13 Classes: Access Control

1 Access control

Each method, constructor, and variable within a class, whether static or dynamic, may be marked as either `public` or `private`. If neither is stated, then the default is public (but Java requires an explicit writing of `public` in the declaration of main(), and in some other situations which we will mention when we meet them). These keywords do not bring any new power, in the sense that they do not enable us to write programs that we couldn't write without them. Their role is stylistic.

Entities in a class that are not marked private are public by default. Entities marked as private cannot be seen outside the class, and so it is not possible for other classes to refer to them directly. If we need to change a class method, say – perhaps we have discovered a more efficient implementation – the change will likely impact on other modules, which will in turn need to be updated, and so on. This can be very time-consuming, especially the time lost in re-testing everything. However, if the method happened to be private, we can be certain that no other classes are affected.

It is standard programming practice to make all the instance variables in a class private. Methods should be marked private if they are written solely to help us to write another method in the same class – in other words, if they were not part of the formal requirements for the class. For example, recall the first example of Date we wrote earlier; it is reproduced in outline below with its variables and methods marked as private where appropriate:

```
class Date {
   private int day; private int month; private int year;
   void getDate() {
        ...
   }
```

```
    private String monthName() { // not intended to be invoked from outside
        ...
    }

    void putDate() {  // in form 23rd March 2006
        ...
    }
}
```

The instance variables `day`, `month`, and `year` are marked private as is usually the case. In addition, method `monthName` is marked private because it was introduced only during the coding process to help us write `putDate`.

It is not easy for beginners to see the value of `public` and `private`. Indeed, if we replace all occurrences of `private` in a program with `public`, then the program will compute the same result. However, if you earn your living updating other peoples programs, you will pray that they have employed private variables and methods as much as possible. Professional programmers always privatise when possible. Regular changes in large working programs are the norm, and it is easier to change programs where the internal details of classes are kept hidden by making them private.

2 The formal rules

The basic rule is that private entities cannot be accessed *outside the class* in which they are defined. For example:

```
class Pair {
   int x;              // x is public by default
   private int y;      //  y is private

   private void meth1(Pair p) {
      .....
     int tempx = p.x; // okay
     int tempy = p.y; // okay (we're still inside the class where y is defined)
      .....
   }
   .....
```

```
}

class UsePair {
   public static void main(String args[] ) {
      Pair r = new Pair ();
      Pair q = new Pair();
      r.x = 1;          // okay as x public
      r.y = 2;          // ILLEGAL as y private and we're outside the class
      r.meth1(p);       // ILLEGAL as meth1 private and we're outside the class
   }
}
```

Remember that the attributes `public` and `private` only constrain code outside the *class* in which they occur – note *class* here, not object. If a class contains a private variable `x`, say, then an instance method in the class can refer not only to its own `x`, but to the corresponding `x` in every other object of the class. For example, in class `Pair` above

```
int tempy = p.y;
```

is allowed in instance method `meth1` even though it is referring to a private variable `y` in another object – as long as the object is of the same *class* the reference is okay.

The private/public trap

Only variables declared at class level may be qualified as private or public. Local variables (i.e. those declared inside methods) are never so marked:

```
class Duration {

   private int counter = 0; // okay

   public static void main(String[] args) {
         private int length = 0;      // WRONG!
         .....
   }
}
```