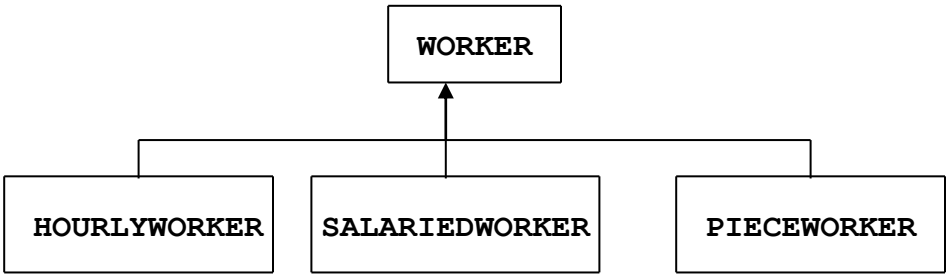
## Unit Nine Lecture Two

### Topic 1: Inheritance Hierarchies

Hierarchies are frequently represented as trees, with the most general concepts at the root of the hierarchy and more specialized ones towards the branches.

When designing a hierarchy of classes you should ask yourself, which features(instance fields) and behaviors (methods) are common to all the classes that you are designing. Those common properties are collected in a superclass(the root of the tree).

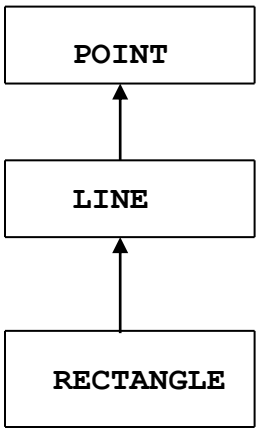
The hierarchy for u10a1, the Worker project, would look something like this...




This hierarchy tells us that the subclasses, HourlyWorker, SalariedWorker, and PieceWorker have instance fields and methods in common and that they are incorporated into the superclass, Worker.

Hierarchies can have more than two levels. This means that a class in the middle of the hierarchy is both a subclass and a superclass. Diagrams of such hierarchies can be found in your textbook on pages 434 & 435.

The hierarchy for u10a2, the Point project, would look something like this...




This hierarchy tells us that Line is a subclass of the superclass Point and that Rectangle is a subclass of the superclass Line. Speaking geometrically, this means that Lines are made up of Points and Rectangles are made of Lines.

**Topic 2: Final classes**

If you want to prevent other programmers from creating subclasses of one of your classes, you use the reserved word *final*. For example...

```
public final class Rectangle extends Line
```

This means that nobody can extend your Rectangle class, which also means they can't override methods of the Rectangle class.

This is done because the compiler can generate more efficient method calls if it knows that it doesn't have to worry about late binding.

If you prefer not to make your class *final*, but don't want one of your methods overridden, you can declare individual methods as *final*.