Public and Private

```
class Hello {
  int y;
  public int x;
}
```

What will happen now?

```
Hello h;
h.x =3;
h.y = 4;
```


Inheritance

We've had points, but what if we wanted coloured points? Would we have to define a whole new class?

```
class ColouredPoint extends point {
  int colour;
}
```


Relationships

- The class Point is a sub-class of object
- The class ColouredPoint is a sub-class of Point
- The class Point is a superclass of ColouredPoint

Class Members

Now with inheritance things become a little more complicated. A class has the following members (methods)

- Members inherited from its direct superclass
- Members declared in the body of the class

Members of a class that are declared private are not inherited by subclasses of that class. Only members of a class that are declared protected or public are inherited by subclasses declared in a package other than the one in which the class is declared.

More on Inheritance

- Defines a new class with the same methods as the superclass.
- Methods in the superclass can be overridden.
- Private methods not inherited.
- Protected methods inherited but not visible outside the class.
- `super()` and `this()`
- Things can get very messy, *think* before you code.

ColouredPoint

So our coloured point has all the methods that our Point had, infact nothing different happens.

```
class ColouredPoint extends point {
 int colour;
 public void plot() {
   super.plot();
   System.out.print("Colour = ");
   System.out.println(colour);
 }
 public void set_colour(int new_c) {
   colour = new_c;
 }
}
```


Things I'm not goingt to tell you about

- Packages
- Interfaces