**Java Inheritance**

A Java class may inherit from another, via the **extends** keyword:

**public class USTEmployee extends Employee {…}**

"Java Inheritance" means two things:

1. Subclass objects automatically contain all of the non-**private** instance fields and instance methods of their superclass: they inherit their parent's structure and behavior.
2. Subclass object references may be assigned to any superclass-typed variable (upcasting). All of the methods of the superclass are either inherited or overridden (replaced) in the subclass, so all superclass method calls against this superclass-typed subclass reference are legal - and run the overridden subclass code.

Subclasses are free to add new instance fields and new methods as well, thus extending the structure and behavior of their superclass.

Java Code and Visualization (extends Example in Java Concept map document)  
  
**Example**

**public class USTEmployee extends Employee   
  
// define new instance field beyond Employee's**

**private int parkingID;**

**// declare USTEmployee constructor (initializer)**

**public USTEmployee (int parkID, int a, double i, String n)**

**{  
 super(a,i,n);   
 parkingID = parkID;**

**}**

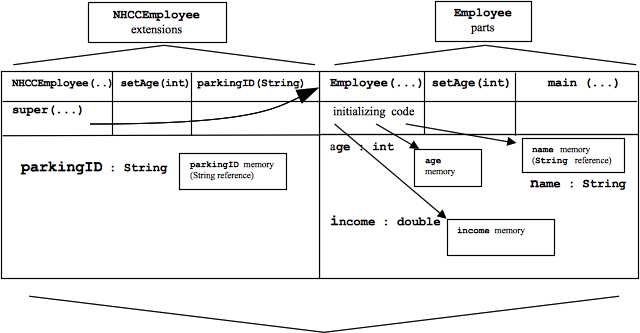
**// declare new method beyond Employee's**

**public void setParkingID (int parkID)**

**{  
  
 parkingID = parkID;  
 }  
  
// USTEmployees \*don't\* age! Overrides Employee's setAge(int)  
  
 public void setAge(int a)**

**{  
 // do nothing!  
 }  
  
public static void main (String[] args) {  
  
 USTEmployee evl =  
 = new Employee (1234, 47, 50000.0, "Eric");  
  
 evl.setAge (48); // Eric ages… or does he?  
 evl.setParkingID (4747); // and changes parking id  
}**

**Memory looks like below   
after main method creates a new USTEmployee…**



The entire **USTEmployee** object

(note subclass has a different name in diagram:

couldn't edit it!)

An **USTEmployee** inherits fields **int age**, **String name**, and **double income** from its superclass **Employee**.

The **USTEmployee** subclass also overrides **Employee**'s **setAge(int)**, replacing it with its own **setAge(int)**.

**USTEmployee** adds a new field **String parkingID**, along with a new method **setParkingID (String)**.

Because **USTEmployee**'s constructor cannot directly access its **Employee**-inherited fields, it must call some **Employee** constructor as its first statement: **super (a, i, n);**

Java forces every constructor in a subclass to call some constructor of its immediate superclass.