# Good Class Design

- very challenging
- maximize
  - cohesiveness
  - consistency
- minimize
  - dependencies (coupling)
  - side effects

# Discover

- nouns in the problem domain tend to be classes
- verbs in the problem domain tend to be methods on classes
- avoid defining classes to perform actions

# Utility Classes

- rare
- contain static methods and constants
- no objects of this class are instantiated
- example: `Math`

# Maximize Cohesiveness

- class represents a single concept
- reflected in public interface
- improves testability, reliability, maintainability, extensibility
- decreases coupling

# Lack of Cohesion

```
public class CashRegister
{
   public static final double QUARTER_VALUE = 0.25;
   public static final double DIME_VALUE = 0.1;
   public static final double NICKEL_VALUE = 0.05;
   . . .
   public void receivePayment(int dollars,
         int quarters, int dimes, int nickels,
         int pennies)
   . . .
}
```

- Why?

# Refactor Responsibilities into Two Classes

```
public class Coin
{
   public Coin(double aValue, String aName) { . . . }
   public double getValue() { . . . }
   . . .
}

public class CashRegister
{
   . . .
   public void receivePayment(int coinCount,Coin coinType)
   {
      payment = payment + coinCount * coinType.getValue();
   }
   . . .
}
```

# Dependencies (Coupling)

- when one class requires another class
  - `CashRegister` *depends on* `Coin`
- strive to minimize dependencies
- strive to have unidirectional dependencies through decoupling

# Immutable Classes

- no mutator methods
- appropriate for classes that model values
- safe to share references to objects of immutable classes
  - no unexpected side effects

# Minimize Side Effects

- modification of data that is apparent outside of the context of the method invoked
- mutators modify the attributes of an object; this is expected

# Examples of Side Effects

- modifying a parameter variable is a side effect
- modifying an external object

```
// Not recommended
public void printBalance()
{
   System.out.println(
      "The balance is now $" +
      balance);
}
```