other directory in the *class path*.

Access Modifiers

- Access modifiers (**private, public, protected**) do not add anything to the power of Java.
- Basically allow a programmer to declare what classes are supposed to need to access ("know about") what declarations.
- In Java, are also part of security—prevent programmers from accessing things that would "break" the runtime system.
- Accessibility always determined by static types.
  - To determine correctness of writing `x.f()`, look at the definition of `f` in the *static type* of `x`.
  - Why? Because the rules are supposed to be enforced by the compiler, which only knows static types of things (static types don't depend on what happens at execution time).

The Access Rules

- Suppose we have two packages (not necessarily distinct) and two distinct classes:

```
package P1;
public class C1 ... {        package P2;
  // A member named M,       class C2 extends C3 {
  𝒜 int M ...                  void f (P1.C1 x) {... x.M ...} // OK?
  void h (C1 x)                // C4 a subtype of C2 (possibly C2 itself)
    { ... x.M ... } // OK.     void g (C4 y) {... y.M ...  } // OK?
}                            }
```

- The access `x.M` is
  - Legal if $\mathcal{A}$ is **public**;
  - Legal if $\mathcal{A}$ is **protected** and $P1$ is $P2$;
  - Legal if $\mathcal{A}$ is *package private* (default—no keyword) and $P1$ is $P2$;
  - Illegal if $\mathcal{A}$ is **private**.

- Furthermore, if $C3$ is $C1$, then `y.M` is also legal under the conditions above, or if $\mathcal{A}$ is **protected** (i.e., even if $P1$ is not the same as $P2$).

What May be Controlled

- Classes and interfaces that are not nested may be public or package private (we haven't talked explicitly about nested types yet).
- Members—fields, methods, constructors, and (later) nested types—may have any of the four access levels.
- May *override* a method only with one that has *at least* as permissive an access level.
  - Reason: avoid inconsistency:

```
package P1;                             | package P2;
public class C1 {                       | class C3 {
  public int f () { ... }               |   void g (C2 y2) {
}                                       |     C1 y1 = y2
                                        |     y2.f (); // Bad???
public class C2 extends C1 {            |     y1.f (); // OK??!!?
  // Actually a compiler error; pretend |   }
  // it's not and see what happens      | }
  int f () { ... }
}
```

  - That is, there's no point in restricting C2.f, because access control depends on static types, and C1.f is public.

Intentions of this Design

- **public** declarations represent *specifications*—what clients of a package are supposed to rely on.
- *package private* declarations are part of the *implementation* of a class that must be known to other classes that assist in the implementation.
- **protected** declarations are part of the implementation that subtypes may need, but that clients of the subtypes generally won't.
- **private** declarations are part of the implementation of a class that only that class needs.

Quick Quiz

```
package SomePack;
public class A1 {
  int f1() {
    A1 a = ...
    a.x1 = 3; // OK?
  }
  protected int y1;
  private int x1;
}
```

```
// Anonymous package

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

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

- **Note:** Last three lines of `h` have implicit **this.**'s in front. Static type of **this** is `B2`.

Quick Quiz

```
package SomePack;
public class A1 {
  int f1() {
    A1 a = ...
    a.x1 = 3; // OK
  }
  protected int y1;
  private int x1;
}
```

```
// Anonymous package

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

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

- **Note:** Last three lines of h have implicit **this.**'s in front. Static type of **this** is B2.

Quick Quiz

```
package SomePack;
public class A1 {
  int f1() {
    A1 a = ...
    a.x1 = 3; // OK
  }
  protected int y1;
  private int x1;
}
```

```
// Anonymous package

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

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

- **Note:** Last three lines of `h` have implicit **this.**'s in front. Static type of **this** is `B2`.

Quick Quiz

```
package SomePack;
public class A1 {
  int f1() {
    A1 a = ...
    a.x1 = 3; // OK
  }
  protected int y1;
  private int x1;
}
```

```
// Anonymous package

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

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

- **Note:** Last three lines of h have implicit **this.**'s in front. Static type of **this** is B2.

Quick Quiz

```
package SomePack;
public class A1 {
  int f1() {
    A1 a = ...
    a.x1 = 3; // OK
  }
  protected int y1;
  private int x1;
}
```

```
// Anonymous package

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

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

- **Note:** Last three lines of h have implicit **this.**'s in front. Static type of **this** is B2.

Quick Quiz

```
package SomePack;
public class A1 {
  int f1() {
    A1 a = ...
    a.x1 = 3; // OK
  }
  protected int y1;
  private int x1;
}
```

```
// Anonymous package

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

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

- **Note:** Last three lines of h have implicit **this.**'s in front. Static type of **this** is B2.

Quick Quiz

```
package SomePack;
public class A1 {
  int f1() {
    A1 a = ...
    a.x1 = 3; // OK
  }
  protected int y1;
  private int x1;
}
```

```
// Anonymous package

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

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

- **Note:** Last three lines of `h` have implicit **this.**'s in front. Static type of **this** is `B2`.

Quick Quiz

```
package SomePack;
public class A1 {
  int f1() {
    A1 a = ...
    a.x1 = 3; // OK
  }
  protected int y1;
  private int x1;
}
```

```
// Anonymous package

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

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

- **Note:** Last three lines of h have implicit **this.**'s in front. Static type of **this** is B2.

Quick Quiz

```
package SomePack;
public class A1 {
  int f1() {
    A1 a = ...
    a.x1 = 3; // OK
  }
  protected int y1;
  private int x1;
}
```

```
// Anonymous package

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

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

- **Note:** Last three lines of h have implicit **this.**'s in front. Static type of **this** is B2.

Access Control Static Only

"Public" and "private" don't apply to dynamic types; it is possible to call methods in objects of types you can't name:

```
package utils;                           | package mystuff;
/** A Set of things. */                  |
public interface Collector {             | class User {
  void add (Object x);                   |    Collector c =
}                                        |      utils.Utils.concat ();
----------------------------             |
package utils;                           |    c.add ("foo");  // OK
public class Utils {                     |    ... c.value (); // ERROR
  public static Collector concat () {    |    ((utils.Collector) c).value ()
    return new Concatenator ();          |                   // ERROR
  }                                      |
}                                        ----------------------------------

/** NON-PUBLIC class that collects strings. */
class Concatenater implements Collector {
  StringBuffer stuff = new StringBuffer ();
  int n = 0;
  public void add (Object x) { stuff.append (x); n += 1; }
  public Object value () { return stuff.toString (); }
}
```