Refactoring

COMP3111/H tutorial

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

Objectives

- After this tutorial, you will learn about
  - Basic concept about refactoring
  - Low-level refactoring by refactoring tools in Eclipse
  - How to use Eclipse tool for refactoring

What is Refactoring?

- **Refactoring** improves the design of existing code
- Improving **Maintainability** and **Extensibility**
  - The internal structure of the code without altering its external behavior
  - Not rewriting from scratch

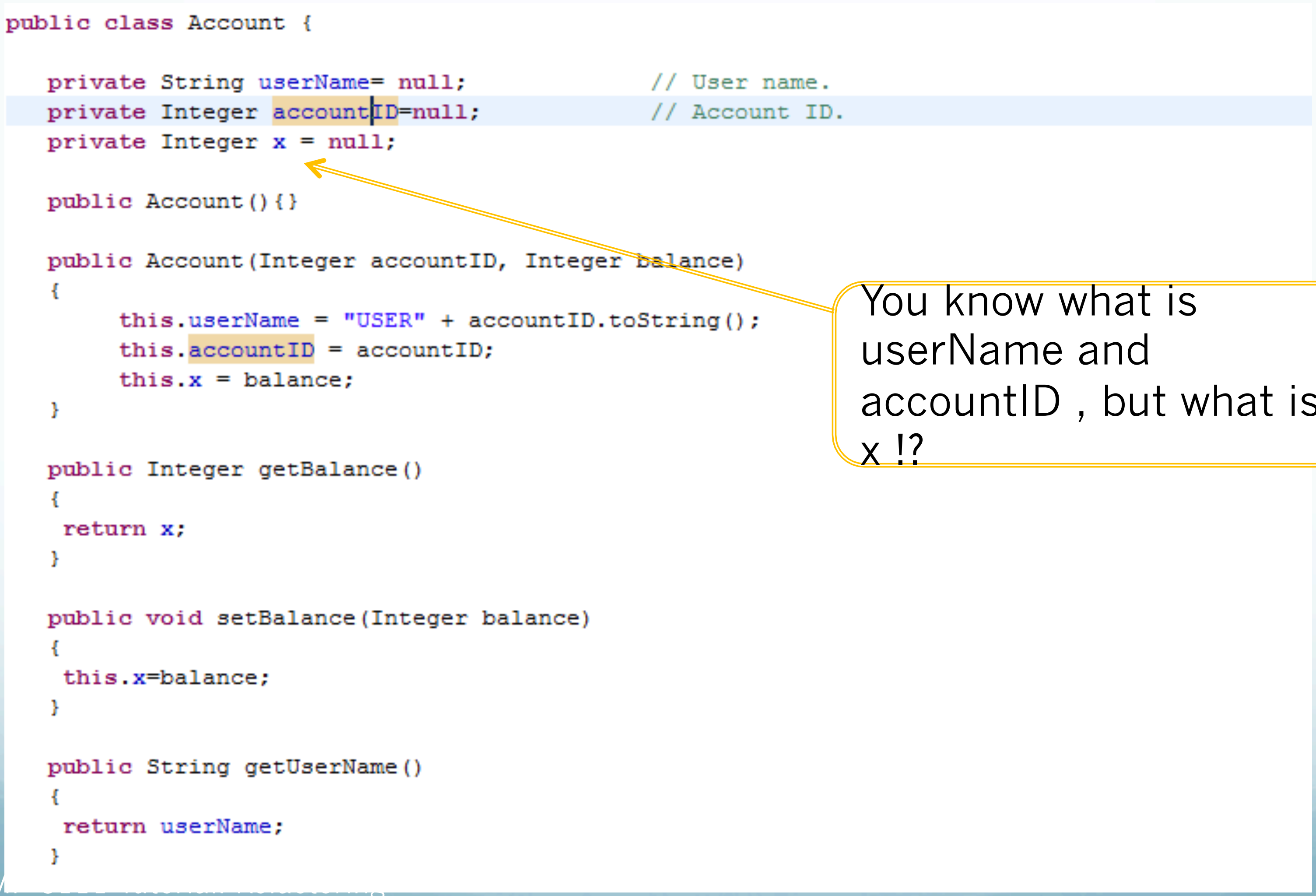
What is Refactoring?

- Why refactoring a piece of code which is functional and bug free?
- Three major purposes:
  - to **execute** its functionality
  - to **allow change**
  - to **communicate** well to developers who read it.
- Good code should do all of them

Low-level Refactoring

- **Names**
  - Renaming (operations, variables)
  - Replace Magic Number with Symbolic Constant
- **Procedures**
  - Extracting code into a method
  - Extracting common functionality (including duplicate code) into a class/method/etc.
  - Inline an Class/Method
  - Changing operation signatures
- **Reordering**
  - Splitting one operation into several to improve cohesion and readability (by reducing its size)
  - Putting statements that semantically belong together near each other

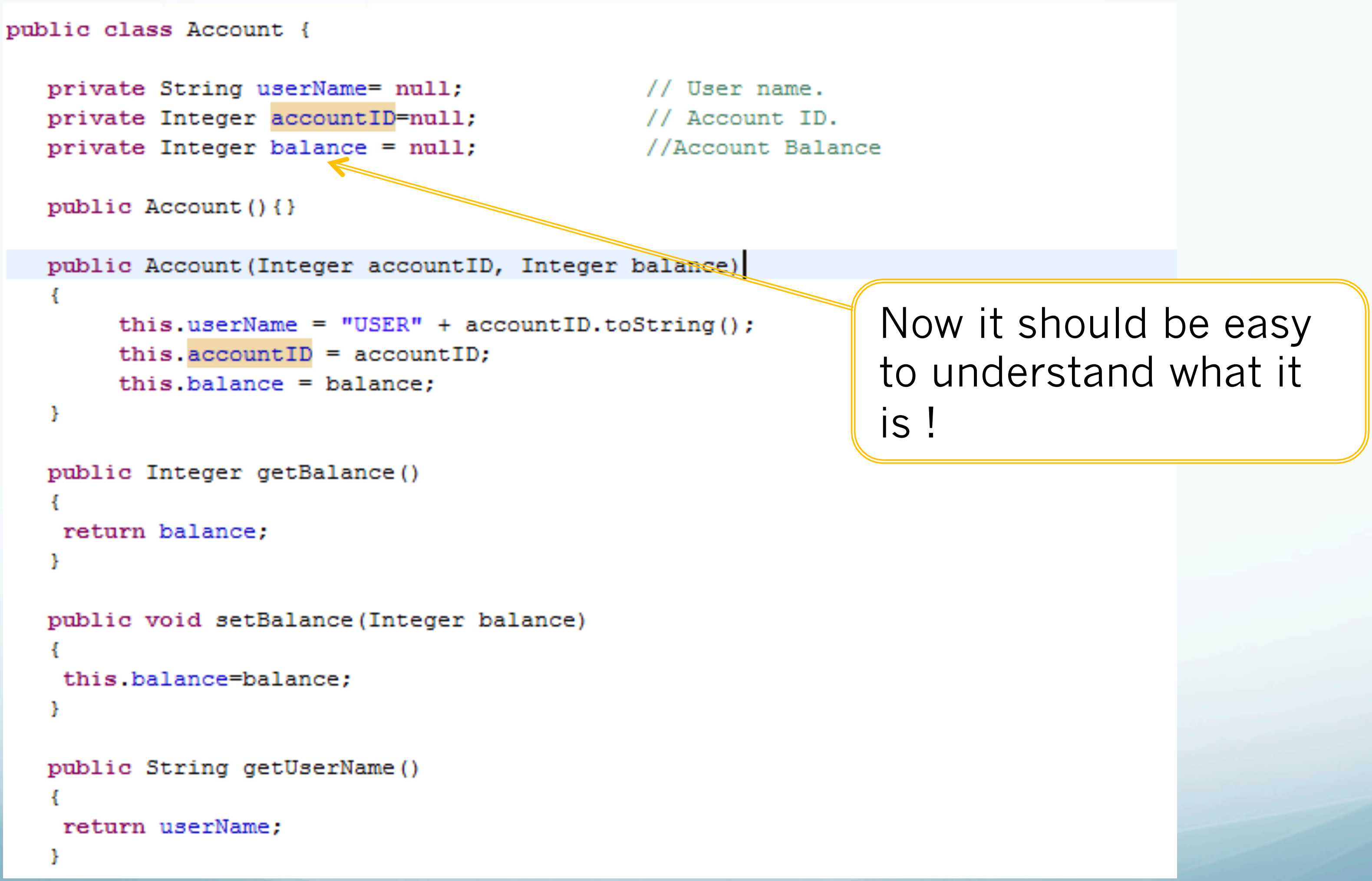
Renaming Example



```
public class Account {

   private String userName= null;                  // User name.
   private Integer accountID=null;                 // Account ID.
   private Integer x = null;

   public Account(){}

   public Account(Integer accountID, Integer balance)
   {
        this.userName = "USER" + accountID.toString();
        this.accountID = accountID;
        this.x = balance;
   }

   public Integer getBalance()
   {
    return x;
   }

   public void setBalance(Integer balance)
   {
    this.x=balance;
   }

   public String getUserName()
   {
    return userName;
   }
```

You know what is userName and accountID , but what is x !?

Renaming Example



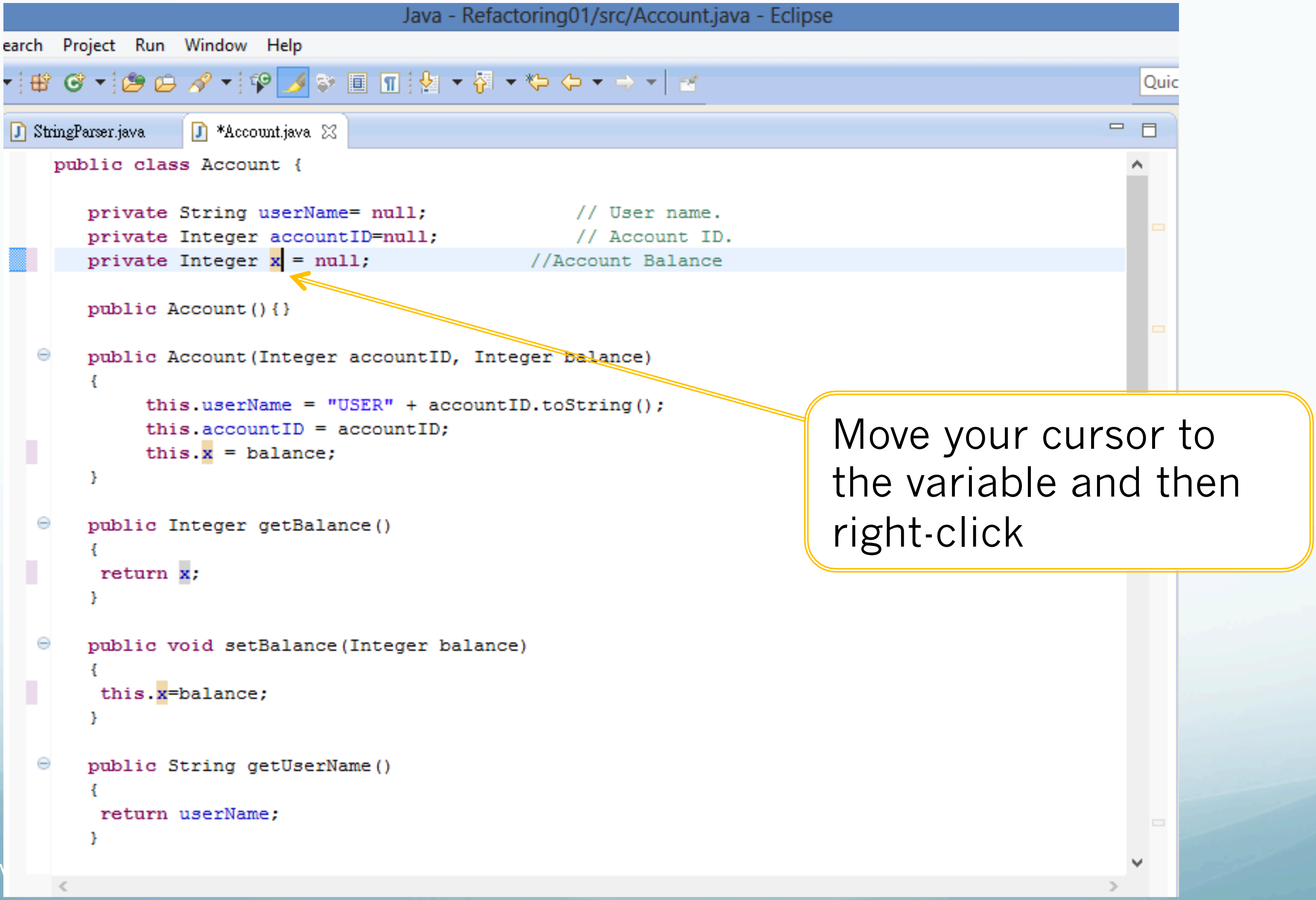
```
public class Account {

   private String userName= null;              // User name.
   private Integer accountID=null;             // Account ID.
   private Integer balance = null;             //Account Balance

   public Account(){}

   public Account(Integer accountID, Integer balance)
   {
        this.userName = "USER" + accountID.toString();
        this.accountID = accountID;
        this.balance = balance;
   }

   public Integer getBalance()
   {
    return balance;
   }

   public void setBalance(Integer balance)
   {
    this.balance=balance;
   }

   public String getUserName()
   {
    return userName;
   }
```

Now it should be easy to understand what it is !

Renaming Example

- Variable names can be non-sense
  - x, y, i, j, temp, dummy…
  - student1, student2, studentXY…
- Renaming helps improve the communication among developers working on the same software project

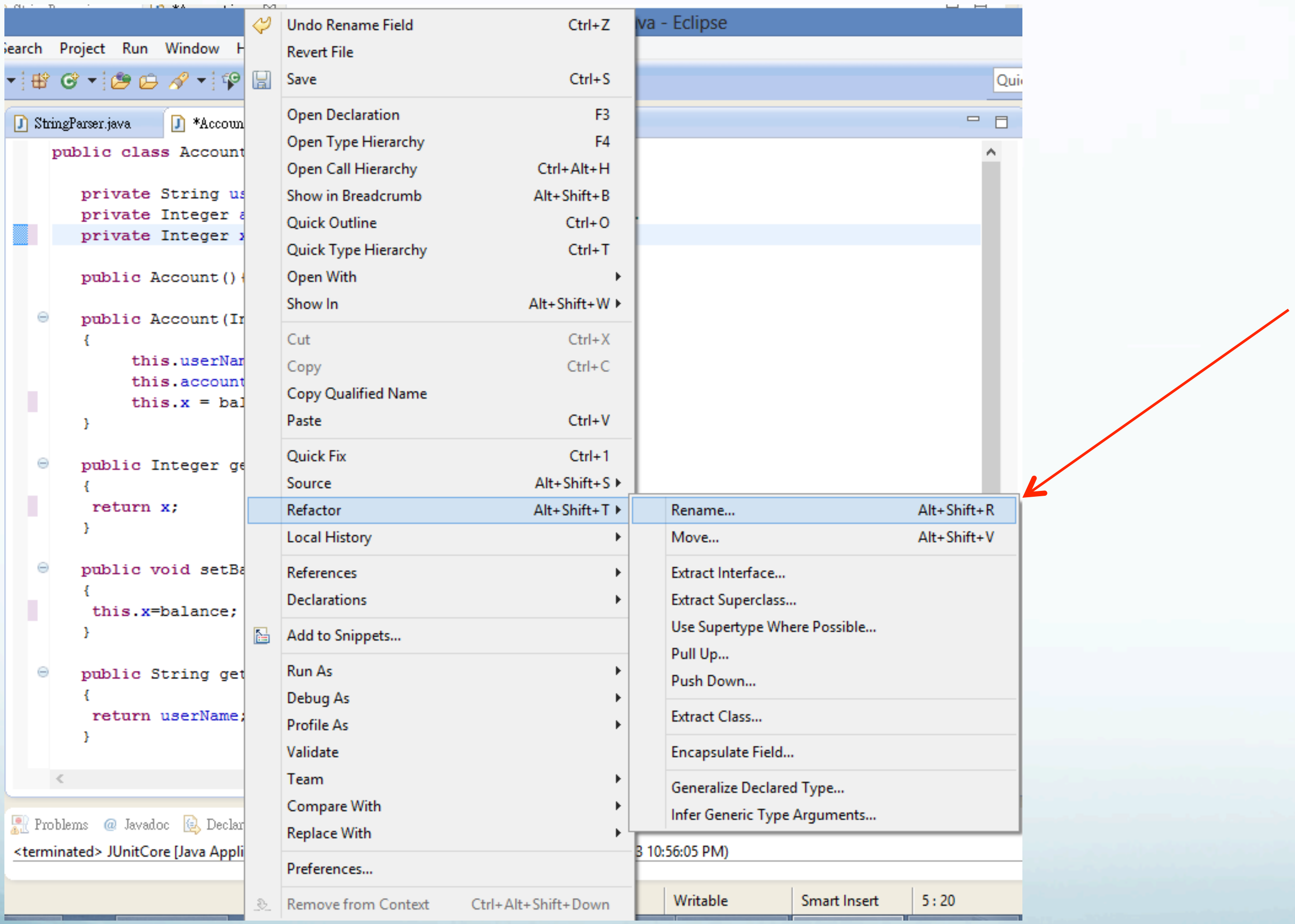
Renaming using Eclipse


Java - Refactoring01/src/Account.java - Eclipse
earch Project Run Window Help
StringParser.java
*Account.java

```
public class Account {

   private String userName= null;             // User name.
   private Integer accountID=null;            // Account ID.
   private Integer x = null;               //Account Balance

   public Account(){}

   public Account(Integer accountID, Integer balance)
   {
        this.userName = "USER" + accountID.toString();
        this.accountID = accountID;
        this.x = balance;
   }

   public Integer getBalance()
   {
    return x;
   }

   public void setBalance(Integer balance)
   {
    this.x=balance;
   }

   public String getUserName()
   {
    return userName;
   }
```

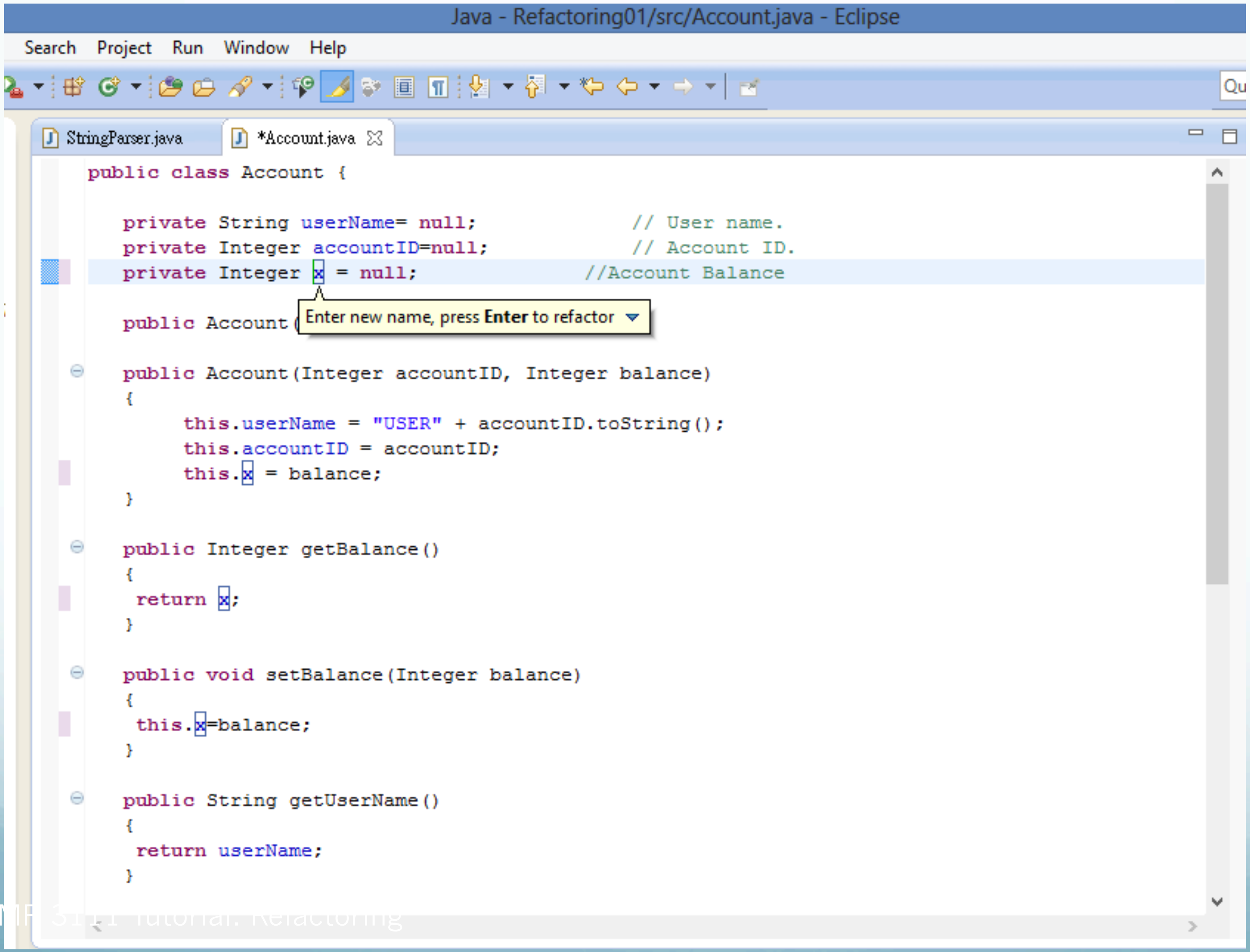
Move your cursor to the variable and then right-click

Renaming using Eclipse



```
public class Account

    private String us
    private Integer a
    private Integer x

    public Account()

    public Account(Ir
    {
        this.userNar
        this.account
        this.x = bal
    }

    public Integer ge
    {
     return x;
    }

    public void setBa
    {
     this.x=balance;
    }

    public String get
    {
     return userName;
    }
```

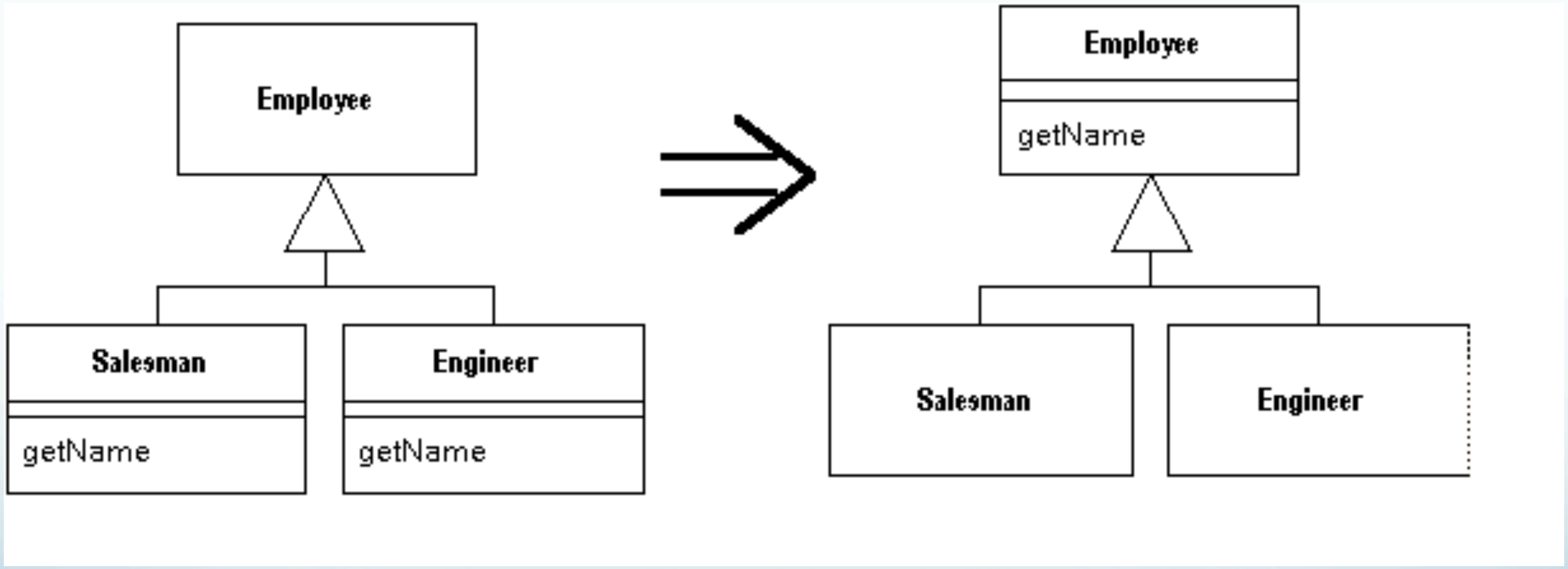
Renaming using Eclipse


Java - Refactoring01/src/Account.java - Eclipse
Search Project Run Window Help
StringParser.java
*Account.java
Enter new name, press Enter to refactor

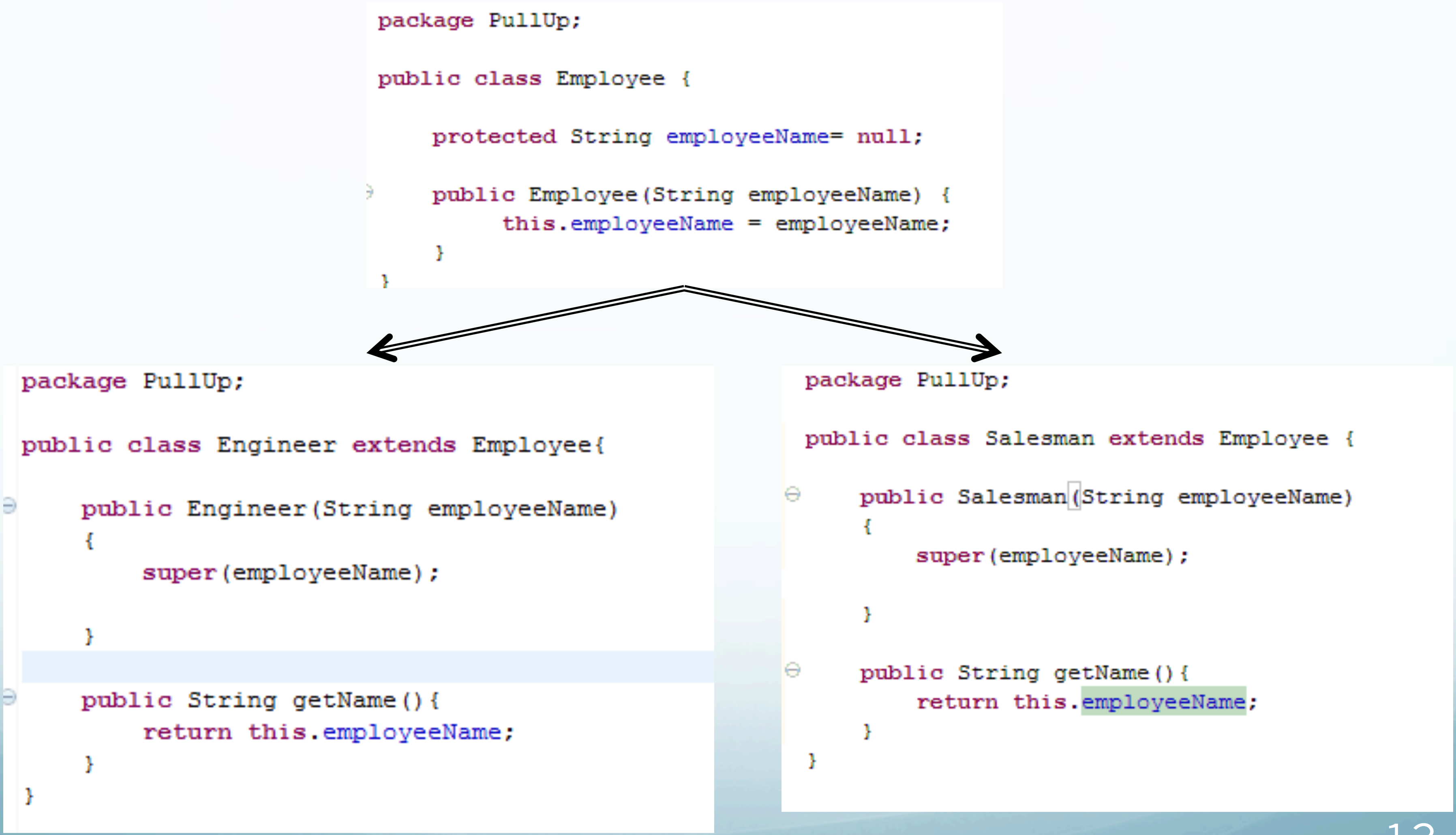
```
public class Account {

    private String userName= null;              // User name.
    private Integer accountID=null;             // Account ID.
    private Integer x = null;               //Account Balance

    public Account(

    public Account(Integer accountID, Integer balance)
    {
         this.userName = "USER" + accountID.toString();
         this.accountID = accountID;
         this.x = balance;
    }

    public Integer getBalance()
    {
     return x;
    }

    public void setBalance(Integer balance)
    {
     this.x=balance;
    }

    public String getUserName()
    {
     return userName;
    }
```

Pull up method

- Duplicated methods in subclasses can be “pulled” up to its superclass



Pull up method Example

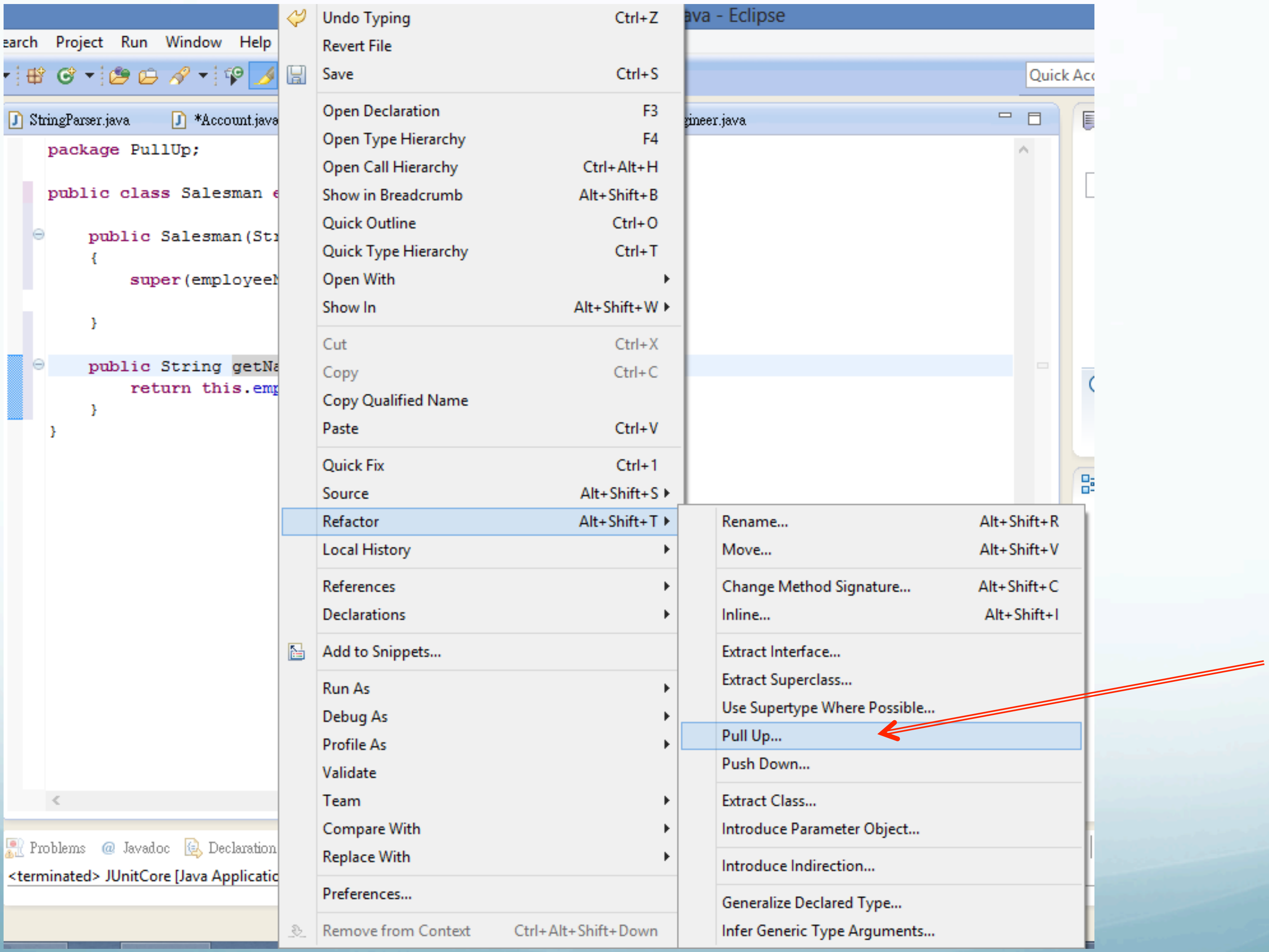


```java
package PullUp;

public class Employee {

    protected String employeeName= null;

    public Employee(String employeeName) {
        this.employeeName = employeeName;
    }
}
```

```java
package PullUp;

public class Engineer extends Employee{

    public Engineer(String employeeName)
    {
        super(employeeName);

    }

    public String getName(){
        return this.employeeName;
    }
}
```

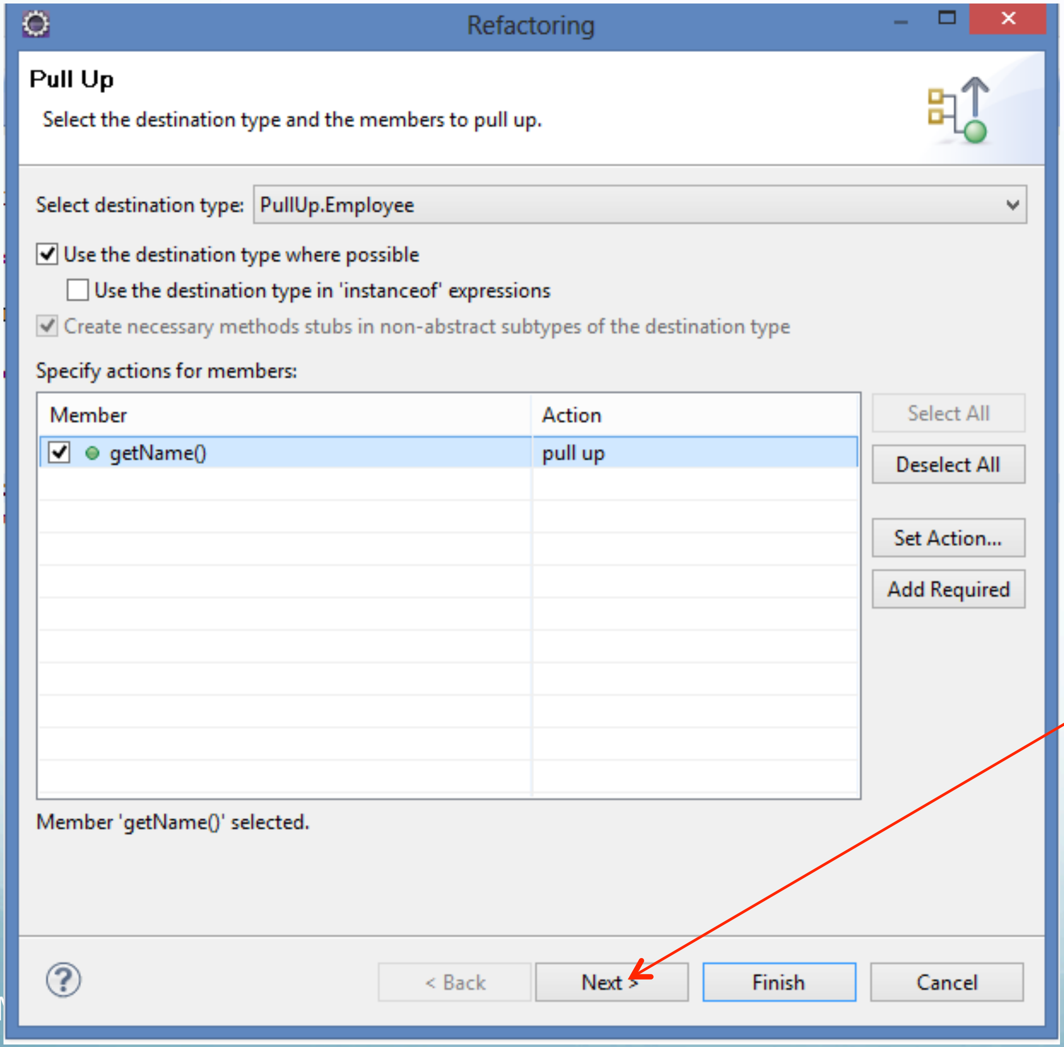
```java
package PullUp;

public class Salesman extends Employee {

    public Salesman(String employeeName)
    {
        super(employeeName);

    }

    public String getName(){
        return this.employeeName;
    }
}
```

Pull up method Example



```
package PullUp;

public class Salesman e

    public Salesman(Str
    {
        super(employeeN

    }

    public String getNa
        return this.emp
    }
}
```

Pull up method Example



Refactoring

Pull Up

Select the destination type and the members to pull up.

Select destination type: PullUp.Employee

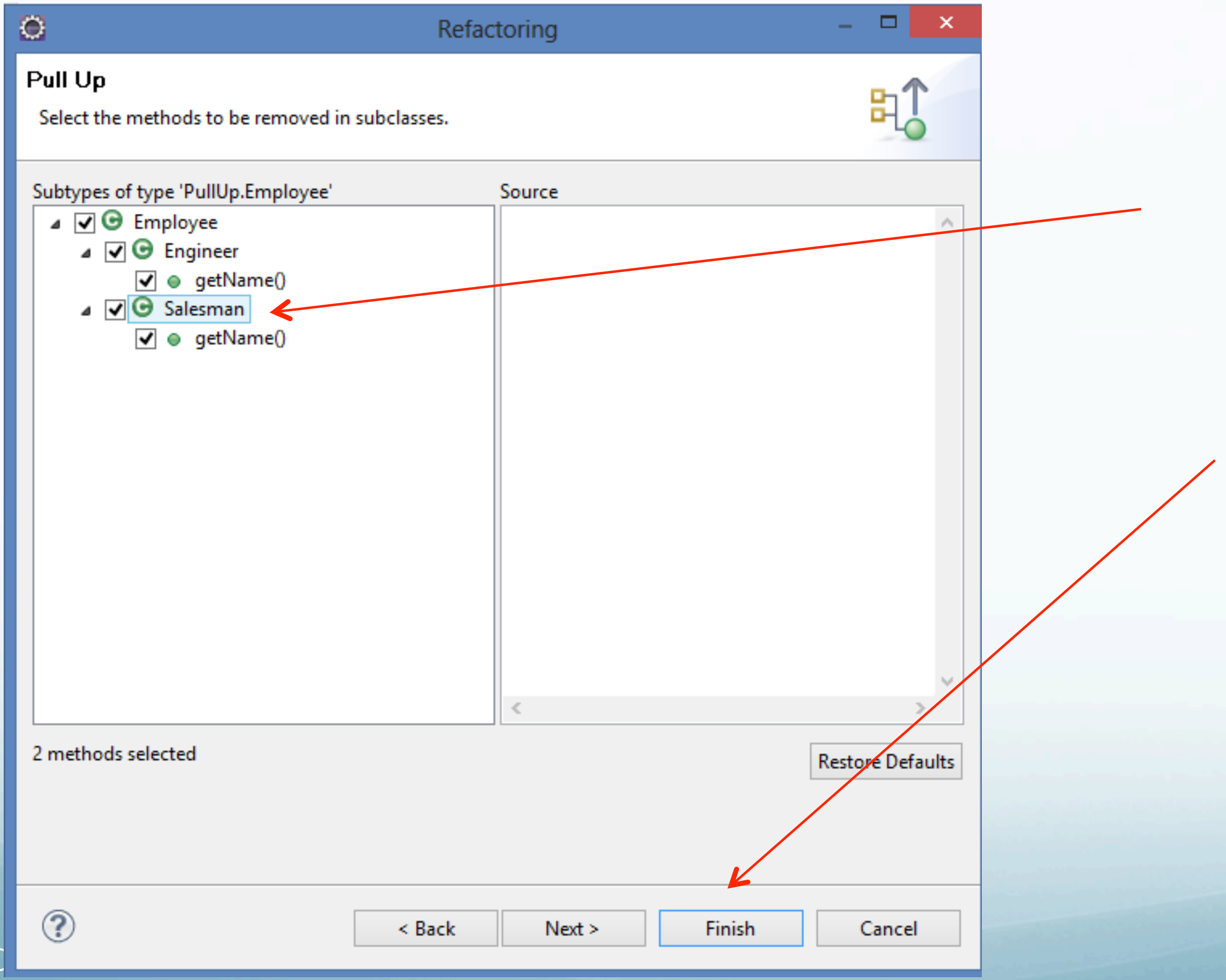
- [✓] Use the destination type where possible
  - [ ] Use the destination type in 'instanceof' expressions
- [✓] Create necessary methods stubs in non-abstract subtypes of the destination type

Specify actions for members:

Member	Action
✓ ● getName()	pull up

Member 'getName()' selected.

Pull up method Example



Refactoring

Pull Up

Select the methods to be removed in subclasses.

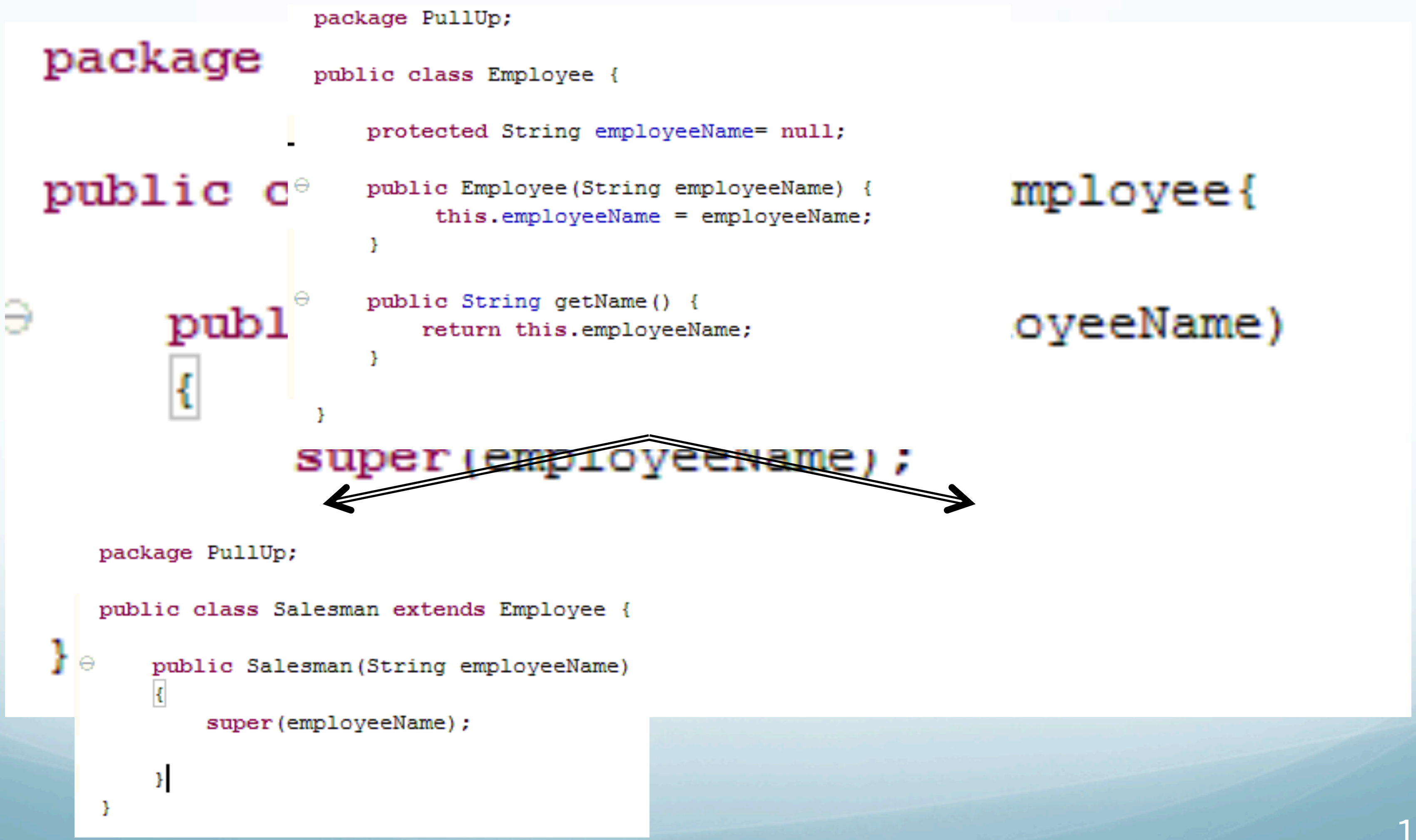
Subtypes of type 'PullUp.Employee'

- [x] Employee
  - [x] Engineer
    - [x] getName()
  - [x] Salesman
    - [x] getName()

Source

2 methods selected

Pull up method Example

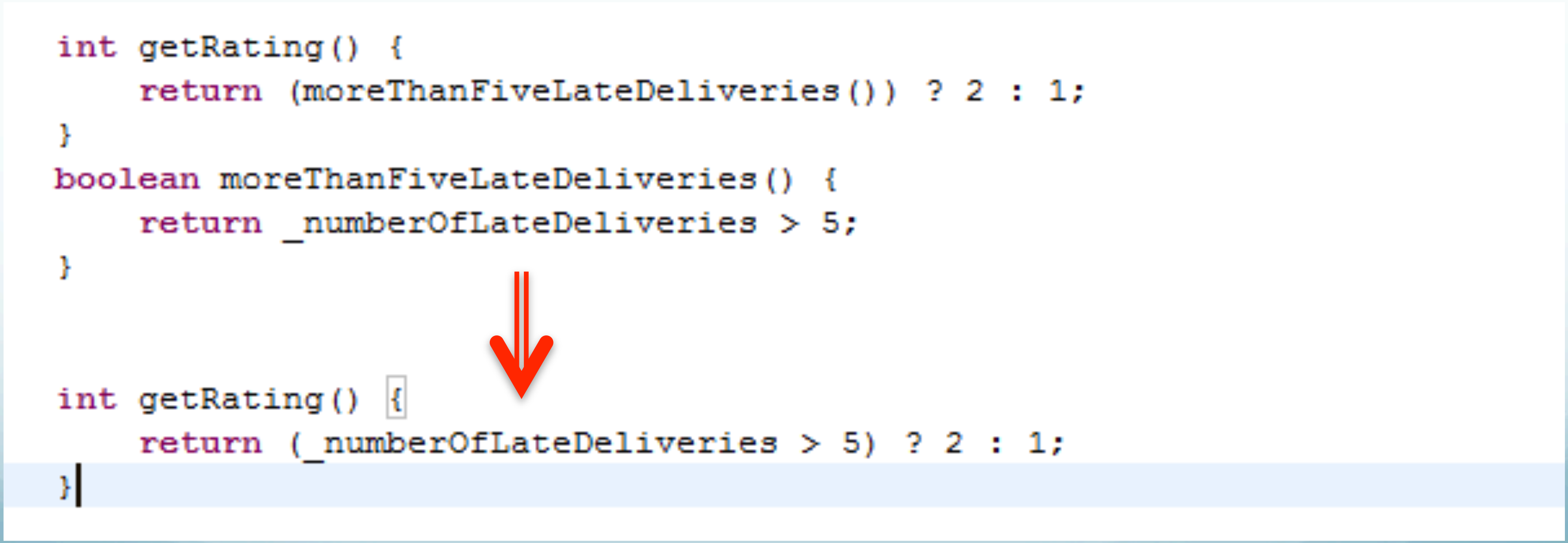


```
package PullUp;

public class Employee {

    protected String employeeName= null;

    public Employee(String employeeName) {
        this.employeeName = employeeName;
    }

    public String getName() {
        return this.employeeName;
    }

}
```

```
package PullUp;

public class Salesman extends Employee {

    public Salesman(String employeeName)
    {
        super(employeeName);

    }
}
```

Inline method

- Replace the method call by copying the content of method body
- For performance purpose

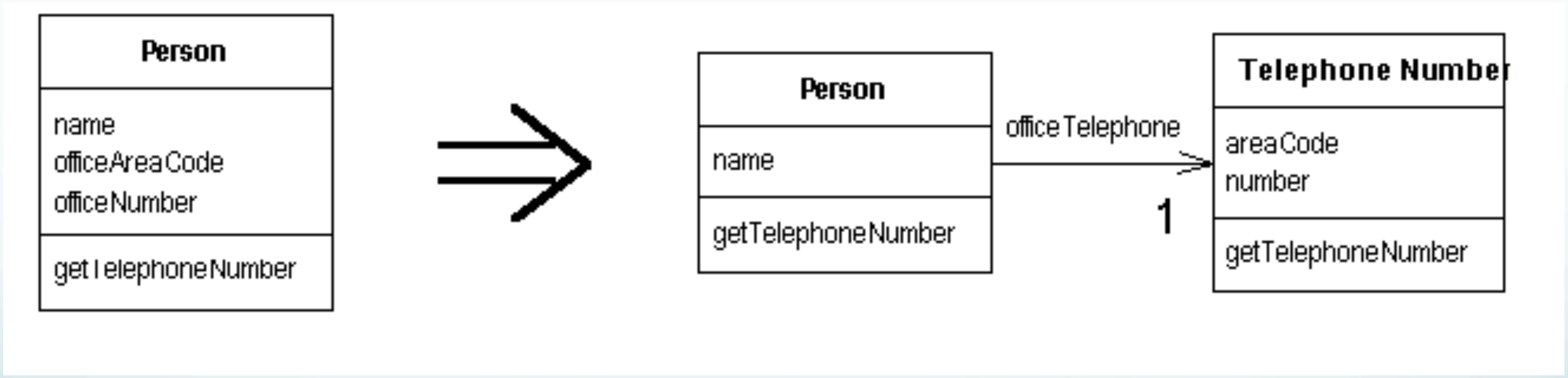


```
int getRating() {
    return (moreThanFiveLateDeliveries()) ? 2 : 1;
}
boolean moreThanFiveLateDeliveries() {
    return _numberOfLateDeliveries > 5;
}
```

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

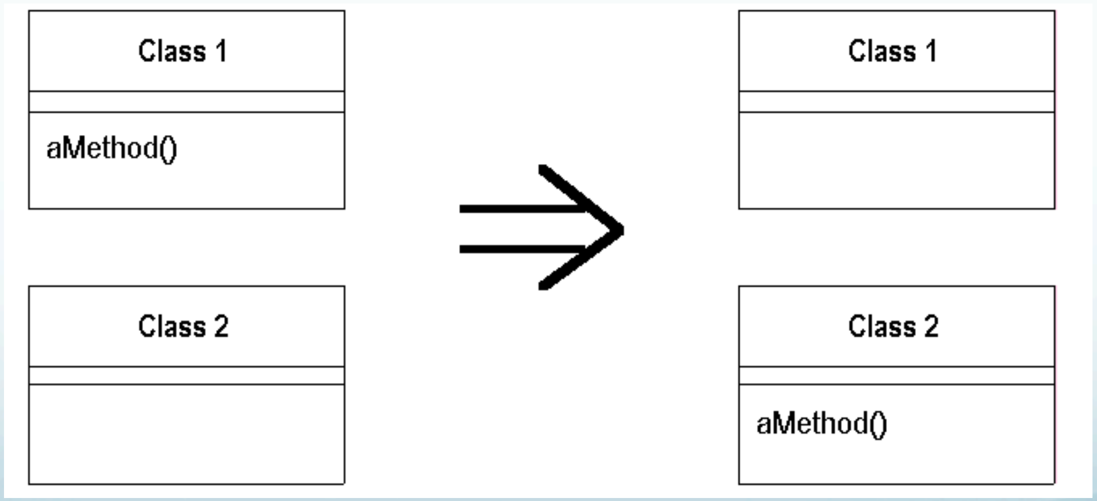
Extract Class

- In this example, two attributes {officeAreaCode and officeNumber} should better be modelled as a separate class "TelephoneNumber"



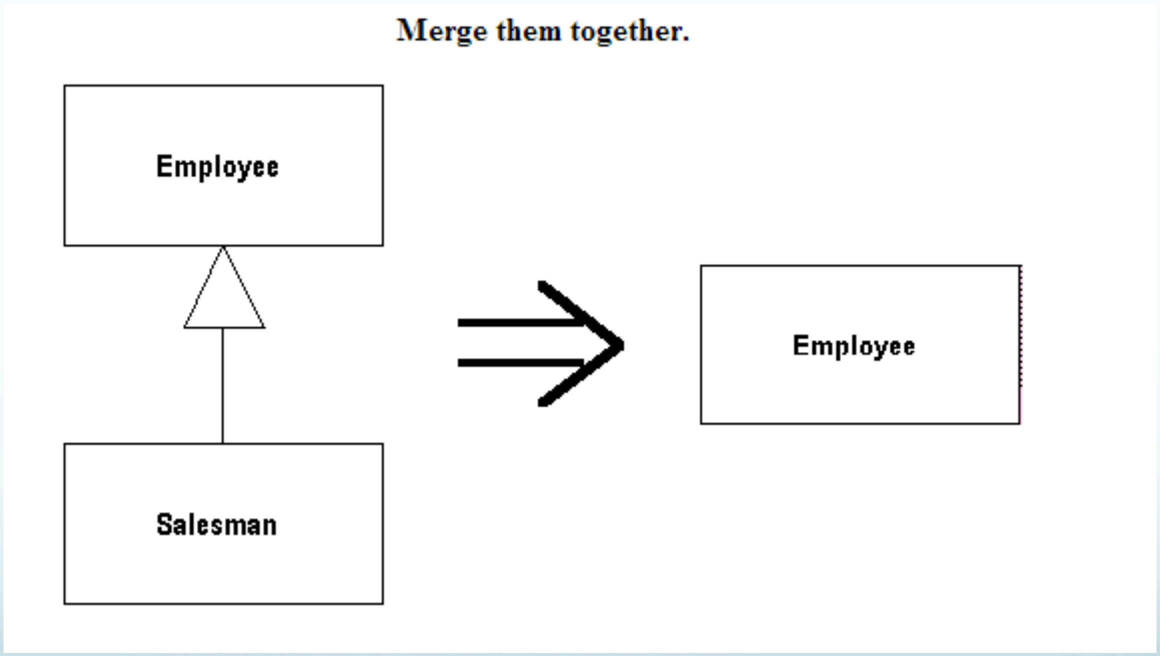
Move Method

- 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



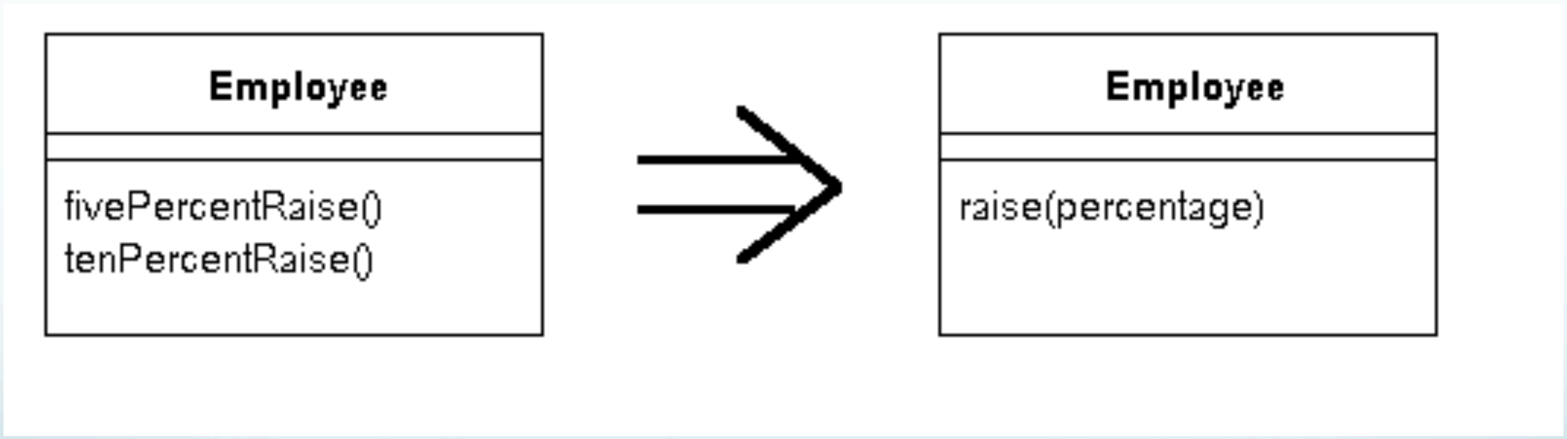
Collapse Hierarchy

- Reduce the complexity of class diagram by merging a superclass with its only subclass



Parameterize Method

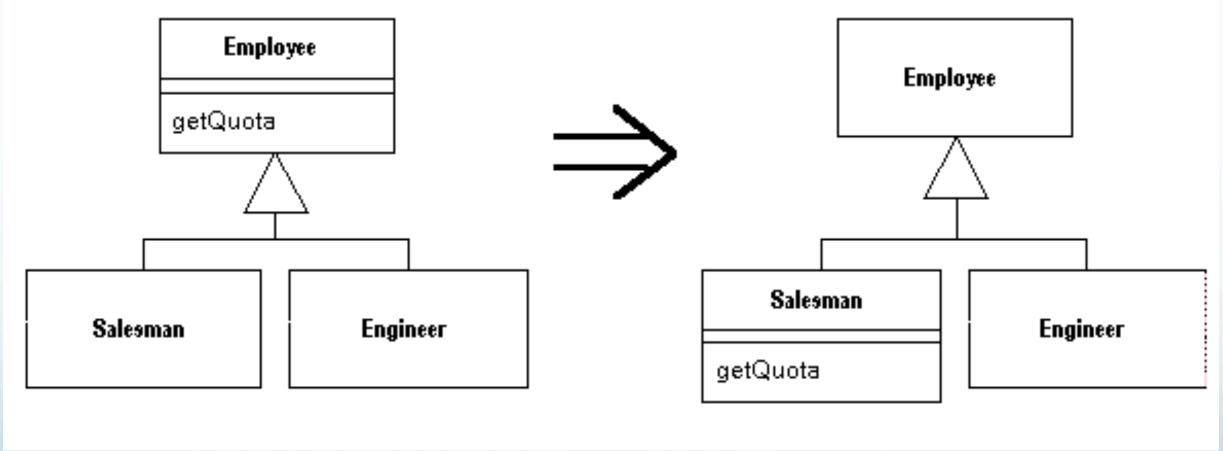
- Create one generalized method that uses a parameter for the different values
- Example:



Push Down Method

- A method may first be defined in the superclass
- In the later phase, that method may not applicable to all subclasses and better be "pushed" down

Move it to those subclasses.



More refactoring

- More refactoring techniques:
  - http://www.refactoring.com/catalog/