Refactoring

Chapter 9

Simplifying Conditional Expressions

Decompose Conditional

- Read this

```
if (date.before (SUMMER_START) || date.after(SUMMER_END))
    charge = quantity * _winterRate + _winterServiceCharge;
else charge = quantity * _summerRate;
```

- and then read this

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

Consolidate Conditional Expression

- Before:

```
double disabilityAmount() {
        if (_seniority < 2) return 0;
        if (_monthsDisabled > 12) return 0;
        if (_isPartTime) return 0;

        // ...
}
```

- After:

```
double disabilityAmount() {
        if (isNotEligableForDisability()) return 0;

        // ...
}
```

Consolidate Duplicate Conditional Fragments

- Before

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

Consolidate Duplicate Conditional Fragments

- After

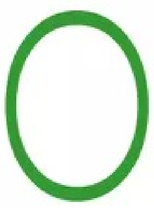
```
doNonRelatedWork();

doSomePreparation();

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

send();
doSomeCleanUp();
```

Remove Control Flag

- Trace this

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

Remove Control Flag



use **`return`**, **`break`**, **`continue`** ...

set a flag and check *somewhere* later

Remove Control Flag

- Alternative 1:

```
void checkSecurity(String[] people) {
    for (int i = 0; i < people.length; i++) {
        if (people[i].equals ("Don")){
            sendAlert();
            break;
        }
        if (people[i].equals ("John")){
            sendAlert();
            break;
        }
    }
}
```

Remove Control Flag

- Alternative 2:

```
void checkSecurity(String[] people) {
     String found = foundMiscreant(people);
     someLaterCode(found);
}
String foundMiscreant(String[] people) {
     for (int i = 0; i < people.length; i++) {
          if (people[i].equals ("Don")){
               sendAlert();
               return "Don";
          }
          if (people[i].equals ("John")){
               sendAlert();
               return "John";
          }
     }
     return "";
}
```

Replace Nested Conditional with Guard Clauses

- Before

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

Replace Nested Conditional with Guard Clauses

- After

```
double getPayAmount() {
    if (_isDead)
        return deadAmount();

    if (_isSeparated)
        return separatedAmount();

    if (_isRetired)
        return retiredAmount();

    return normalPayAmount();
};
```

Replace Nested Conditional with Guard Clauses

Mind implicit semantics

- Guard clause

  → (somehow) exceptional condition

- `if else`

  → two equivalent normal conditions

Replace Nested Conditional with Guard Clauses

- However, you might have seen this ...

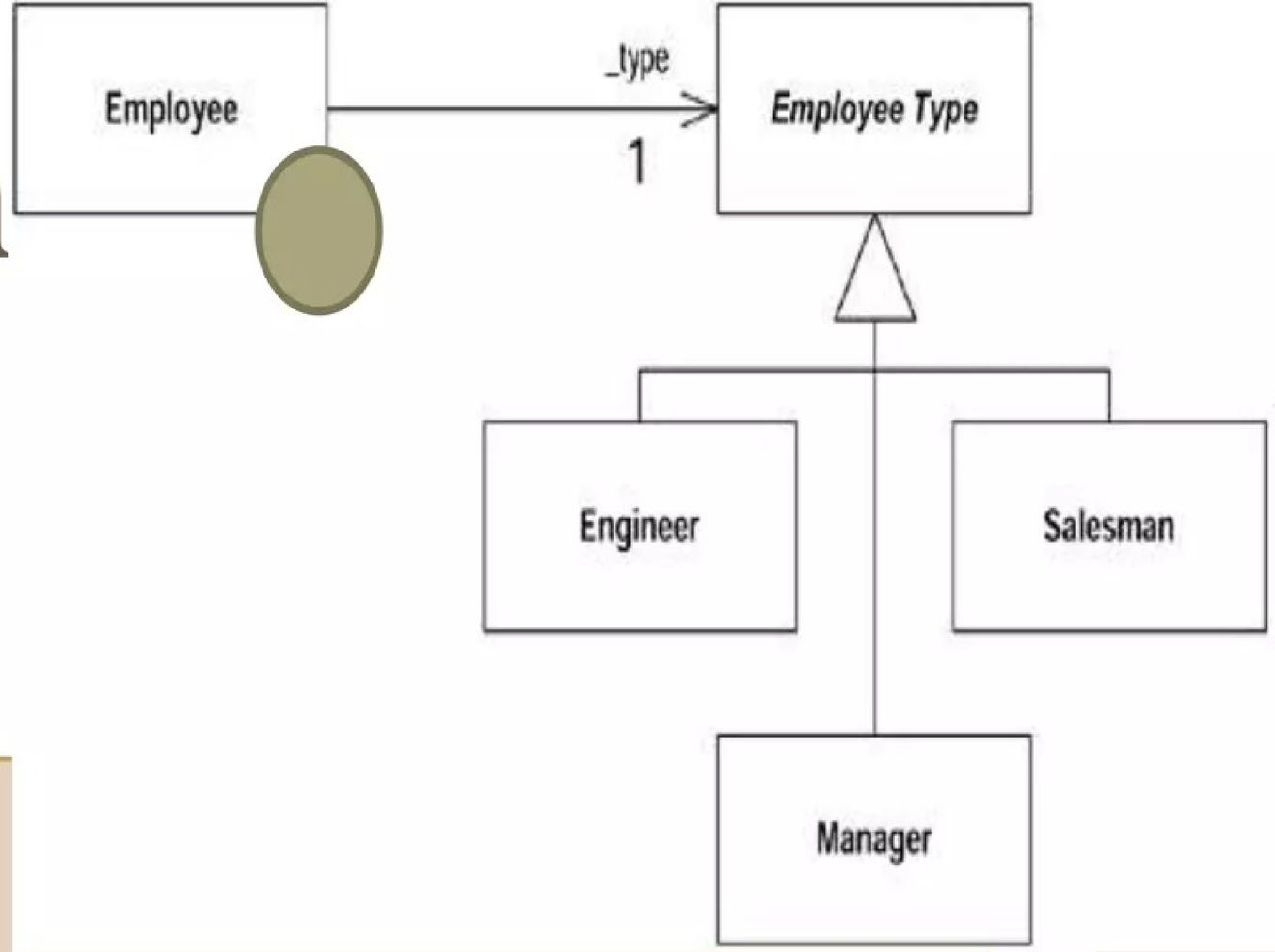
```
if (aquireResource1()) {
    if (aquireResource2()) {
        if (aquireResource3()) {
            if (aquireResource4()) {
                doRealWork();
                releaseResource4();
            }
            releaseResource3();
        }
        releaseResource2();
    }
    releaseResource1();
}
```

Replace Nested Conditional with Guard Clauses

- or this ...

```
if (false == aquireResource1())
      return;
if (false == aquireResource2()) {
      releaseResource1();
      return;
}
if (false == aquireResource3()) {
      releaseResource2();
      releaseResource1();
      return;
}
doRealWork();
releaseResource3();
releaseResource2();
releaseResource1();
return;
```

Replace Condition Polymorphism



- Before

```
class Employee...
    int payAmount(Employee) {
        switch (getType()) {
            case EmployeeType.ENGINEER:
               return _monthlySalary;
            case EmployeeType.SALESMAN:
               return _monthlySalary + _commission;
            case EmployeeType.MANAGER:
               return _monthlySalary + _bonus;
            default:
               throw new RuntimeException("Incorrect Employee");
        }
    }
```

Replace Conditional with Polymorphism

!

- Polymorphism might be better, usually

Replace Conditional with Polymorphism

- After

```
class Employee...
    int payAmount() {
        return _type.payAmount(this);
    }
```

```
class Salesman...
    int payAmount(Employee emp) {
        return emp.getMonthlySalary() + emp.getCommission();
    }
class Manager...
    int payAmount(Employee emp) {
        return emp.getMonthlySalary() + emp.getBonus();
    }
```

Introduce Null Object

- Same with *Replace Conditional with Polymorphism*
  - deal with special cases extensively checked
  - check type → check NULL

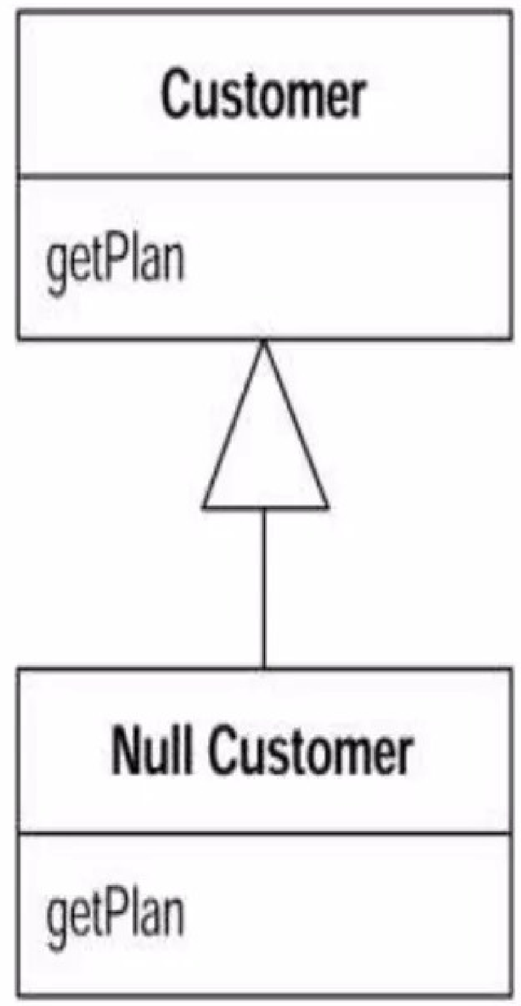
- Before:

```
class Customer...
    public String getName() {...}
    public BillingPlan getPlan() {...}
    public PaymentHistory getHistory() {...}

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

Introduce Null Object

- After

```
plan = customer.getPlan();



class NullCustomer...
    public BillingPlan getPlan(){
        return BillingPlan.basic();
    }
```



Introduce Assertion

Write your assumption explicitly and clearly

```
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 Assertion

- Do NOT overuse
- If code does work without the assertion, remove it.

Notes

- Save brain power
  - Break/prune eye-tracing as early as possible

- ***D***on't ***R***epeat ***Y***ourself