MAKE SCHOOL

Refactoring

simplifying conditional expressions

SPD 2.31

Warm up

Please improve this code snippet to make it easier to understand.

10 mins

```
# Adapted from a Java code in the "Refactoring" book by Martin Fowler.

def send_alert():
  print("alert!")

def check_security(people):
  found = False
  for p in people:
      if not found:
          if p == "Naruto":
              send_alert()
              found = True

          if p == "Goku":
              send_alert()
              found = True


check_security(["Kami", "Naruto"])
```

Learning Outcomes

By the end of today, you should be able to…

1. Compare and contrast different refactoring techniques for simplifying conditional expressions
2. Identify code smells and apply refactoring techniques to improve code quality.

Decompose Conditional

```
if date.before(SUMMER_START) or date.after(SUMMER_END):
  charge = quantity * winter_rate + winter_service_charge
else:
   charge = quantity * summer_rate
```

Decompose Conditional (interim)

```
# Adapted from a Java code in the "Refactoring" book by Martin Fowler.
# Decompose conditional.
# Code snippet. Non-runnable code.

def not_summer(date):
   return date.before(SUMMER_START) or date.after(SUMMER_END)
def winter_charge(quantity):
   return quantity * winter_rate  + winter_service_charge
def summer_charge(quantity):
   return quantity * summer_charge



if (not_summer(date)):
   charge = winter_charge(quantity)
else:
   charge = summer_charge(quantity)
```

Decompose Conditional (refactored)

```
# Adapted from a Java code in the "Refactoring" book by Martin Fowler.
# Decompose conditional.
# Code snippet. Non-runnable code.

def summer(date):
   return not (date.before(SUMMER_START) or date.after(SUMMER_END))
def winter_charge(quantity):
   return quantity * winter_rate  + winter_service_charge
def summer_charge(quantity):
   return quantity * summer_charge



if (summer(date)):
   charge = summer_charge(quantity)
else:
   charge = winter_charge(quantity)
```

Decompose Conditional

You have a complicated conditional (if-then-else) statement → Extract methods from the condition, then part, and else parts.

Decompose Conditional

12 mins

Solve Exercise 12: 'Decompose Conditional' Technique

Consolidate Conditional Expression

Consolidate Conditional Expression

```python
# Adapted from a Java code in the "Refactoring" book by Martin Fowler.
# Code snippet. Non-runnable code.
def disability_amount():
   if seniority < 2:
       return 0
   if months_disabled > 12:
       return 0
   if is_part_time:
       return 0
   # ...Compute the disability amount
```

Consolidate Conditional Expression (refactored)

```
# Adapted from a Java code in the "Refactoring" book by Martin Fowler.
# Code snippet. Non-runnable code.
# Refactored.
def is_not_eligable_for_disability():
   return (seniority < 2 or months_disabled > 12 or is_part_time)

def disability_amount():
   if is_not_eligable_for_disability():
       return 0
   # Compute the disability amount
```

Consolidate Conditional Expression

10 min

Solve Exercise 14: 'Consolidate Conditional Expression' Technique

Consolidate duplicate conditional fragments

```
# Adapted from a Java code in the "Refactoring" book by Martin Fowler.
# Consolidate duplicate conditional fragments
# Code snippet. Not runnable.

if (is_special_deal()):
   total = price * 0.95
   send()
else:
   total = price * 0.98
   send()
```

```
# Adapted from a Java code in the "Refactoring" book by Martin Fowler.
# Consolidate duplicate conditional fragments
# Code snippet. Not runnable.

if (is_special_deal()):
   total = price * 0.95
else:
   total = price * 0.98

send()
```

Consolidate duplicate conditional fragments

The same fragment of code is in all branches of a conditional expression →

Move it outside of the expression.

Consolidate Duplicate Conditional Fragments

10 min

Solve Exercise 13: 'Consolidate Duplicate Conditional Fragments' Technique.

Break for 10 mins

Get up, stretch, get some water, and relax your mind. ☁️

Remove Control Flag

Remove Control Flag

```
# Adapted from a Java code in the "Refactoring" book by Martin Fowler.
# Remove control flag.
def send_alert():
   print("alert!")

def check_security(people):
   found = False
   for p in people:
       if not found:
           if p == "Naruto":
               send_alert()
               found = True

           if p == "Goku":
               send_alert()
               found = True


check_security(["Kami", "Naruto"])
```

Remove Control Flag (refactored)

```
# Adapted from a Java code in the "Refactoring" book by Martin Fowler.
# Refactored.
def send_alert():
  print("alert!")

def check_security(people):
  for p in people:
        if p == "Naruto":
           send_alert()
           return
        if p == "Goku":
           send_alert()
           return


check_security(["Kami", "Naruto"])
```

Remove Control Flag

You have a variable that is acting as a control flag for a series of boolean expressions → use a ***break*** or ***return*** instead.

Remove Control Flag

10 mins

Solve Exercise 15: 'Remove Control Flag' Technique

MAKE SCHOOL

Remove Nested Conditional with Guard Clauses

Remove Nested Conditional with Guard Clauses

```python
# Adapted from a Java code in the "Refactoring" book by Martin Fowler.
# Code snippet. Not runnable.
def getPayAmount():
   result = 0
   if is_dead:
       result = dead_amount()
   else:
       if is_separated:
           result = separated_amount()
       else:
           if is_retired:
               result = retired_amount()
           else:
               result = normal_pay_amount()

   return result
```

Remove Nested Conditional with Guard Clauses

```python
# Adapted from a Java code in the "Refactoring" book by Martin Fowler.
# Code snippet. Not runnable.
# Refactored.
def get_pay_amount():
  if is_dead:
      return dead_amount()
  if is_separated:
      return separated_amount()
  if is_retired:
      return retired_amount()

  return normal_pay_amount()
```

Remove Nested Conditional with Guard Clauses

A method has conditional behaviour that does not make clear the normal path of execution. → **use guard clauses for all the special cases.**

Remove Nested Conditional with Guard Clauses

10 mins

Solve Exercise 16: 'Replace Nested Conditional with Gaurd Clauses' Technique

Summary

1. Decompose Conditional
2. Consolidate Conditional Expression
3. Consolidate Duplicate Conditional Fragments
4. Remove Control Flag
5. Remove Nested Conditional with Guard Clauses

References and Further Study

MAKE SCHOOL

1. "Refactoring: Improving the Design of Existing Code" (1st edition) by Martin Fowler
2. https://en.wikipedia.org/wiki/Code_refactoring
3. https://myanimelist.net/featured/773/Naruto_Hand_Signs_and_Jutsu