

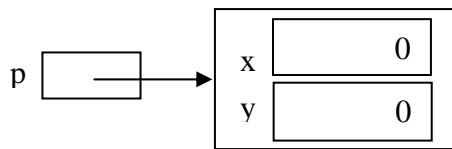
3b

Classes: Constructors

1 No-args constructors

For a class called `MyClass`, say, the phrase `new MyClass()` is an expression which yields a reference to a freshly created object of type `MyClass`. `MyClass()` is an example of a “constructor”. Whenever we introduce a class, the system automatically provides a default constructor for it, as typified by `MyClass()` above.

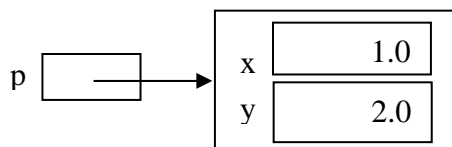
Instance variables are always given initial values by the constructor, according to certain default rules – e.g. integer components of objects are initialised to 0. For example, the effect of executing `Point p = new Point()` can be visualised as:



It is possible to change the default initial values, as follows:

```
class Point {  
    double x = 1.0;  
    double y = 2.0;  
}
```

Now the effect of executing `Point p = new Point()` can be visualised as:



There is another way to provide specific initial values for instance variables: by writing your own constructor. (In contrast, the constructors supplied automatically by the system are said to be

default no-args constructors). The following example illustrates a programmer-supplied constructor:

```
class Point {
    double x, y;

    Point() { // constructor with no parameters
        x = 1.0; y = 2.0;
    }
}
```

The effect of executing `Point p = new Point()` is exactly as in the preceding diagram.

2 Constructors with parameters

We write constructors just as we write methods, but *there is no return type in the header, nor any return statements in the body*. They *must* have the same name as the class. They may have parameters, just like methods, as in:

```
class Point {
    double x, y;

    Point(double x0, double y0) { // constructor with two parameters
        x = x0; y = y0;
    }
}
```

In this case, we must explicitly supply the desired initial values for the instance variables each time we construct the object, as in

```
p = new Point(2.5, 3.14);
```

The effect of this statement is to construct an object of type `Point`, initialise its `x` and `y` instance variables to 2.5 and 3.4, respectively, and assign a reference to the object to `p`. This constructor is said to be an *all-args* constructor. Its use is illustrated in the following trivial program:

```
class Point {
    double x, y;
```

```

    Point(double x0, double y0 { // constructor with two parameters
        x = x0; y = y0;
    }
}

class PointDemo {
    public static void main(String args[] ) {
        Point p = new Point(2.0,4.5);
        double dist = Math.sqrt(p.x*p.x+ p.y*p.y); // distance of p from origin
        System.out.println("The point is " + dist + " from the origin.");
    }
}

```

Remember not to confuse constructors with methods: for constructors *there is no return type in the header, nor any return statements in the body.*

Class definitions can be composed using a combination of specifying default initial values for instance variables, and supplying zero or more constructors with varying numbers of arguments. This is illustrated below:

```

class Point {
    double x = 5.0; // default initial value of x components is 5.0
    double y;

    Point(double y0) {
        y = y0;
    }

    Point(double x0, double y0) {
        x = x0; y = y0;
    }
}

class ConstructorsTest {

    public static void main(String args[] ) {
        Point q, r;
        q = new Point(4.5); // q initialised to (5.0, 4.5)
        r = new Point(3.41, 4.5); // r initialised to (3.41, 4.5)
        .....
    }
}

```

```
}  
}
```

The various constructors in a class definition must differ from one another in the number and types of their parameters. Constructors may invoke methods, whether they are defined in the same class or not. Actually, it is possible for a constructor to call another constructor within its class. The called constructor is invoked as though it were a void method (i.e. a procedure), and the invocation must be the *first* action in the calling constructor. This facility is not much used in small programs.

The default constructor trap

If you define your own constructor(s), then the default no-args constructor is no longer made available. If you really need it you have to write it, as for example in :

```
class Point {  
    double x, y;  
  
    Point(double x0, double y0) { // all-args constructor  
        x = x0; y = y0;  
    }  
  
    Point() { } // reinstates the default no-args constructor  
}
```

We have not so far written any interesting programs using classes. This is because interesting programs that use classes nearly always employ dynamic methods which we haven't yet studied.