

# Ingeniería de Software II (CI-4713)

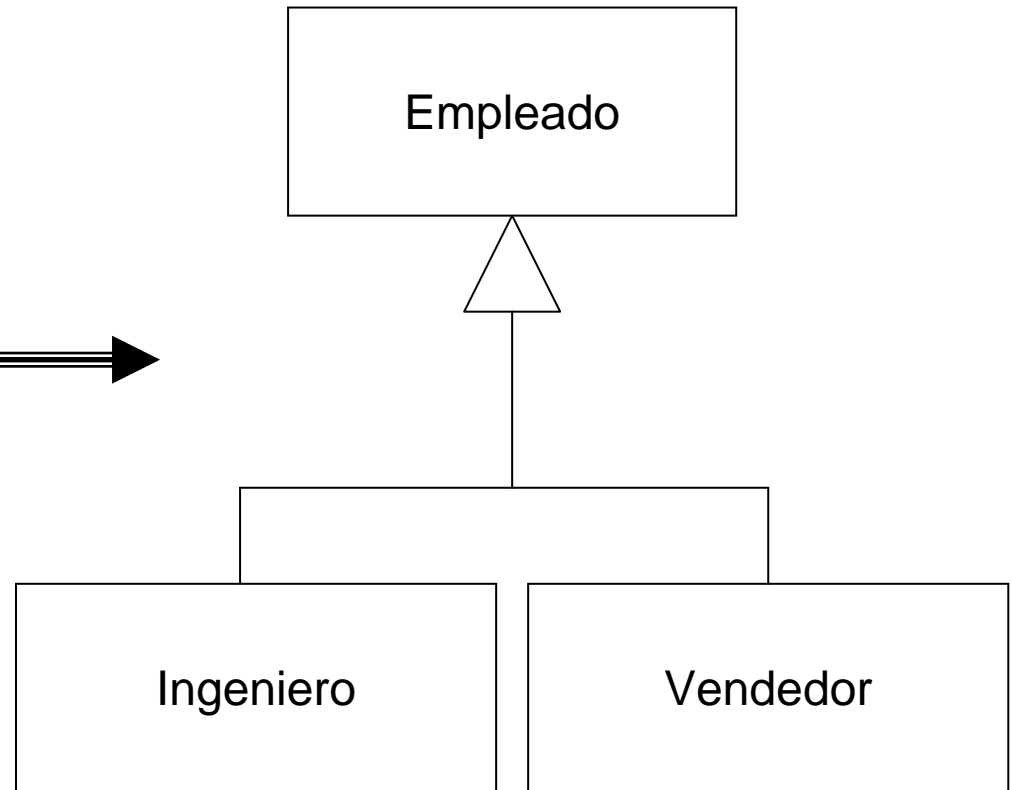
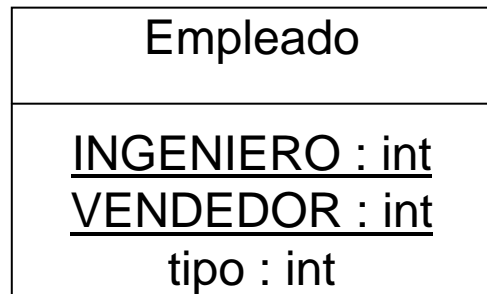
## Refactoring

mayo de 2008

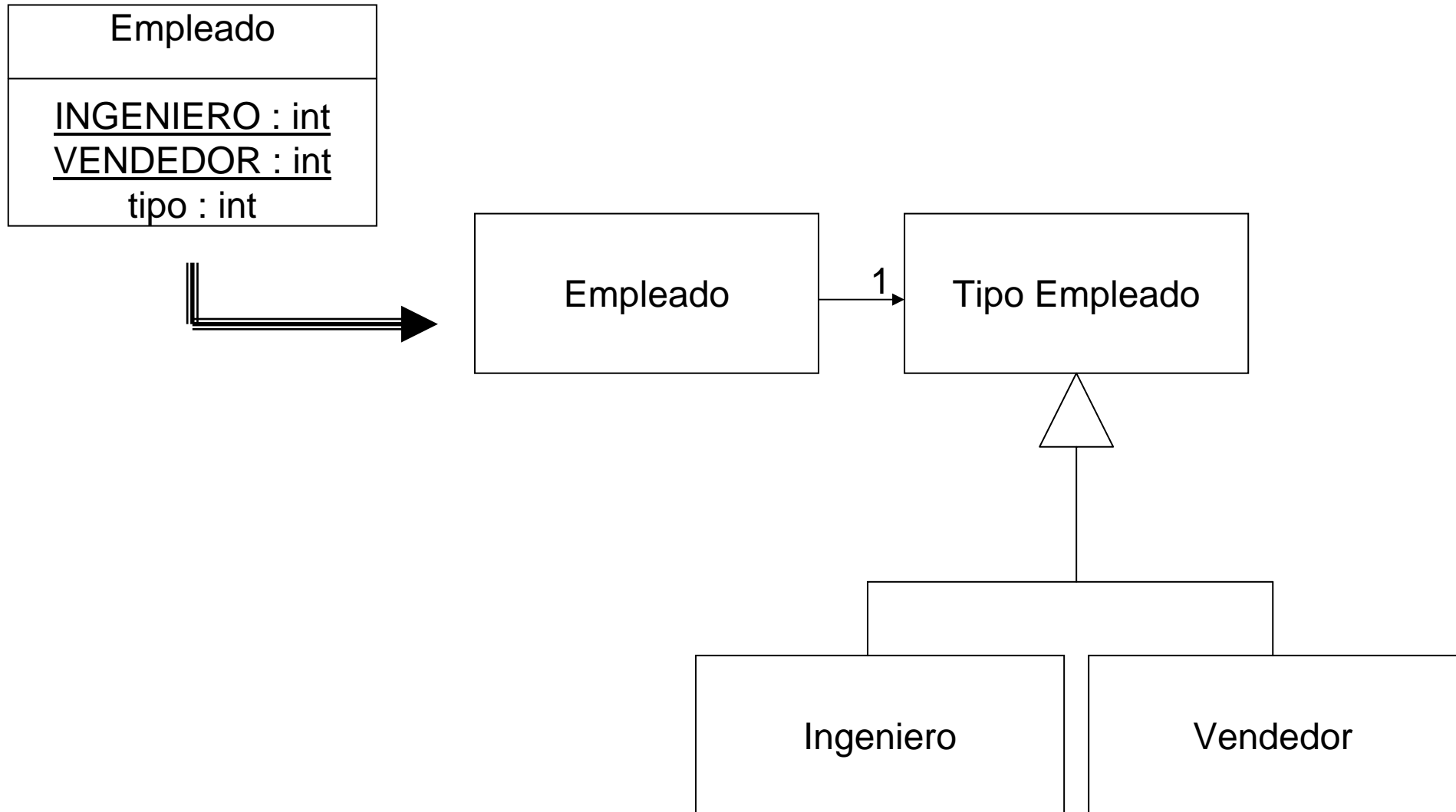
# Agenda

- **Organización de Datos**
  - Reemplazar Tipo con Subclases
  - Reemplazar Tipo con Estado/Estrategia
  - Reemplazar Suclases con Campos
- **Simplificando Condicionales**
- **Generalización**

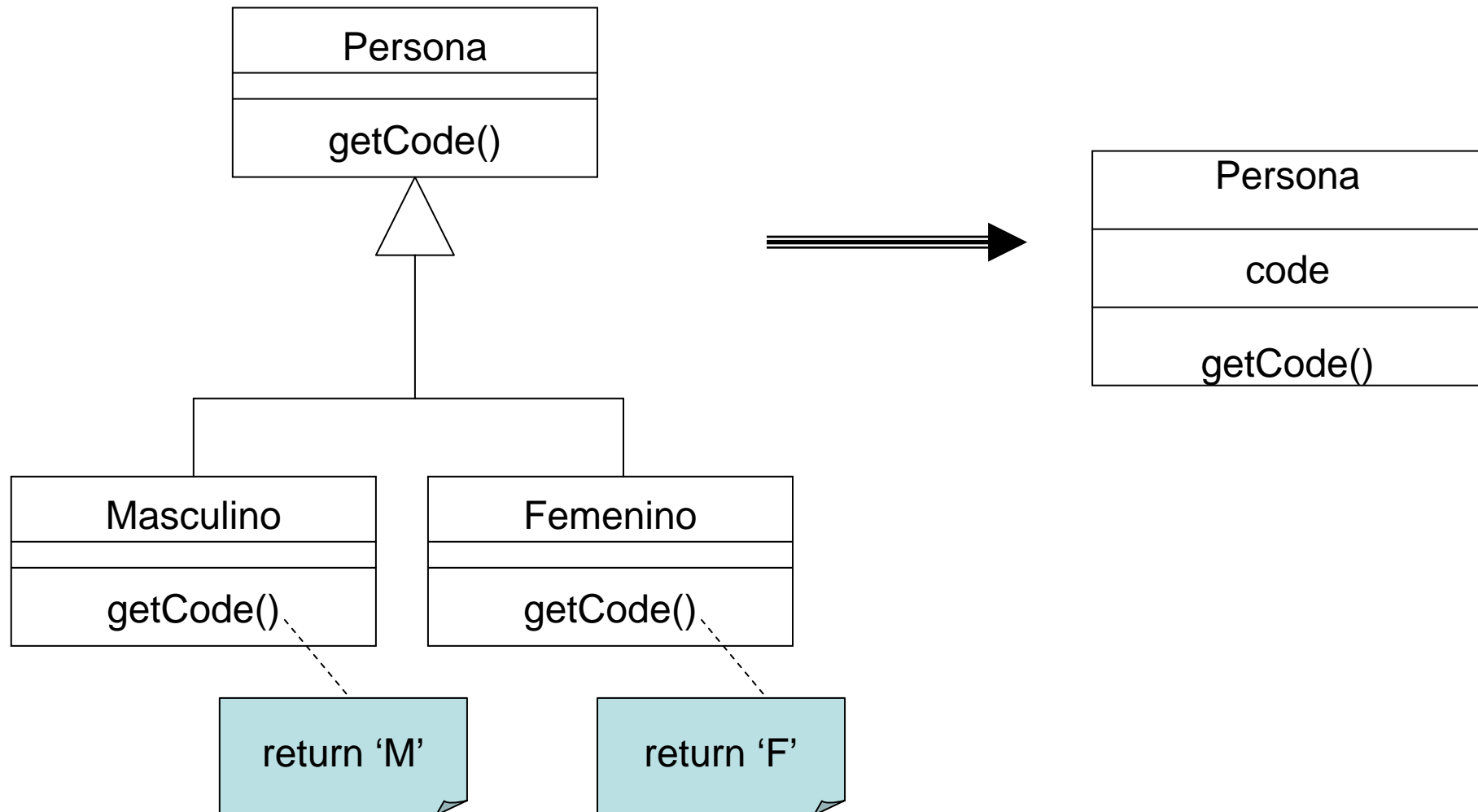
# Replace Type Code with Subclasses



# Replace Type Code with State/Strategy



# Replace Subclass with Fields



# Decompose Conditional

```
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);
```

# Consolidate Conditional Expression

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



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

# Consolidate Duplicate Conditional Fragments

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



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

# Remove Control Flag

**set done to false**

while not done

if **(condition)**

do something

**set done to true**

next step of loop

- Método:
  1. Encontrar el valor del control flag que permite salir del ciclo.
  2. Reemplazar con un “break” o “continue”.
  3. Compilar y probar cada reemplazo.

# Actividad

```
void checkSecurity(String[] people) {  
    boolean found = false;  
    for (int i = 0; i < people.length; i++) {  
        if (! found) {  
            if (people[i].equals ("Don")){  
                showAlert(); found = true;  
            }  
            if (people[i].equals ("John")){  
                showAlert(); found = true;  
            }  
        }  
    }  
}
```

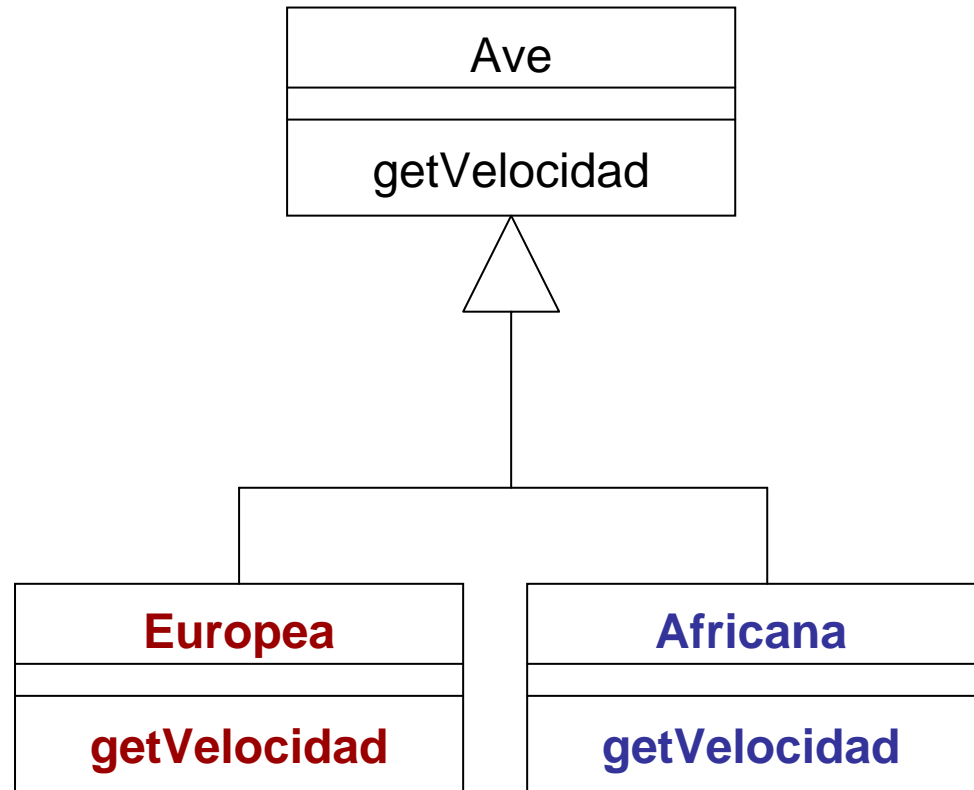
# Replace Nested Conditional with Guard Clauses

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

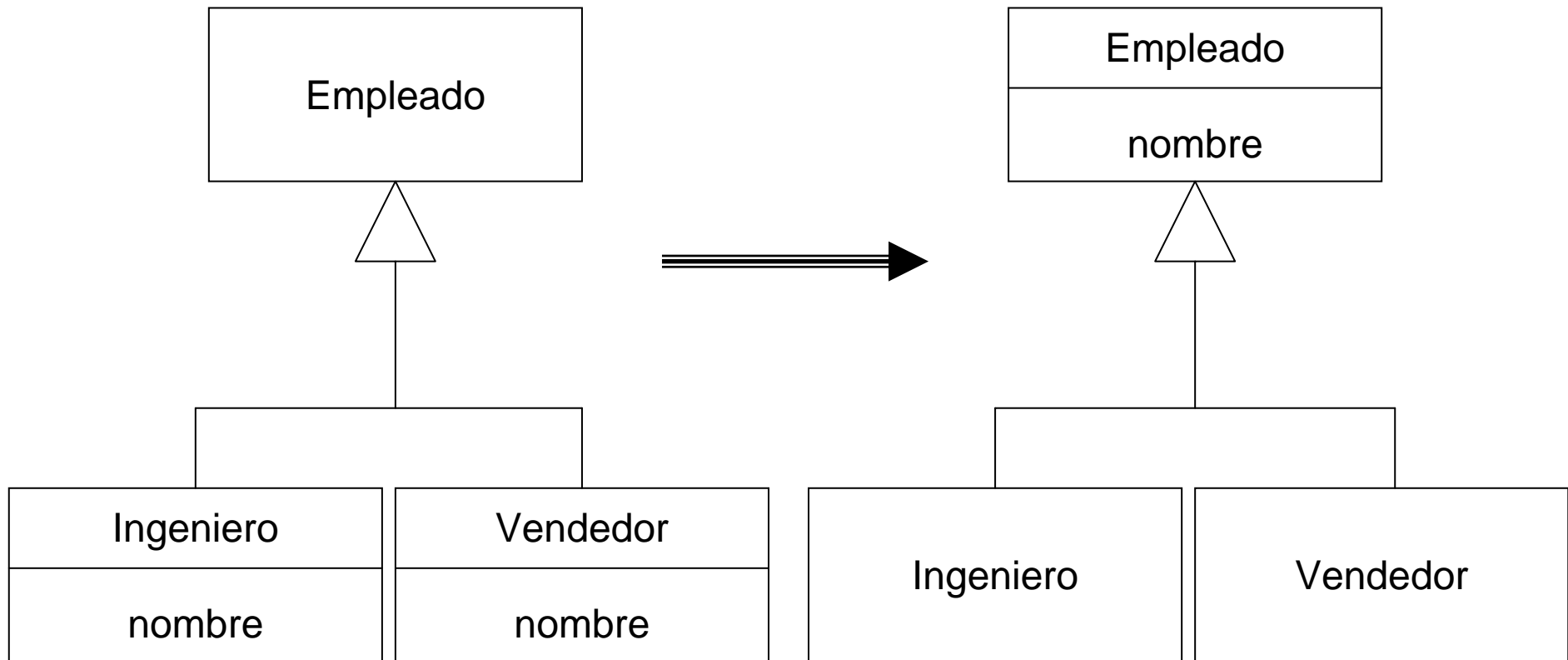
```
double getPayAmount() {
    if (_isDead) return deadAmount();
    if (_isSeparated) return separatedAmount();
    if (_isRetired) return retiredAmount();
    return normalPayAmount();
};
```

# Replace Conditional with Polymorphism

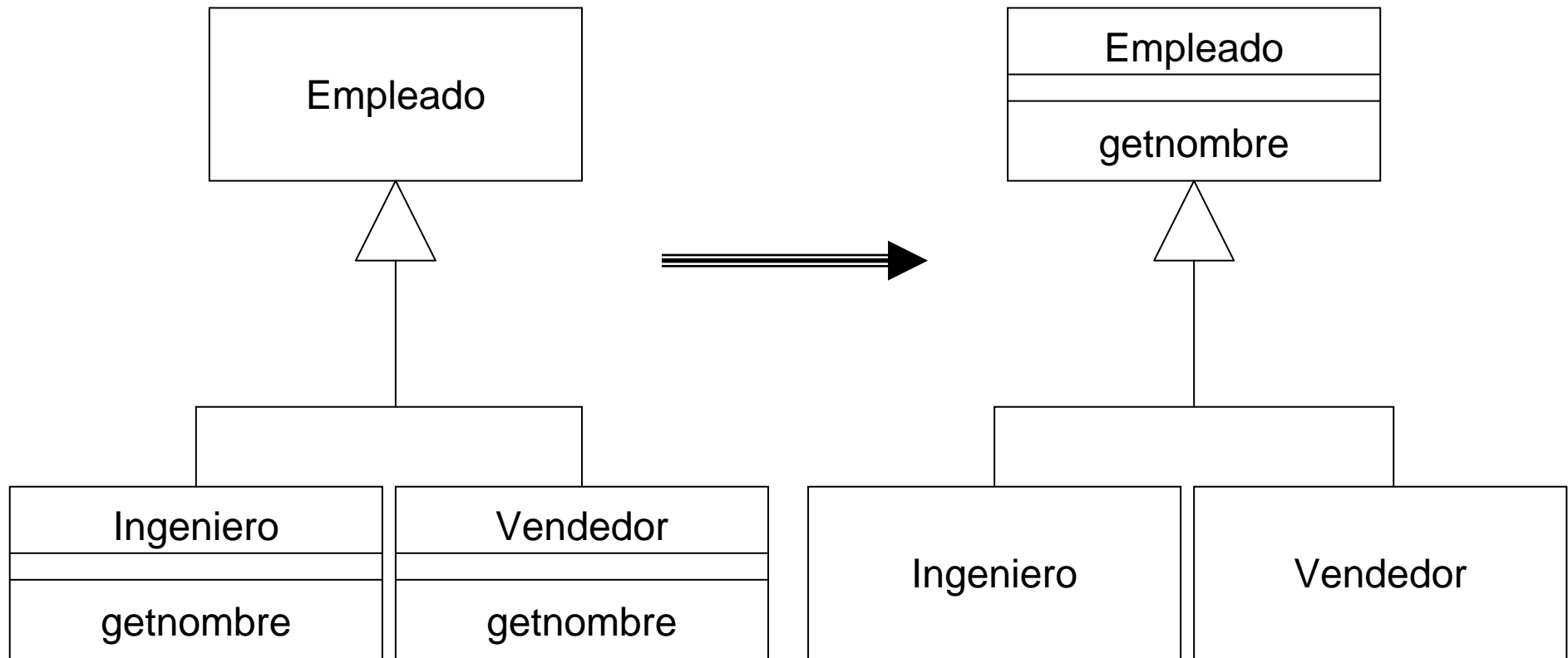
```
double getVelocidad() {  
    switch (_tipo) {  
        case EUROPEA:  
            return  
            getVelocidadBase();  
        case AFRICANA:  
            return  
            getVelocidadBase() -  
            getLoadFactor() *  
            _numberOfCoconuts;  
    }  
};
```



# Pull Up Field



# Pull Up Method



# Pull Up Constructor Body

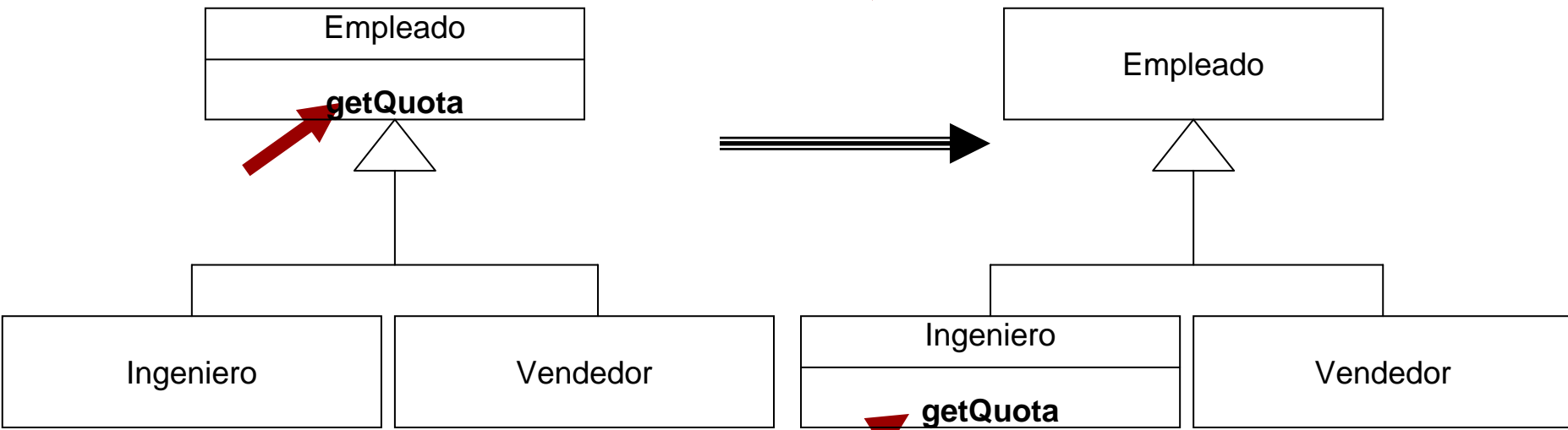
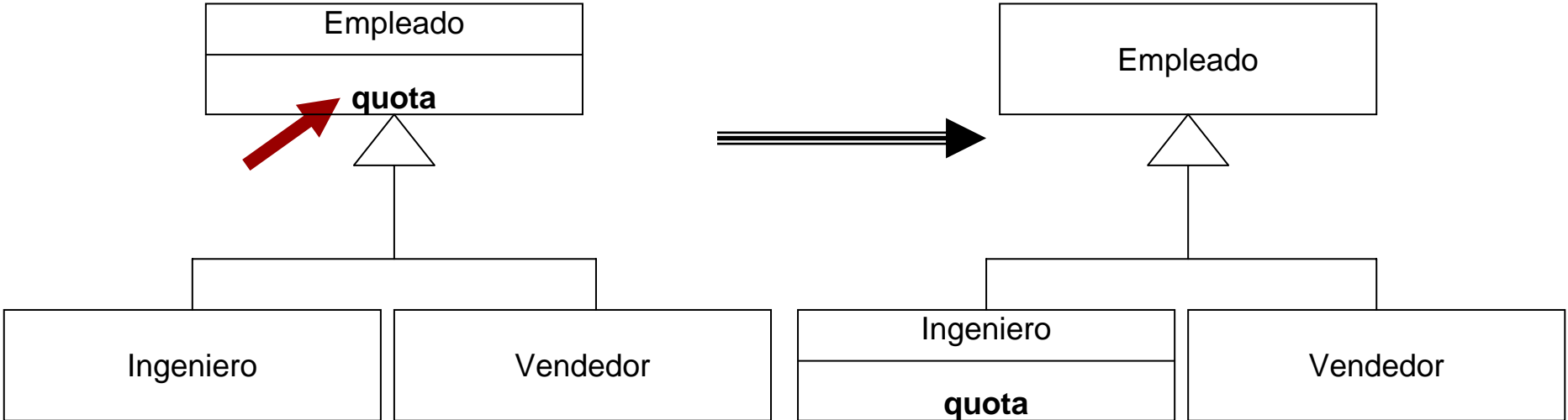
class Manager extends Employee...

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

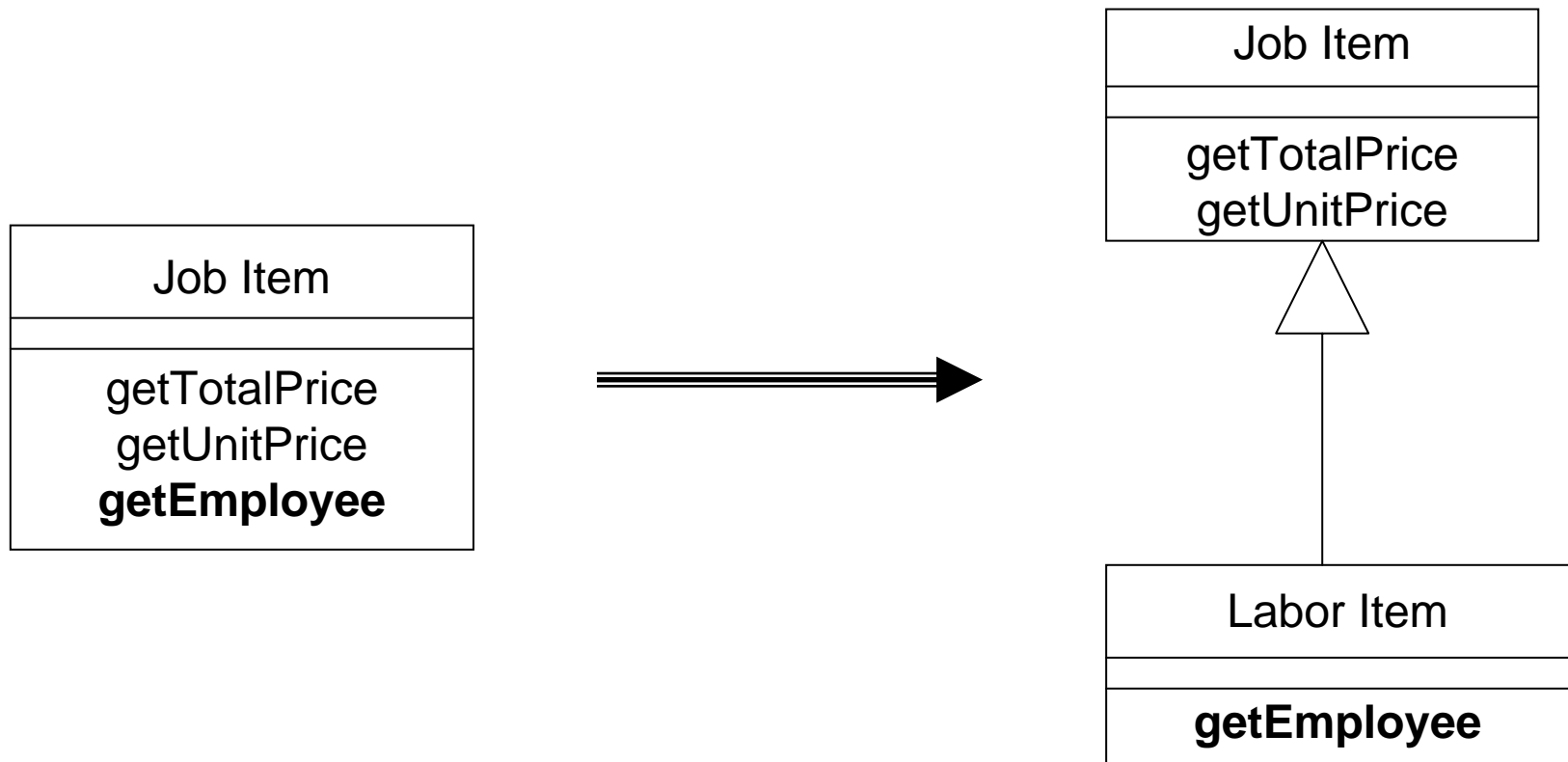


```
public Manager (String name, String id, int grade) {  
    super (name, id);  
    _grade = grade;  
}
```

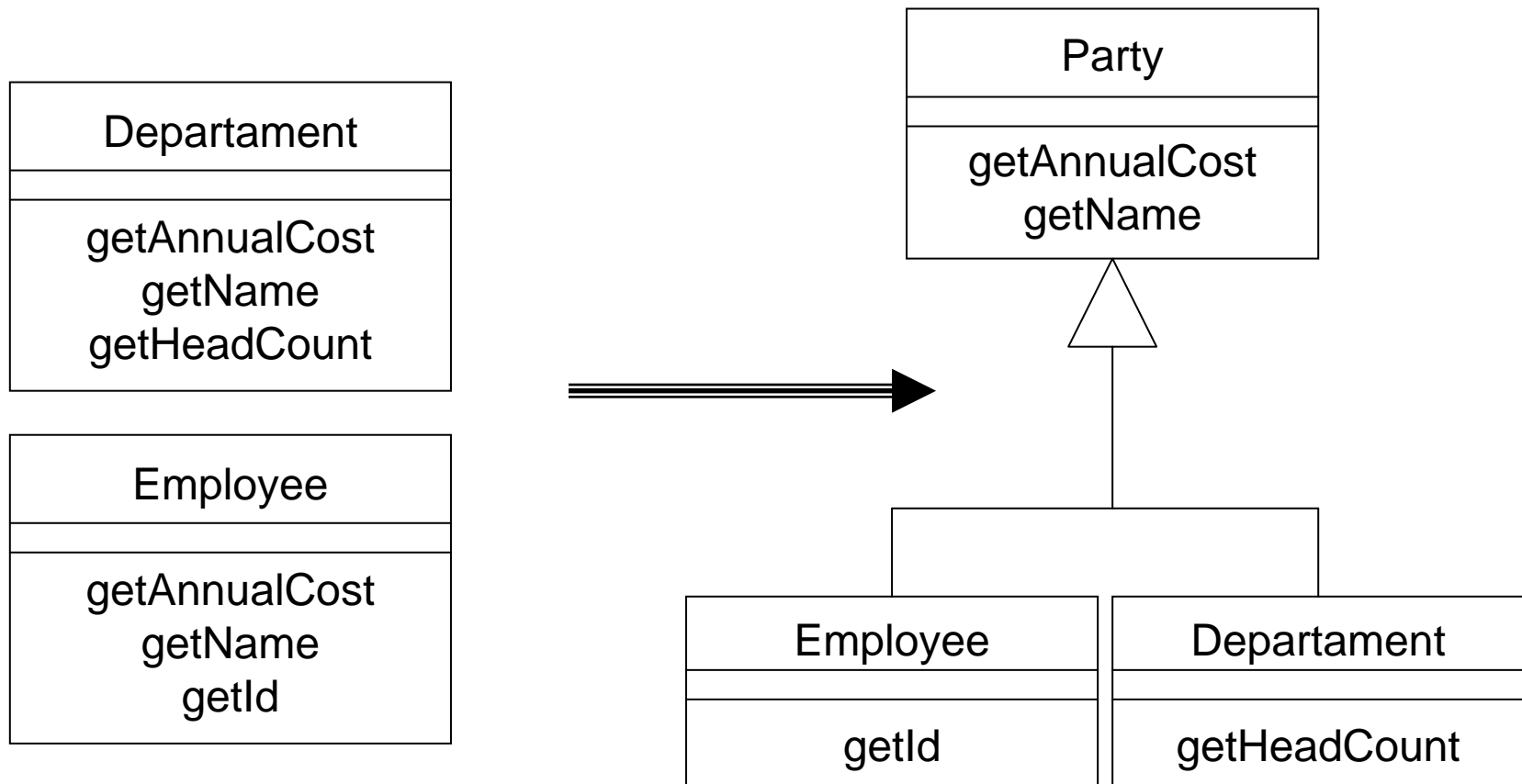
# Pull Down Method/Field



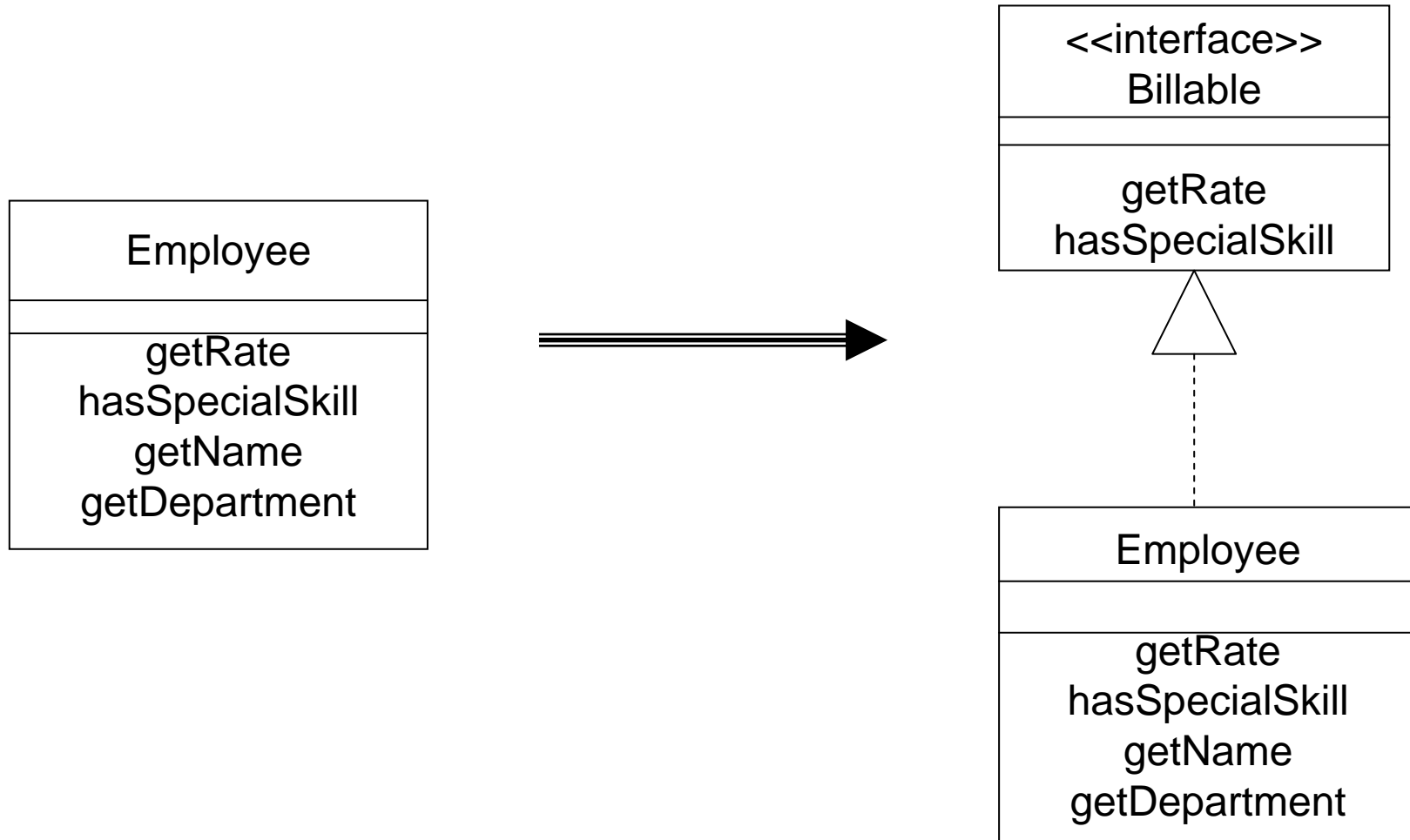
# Extract SubClass



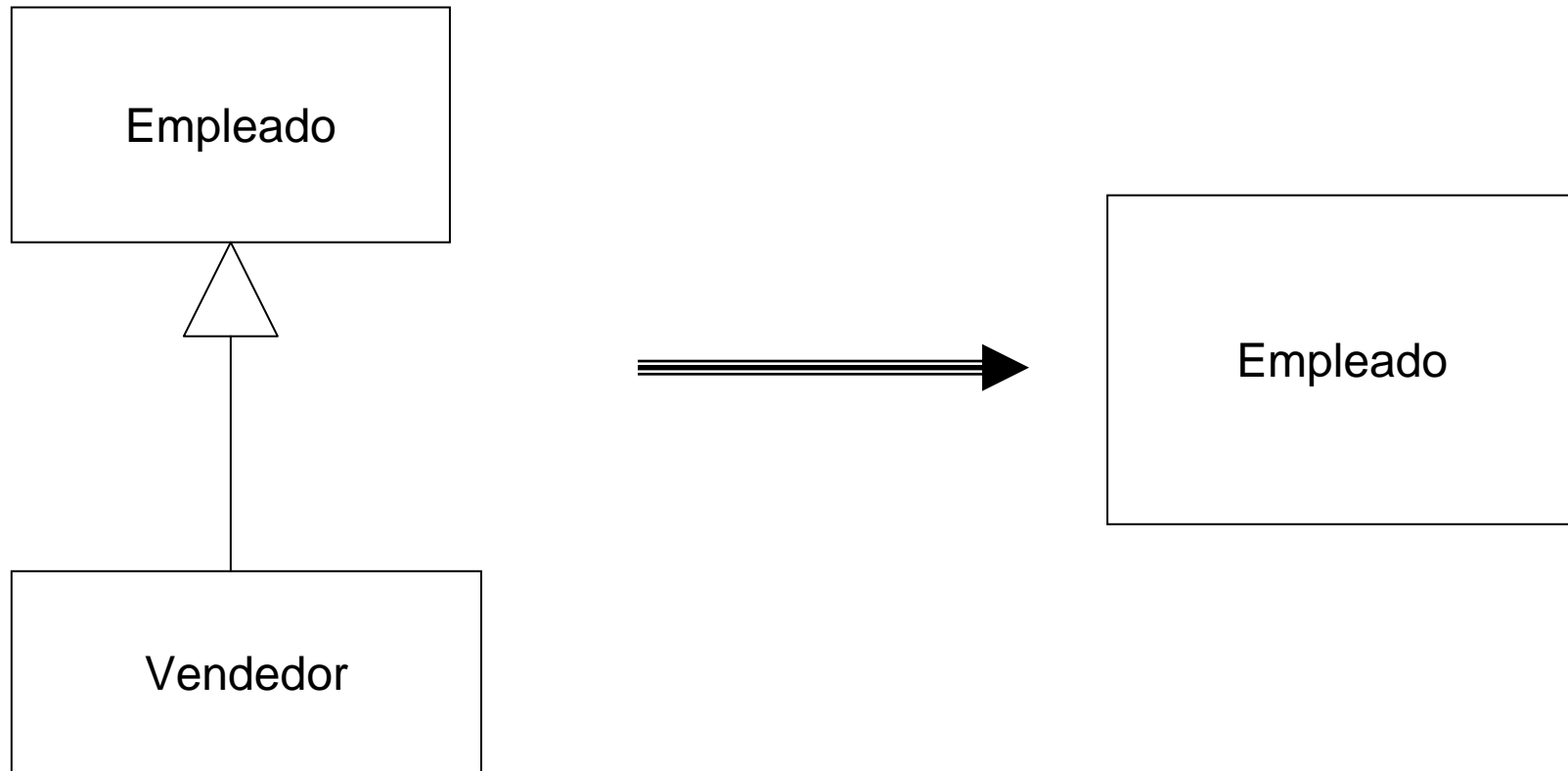
# Extract SuperClass



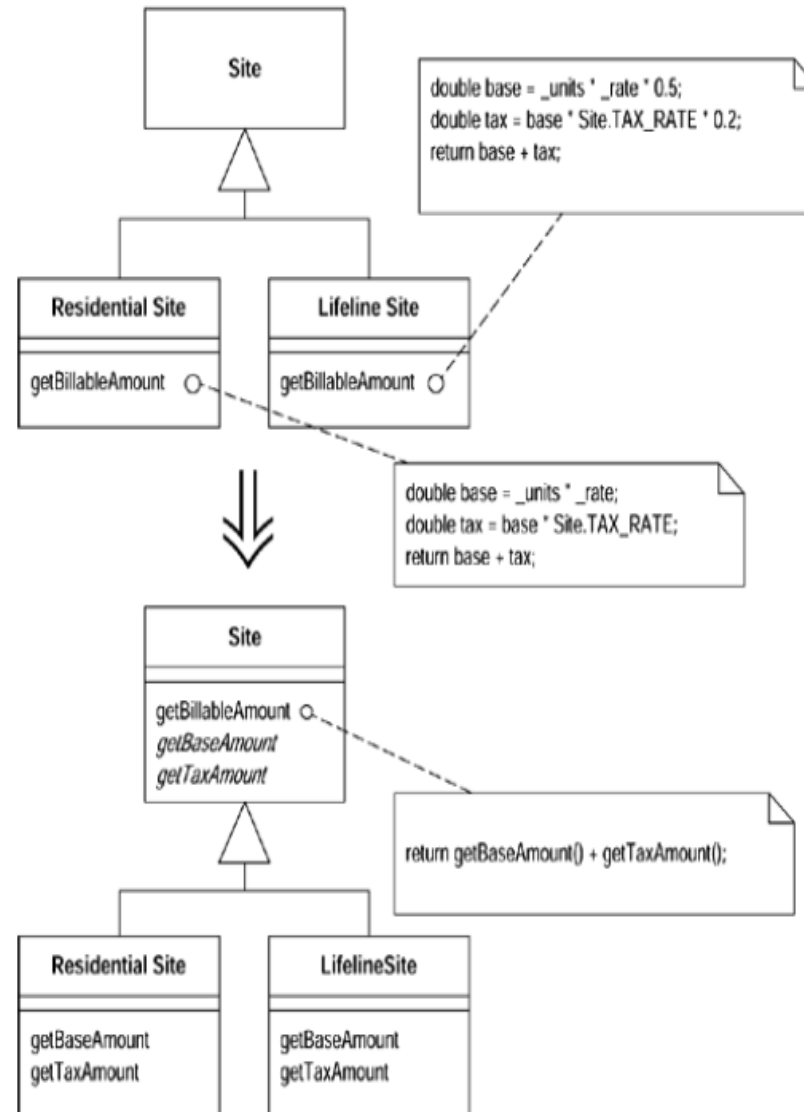
# Extract Interface



# Collapse Hierarchy



# Form Template Method



# Referencias

- **Refactoring. Improving the design of existing code. Fowler, M. Addison Wesley 2000**