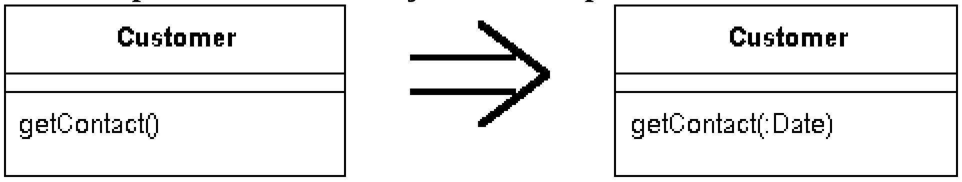
Refactoring Examples

Extracted from http://www.refactoring.com/

Add Parameter

A method needs more information from its caller.

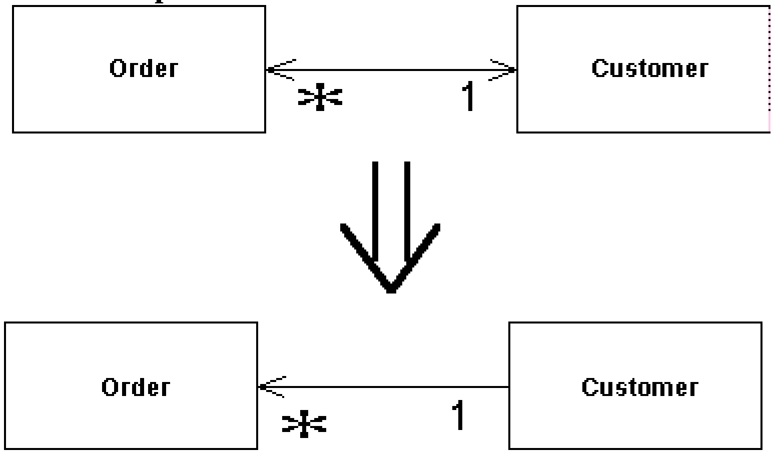
Add a parameter for an object that can pass on this information.



Change Bidirectional Association to Unidirectional

You have a two-way association but one class no longer needs features from the other.

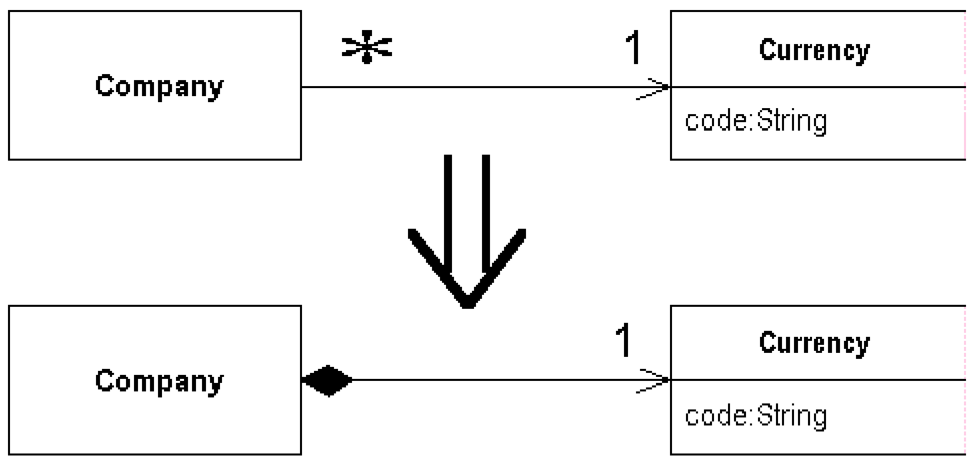
Drop the unneeded end of the association.



Change Reference to Value

You have a reference object that is small, immutable, and awkward to manage.

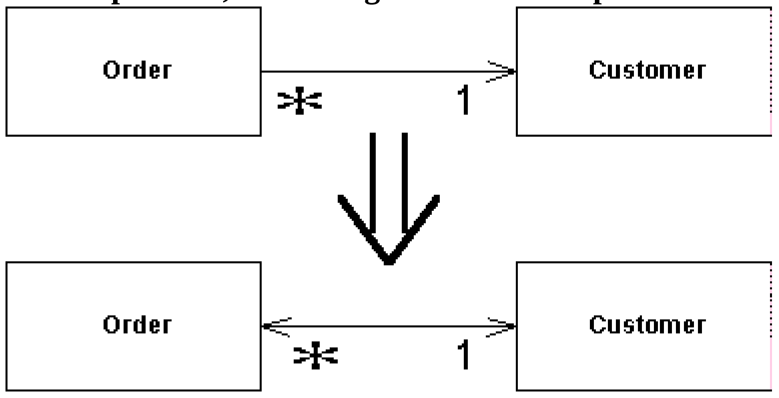
Turn it into a value object.



Change Unidirectional Association to Bidirectional

You have two classes that need to use each other's features, but there is only a one-way link.

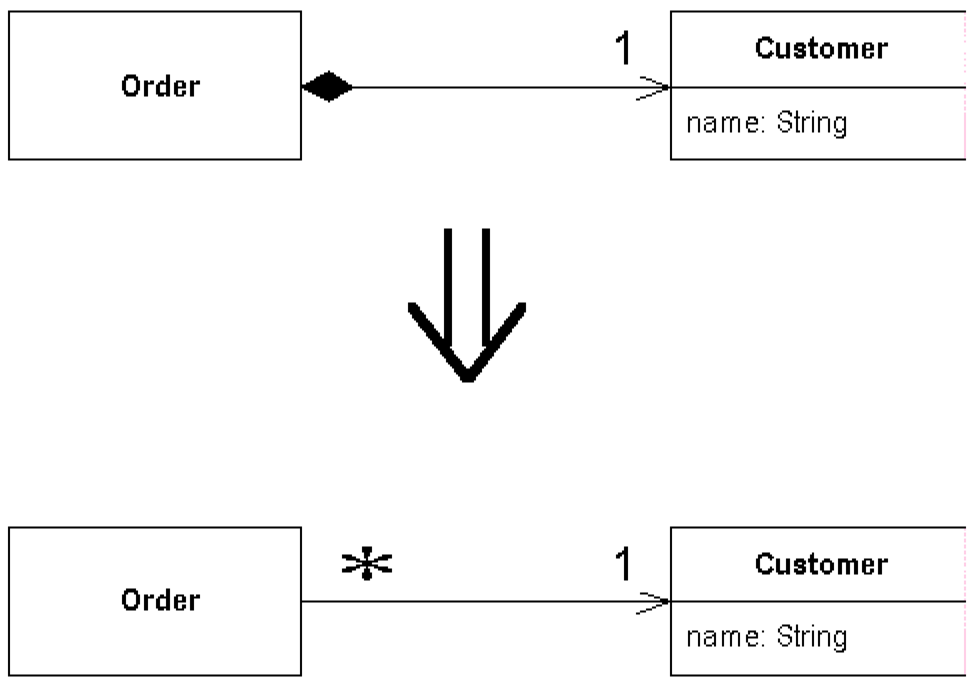
Add back pointers, and change modifiers to update both sets.



Change Value to Reference

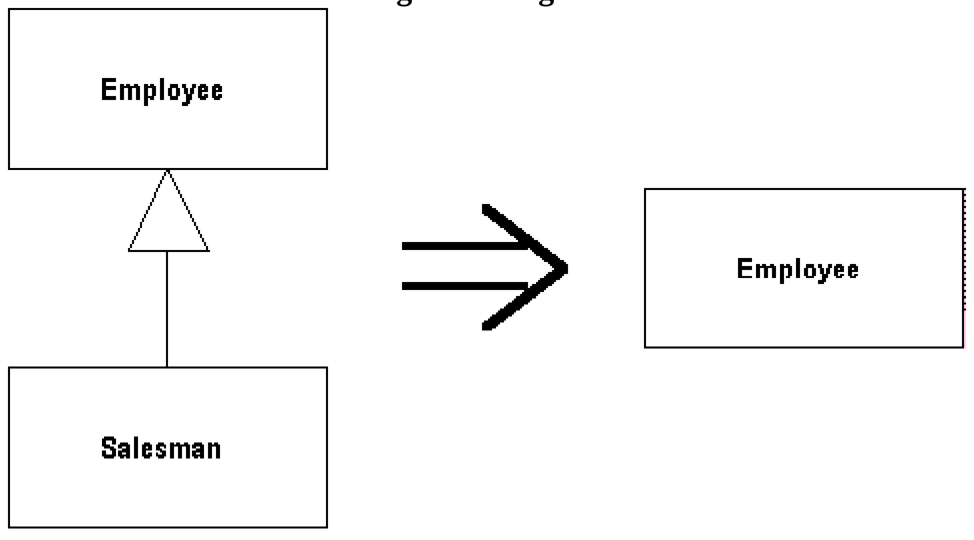
You have a class with many equal instances that you want to replace with a single object.

Turn the object into a reference object.



Collapse Hierarchy

A superclass and subclass are not very different.
Merge them together.



Consolidate Conditional Expression

You have a sequence of conditional tests with the same result.

Combine them into a single conditional expression and extract it.

```
double disabilityAmount() {
            if (_seniority < 2) return 0;
            if (_monthsDisabled > 12) return 0;
            if (_isPartTime) return 0;
            // compute the disability amount
```

$$\Downarrow$$

```
double disabilityAmount() {
            if (isNotEligableForDisability()) return 0;
            // compute the disability amount
```

Consolidate Duplicate Conditional Fragments

The same fragment of code is in all branches of a conditional expression.

Move it outside of the expression.

```
if (isSpecialDeal()) {
      total = price * 0.95;
      send();
}
else {
      total = price * 0.98;
      send();
}
```

$$\Downarrow$$

```
if (isSpecialDeal())
      total = price * 0.95;
else
      total = price * 0.98;
send();
```

Convert Dynamic to Static Construction by Gerard M. Davison

You have code that loads other classes dynamically. This can introduce a un-warranted overhead and can produce code that is more fragile.

Replace the dynamic class loading with static code.

```
try
{
         DataProvider dp = (DataProvider)
Class.forName("org.davison.data.jdbc.JDBCProvider").newInstance();
}
catch (IllegalAccessException iae)
{
         // Convert exception to error to preseve the interface.
         //

         throw new IllegalAccessError(iae.getMessage());
}
catch (InstantiationException ie)
{
         // Convert exception to error to preseve the interface.
         //

         throw new InstantiationError(ie.getMessage());
}
catch (ClassNotFoundException cnfe)
{
         // Convert exception to error to preseve the interface.
         //

         throw new NoClassDefFoundError(cnfe.getMessage());
}
```

$$\Downarrow$$

```
import org.davison.data.jdbc.JDBCProvider;

.
.
.

DataProvider dp = new JDBCProvider();
```

Convert Static to Dynamic Construction by Gerard M. Davison NEW

You have classes that have static compile time dependencies on classes that can only be built on a specific platform.

Make use of the java.lang.reflect to break the static dependency.

```
import org.davison.data.jdbc.JDBCProvider;

.
.
.

DataProvider dp = new JDBCProvider();
```

$$\Downarrow$$

```
try
{
         DataProvider dp = (DataProvider)
Class.forName("org.davison.data.jdbc.JDBCProvider").newInstance();
}
catch (IllegalAccessException iae)
{
         // Convert exception to error to preseve the interface.
         //

         throw new IllegalAccessError(iae.getMessage());
}
catch (InstantiationException ie)
{
         // Convert exception to error to preseve the interface.
         //

         throw new InstantiationError(ie.getMessage());
}
catch (ClassNotFoundException cnfe)
{
         // Convert exception to error to preseve the interface.
         //

         throw new NoClassDefFoundError(cnfe.getMessage());
}
```

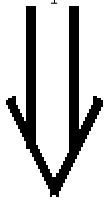
Decompose Conditional

You have a complicated conditional (if-then-else) statement.

Extract methods from the condition, then part, and else parts.

```
if (date.before (SUMMER_START) ||
date.after(SUMMER_END))
```

```
            charge = quantity * _winterRate +
            _winterServiceCharge;
      else charge = quantity * _summerRate;
```

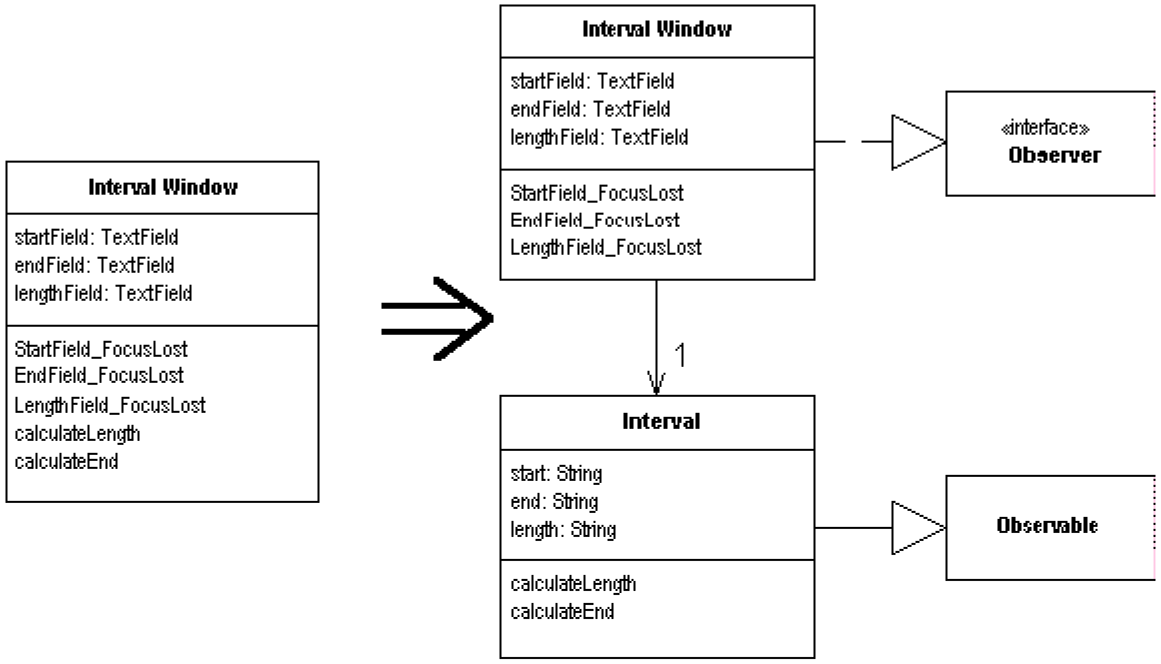


```
      if (notSummer(date))
            charge = winterCharge(quantity);
      else charge = summerCharge (quantity);
```

Duplicate Observed Data

You have domain data available only in a GUI control, and domain methods need access.

Copy the data to a domain object. Set up an observer to synchronize the two pieces of data.



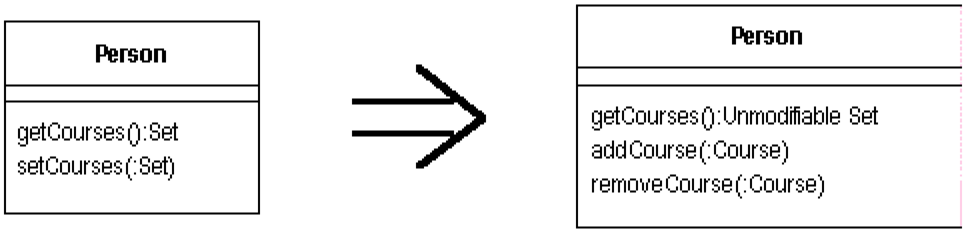
Eliminate Inter-Entity Bean Communication (Link Only)

Reduce or eliminate the inter-entity bean relationships by using coarse-grained entity bean (Composite Entity) with dependent objects

Encapsulate Collection

A method returns a collection.

Make it return a read-only view and provide add/remove methods.



Encapsulate Downcast

A method returns an object that needs to be downcasted by its callers.

Move the downcast to within the method.

```
Object lastReading() {
        return readings.lastElement();
}
```

$$\Downarrow$$

```
Reading lastReading() {
        return (Reading) readings.lastElement();
}
```

Encapsulate Field

There is a public field.

Make it private and provide accessors.

```
public String _name
```

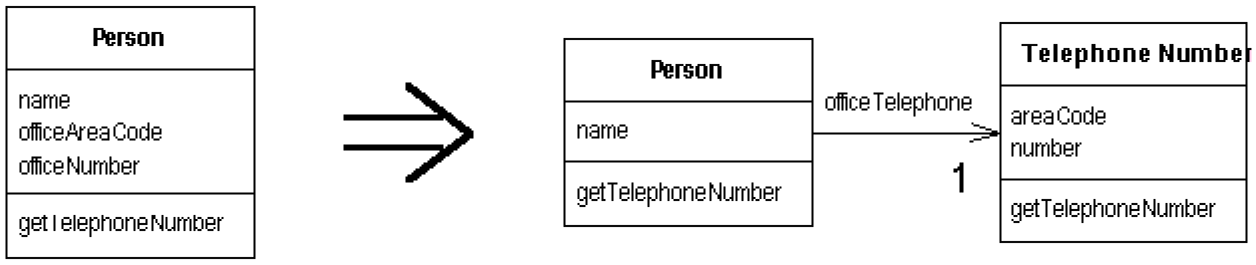
$$\Downarrow$$

```
private String _name;
public String getName() {return _name;}
public void setName(String arg) {_name = arg;}
```

Extract Class

You have one class doing work that should be done by two.

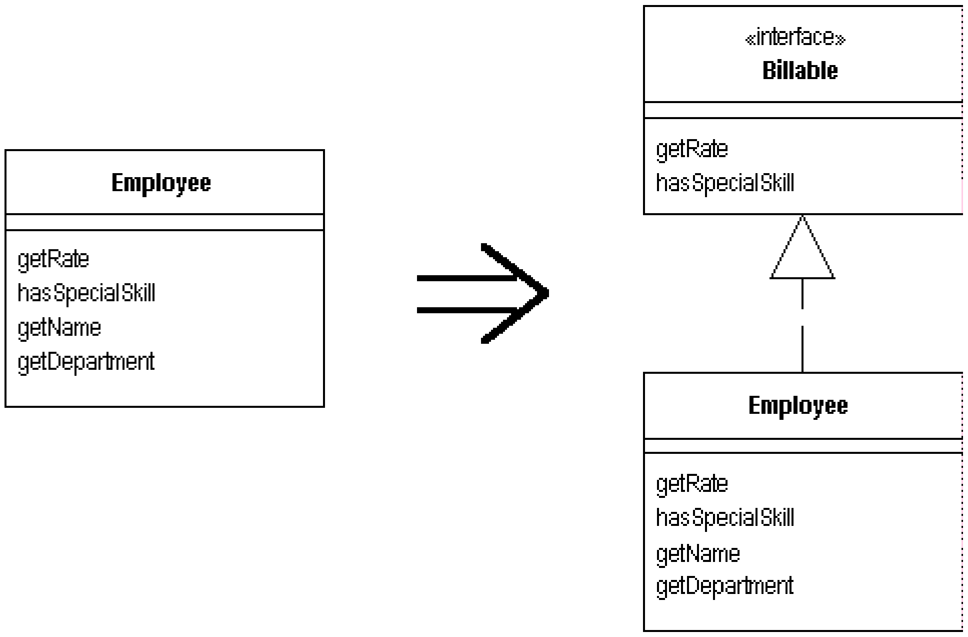
Create a new class and move the relevant fields and methods from the old class into the new class.



Extract Interface

Several clients use the same subset of a class's interface, or two classes have part of their interfaces in common.

Extract the subset into an interface.



Extract Method

You have a code fragment that can be grouped together.

Turn the fragment into a method whose name explains the purpose of the method.

```
void printOwing() {
        printBanner();

        //print details
        System.out.println ("name:      " + _name);
        System.out.println ("amount     " +
        getOutstanding());
}
```

⇓

```
void printOwing() {
            printBanner();
            printDetails(getOutstanding());
}

void printDetails (double outstanding) {
            System.out.println ("name:       " + _name);
            System.out.println ("amount      " + outstanding);
}
```

Extract Package by Gerard M. Davison

A package either has too many classes to be easily understandable or it suffers from the 'Promiscuous packages' smell.

Extract a sub package depending on their gross dependencies or usages.

```
interface org.davison.data.DataProvider
class org.davison.data.DataFactory

// Database classes

class org.davison.data.JDBCProvider
class org.davison.data.JDBCHelper
class org.davison.data.JDBCUtils
```

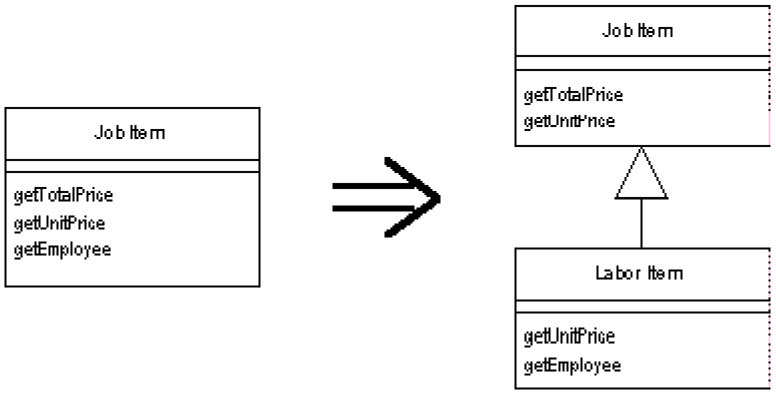
⇓

```
interface org.davison.data.DataProvider
class org.davison.data.DataFactory

// Database classes

class org.davison.data.jdbc.JDBCProvider
class org.davison.data.jdbc.JDBCHelper
class org.davison.data.jdbc.JDBCUtils
```

Extract Subclass

A class has features that are used only in some instances.

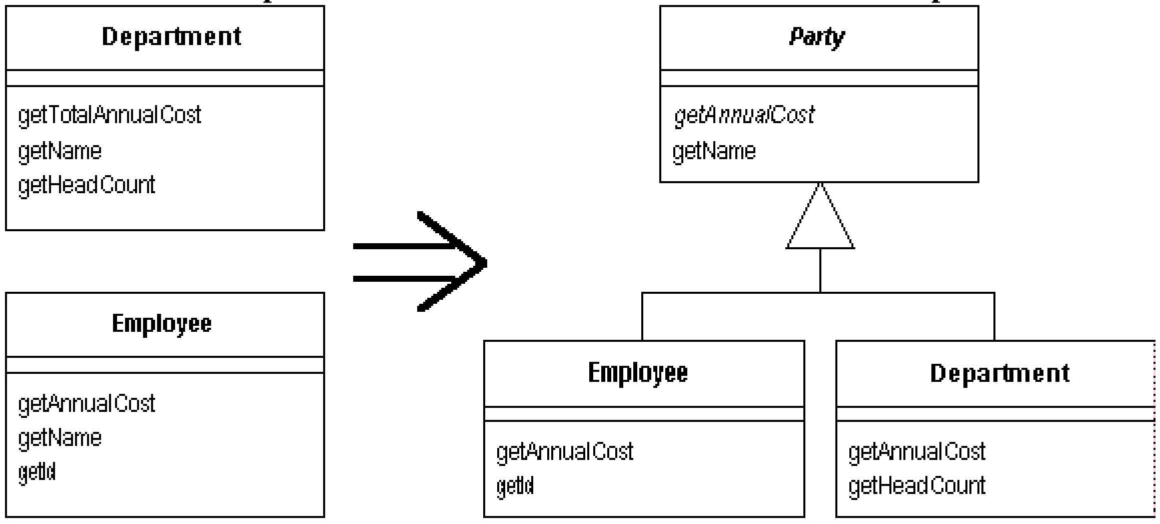
Create a subclass for that subset of features.



Extract Superclass

You have two classes with similar features.

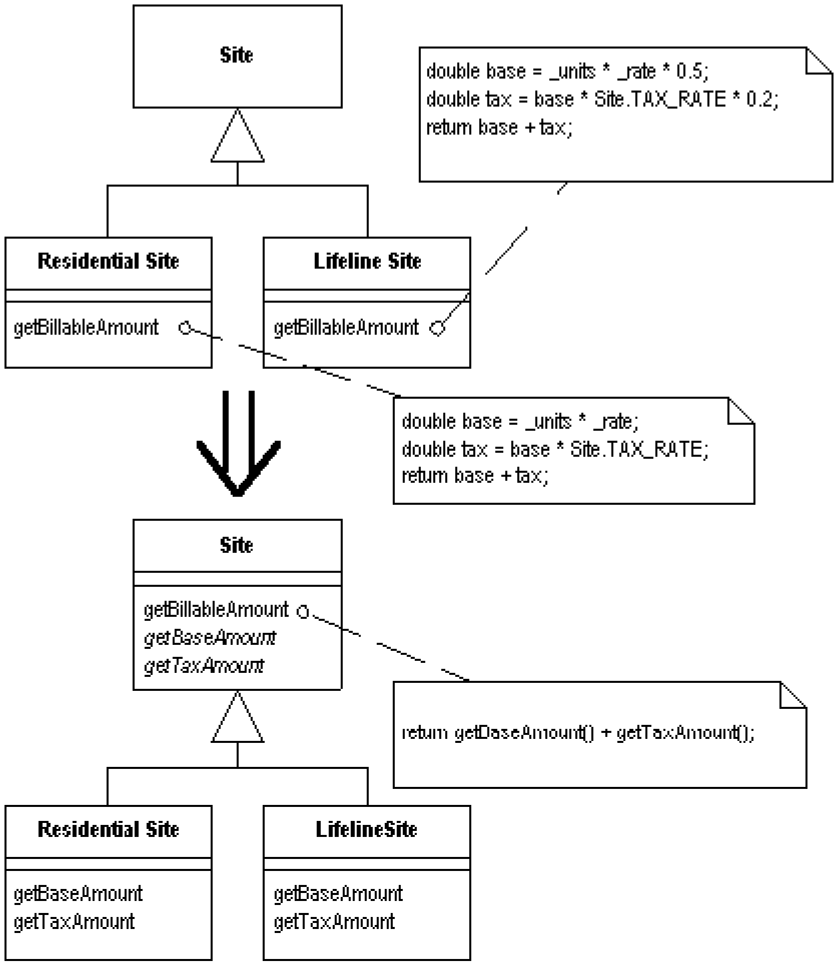
Create a superclass and move the common features to the superclass.



Form Template Method

You have two methods in subclasses that perform similar steps in the same order, yet the steps are different.

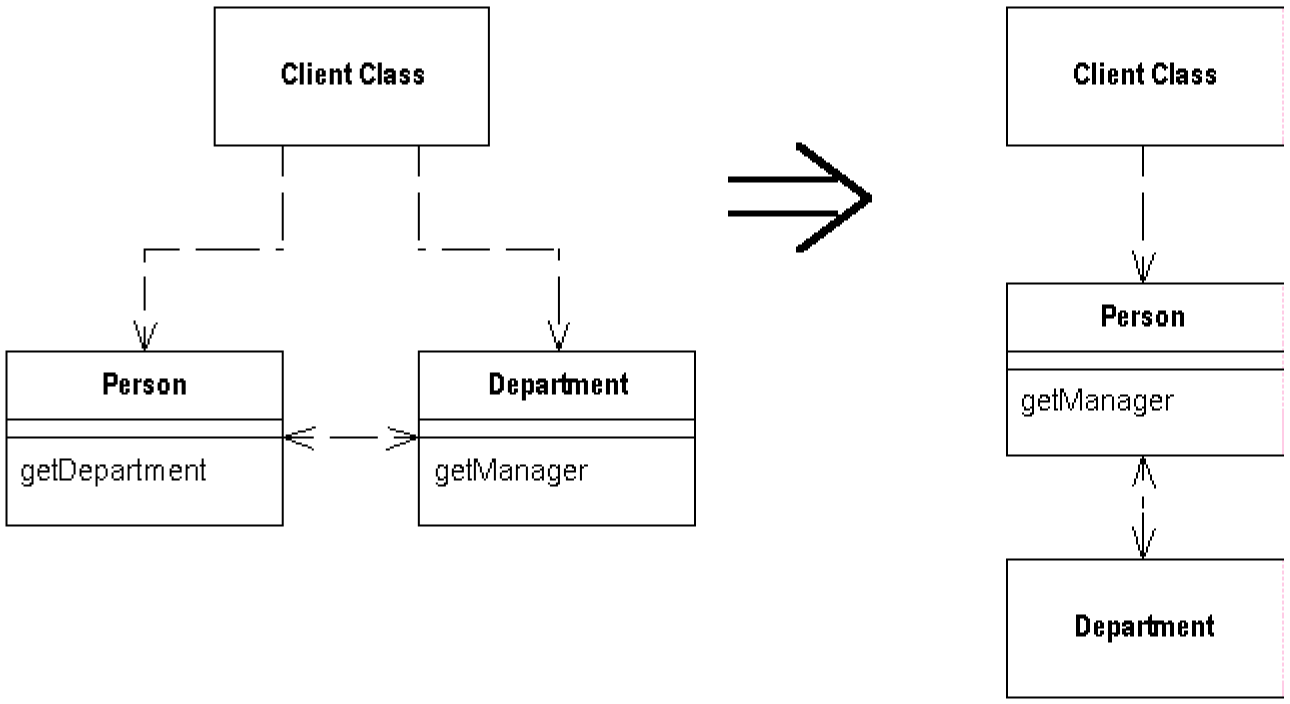
Get the steps into methods with the same signature, so that the original methods become the same. Then you can pull them up.



Hide Delegate

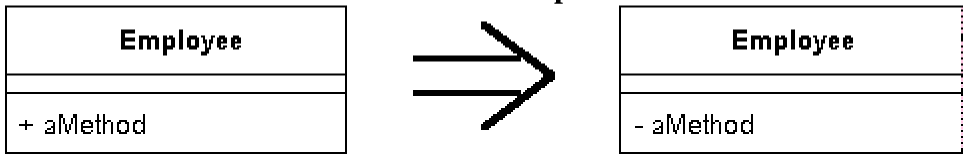
A client is calling a delegate class of an object.

Create methods on the server to hide the delegate.



Hide Method

A method is not used by any other class.
Make the method private.



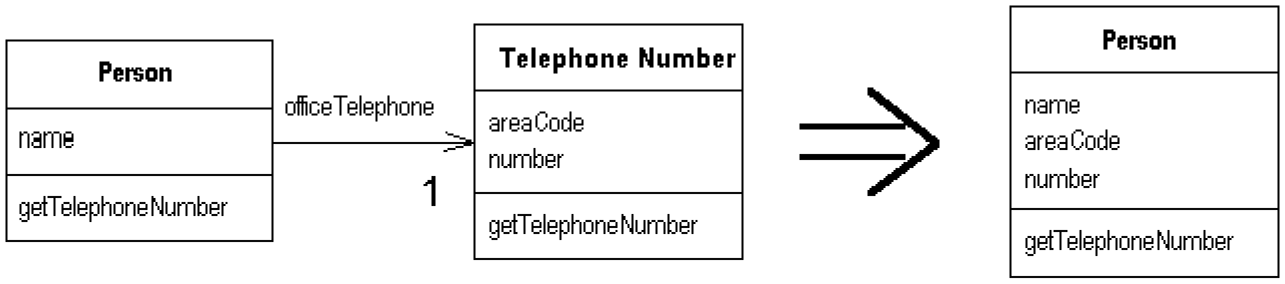
Hide presentation tier-specific details from the business tier (Link Only)

Request handling and/or protocol-related data structures are exposed from the presentation tier to the business tier
Remove all references to request handling and protocol-related presentation tier data structures from the business tier. Pass values between tiers using more generic data structures.

Inline Class

A class isn't doing very much.

Move all its features into another class and delete it.



Inline Method

A method's body is just as clear as its name.

Put the method's body into the body of its callers and remove the method.

```
int getRating() {
            return (moreThanFiveLateDeliveries()) ? 2 : 1;
}
boolean moreThanFiveLateDeliveries() {
            return _numberOfLateDeliveries > 5;
}
```

$$\Downarrow$$

```
int getRating() {
            return (_numberOfLateDeliveries > 5) ? 2 : 1;
}
```

Inline Temp

You have a temp that is assigned to once with a simple expression, and the temp is getting in the way of other refactorings.

Replace all references to that temp with the expression.

```
            double basePrice = anOrder.basePrice();
            return (basePrice > 1000)
```

$$\Downarrow$$

```
            return (anOrder.basePrice() > 1000)
```

Introduce A Controller (Link Only)

Control logic is scattered throughout the application, typically duplicated in multiple Java Server Page (JSP) views

Extract control logic into one or more controller classes that serve as the initial contact point for handling a client request

Introduce Assertion

A section of code assumes something about the state of the program.

Make the assumption explicit with an assertion.

```
double getExpenseLimit() {
            // should have either expense limit or a primary
            project
            return (_expenseLimit != NULL_EXPENSE) ?
                  _expenseLimit:
                  _primaryProject.getMemberExpenseLimit();
}
```

⇓

```
double getExpenseLimit() {
            Assert.isTrue (_expenseLimit != NULL_EXPENSE ||
            _primaryProject != null);
            return (_expenseLimit != NULL_EXPENSE) ?
                  _expenseLimit:
                  _primaryProject.getMemberExpenseLimit();
}
```

Introduce Business Delegate (Link Only)

Session beans in the business tier are exposed to clients in other tiers

Use a business delegate to decouple the tiers and to hide the implementation details

Introduce Explaining Variable

You have a complicated expression.

Put the result of the expression, or parts of the expression, in a temporary variable with a name that explains the purpose.

```
if ( (platform.toUpperCase().indexOf("MAC") > -1) &&
         (browser.toUpperCase().indexOf("IE") > -1) &&
         wasInitialized() && resize > 0 )
{
// do something
}
```

⇓

```
         final boolean isMacOs     =
         platform.toUpperCase().indexOf("MAC") > -1;
         final boolean isIEBrowser =
         browser.toUpperCase().indexOf("IE")  > -1;
         final boolean wasResized  = resize > 0;

         if (isMacOs && isIEBrowser && wasInitialized() && wasResized)
         {
                     // do something
         }
```

Introduce Foreign Method

A server class you are using needs an additional method, but you can't modify the class.
Create a method in the client class with an instance of the server class as its first argument.

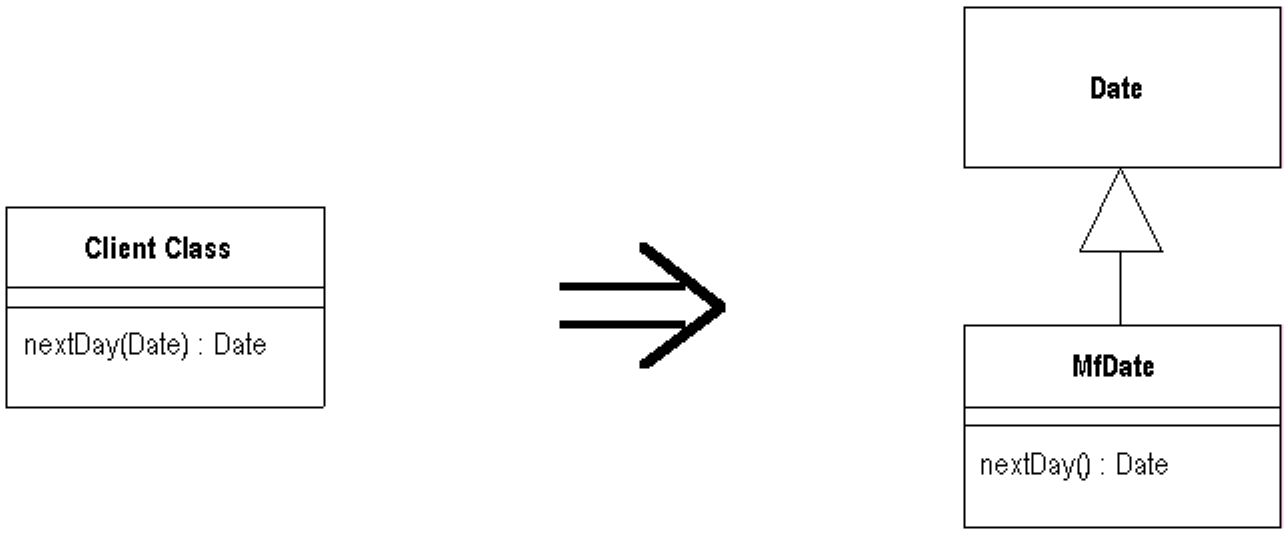
```
Date newStart = new Date (previousEnd.getYear(),
                                                   previousEnd.getMonth
                                                   (),
                                                   previousEnd.getDate(
                                                   ) + 1);
```

⇓

```
Date newStart = nextDay(previousEnd);

private static Date nextDay(Date arg) {
         return new Date (arg.getYear(),arg.getMonth(), arg.getDate() +
         1);
}
```

Introduce Local Extension

A server class you are using needs several additional methods, but you can't modify the class.
Create a new class that contains these extra methods. Make this extension class a subclass or a wrapper of the original.

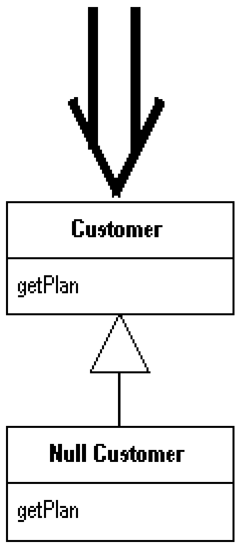


Introduce Null Object

You have repeated checks for a null value.

Replace the null value with a null object.

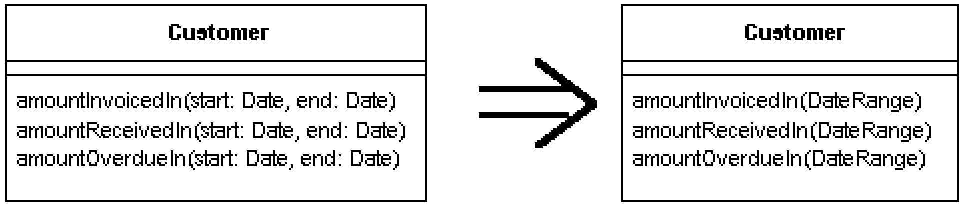
```
if (customer == null) plan =
BillingPlan.basic();
else plan = customer.getPlan();
```



Introduce Parameter Object

You have a group of parameters that naturally go together.

Replace them with an object.


Customer
amountInvoicedIn(start: Date, end: Date)
amountReceivedIn(start: Date, end: Date)
amountOverdueIn(start: Date, end: Date)
Customer
amountInvoicedIn(DateRange)
amountReceivedIn(DateRange)
amountOverdueIn(DateRange)

Introduce Synchronizer Token (Link Only)

Clients make duplicate resource requests that should be monitored and controlled, or clients access certain views out of order by returning to previously bookmarked pages

Use a shared token to monitor and control the request flow and client access to certain resources

Localize Disparate Logic (Link Only)

Business logic and presentation formatting are intermingled within a JSP view

Extract business logic into one or more helper classes that can be used by the JSP or by a controller

Merge Session Beans (Link Only)

Create a one-to-one mapping between session beans and entity beans

Map coarse-grained business services to session beans. Eliminate or combine session beans that act solely as entity bean proxies into session beans that represent coarse grained business services

Move Business Logic to Session (Link Only)

Inter-entity bean relationships introduce overhead in the model

Encapsulate the workflow related to inter-entity bean relationships in a session bean (Session Facade)

Move Class by Gerard M. Davison NEW

You have a class that is in a package that contains other classes that it is not related to in function.

Move the class to a more relevant package. Or create a new package if required for future use.

```
class org.davison.ui.TextThing
class org.davison.ui.TextProcessor
class org.davison.log.Logger

depends on

class org.davison.ui.StringUtil
```

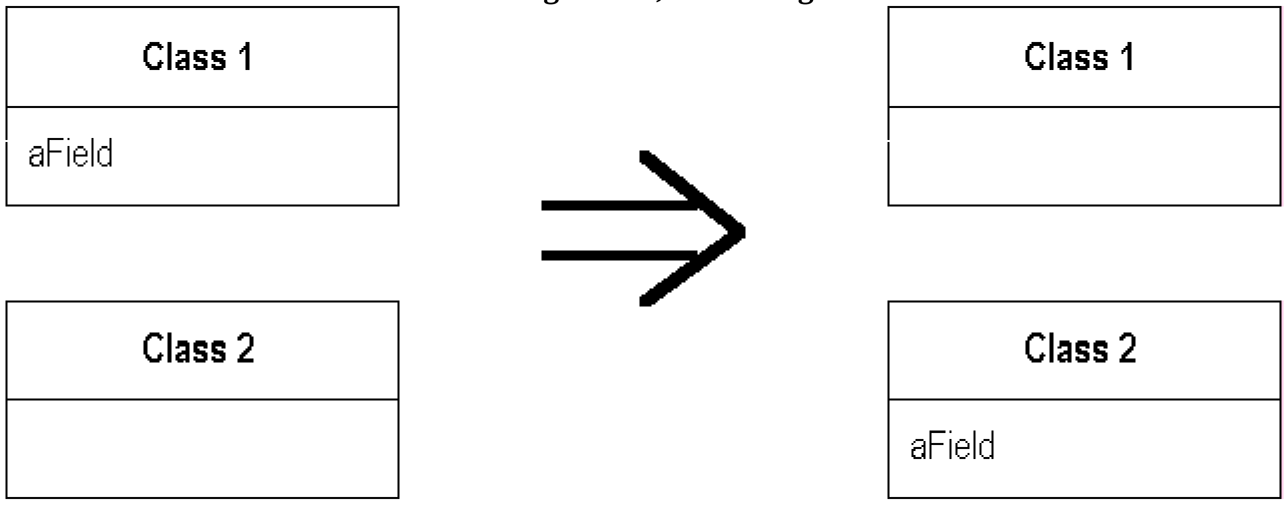
$$\Downarrow$$

```
class org.davison.ui.TextThing
class org.davison.ui.TextProcessor
class org.davison.log.Logger

depends on

class org.davison.util.StringUtil
```

Move Field

A field is, or will be, used by another class more than the class on which it is defined.

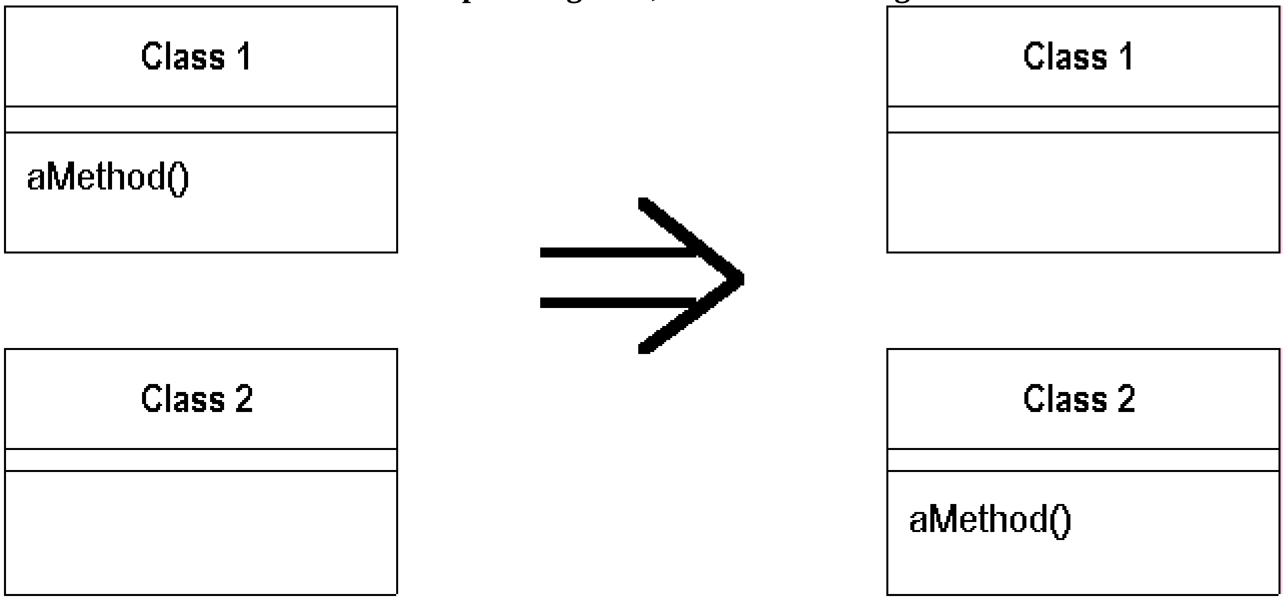
Create a new field in the target class, and change all its users.



Move Method

A method is, or will be, using or used by more features of another class than the class on which it is defined.

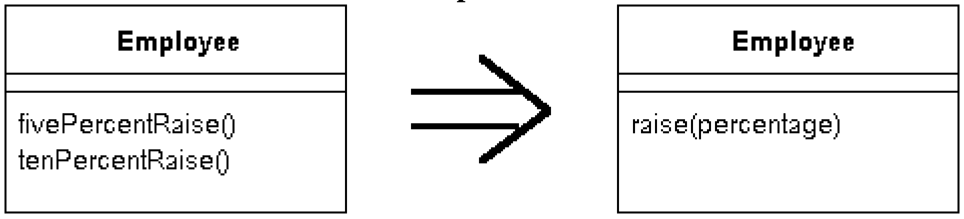
Create a new method with a similar body in the class it uses most. Either turn the old method into a simple delegation, or remove it altogether.



Parameterize Method

Several methods do similar things but with different values contained in the method body.

Create one method that uses a parameter for the different values.



Preserve Whole Object

You are getting several values from an object and passing these values as parameters in a method call.

Send the whole object instead.

```
int low = daysTempRange().getLow();
int high = daysTempRange().getHigh();
withinPlan = plan.withinRange(low, high);
```

⇓

```
                        withinPlan = plan.withinRange(daysTempRange());
```

Pull Up Constructor Body

You have constructors on subclasses with mostly identical bodies.

Create a superclass constructor; call this from the subclass methods.

```
class Manager extends Employee...
         public Manager (String name, String id, int grade) {
                     _name = name;
                     _id = id;
                     _grade = grade;
         }
```

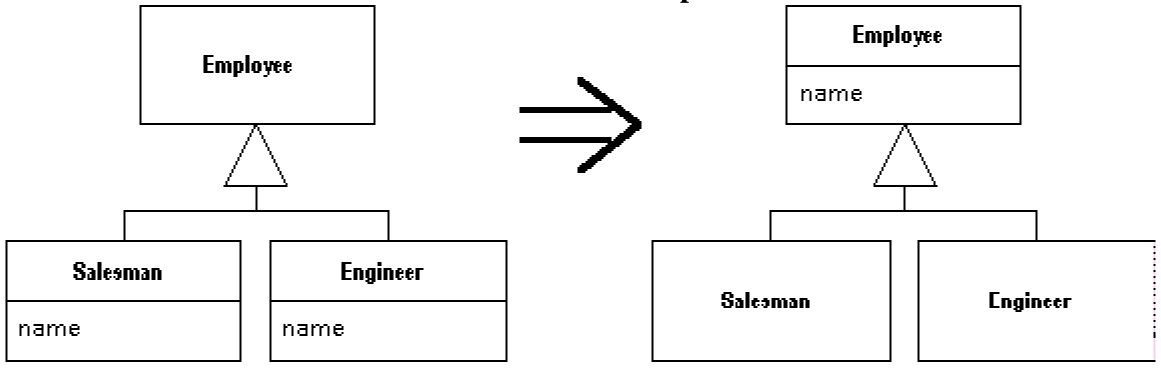
⇓

```
         public Manager (String name, String id, int grade) {
                     super (name, id);
                     _grade = grade;
         }
```

Pull Up Field

Two subclasses have the same field.

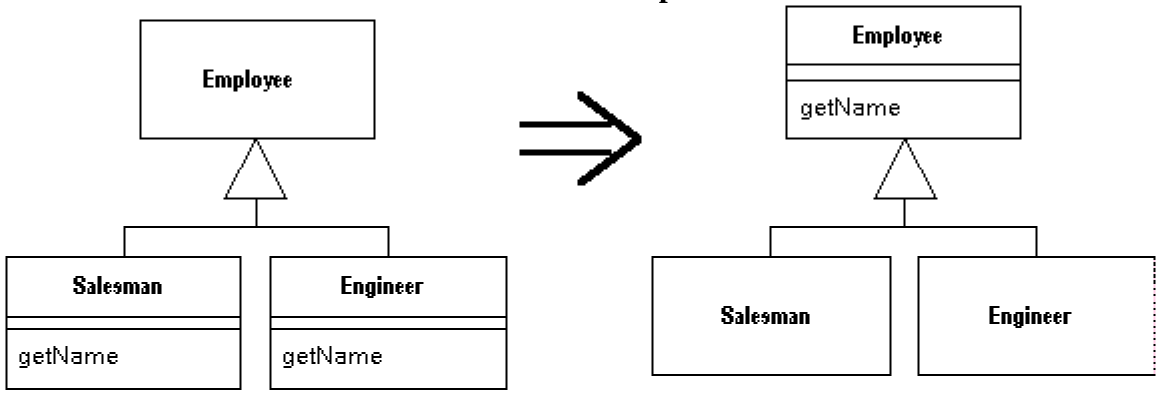
Move the field to the superclass.



Pull Up Method

You have methods with identical results on subclasses.

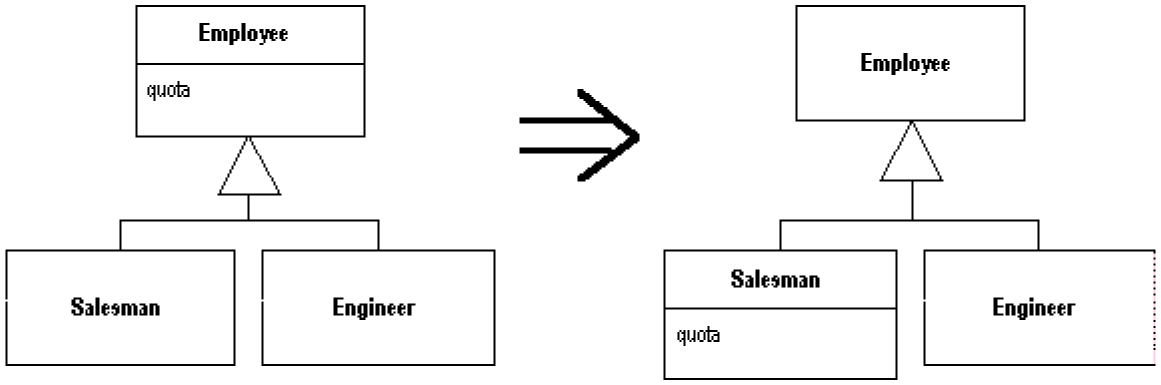
Move them to the superclass.



Push Down Field

A field is used only by some subclasses.

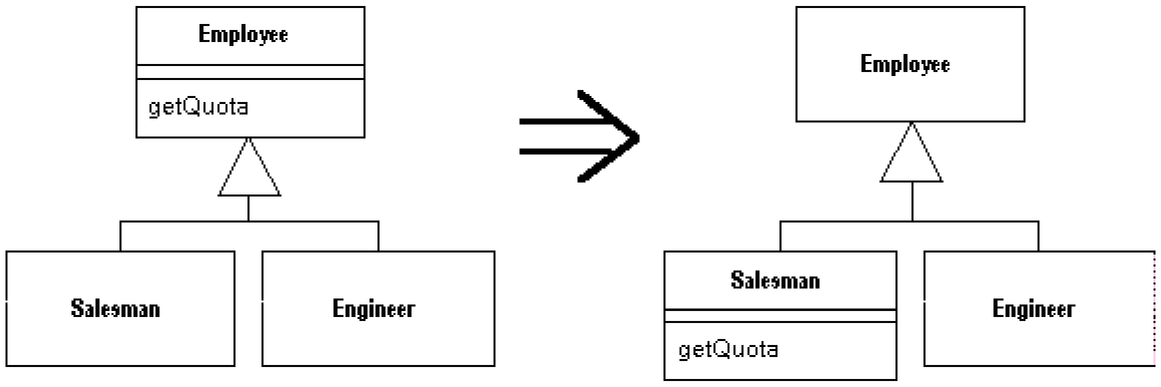
Move the field to those subclasses.



Push Down Method

Behavior on a superclass is relevant only for some of its subclasses.

Move it to those subclasses.

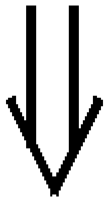


Reduce Scope of Variable by Mats Henricson

You have a local variable declared in a scope that is larger than where it is used

Reduce the scope of the variable so that it is only visible in the scope where it is used

```
void foo()
{
int i = 7;

// i is not used here

if (someCondition)
{
         // i is used only within this block
}

// i is not used here
}
```



```
void foo()
{
// i can not be used here

if (someCondition)
{
         int i = 7;

         // i is used only within this block
}

// i can not be used here
}
```

Refactor Architecture by Tiers (Link Only)

Increasing architectural sophistication requires changing the localization of data access logic and processing logic

Move Data Access code logically and/or physically closer to the actual data source. Move processing logic out of the client and presentation tiers into the business tier.

Remove Assignments to Parameters

The code assigns to a parameter.

Use a temporary variable instead.

```
int discount (int inputVal, int quantity, int yearToDate) {
        if (inputVal > 50) inputVal -= 2;
```

⇓

```
int discount (int inputVal, int quantity, int yearToDate) {
        int result = inputVal;
        if (inputVal > 50) result -= 2;
```

Remove Control Flag

You have a variable that is acting as a control flag for a series of boolean expressions.

Use a break or return instead.

Remove Double Negative by Ashley Frieze and Martin Fowler

You have a double negative conditional.

Make it a single positive conditional

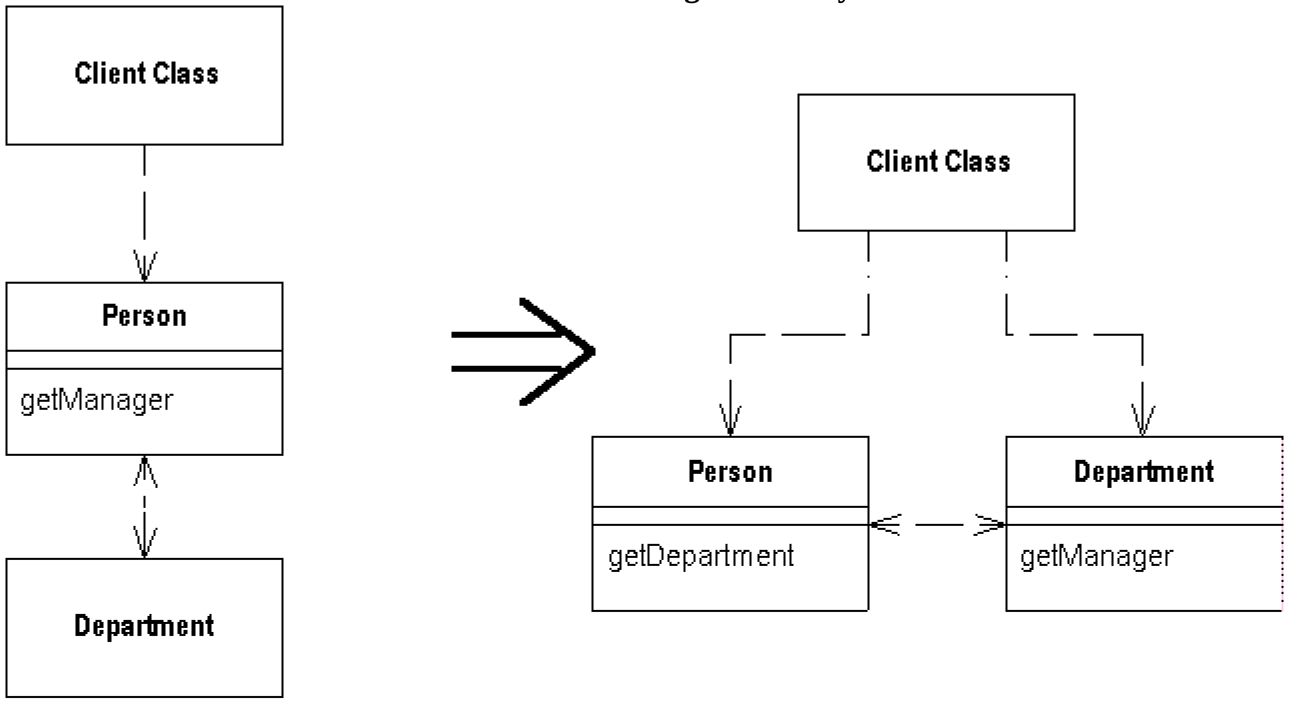
```
if ( !item.isNotFound() )
```

⇓

```
if ( item.isFound() )
```

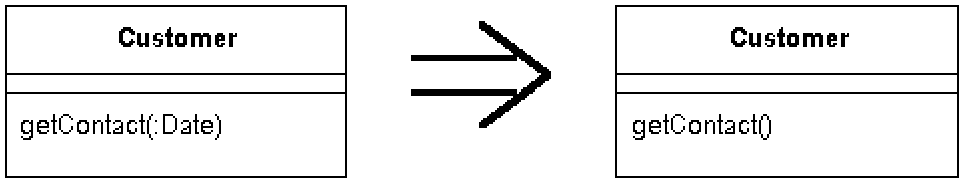
Remove Middle Man

A class is doing too much simple delegation.
Get the client to call the delegate directly.



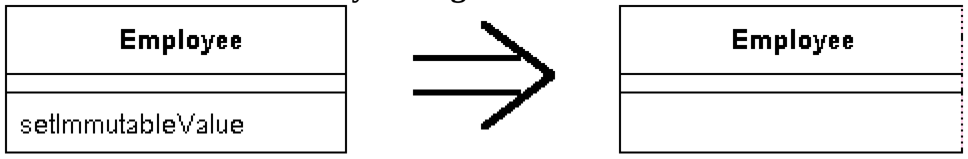
Remove Parameter

A parameter is no longer used by the method body.
Remove it.



Remove Setting Method

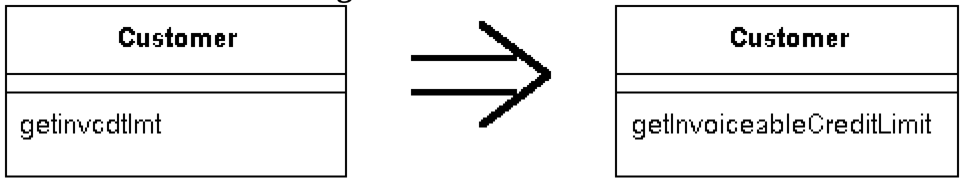
A field should be set at creation time and never altered.
Remove any setting method for that field.



Rename Method

The name of a method does not reveal its purpose.

Change the name of the method.



Replace Array with Object

You have an array in which certain elements mean different things.

Replace the array with an object that has a field for each element.

```
String[] row = new String[3];
row [0] = "Liverpool";
row [1] = "15";
```

⇓

```
Performance row = new Performance();
row.setName("Liverpool");
row.setWins("15");
```

Replace Assignment with Initialization by Mats Henricson

You have code that first declares a variable and then assigns a value to it

Make it into a direct initialization instead

```
void foo() {
int i;
// ....
i = 7;
}
```

⇓

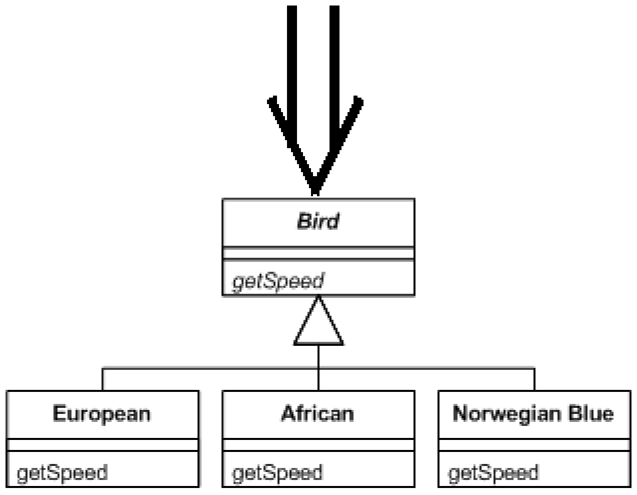
```
void foo() {
// ...
int i = 7;
}
```

Replace Conditional with Polymorphism

You have a conditional that chooses different behavior depending on the type of an object.

Move each leg of the conditional to an overriding method in a subclass. Make the original method abstract.

```
double getSpeed() {
switch (_type) {
         case EUROPEAN:
         return getBaseSpeed();
         case AFRICAN:
         return getBaseSpeed() - getLoadFactor() * _numberOfCoconuts;
         case NORWEGIAN_BLUE:
         return (_isNailed) ? 0 : getBaseSpeed(_voltage);
}
throw new RuntimeException ("Should be unreachable");
}
```



Replace Conditional with Visitor by Ivan Mitrovic

You have an "aggressive" Conditional that chooses different behaviour depending on the type of an object and repeats itself in a large number throughout the code.

Create concrete instance of Visitable object for each data type in Conditional. Create concrete instance of Visitor that encapsulates logic of each Conditional. Visit Visitable by Visitor.

Replace Constructor with Factory Method

You want to do more than simple construction when you create an object.

Replace the constructor with a factory method.

```
Employee (int type) {
```

```
            _type = type;
}
```

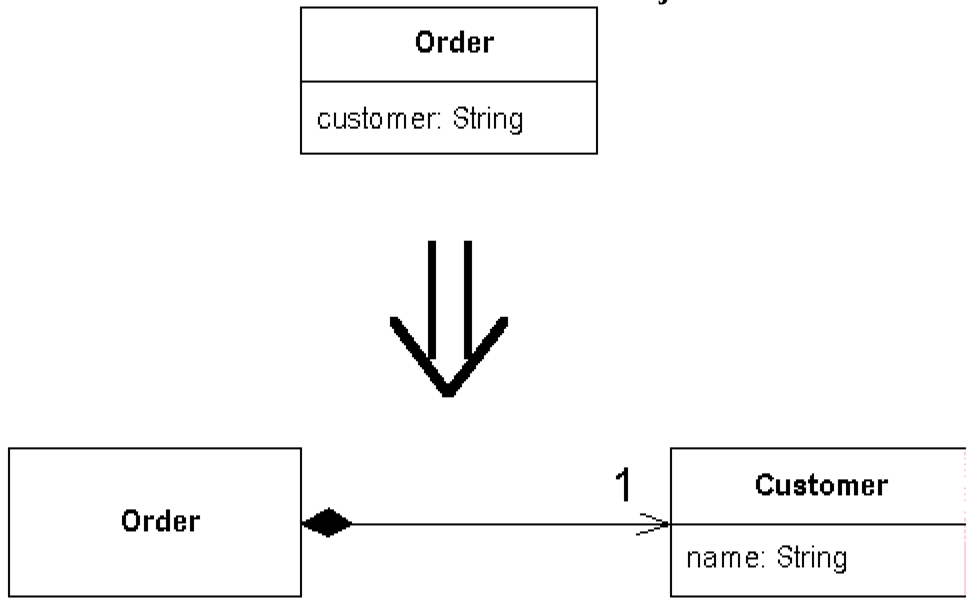
⇓

```
static Employee create(int type) {
            return new Employee(type);
}
```

Replace Data Value with Object

You have a data item that needs additional data or behavior.

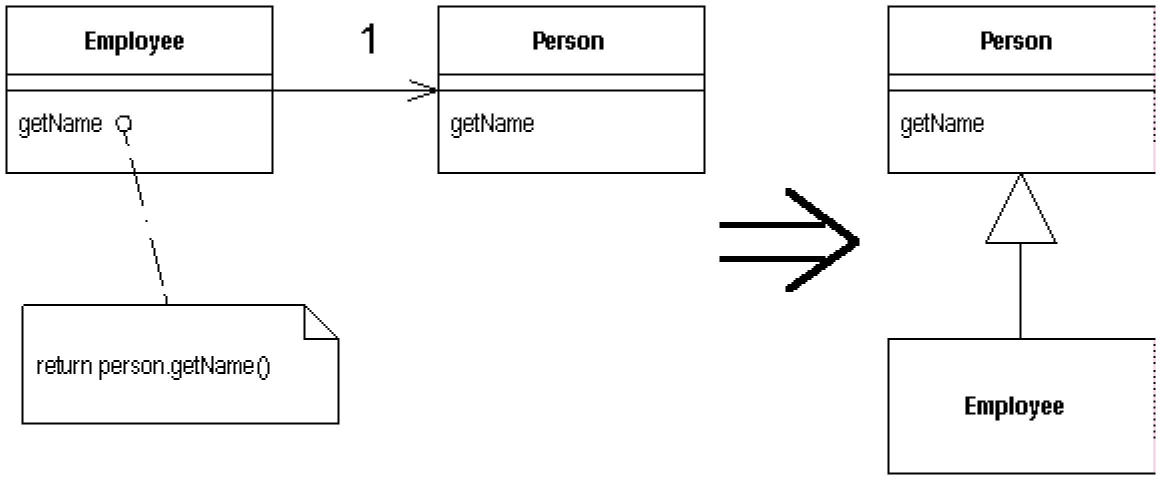
Turn the data item into an object.



Replace Delegation with Inheritance

You're using delegation and are often writing many simple delegations for the entire interface.

Make the delegating class a subclass of the delegate.

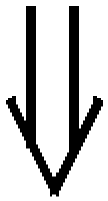


Replace Error Code with Exception

A method returns a special code to indicate an error.

Throw an exception instead.

```
int withdraw(int amount) {
            if (amount > _balance)
                  return -1;
            else {
                  _balance -= amount;
                  return 0;
            }
}
```



```
void withdraw(int amount) throws BalanceException {
            if (amount > _balance) throw new
            BalanceException();
            _balance -= amount;
}
```

Replace Exception with Test

You are throwing an exception on a condition the caller could have checked first.

Change the caller to make the test first.

```
double getValueForPeriod (int periodNumber) {
            try {
```

```
                        return _values[periodNumber];
                } catch (ArrayIndexOutOfBoundsException e) {
                        return 0;
                }
        }
```

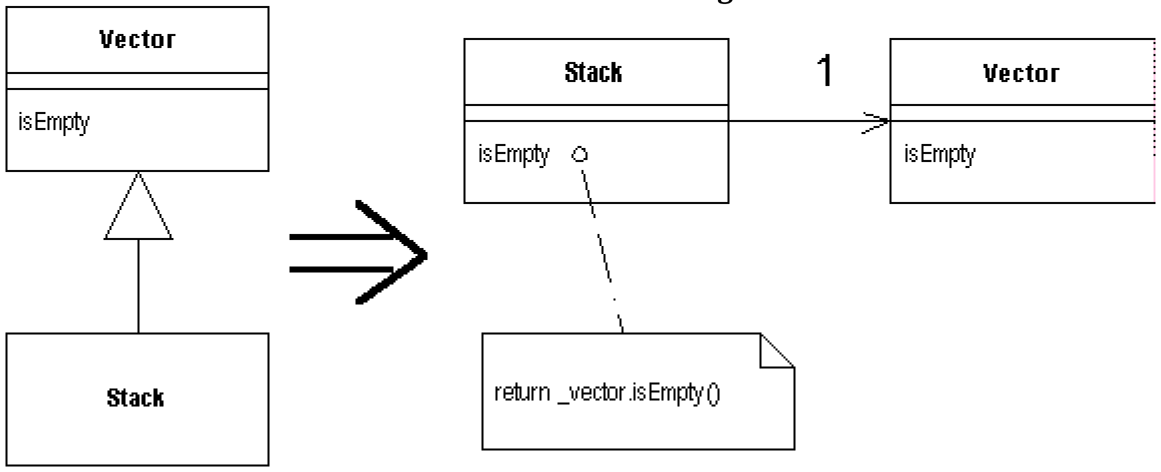
$$\Downarrow$$

```
        double getValueForPeriod (int periodNumber) {
                if (periodNumber >= _values.length) return 0;
                return _values[periodNumber];
        }
```

Replace Inheritance with Delegation

A subclass uses only part of a superclasses interface or does not want to inherit data.

Create a field for the superclass, adjust methods to delegate to the superclass, and remove the subclassing.



Replace Iteration with Recursion by Dave Whipp

You have a loop, and it is not obvious what each iteration is doing

Replace Iteration with Recursion

```
unsigned greatest_common_divisor (unsigned a, unsigned b)
{
while (a != b)
{
if (a > b)
{
         a -= b;
}
else if (b > a)
{
         b -= a;
}
}
}
```

$$\Downarrow$$

```
unsigned greatest_common_divisor (unsigned a, unsigned b)
{
if (a > b)
{
return greatest_common_divisor ( a-b, b );
}
else if (b > a)
{
return greatest_common_divisor ( a, b-a );
}
else // (a == b)
{
return a;
}
}
```

Replace Magic Number with Symbolic Constant

You have a literal number with a particular meaning.

Create a constant, name it after the meaning, and replace the number with it.

```
         double potentialEnergy(double mass, double height) {
                    return mass * height * 9.81;
         }
```

$$\Downarrow$$

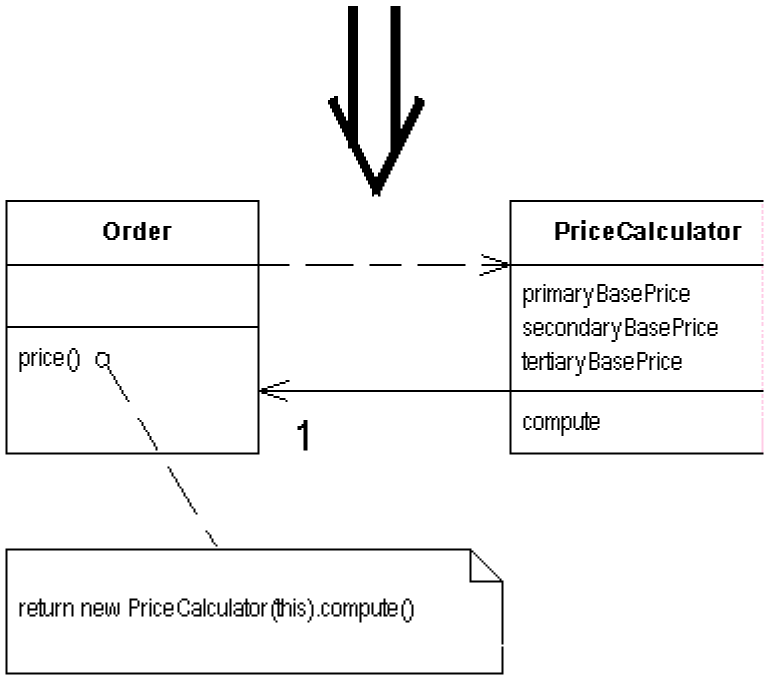
```
         double potentialEnergy(double mass, double height) {
                return mass * GRAVITATIONAL_CONSTANT * height;
        }
        static final double GRAVITATIONAL_CONSTANT = 9.81;
```

Replace Method with Method Object

You have a long method that uses local variables in such a way that you cannot apply Extract Method

Turn the method into its own object so that all the local variables become fields on that object. You can then decompose the method into other methods on the same object.

```
class Order...
        double price() {
                double primaryBasePrice;
                double secondaryBasePrice;
                double tertiaryBasePrice;
                // long computation;
                ...
        }
```



Replace Nested Conditional with Guard Clauses

A method has conditional behavior that does not make clear what the normal path of execution is

Use Guard Clauses for all the special cases

```
double getPayAmount() {
        double result;
        if (_isDead) result = deadAmount();
        else {
                if (_isSeparated) result = separatedAmount();
                else {
                    if (_isRetired) result = retiredAmount();
                    else result = normalPayAmount();
                };
        }
return result;
};
```

⇓

```
double getPayAmount() {
        if (_isDead) return deadAmount();
        if (_isSeparated) return separatedAmount();
        if (_isRetired) return retiredAmount();
        return normalPayAmount();
};
```

Replace Parameter with Explicit Methods

You have a method that runs different code depending on the values of an enumerated parameter.

Create a separate method for each value of the parameter.

```
        void setValue (String name, int value) {
                if (name.equals("height")) {
                    _height = value;
                    return;
                }
                if (name.equals("width")) {
                    _width = value;
                    return;
                }
                Assert.shouldNeverReachHere();
        }
```

⇓

```
        void setHeight(int arg) {
                _height = arg;
        }
        void setWidth (int arg) {
                _width = arg;
        }
```

Replace Parameter with Method

An object invokes a method, then passes the result as a parameter for a method. The receiver can also invoke this method.

Remove the parameter and let the receiver invoke the method.

```
int basePrice = _quantity * _itemPrice;
discountLevel = getDiscountLevel();
double finalPrice = discountedPrice (basePrice,
discountLevel);
```

$\Downarrow$

```
int basePrice = _quantity * _itemPrice;
double finalPrice = discountedPrice (basePrice);
```

Replace Record with Data Class

You need to interface with a record structure in a traditional programming environment.

Make a dumb data object for the record.

Replace Recursion with Iteration by Ivan Mitrovic

You have code that uses Recursion and is hard to understand.

Replace Recursion with Iteration.

```
public void countDown(int n) {
         if(n == 0) return;

         System.out.println(n + "...");
         waitASecond();
         countDown(n-1);
}
```

$\Downarrow$

After Refactoring:

```
public void countDown(int n) {
         while(n > 0) {
                     System.out.println(n + "...");
                     waitASecond ();
                     n -= 1;
         }
```

Replace Static Variable with Parameter by Marian Vittek

A function depending on a static variable needs to be reused in more general context.
Add a new parameter to the function and replace all references of the static variable within the function by this new parameter.

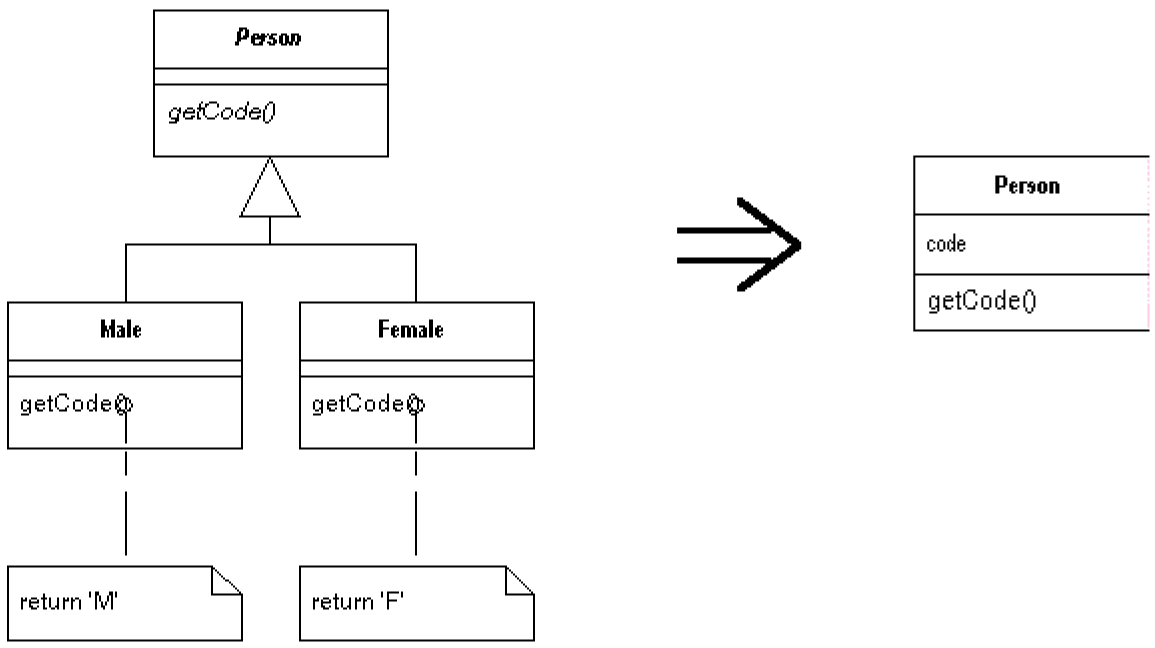
```
void printValues() {
         for (int i = 0; i < people.length; i++) {
                     System.out.println(people[i].name+" has salary
                     "+people[i].salary);
         }
}

public static void main(String args[]) {
         ...
         printValues();
}
```

$$\Downarrow$$

```
void printValues(PrintStream outfile) {
         for (int i = 0; i < people.length; i++) {
                     outfile.println(people[i].name+" has salary
                     "+people[i].salary);
         }
}

public static void main(String args[]) {
         ...
         printValues(System.out);
}
```

Replace Subclass with Fields

You have subclasses that vary only in methods that return constant data.
Change the methods to superclass fields and eliminate the subclasses.



Replace Temp with Query

ou are using a temporary variable to hold the result of an expression.

Extract the expression into a method. Replace all references to the temp with the expression. The new method can then be used in other methods.

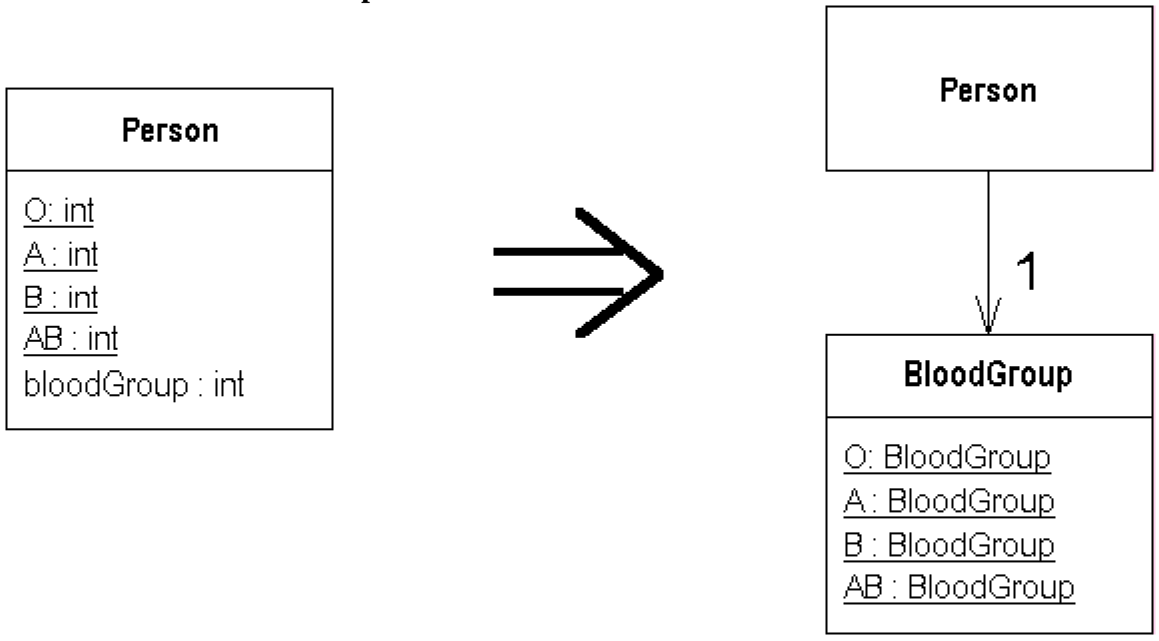
```
                        double basePrice = _quantity * _itemPrice;
                        if (basePrice > 1000)
                              return basePrice * 0.95;
                        else
                              return basePrice * 0.98;
```

$$\Downarrow$$

```
                        if (basePrice() > 1000)
                              return basePrice() * 0.95;
                        else
                              return basePrice() * 0.98;
...
            double basePrice() {
                        return _quantity * _itemPrice;
            }
```

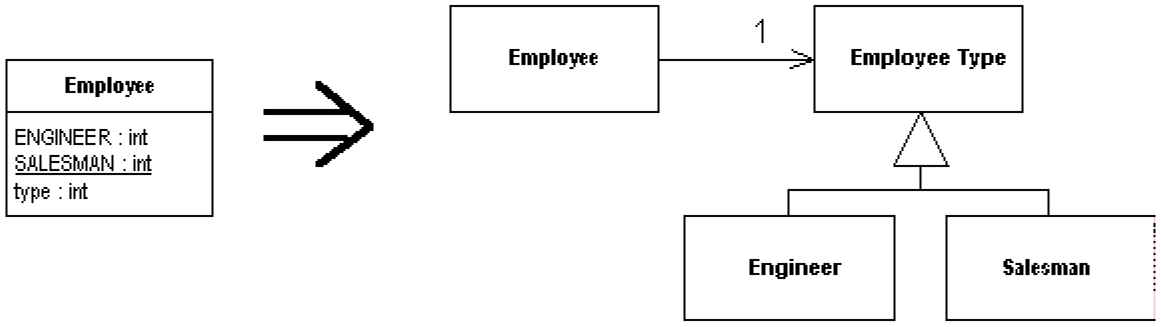
Replace Type Code with Class

A class has a numeric type code that does not affect its behavior.
Replace the number with a new class.



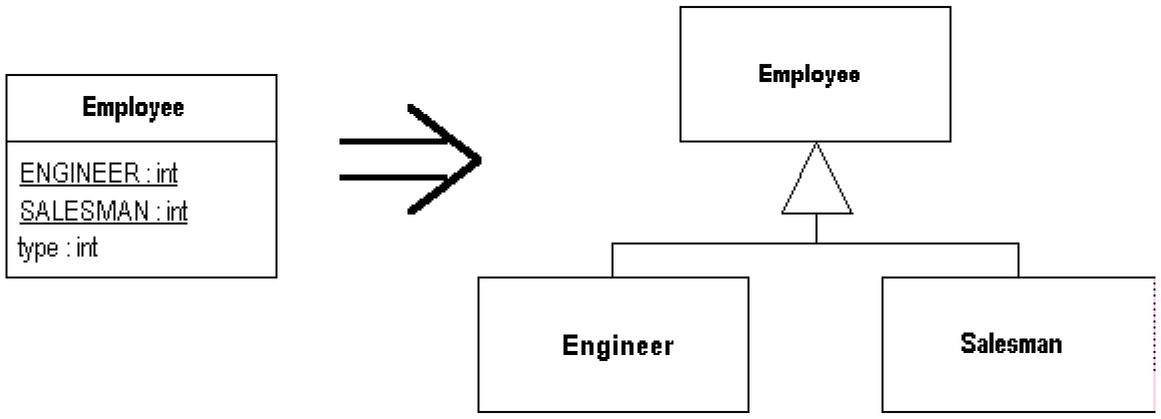
Replace Type Code with State/Strategy

You have a type code that affects the behavior of a class, but you cannot use subclassing.
Replace the type code with a state object.



Replace Type Code with Subclasses

You have an immutable type code that affects the behavior of a class.
Replace the type code with subclasses.



Reverse Conditional by Bill Murphy and Martin Fowler

You have a conditional that would be easier to understand if you reversed its sense.

Reverse the sense of the conditional and reorder the conditional's clauses.

```
if ( !isSummer( date ) )
charge = winterCharge( quantity );
else
charge = summerCharge( quantity );
```

$$\Downarrow$$

```
if ( isSummer( date ) )
charge = summerCharge( quantity );
else
charge = winterCharge( quantity );
```

Self Encapsulate Field

You are accessing a field directly, but the coupling to the field is becoming awkward.

Create getting and setting methods for the field and use only those to access the field.

```
        private int _low, _high;
        boolean includes (int arg) {
                    return arg >= _low && arg <= _high;
        }
```

$$\Downarrow$$

```
         private int _low, _high;
         boolean includes (int arg) {
                     return arg >= getLow() && arg <= getHigh();
         }
         int getLow() {return _low;}
         int getHigh() {return _high;}
```

Separate Data Access Code (Link Only)

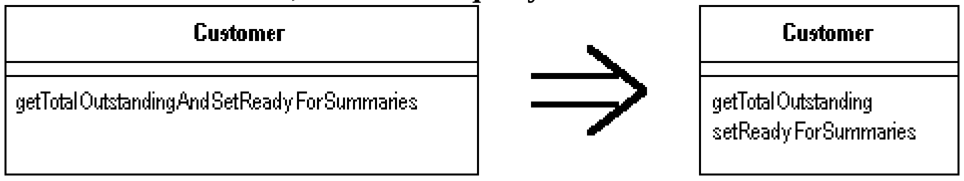
Data access code is embedded directly within a class that has other unrelated responsibilities

Extract the data access code into a new class and move the new class logically and/or physically closer to the data source

Separate Query from Modifier

You have a method that returns a value but also changes the state of an object.

Create two methods, one for the query and one for the modification.



Split Loop by Martin Fowler

You have a loop that is doing two things

Duplicate the loop

```
void printValues() {
         double averageAge = 0;
         double totalSalary = 0;
         for (int i = 0; i < people.length; i++) {
                          averageAge += people[i].age;
                          totalSalary += people[i].salary;
         }
         averageAge = averageAge / people.length;
         System.out.println(averageAge);
         System.out.println(totalSalary);
}
```

$\Downarrow$

```
void printValues() {
         double totalSalary = 0;
         for (int i = 0; i < people.length; i++) {
                           totalSalary += people[i].salary;
         }

         double averageAge = 0;
         for (int i = 0; i < people.length; i++) {
                           averageAge += people[i].age;
         }
         averageAge = averageAge / people.length;

         System.out.println(averageAge);
         System.out.println(totalSalary);
}
```

Split Temporary Variable

You have a temporary variable assigned to more than once, but is not a loop variable nor a collecting temporary variable.

Make a separate temporary variable for each assignment.

```
double temp = 2 * (_height + _width);
System.out.println (temp);
temp = _height * _width;
System.out.println (temp);
```

$$\Downarrow$$

```
final double perimeter = 2 * (_height + _width);
System.out.println (perimeter);
final double area = _height * _width;
System.out.println (area);
```

Substitute Algorithm

You want to replace an algorithm with one that is clearer.

Replace the body of the method with the new algorithm.

```
String foundPerson(String[] people){
            for (int i = 0; i < people.length; i++) {
                  if (people[i].equals ("Don")){
                        return "Don";
                  }
                  if (people[i].equals ("John")){
                        return "John";
                  }
                  if (people[i].equals ("Kent")){
                        return "Kent";
                  }
            }
            return "";
}
```

⇓

```
String foundPerson(String[] people){
            List candidates = Arrays.asList(new String[]
            {"Don", "John", "Kent"});
            for (int i=0; i<people.length; i++)
                  if (candidates.contains(people[i]))
                        return people[i];
            return "";
}
```

Use a Connection Pool

Database connections are not shared. Instead, clients manage their own connections for making database invocations
Use a connection pool to pre-initialize multiple connections, improving scalability and performance

Wrap entities with session

Entity beans from the business tier are exposed to clients in another tier
Use a Session Facade to encapsulate the entity beans