Recap: References vs Values
- You copy a CD for your friend. Her dog chews it up. Does that affect your CD?
  - no: different values
  - like primitive types
- You and your friend start eating a slice of cake on one shared plate. You get up to make a cup of tea. Her dog jumps on the table and eats the cake. Does that affect your half of the dessert?
  - yes: both forks reference the same plate
  - like objects

Recap: Designer
- Decide on inner workings
  - implementation of class
- Objects need state
  - attributes that distinguish one instance from another
    - many names for these
  - state variables
  - fields
  - attributes
  - data members
  - what fields should we create for Die?

Recap: Designing Die Class
- Blueprint for constructing objects of type Die
- Think of manufacturing airplanes or dresses or whatever
  - design one blueprint or pattern
  - manufacture many instances from it
- Consider two viewpoints
  - client programmer: wants to use Die object in a program
  - designer: creator of Die class

Implementing Die
```java
/**
 * Provides a simple model of a die (as in pair of dice).
 */
public class Die {
    //
}
```

Random Numbers
```java
Random Numbers
```

Implementing Die
```java
/**
 * Provides a simple model of a die (as in pair of dice).
 */
public class Die {
    int addTwoInts(int a, int b) {
        return a + b;
    }
}
```

Information Hiding
```java
Information Hiding
```

Public vs. Private
```java
Public vs. Private Example
```
Public vs. Private Example

Die myDie = new Die();
int result = myDie.roll(); // OK
myDie.cheat(6); // not allowed!

Implementing Die

/**
 * Provides a simple model of a die
 * (as in pair of dice).
 */
public class Die{
    // methods
}

Trying It Out!

- Die class has no main method.
- Best is to write another class that instantiates some objects of your new class and tries them out.
- Sometimes called a "tester" or "testbench"

Implementing RollDice

public class RollDice{
    public static void main ( String [] args)
    {
    }
}