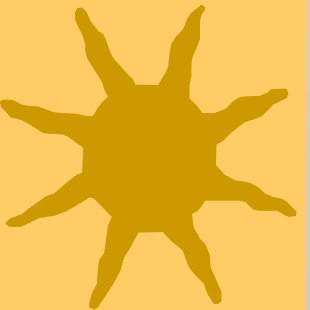




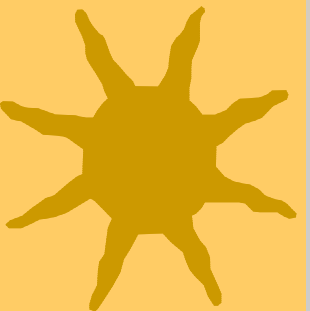
# *Introduction to Refactoring*

---



Cakes Talk – 28 February 2002

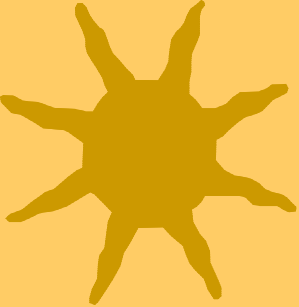
by Rani Ettinger





## *A definition of Refactoring*

---



The process of gradually improving the design of an existing software system





## *A definition of Refactoring*

---

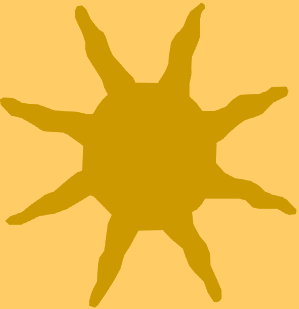
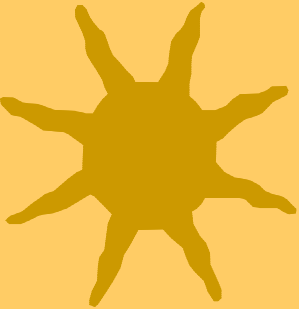


The process of gradually improving the design of an existing software system by performing source code transformations that improve its quality in such a way that it becomes easier to maintain the system and reuse parts of it

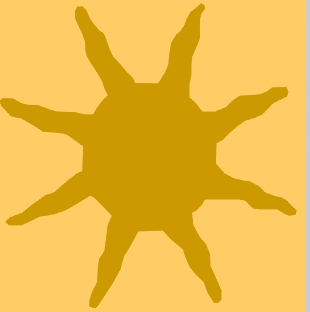
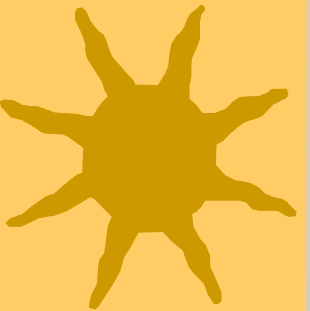


## *A definition of Refactoring*

---



The process of gradually improving the design of an existing software system by performing source code transformations that improve its quality in such a way that it becomes easier to maintain the system and reuse parts of it, while preserving the behavior of the original system.



## *Some Examples...*

---

Taken from Martin Fowler's book:  
"Refactoring – Improving the Design of  
Existing Code", Addison Wesley, 2000

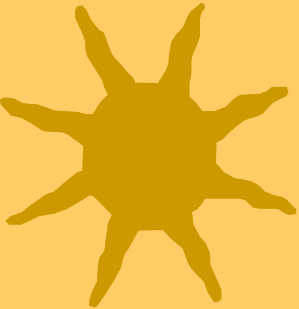


## *Extract Method*

---



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



```
// print details
```

```
System.out.println("name:" + _name);
```

```
System.out.println("amount:" + amount);
```

```
}
```



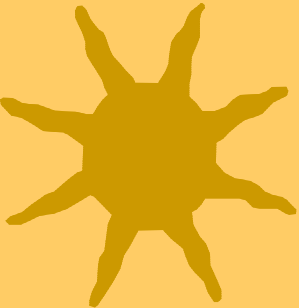


## *Extract Method (2)*

---



```
void printOwing(double amount) {  
    printBanner();  
    printDetails(amount);  
}
```



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





## *Add Parameter*

---



“A method needs more information from its caller.”



**Customer**

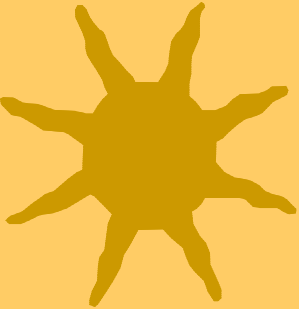


 getContact()



## *Add Parameter (2)*

---

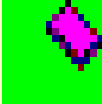


“Add a parameter to an object that can pass on this information.”



# Customer



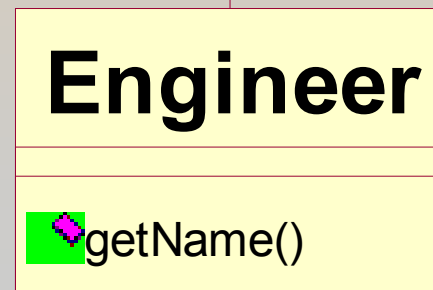
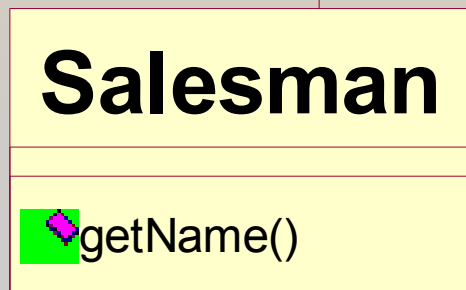
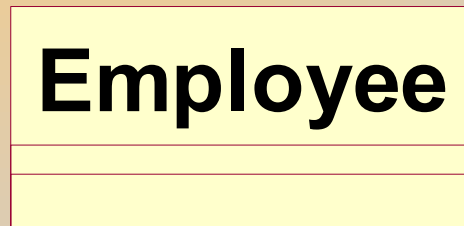
 getContact(date : Date)



# *Pull Up Method*



“You have methods with identical results on subclasses”

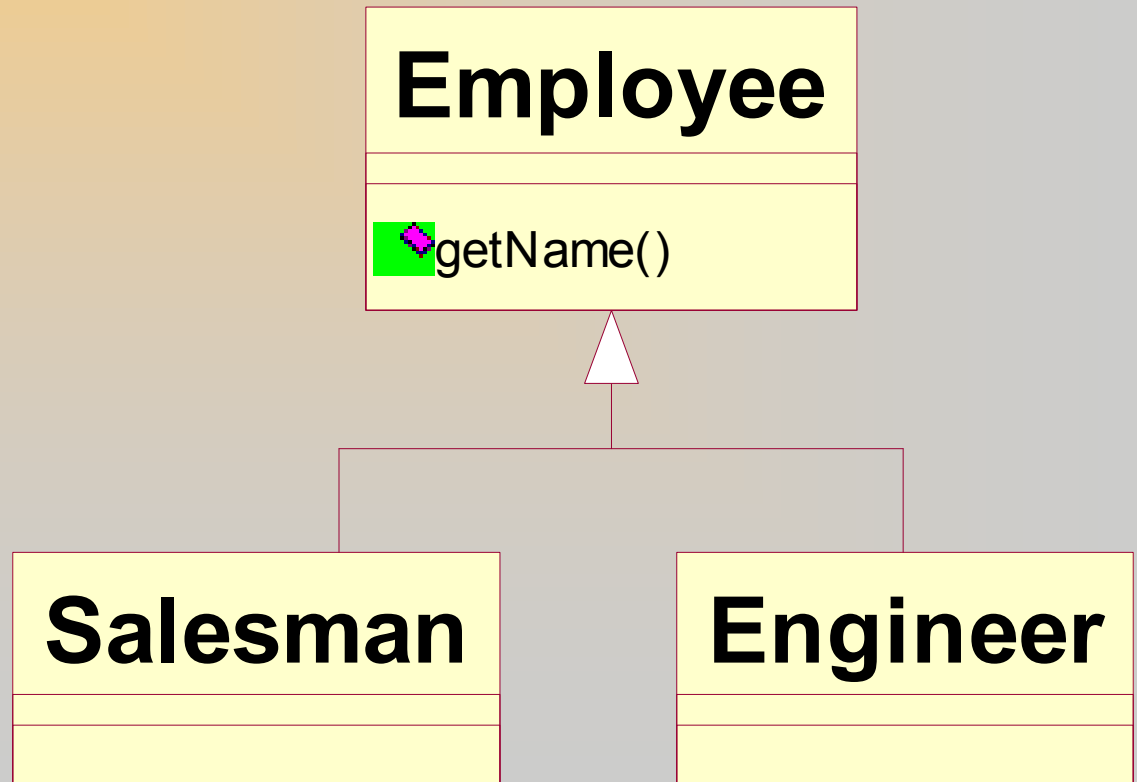




## *Pull Up Method (2)*

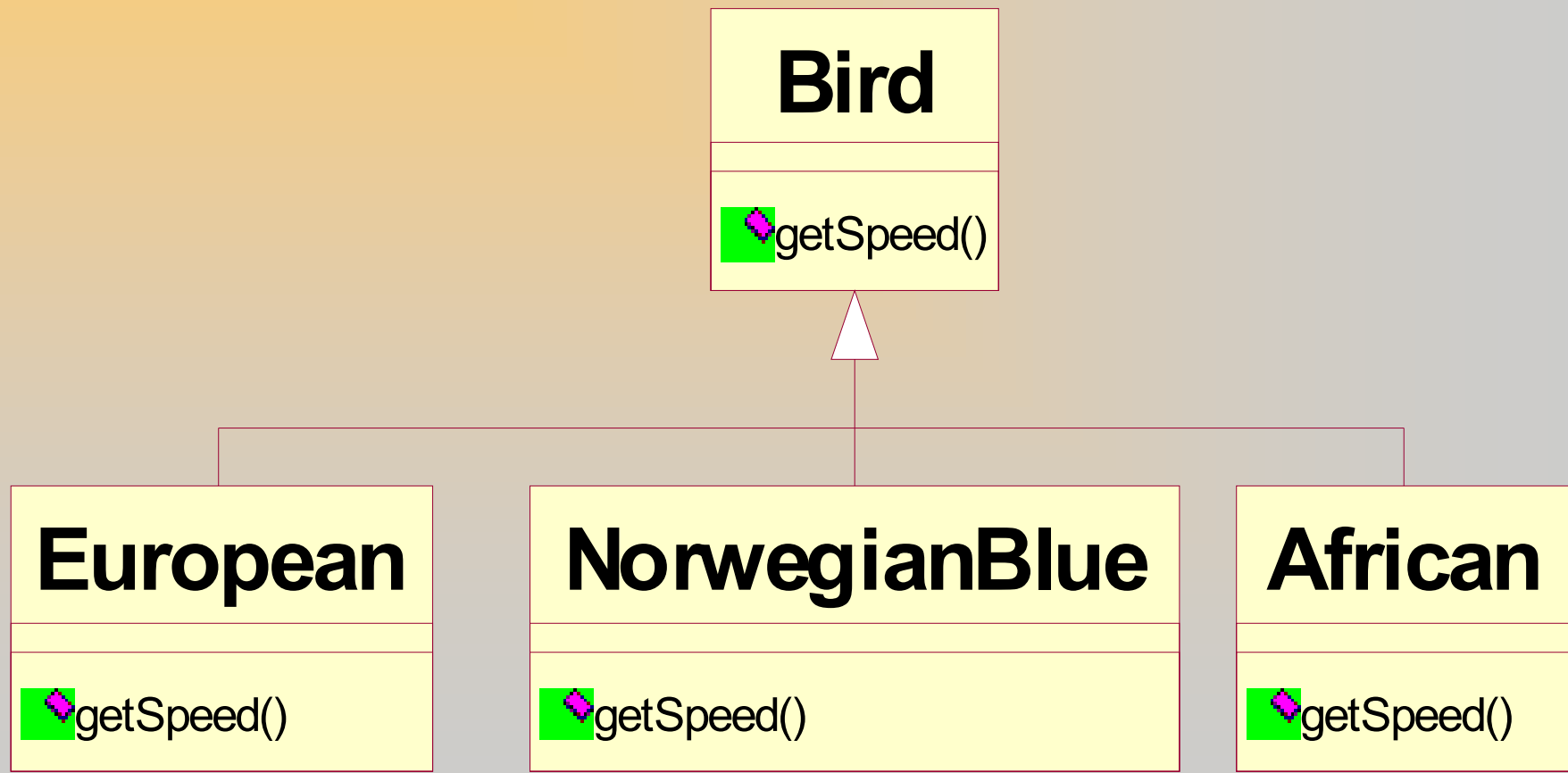


“Move them to the superclass”





# *Replace conditional with Polymorphism (2)*





## *The trick*

---

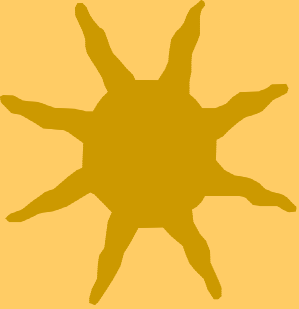


- Quick and not dirty
- Design mistakes become cheaper to fix
- Iterative and incremental design
- Don't tell the boss
- Two hats metaphor

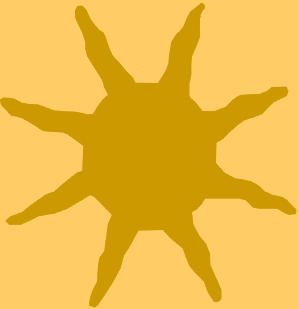


## *Tools (can help)*

---



- Identify bad smells
- Transform
- Verify behavior preservation





## *Tools (existing + in research)*

---



- Smalltalk's RefactoringBrowser
- Java (IntelliJ, XPTTools, Borland, IBM, ...)
- Functional Programming (UKC)



# *The Challenge of an Automatic Tool*

---



- Correct transformations
- Extensibility
- Quicker than manual transformation



## *A definition of Refactoring (revisited)*

---



The process of gradually improving the design of an existing software system by performing source code transformations that improve its quality in such a way that it becomes easier to maintain the system and reuse parts of it, while preserving the behavior of the original system.