class only if you want to express "non-existence". Assign null to a variable of some null represents "non-existence." It is automatically assigned the value null.

When a variable of some class is declared without any initial instantiation,

```
AsShape s;
/*
 * is given the value null (similar to null in Scheme).
 */
```

What is null?
Instance Fields and Instance Methods
• It can only be called on an existing instance of a class.

• double area() is said to be an instance method of s or rectangle.

Instance Fields and Instance Methods (cont.)
the keyword static.

and that can be accessed by all methods in Rectangle. In Java, we use

Answer: a field that is unique and global to all instances of Rectangle

during the course of our program. What do we need?

Suppose we want to keep track of how many Rectangles are being created

Static Fields and Static Methods
class Rectangle extends Shape{

  ... other code ...  //
  
  -iCOUNT++;
  
  -iCOUNT is incremented by 1.  //
  Each time this constructor is called,
  
  }

public Rectangle(double width, double height)

  ... other code ...  //

  private static int -iCOUNT;  // Initial value is 0.

  }

Static Fields and Static Methods (cont.)

January 24, 2001
Comp 212
Example: System.out.println("crpzza");

System.out.println("crpzza");

(Recall main.)

• Static methods can be called before any instantiation of the class.

• Static fields.

• Static methods cannot access instance fields. They can only access

Static Fields and Static Methods (cont.)
subclasses), namely Rectangle and Circle. The UML diagram shows the taxonomy tree of a Shape and its variants (or subclasses) all of its subclasses form a taxonomy tree. The above
A variable of class AShape can be assigned any instance of subclasses of AShape at any time in a program. AShape is said to be polymorphic.

```java
Circle u = new Rectangle(5, 6); // NO!

int t = s; // OK, the old Rectangle is gone.

Shape t = new Rectangle(3, 4); // OK.
Shape s = new Circle(2.7); // OK.
```

**Polymorphism**
We can think of polymorphism as viewing the taxonomy tree from the top down.

Class can be represented any of its subclasses.

In general, a variable of a superclass can be assigned an instance of any

Polymorphism (cont.)
We can think of inheritance as viewing the taxonomy tree from the bottom up.

B is said to be a specialization of A. As such, B may also have additional fields and methods that A has absolutely no knowledge of. In Java, B can override a non-private method of A, and provide new behavior by redefining it with new code. B can override a non-private method of A, and perform all the non-private methods of A.

Class B inherits class A means that B can automatically access all the non-private fields of A and perform all the non-private methods of A.

The "is-a" relationship between two classes is called inheritance.

Inheritance