CSE 470 – Refactoring Code Smells

BRAC UNIVERSITY

Python Example Sheet: [Code Smells (SHRR Version).ipynb]

What is Refactoring?

❑ A series of **small steps**, each of which changes the program's **internal structure** without changing its **external behavior** - Martin Fowler

Verify no change in external behavior by -

- Testing
- Using the right tool - IDE
- Formal code analysis by any tool or team
- Being very, very careful

What if you hear...

- We’ll just refactor the code to support logging
- Can you refactor the code so that it authenticates against LDAP instead of Database?
- We have too much duplicate code, we need to refactor the code to eliminate duplication
- This class is too big, we need to refactor it
- Caching?

Why do we Refactor?

- ✓ Helps us deliver **more business value faster**
- ✓ Improves the **design** of our software
- ✓ Minimizes **technical debt**
- ✓ Keep **development** at *speed*
- ✓ To make the software easier to **understand**
- ✓ Write for people, not the compiler
- ✓ To help find **bugs**

BRAC UNIVERSITY
Inspiring Excellence

Readability

Which code segment is easier to read?

Sample 1

```
if (date.Before(Summer_Start) || date.After(Summer_End)){
        charge = quantity * winterRate + winterServiceCharge;
else charge = quantity * summerRate;

}
```

Sample 2

```
if (IsSummer(date)) {
        charge = SummerCharge(quantity);
else
        charge =
}
        WinterCharge(quantity);
```

Inspiring Excellence

When should you refactor?

- To add **new functionality**
  - refactor existing code until you understand it
  - refactor the design to make it simple to add
- To find **bugs**
  - refactor to understand the code
- For **code reviews**
  - immediate effect of code review
  - allows for higher level suggestions

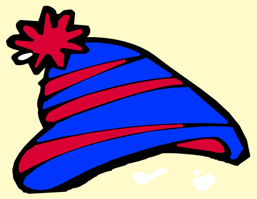
Like championship snooker players we are setting ourselves up for our next shot

The Two Hats

Adding Function

- ☑ Add new capabilities to the system
- ☑ Adds new tests
- ☑ Get the test working

Refactoring



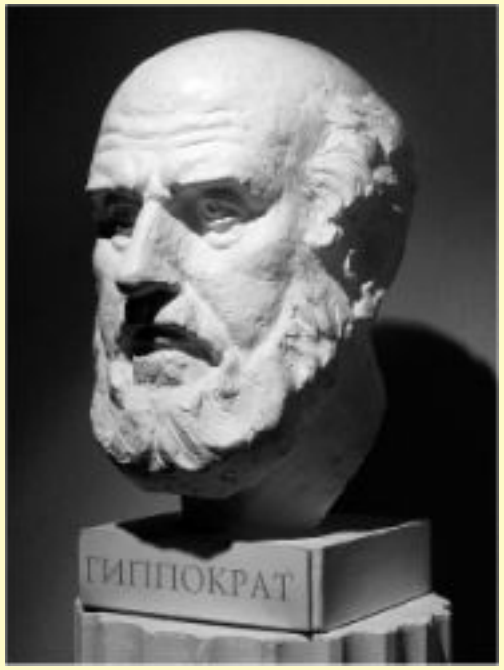
- ☑ Does not add any new features
- ☑ Does not add tests (but may change some)
- ☑ Restructure the code to remove redundancy

How do we Refactor?

- We look for **Code-Smells**
- Things that we suspect are not quite right or will cause us severe pain if we do not fix

2 Piece of Advice before Refactoring

Baby Steps



The Hippocratic Oath

First Do No Harm!

Code Smells?

Code Smells identify *frequently* occurring **design problems** in a way that is more *specific or targeted* than general design guidelines (like “loosely coupled code” or “duplication-free code”). - Joshua K

A code smell is a design that duplicates, complicates, bloats or tightly couples code.

A short history of Code Smells

- If it stinks, change it!
- Kent Beck coined the term code smell to signify something in code that needed to be changed.

Common Code Smells

- ✓ Inappropriate Naming
- ✓ Comments
- ✓ Dead Code
- ✓ Duplicated code
- ✓ Primitive Obsession
- ✓ Large Class
- ✓ Lazy Class
- ✓ Alternative Class with Different Interface
- ✓ Long Method
- ✓ Long Parameter List
- ✓ Switch Statements
- ✓ Speculative Generality
- ✓ Oddball Solution
- ✓ Feature Envy
- ✓ Refused Bequest
- ✓ Black Sheep
- ✓ Train Wreck

BRAC UNIVERSITY
Inspiring Excellence

Code Smell - Inappropriate Naming

- Names given to variables (fields) and methods should be clear and meaningful.
- A variable name should say exactly what it is.
  - Which is better?
    - private string s; OR **private string salary;**
- A method should say exactly what it does.
  - Which is better?
    - public double calc (double s)
    - **public double calculateFederalTaxes (double salary)**

BRAC UNIVERSITY
Inspiring Excellence

Code Smell - Comments

- Comments are often used as deodorant
- Comments represent a *failure to express an idea in the code*. Try to make your code self-documenting or intention-revealing
- When you feel like writing a comment, first try "to refactor so that the comment becomes superfluous
- **Remedies:**
  - Extract Method
  - Rename Method
  - Introduce Assertion

Comment: “Grow the Array” smells

```
public class MyList
  {
    int INITIAL_CAPACITY = 10;
    bool m_readOnly;
    int m_size = 0;
    int m_capacity;
    string[] m_elements;

    public MyList()
    {
      m_elements = new string[INITIAL_CAPACITY];
      m_capacity = INITIAL_CAPACITY;
    }

    int GetCapacity() {
      return m_capacity;
    }
}
```

```
void AddToList(string element)
{
  if (!m_readOnly)
  {
    int newSize = m_size + 1;
    if (newSize > GetCapacity())
    {
      // grow the array
      m_capacity += INITIAL_CAPACITY;
      string[] elements2 = new string[m_capacity];
      for (int i = 0; i < m_size; i++)
        elements2[i] = m_elements[i];

      m_elements = elements2;
    }
    m_elements[m_size++] = element;
  }
}
```

BRAC UNIVERSITY
Inspiring Excellence

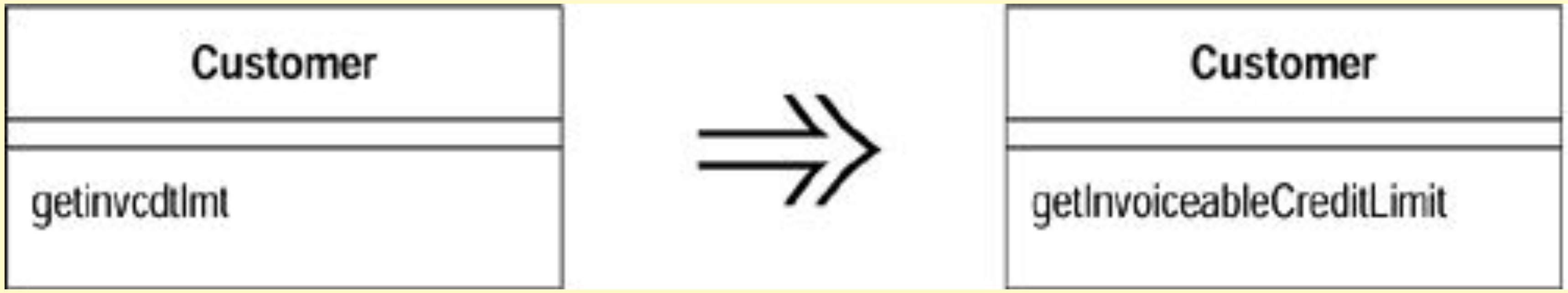
Comment Smells Make-over

```
void AddToList(string element)
{
   if (m_readOnly)
      return;
   if (ShouldGrow())
   {
      Grow();
   }
   StoreElement(element);
}
```

```
private void Grow()
{
   m_capacity += INITIAL_CAPACITY;
   string[] elements2 = new string[m_capacity];
   for (int i = 0; i < m_size; i++)
      elements2[i] = m_elements[i];

   m_elements = elements2;
}

private void StoreElement(string element)
{
   m_elements[m_size++] = element;
}
```

```
private bool ShouldGrow()
{
   return (m_size + 1) > GetCapacity();
}
```

Rename Method



Extract Method

```
void PrintOwning(double amount){
    PrintBanner();

    // print details
    System.Console.Out.WriteLine(“name: “+ name);
    System.Console.Out.WriteLine(“amount: “+ amount);
}
```

Extract Method

```
void PrintOwning(double amount){
    PrintBanner();

    // print details
    System.Console.Out.WriteLine("name: "+ name);
    System.Console.Out.WriteLine("amount: "+ amount);
}
```

```
void PrintOwning(double amount){
    PrintBanner();
    PrintDetails(amount);
}

void PrintDetails(double amount){
    System.Console.Out.WriteLine("name: "+ name);
    System.Console.Out.WriteLine("amount: "+ amount);
}
```

BRAC UNIVERSITY

Inspiring Excellence

Introduce Assertion

Introduce Assertion

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

Introduce Assertion

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

```
double getExpenseLimit() {
    Assert(_expenseLimit != NULL_EXPENSE || _primaryProject != null,
    “Both Expense Limit and Primary Project must not be null”);

    return (_expenseLimit != NULL_EXPENSE) ? _expenseLimit :
        _primaryProject.GetMemberExpenseLimit();
}
```

https://www.w3schools.com/python/ref_keyword_assert.asp

Code Smell - Long Method

- A method is long when it is too hard to quickly comprehend.
- Long methods tend to hide behavior that ought to be shared, which leads to duplicated code in other methods or classes.
- Good Object Oriented code is easiest to understand and maintain with shorter methods
- with good names

Remedies:

- Extract Method
- Replace Temp with Query
- Introduce Parameter Object
- Preserve Whole Object
- Decompose Conditional

BRAC UNIVERSITY
Inspiring Excellence

Long Method Example

```
private String toStringHelper(StringBuffer result)
{
   result.append("<");
   result.append(name);
   result.append(attributes.toString());
   result.append(">");
   if (!value.equals(""))
      result.append(value);
   Iterator it = children().iterator();
   while (it.hasNext())
   {
      TagNode node = (TagNode)it.next();
      node.toStringHelper(result);
   }
   result.append("</");
   result.append(name);
   result.append(">");
   return result.toString();
}
```

Example Html tag:
<name> Jannet Jhonson </name>

Inspiring Excellence

Long Method Makeover (Extract Method)

```
private String toStringHelper(StringBuffer result)
{
   writeOpenTagTo(result);
   writeValueTo(result);
   writeChildrenTo(result);
   writeEndTagTo(result);
   return result.toString();
}

private void writeOpenTagTo(StringBuffer result)
{
   result.append("<");
   result.append(name);
   result.append(attributes.toString());
   result.append(">");
}
private void writeEndTagTo(StringBuffer result)
{
   result.append("</");
   result.append(name);
   result.append(">");
}
```

```
private void writeValueTo(StringBuffer result)
{
   if (!value.equals(""))
      result.append(value);
}

private void writeChildrenTo(StringBuffer result)
{
   Iterator it = children().iterator();
   while (it.hasNext())
   {
      TagNode node = (TagNode)it.next();
      node.toStringHelper(result);
   }
}
```

BRAC UNIVERSITY
Inspiring Excellence

Replace Temp with Query

```
Method1(){
  double basePrice = _quanity * _itemPrice;
if(basePrice > 1000) {
   return basePrice * 0.95
  }
else{
   return basePrice*0.98
  }
}
 Method2(){
  double basePrice = _quanity * _itemPrice;
  return basePrice + 100;
 }
```

What if the basePrice calculation equation changes ??

-- We would need to change two lines in the code

BRAC UNIVERSITY
Inspiring Excellence

Replace Temp with Query

```
Method1(){
  double basePrice = _quanity * _itemPrice;
if(basePrice > 1000) {
   return basePrice * 0.95
  }
else{
   return basePrice*0.98
  }
}
```

```
Method1(){
if(getBasePrice() > 1000) {
     return getBasePrice() * 0.95;
}
else {
       return getBasePrice() * 0.98;
  }

}
double getBasePrice()  {
       return _quanitiy * itemPrice;
}
```

Replace Temp with Query

```
Method2(){
 double basePrice = _quanity * _itemPrice;
 return basePrice + 100;
}
```

double getBasePrice() {

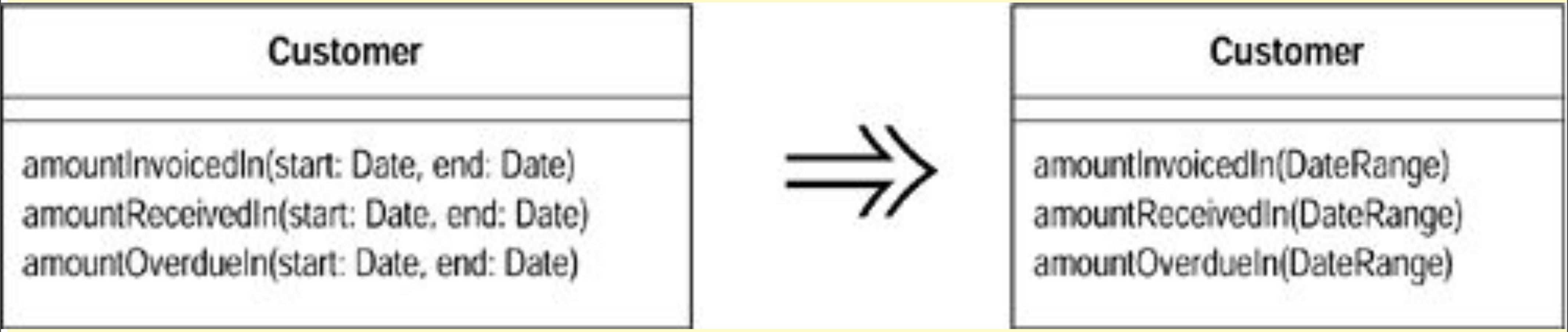
```
double getBasePrice()  {
      return _quanitiy * itemPrice;
}
```

```
Method2(){
 return getBasePrice() + 100;
}
```

Introduce Parameter Object

```
int MethodTooManyParameter (Date start, Date end, int value, string
                            month, string yearStart, string yearEnd)
{
        // method body
}
```

Introduce Parameter Object


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

```
Class DateRange{
    Date start;
    Date end;
}
```

Preserve Whole Object

```
int low = daysTempRange().getLow();

int high = daysTempRange().getHigh();

withinPlan = plan.withinRange(low, high);
```

```
daysTempRange(){
  return someObject;
}
```

Preserve Whole Object

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

$\Downarrow$

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

Decompose Conditional

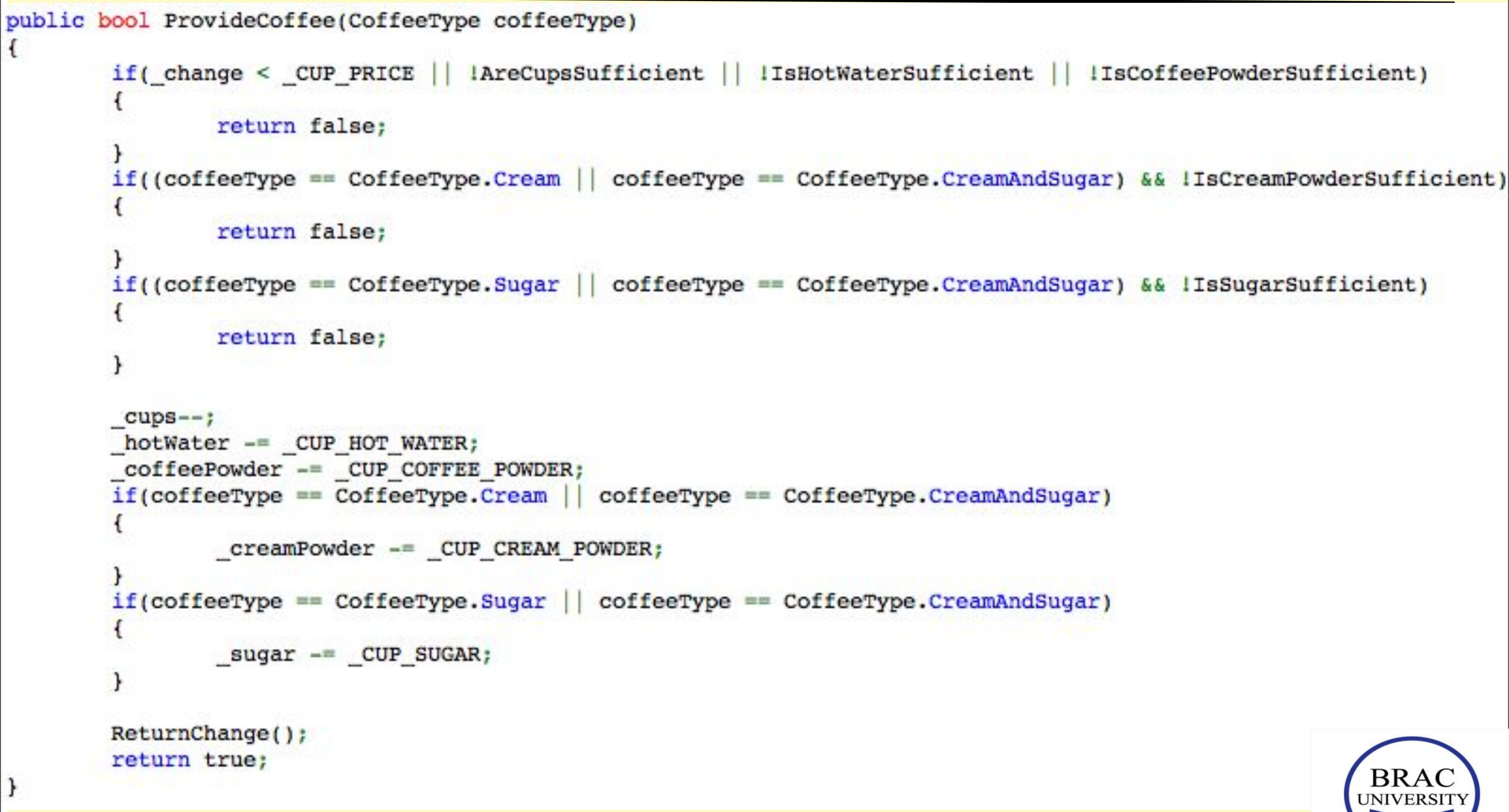
You have a complicated conditional (if-then-else) statement.

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);
```

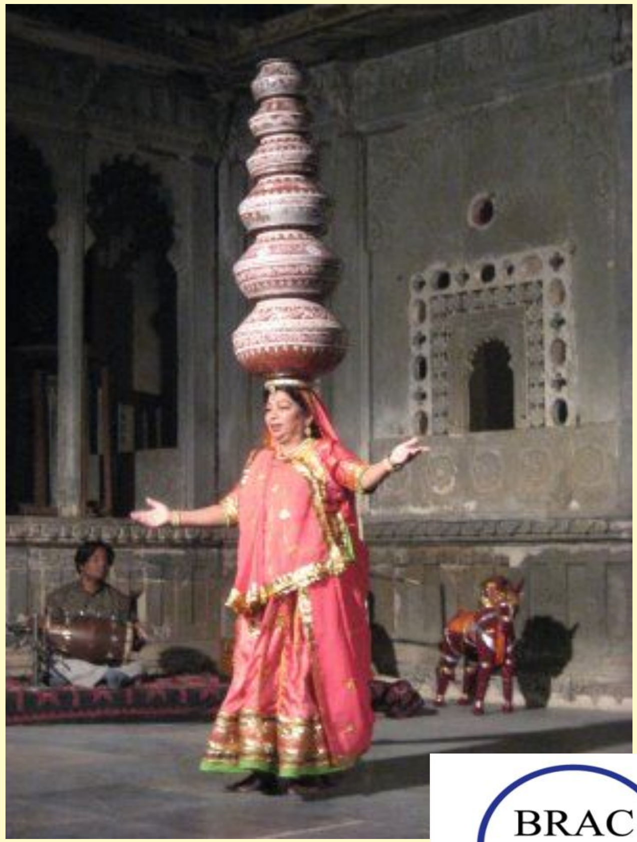
Example of Conditional Complexity



```
public bool ProvideCoffee(CoffeeType coffeeType)
{
        if(_change < _CUP_PRICE || !AreCupsSufficient || !IsHotWaterSufficient || !IsCoffeePowderSufficient)
        {
                return false;
        }
        if((coffeeType == CoffeeType.Cream || coffeeType == CoffeeType.CreamAndSugar) && !IsCreamPowderSufficient)
        {
                return false;
        }
        if((coffeeType == CoffeeType.Sugar || coffeeType == CoffeeType.CreamAndSugar) && !IsSugarSufficient)
        {
                return false;
        }

        _cups--;
        _hotWater -= _CUP_HOT_WATER;
        _coffeePowder -= _CUP_COFFEE_POWDER;
        if(coffeeType == CoffeeType.Cream || coffeeType == CoffeeType.CreamAndSugar)
        {
                _creamPowder -= _CUP_CREAM_POWDER;
        }
        if(coffeeType == CoffeeType.Sugar || coffeeType == CoffeeType.CreamAndSugar)
        {
                _sugar -= _CUP_SUGAR;
        }

        ReturnChange();
        return true;
}
```

Code Smell- Long Parameter List

- Methods that take too many parameters produce client code that is awkward and difficult to work with.
- **Remedies:**
  - Introduce Parameter Object
  - Replace Parameter with Method
  - Preserve Whole Object



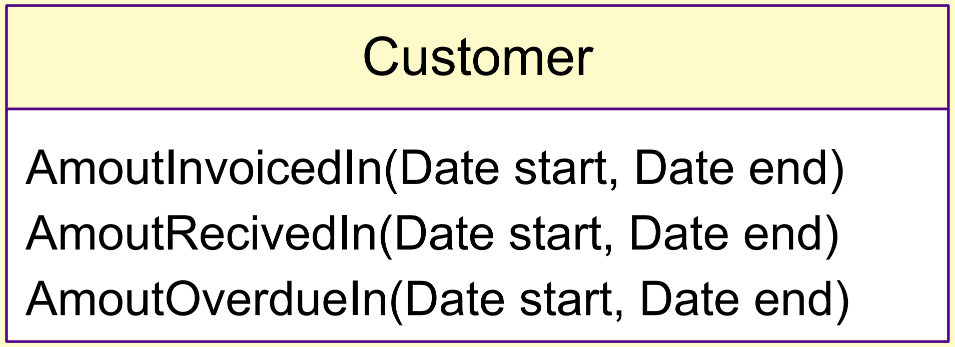
BRAC UNIVERSITY
Inspiring Excellence

Example

```
private void createUserInGroup() {
    GroupManager groupManager = new GroupManager();
    Group group = groupManager.create(TEST_GROUP, false,
      GroupProfile.UNLIMITED_LICENSES, "",
                    GroupProfile.ONE_YEAR, null);
    user = userManager.create(USER_NAME, group, USER_NAME, "jack",
              USER_NAME, LANGUAGE, false, false, new Date(),
                    "blah", new Date());
}
```

BRAC UNIVERSITY
Inspiring Excellence

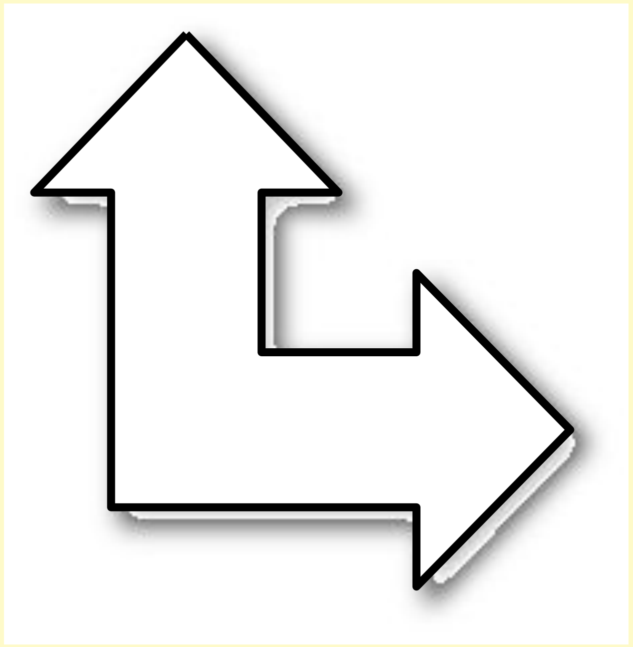
Introduce Parameter Object


Customer
AmoutInvoicedIn(Date start, Date end)
AmoutRecivedIn(Date start, Date end)
AmoutOverdueIn(Date start, Date end)

Introduce Parameter Object

Customer

AmoutInvoicedIn(Date start, Date end)
AmoutRecivedIn(Date start, Date end)
AmoutOverdueIn(Date start, Date end)



Customer

AmoutInvoicedIn(DateRange range)
AmoutRecivedIn(DateRange range)
AmoutOverdueIn(DateRange ran

Replace Parameter with Method

```
public double getPrice() {
   int basePrice = _quantity * _itemPrice;
   int discountLevel;
   if (_quantity > 100)
     discountLevel = 2;
   else
     discountLevel = 1;
   double finalPrice = discountedPrice (basePrice, discountLevel);
   return finalPrice;
}

private double discountedPrice (int basePrice, int discountLevel) {
   if (discountLevel == 2)
     return basePrice * 0.1;
   else
     return basePrice * 0.05;
}
```

Replace Parameter with Method

```
public double getPrice() {
   int basePrice = _quantity * _itemPrice;
   int discountLevel = getDiscountLevel();
   double finalPrice = discountedPrice (basePrice, discountLevel);
   return finalPrice;
}

private int getDiscountLevel() {
   if (_quantity > 100) return 2;
   else return 1;
}
private double discountedPrice (int basePrice, int discountLevel) {
   if (getDiscountLevel() == 2) return basePrice * 0.1;
   else return basePrice * 0.05;
}
```

Replace Parameter with Method

```
public double getPrice() {
    int basePrice = _quantity * _itemPrice;
    int discountLevel = getDiscountLevel();
    double finalPrice = discountedPrice (basePrice);
    return finalPrice;
}

private double discountedPrice (int basePrice) {
    if (getDiscountLevel() == 2) return basePrice * 0.1;
    else return basePrice * 0.05;
}
```

Preserve Whole Object

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

Preserve Whole Object

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

⇩

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

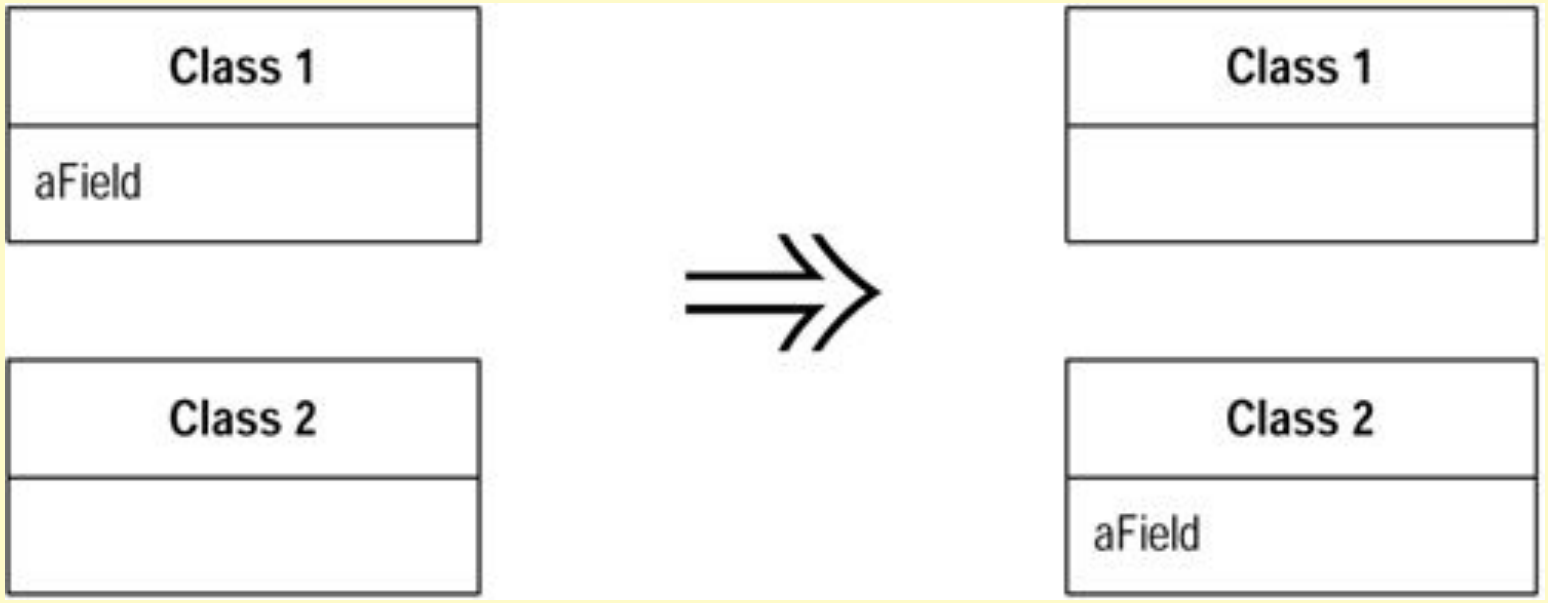
Feature Envy

- A method that seems more interested in some other class than the one it is in.
- Data and behavior that acts on that data belong together. When a method makes too many calls to other classes to obtain data or functionality, Feature Envy is in the air.
- **Remedies:**
  - Move Field
  - Move Method
  - Extract Method

Example

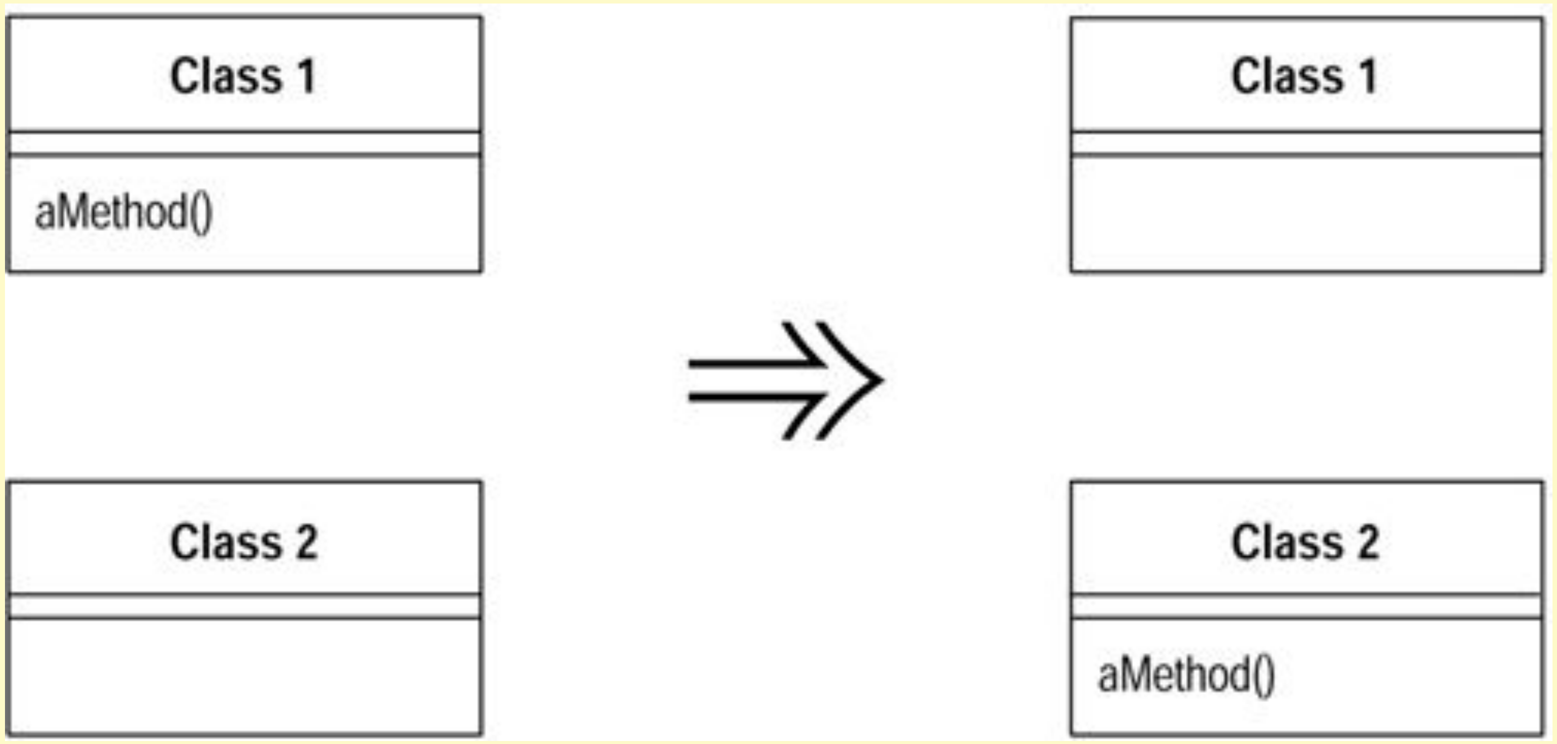
```
Public class CapitalStrategy{
    double capital(Loan loan)
    {
      if (loan.getExpiry() == NO_DATE && loan.getMaturity() != NO_DATE)
        return loan.getCommitmentAmount() * loan.duration() * loan.riskFactor();

      if (loan.getExpiry() != NO_DATE && loan.getMaturity() == NO_DATE)
      {
        if (loan.getUnusedPercentage() != 1.0)
          return loan.getCommitmentAmount() * loan.getUnusedPercentage() *
    loan.duration() * loan.riskFactor();
        else
          return (loan.outstandingRiskAmount() * loan.duration() * loan.riskFactor()) +
            (loan.unusedRiskAmount() * loan.duration() * loan.unusedRiskFactor());
      }

      return 0.0;
    }
}
```

Move Field



Inspiring Excellence

Move Method

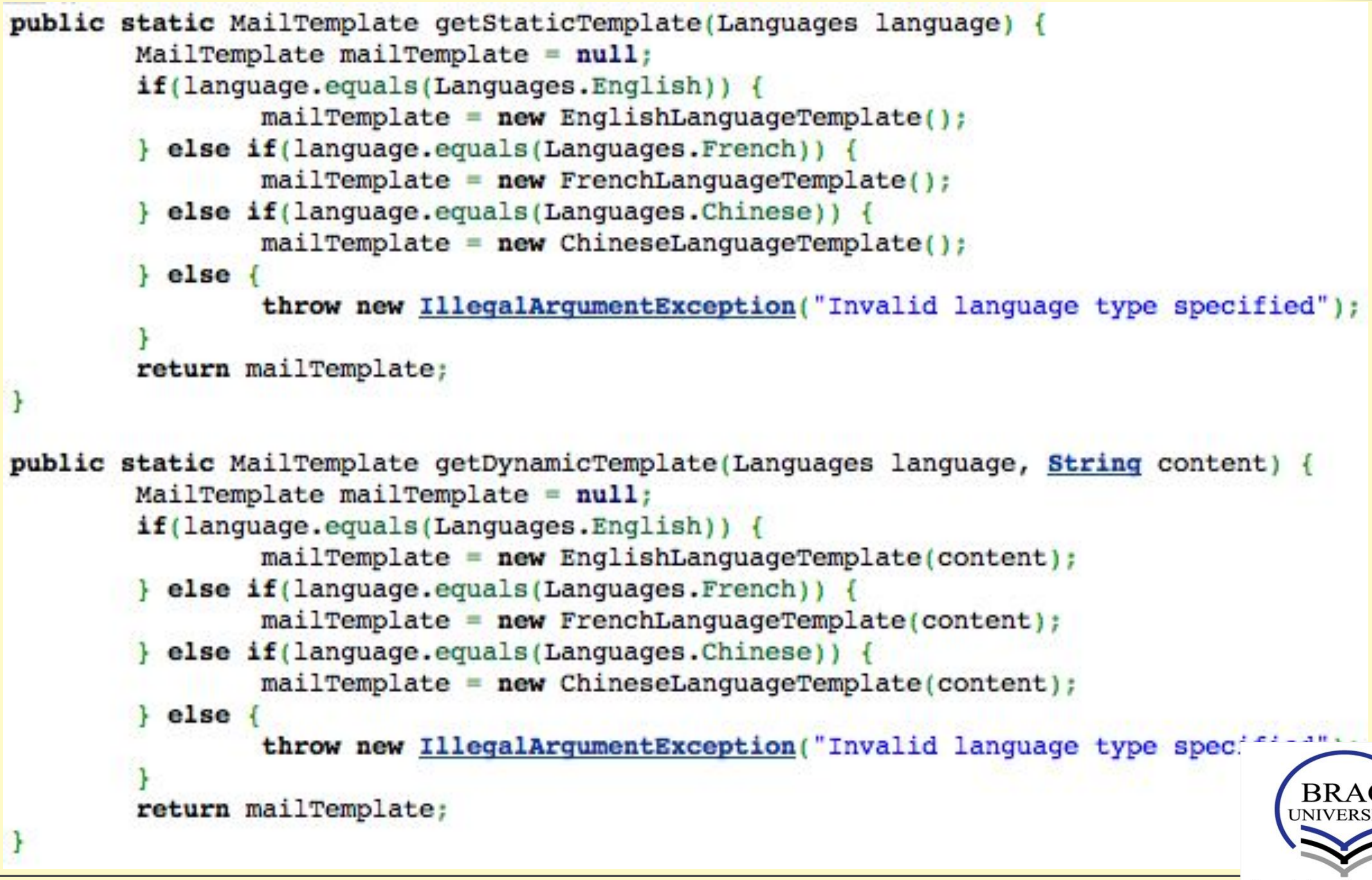


Duplicated Code

- The *most pervasive and pungent smell* in software
- There is obvious or blatant duplication
  - Such as copy and paste
- There are subtle or non-obvious duplications
  - Similar algorithms
- **Remedies**
  - Extract Method
  - Pull Up Field
  - Form Template Method
  - Substitute Algorithm

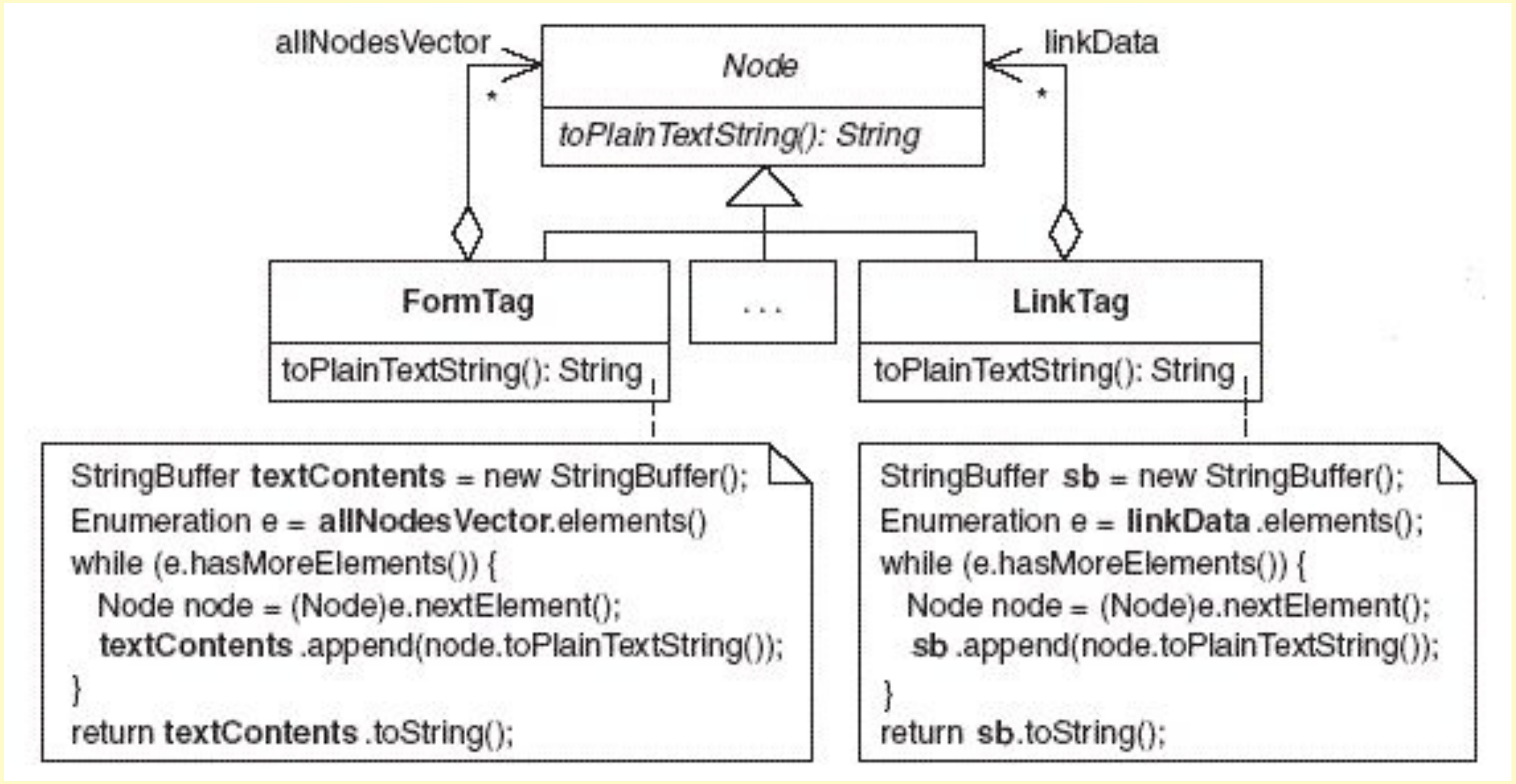
BRAC UNIVERSITY
Inspiring Excellence

Ctl+C Ctl+V Pattern



```
public static MailTemplate getStaticTemplate(Languages language) {
        MailTemplate mailTemplate = null;
        if(language.equals(Languages.English)) {
                mailTemplate = new EnglishLanguageTemplate();
        } else if(language.equals(Languages.French)) {
                mailTemplate = new FrenchLanguageTemplate();
        } else if(language.equals(Languages.Chinese)) {
                mailTemplate = new ChineseLanguageTemplate();
        } else {
                throw new IllegalArgumentException("Invalid language type specified");
        }
        return mailTemplate;
}

public static MailTemplate getDynamicTemplate(Languages language, String content) {
        MailTemplate mailTemplate = null;
        if(language.equals(Languages.English)) {
                mailTemplate = new EnglishLanguageTemplate(content);
        } else if(language.equals(Languages.French)) {
                mailTemplate = new FrenchLanguageTemplate(content);
        } else if(language.equals(Languages.Chinese)) {
                mailTemplate = new ChineseLanguageTemplate(content);
        } else {
                throw new IllegalArgumentException("Invalid language type spec
        }
        return mailTemplate;
}
```

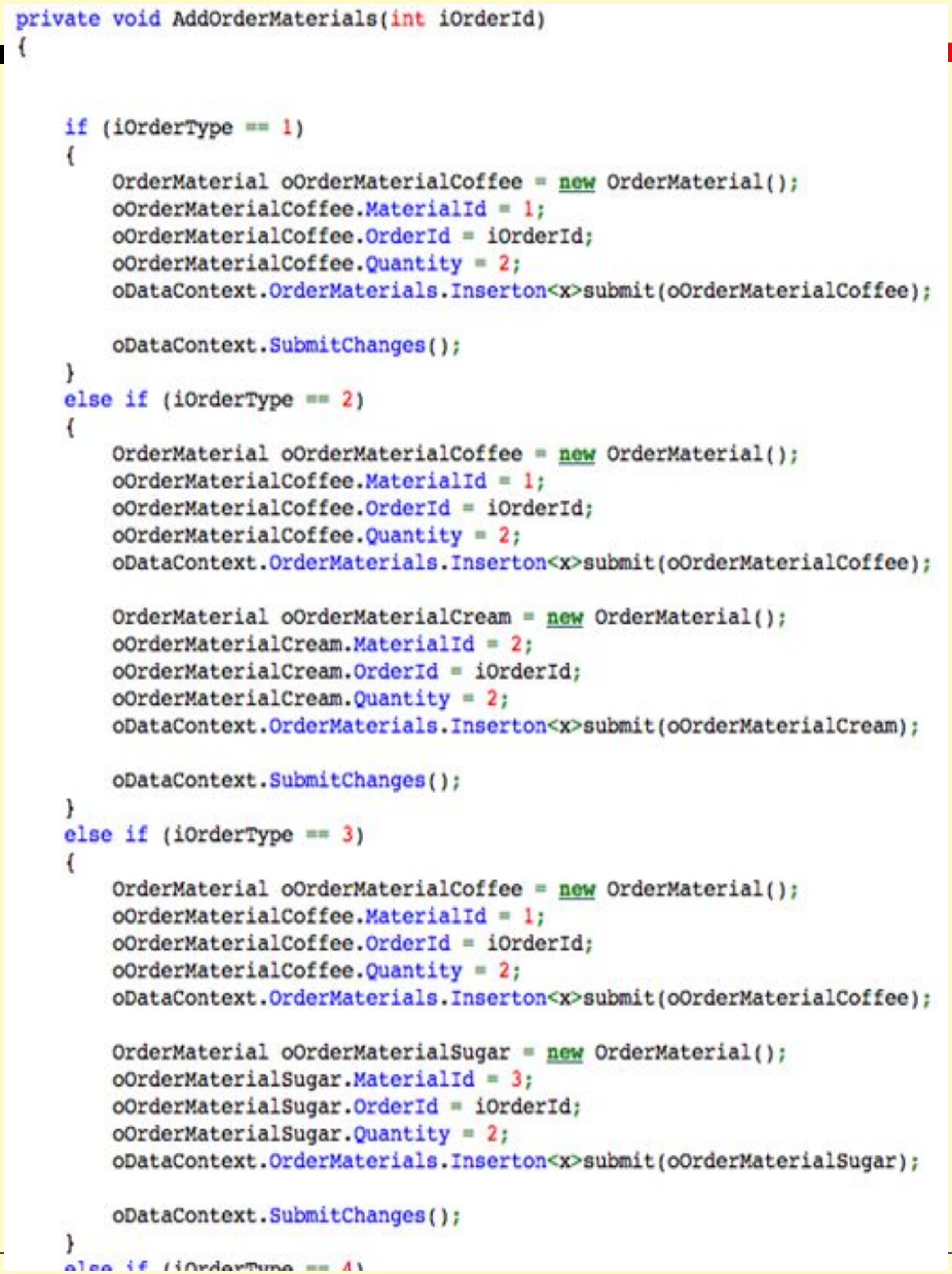
Example Of Obvious Duplication


allNodesVector
*
Node
toPlainTextString(): String
linkData
*
FormTag
toPlainTextString(): String
. . .
LinkTag
toPlainTextString(): String

```
StringBuffer textContents = new StringBuffer();
Enumeration e = allNodesVector.elements()
while (e.hasMoreElements()) {
  Node node = (Node)e.nextElement();
  textContents .append(node.toPlainTextString());
}
return textContents .toString();
```

```
StringBuffer sb = new StringBuffer();
Enumeration e = linkData .elements();
while (e.hasMoreElements()) {
  Node node = (Node)e.nextElement();
  sb .append(node.toPlainTextString());
}
return sb.toString();
```

Inspiring Excellence



```
private void AddOrderMaterials(int iOrderId)
{


    if (iOrderType == 1)
    {
        OrderMaterial oOrderMaterialCoffee = new OrderMaterial();
        oOrderMaterialCoffee.MaterialId = 1;
        oOrderMaterialCoffee.OrderId = iOrderId;
        oOrderMaterialCoffee.Quantity = 2;
        oDataContext.OrderMaterials.Inserton<x>submit(oOrderMaterialCoffee);

        oDataContext.SubmitChanges();
    }
    else if (iOrderType == 2)
    {
        OrderMaterial oOrderMaterialCoffee = new OrderMaterial();
        oOrderMaterialCoffee.MaterialId = 1;
        oOrderMaterialCoffee.OrderId = iOrderId;
        oOrderMaterialCoffee.Quantity = 2;
        oDataContext.OrderMaterials.Inserton<x>submit(oOrderMaterialCoffee);

        OrderMaterial oOrderMaterialCream = new OrderMaterial();
        oOrderMaterialCream.MaterialId = 2;
        oOrderMaterialCream.OrderId = iOrderId;
        oOrderMaterialCream.Quantity = 2;
        oDataContext.OrderMaterials.Inserton<x>submit(oOrderMaterialCream);

        oDataContext.SubmitChanges();
    }
    else if (iOrderType == 3)
    {
        OrderMaterial oOrderMaterialCoffee = new OrderMaterial();
        oOrderMaterialCoffee.MaterialId = 1;
        oOrderMaterialCoffee.OrderId = iOrderId;
        oOrderMaterialCoffee.Quantity = 2;
        oDataContext.OrderMaterials.Inserton<x>submit(oOrderMaterialCoffee);

        OrderMaterial oOrderMaterialSugar = new OrderMaterial();
        oOrderMaterialSugar.MaterialId = 3;
        oOrderMaterialSugar.OrderId = iOrderId;
        oOrderMaterialSugar.Quantity = 2;
        oDataContext.OrderMaterials.Inserton<x>submit(oOrderMaterialSugar);

        oDataContext.SubmitChanges();
    }
    else if (iOrderType == 4)
```

Levels of Duplication

Literal Duplication

Same for loop in 2 places

Semantic Duplication

1st Level - For and For Each Loop

2nd Level - Loop v/s Lines repeated

```
for(int i : asList(1,3,5,10,15))
stack.push(i);

v/s

for(int i=0;i<5;i++){
  stack.push(asList(i));

}
```

```
stack.push(1); stack.push(3);
stack.push(5); stack.push(10);
stack.push(15);

v/s

for(int i : asList(1,3,5,10,15))
stack.push(i);
```

Data Duplication

Some constant declared in 2 classes (test and production)

Conceptual Duplication

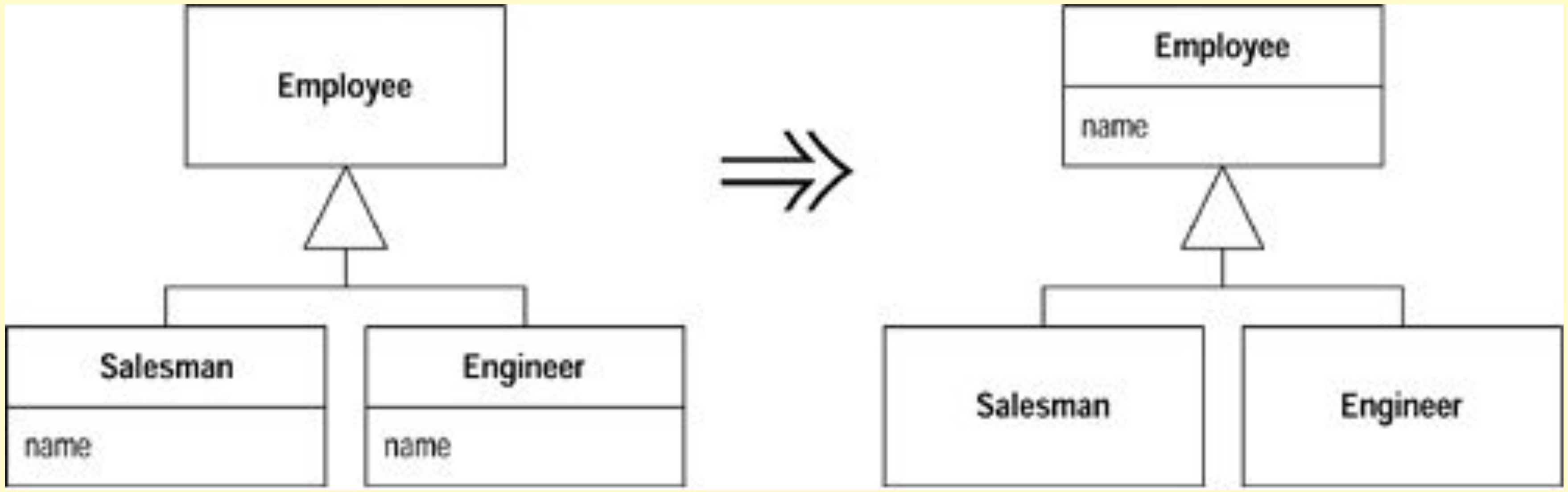
2 Algorithm to Sort elements (Bubble sort and Quick sort)

Logical Steps - Duplication

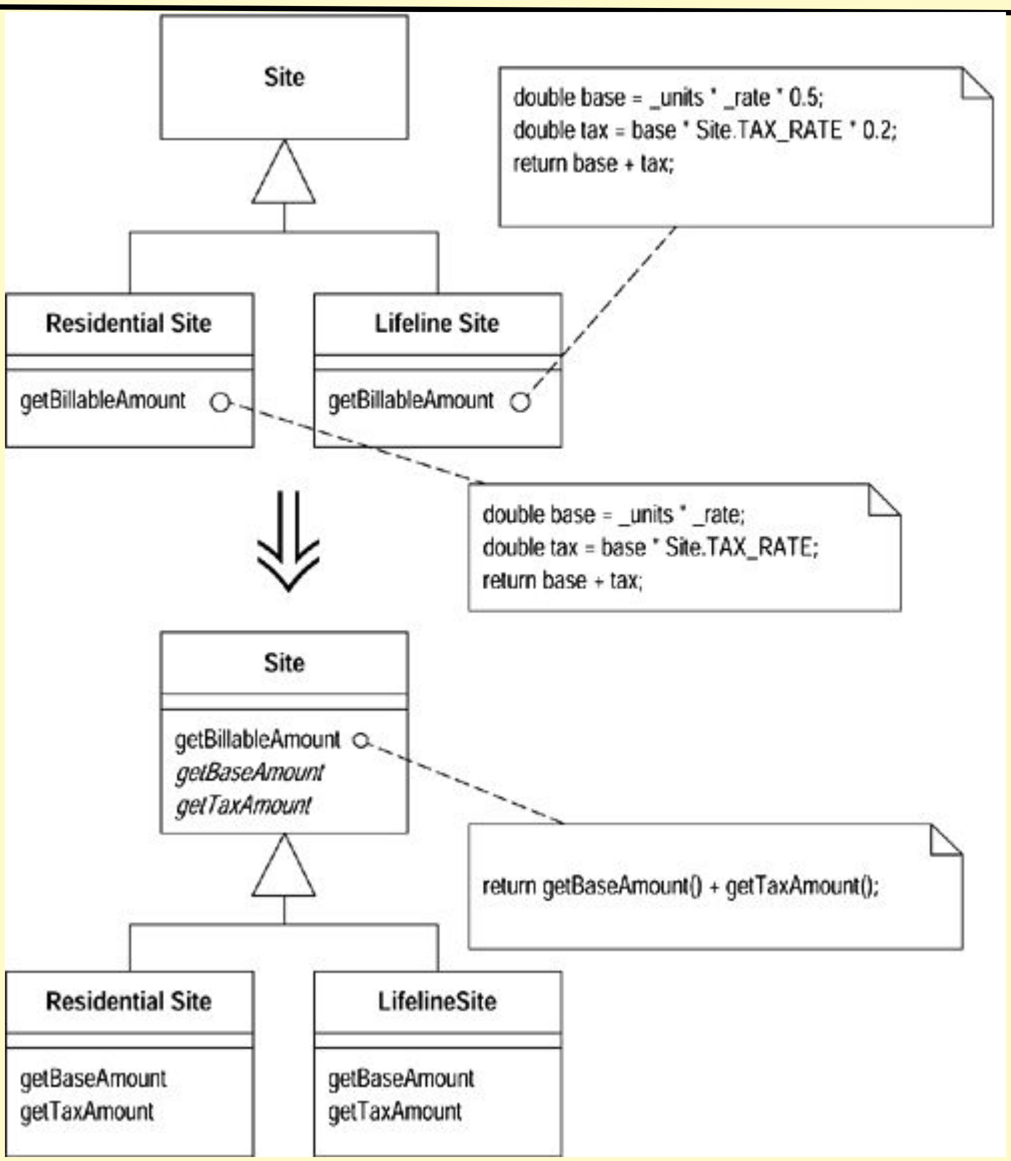
Same set of steps repeat in different scenarios.

Ex: Same set of validations in various points in your applications

Pull Up Field



Form Template Method



```
double base = _units * _rate * 0.5;
double tax = base * Site.TAX_RATE * 0.2;
return base + tax;
```

```
double base = _units * _rate;
double tax = base * Site.TAX_RATE;
return base + tax;
```

```
return getBaseAmount() + getTaxAmount();
```

Substitute 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 ""; }
```

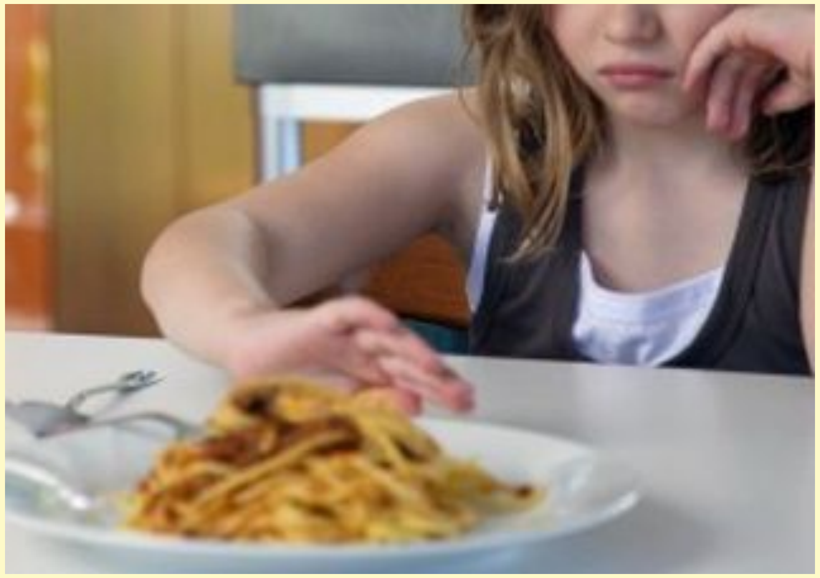
Substitute Algorithm

```
 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 "";
}
```

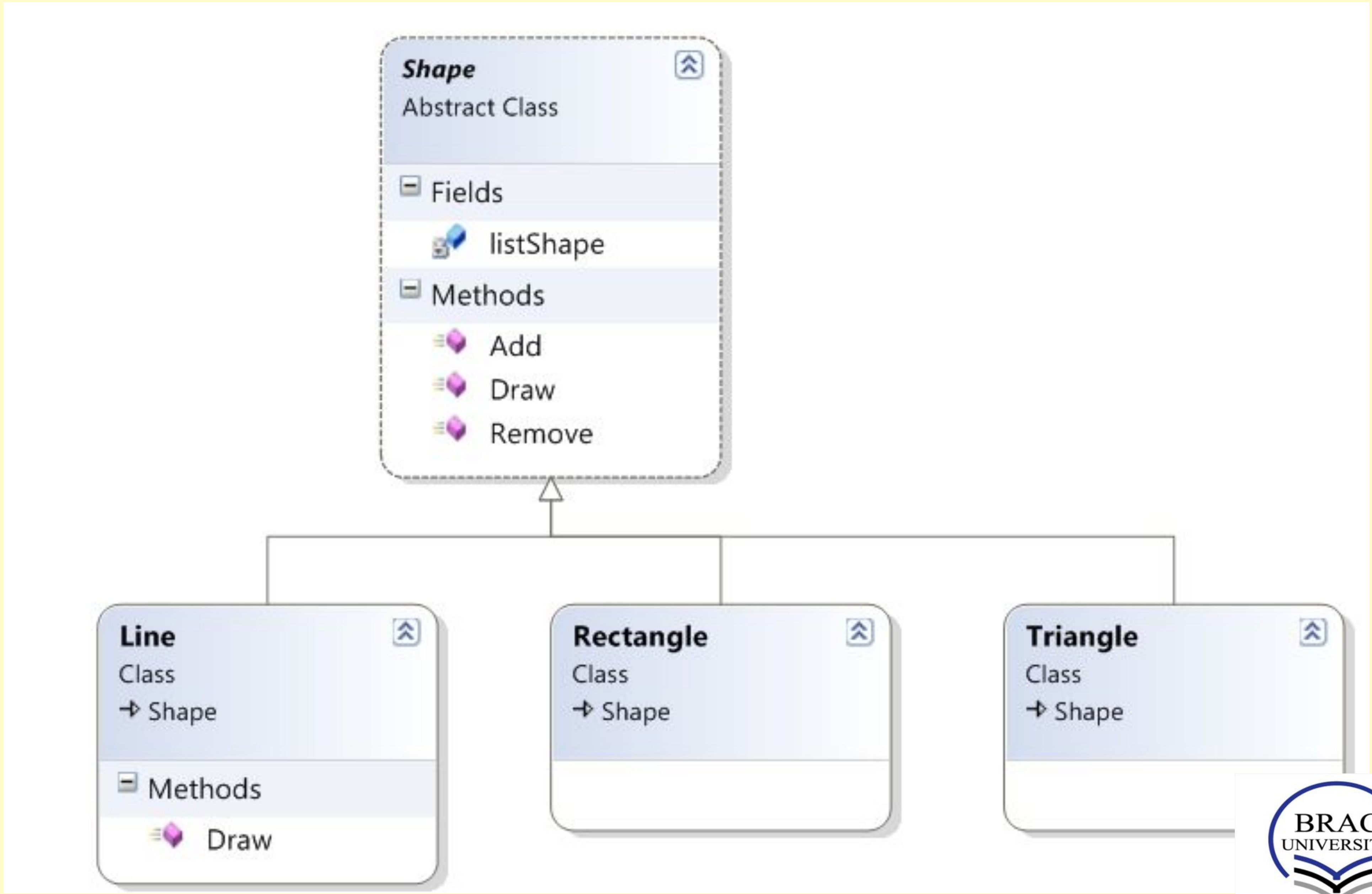
```
String foundPerson(String[] people){
    List candidates = Arrays.asList(new String[]  {"Don",
"John", "Kent"});
    for (String person : people)
        if (candidates.contains(person))
            return person;
    return "";
}
```

Refused Bequest

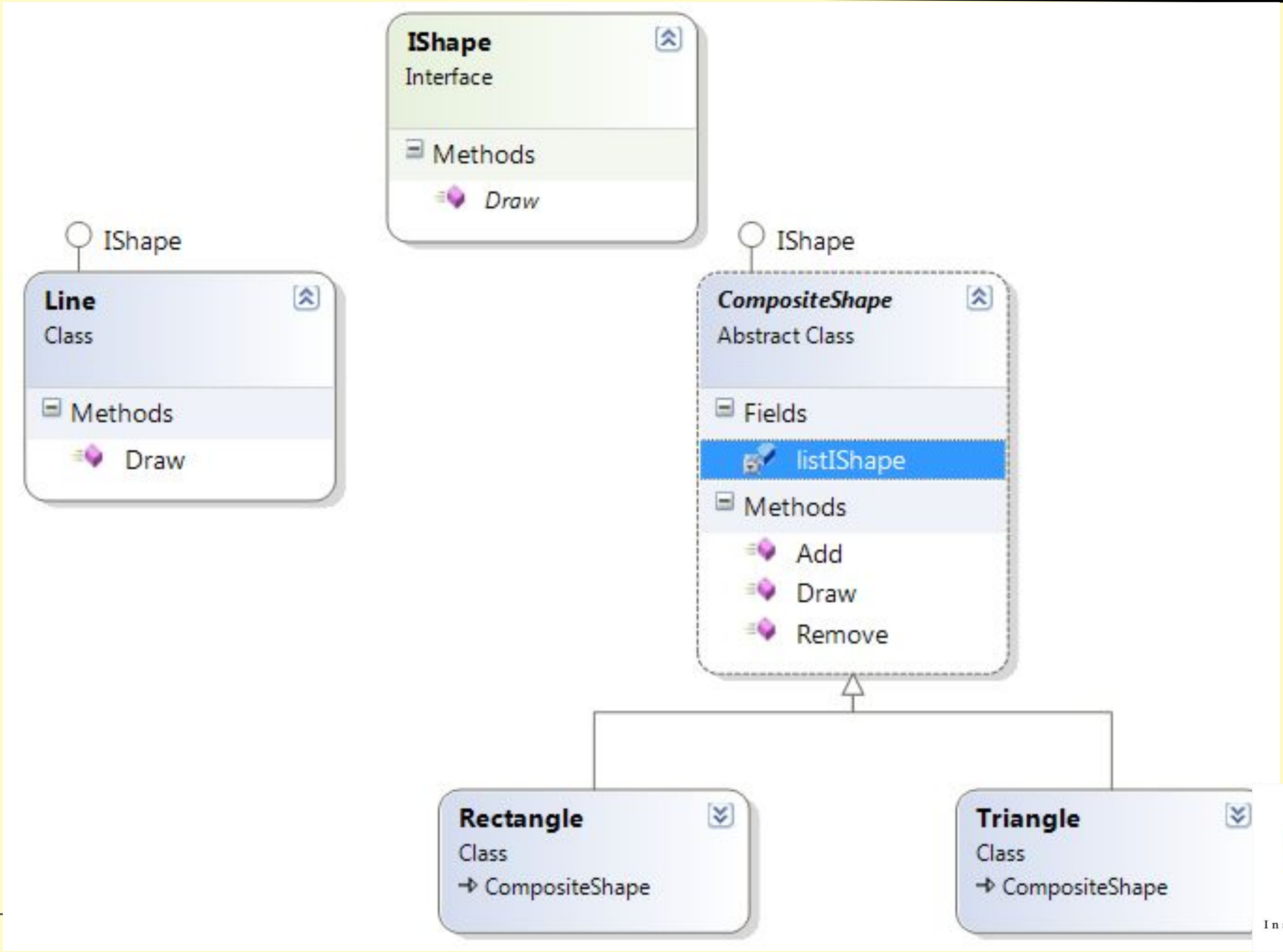
- This rather potent odor results when *subclasses inherit code that they don't want*. In some cases, a subclass may "refuse the bequest" by providing a *do-nothing implementation* of an inherited method.
- Remedies
  - Push Down Field
  - Push Down Method



Example of Refused Bequest



Refused Bequest Make Over



References

[F] Fowler, Martin. *Refactoring: Improving the Design of Existing Code*. Boston, MA: Addison-Wesley, 2000

[K] Kerievsky, Joshua. *Refactoring to Patterns*. Boston, MA: Addison-Wesley, 2005