Quiz 4 - Pre Lecture Review Quiz (/tests/5)

Question 1

Which of the following is not a suitable recommendation of when to refactor?

Option 1: When you add functionality

Option 2: When you code review

Option 3: When you repeat thesame thing 3 or more times

Option 4: Only after your software is compiled

Edit Question

Question 2

Which of the following is NOT a code smell?

Option 1: Long method

Option 2: Duplicate code

Option 3: Extract method

Option 4: case statements

Edit Question

Question 3

Which of the following is NOT a refactoring?

Option 1: Extract method

Option 2: Pull up method

Option 3: Duplicate Code

Option 4: Move method

Question 1

Which of the following is not a suitable recommendation of when to refactor?

Option 1: When you add functionality

Option 2: When you code review

Option 3: When you repeat thesame thing 3 or more times

Option 4: Only after your software is compiled

Answer: Only after your software is compiled

Edit Question

Question 2

Which of the following is NOT a code smell?

Option 1: Long method

Option 2: Duplicate code

Option 3: Extract method

Option 4: case statements

Answer: Extract method

Edit Question

Question 3

Which of the following is NOT a refactoring?

Option 1: Extract method

Option 2: Pull up method

Option 3: Duplicate Code

Option 4: Move method

Answer: Duplicate Code

Edit Question

Question 4

On the right, we have a refactored version of the code segment to the left. Please fill in the blank using the drop down options.

A: ▼

B: ▼

C: ▼

D: ▼

```
def print_owing
   outstanding = 0.0

      # print banner
      puts "*************************"
      puts "**Customer Owes ***"
      puts "*************************"

      #calculate outstanding
      @orders.each do |order|
     outstanding += order.amount
      end

   #print details
   puts("name: #{@name})
   puts("amount: #{outstanding})

   puts "*************************"
      puts "**Customer Owes ***"
      puts "*************************"
end
```

```
def print_owing
   printBanner
      outstanding = A
   B
   printBanner
end
def get_outstanding          #calculate outstanding
      outstanding = 0.0
      @orders.each do |order|
     outstanding += order.amount
      end
   C
end

D                            #print details
   puts("name: #{_name})
   puts("amount: #{amount})
end

def print_banner                    # print banner
      puts "*************************"
      puts "**Customer Owes ***"
      puts "*************************"
end
```

Answer:

A: get_outstanding

B: print_details(@name, outstanding)

C: return outstanding

D: def print_details(_name, amount)

Question 5

Edit Question

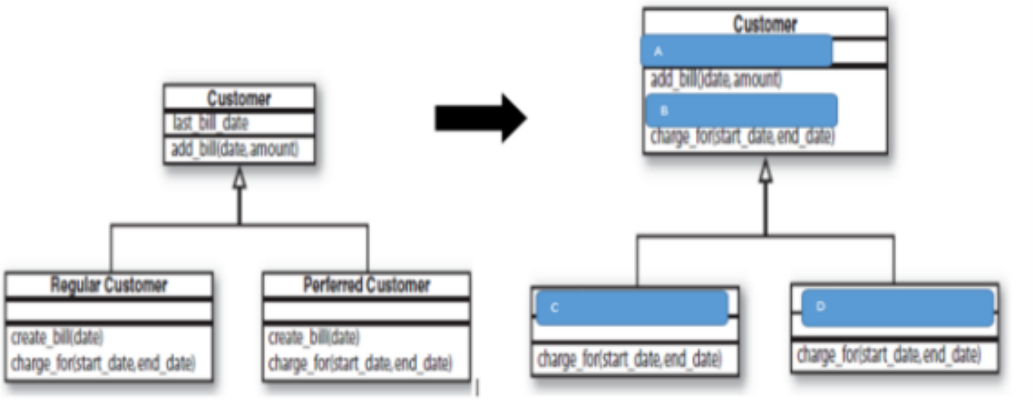
After applying "pull up method" to the diagram on the left, we end up with the diagram on the right. Fill in the missing pieces.

A: ▼

B: ▼

C: ▼

D: ▼



Answer:

A: last_bill_date

B: create_bill(date)

C: Regular Customer

D: Preferred Customer

Edit Question

Quiz 5 - Move Method (/tests/6)

Create Descriptive Question | Create Multi Fill Question | Create Multiple Choice Question | Create Parsons Problem Question

---

Question: Given the code on the board, arrange your code blocks so that it would reflect move method being applied to the "get_formatted" method.

```ruby
class Phone

    attr_reader :unformattedNumber
    def initialize(unformattedNumber)
        @unformatedNumber = unformattedNumber
    end

    def get_area_code
        @unformatedNumber[0,3]
    end

    def get_middle_three
        @unformatedNumber[3,3]
    end

    def get_last_four
        @unformatedNumber[6,4]
    end
end


class Customers
    attr_reader :customerinfo

    def get_formatted
        @mobile = Phone.new("6159831279")
        "("+ @mobile.get_area_code + ") " + @mobile.get_middle_three + "-" + @mobile.get_last_four
    end
end

Test = Customers.new
puts Test.get_formatted
```

Drag from here

```
def get_last_four
  @unformatedNumber[6,4]
end
```

```
def get_middle_three
  @unformatedNumber[3,3]
end
```

```
def get_area_code
```

```
class Customers
  attr_reader :customerinfo
@mobile = Phone.new("6159831279")
puts @mobile.get_formatted
end
```

```
end
```

```
class Phone
```

```
end
```

```
Test = Customers.new
Test
```

```
attr_reader :unformattedNumber
def initialize(unformattedNumber)
  @unformatedNumber = unformattedNumber
end
```

```
"("+ get_area_code + ") " + get_middle_three + "-" + get_last_four
```

```
@unformatedNumber[0,3]
```

```
def get_formatted
```

```
end
```

```
"("+ @mobile.get_area_code + ") " + @mobile.get_middle_three + "-" + @mobile.get_last_four
```

Construct your solution here

Reset Get feedback

Answer:

```ruby
class Phone

    attr_reader :unformattedNumber
    def initialize(unformattedNumber)
        @unformatedNumber = unformattedNumber
    end

    def get_area_code
        @unformatedNumber[0,3]
    end

    def get_middle_three
        @unformatedNumber[3,3]
    end

    def get_last_four
        @unformatedNumber[6,4]
    end
    def get_formatted
        "("+ get_area_code + ") " + get_middle_three + "-" + get_last_four
    end
end


class Customers
    attr_reader :customerinfo
    @mobile = Phone.new("6159831279")
    puts @mobile.get_formatted
end

Test = Customers.new
Test
```

Edit Question

Question 2

Which of the following corresponds to a bad smell that invokes too many getters from another class?

Option 1: Shotgun surgery

Option 2: Feature Envy

Option 3: Duplicate Code

Option 4: Long method

Answer: Feature Envy

Edit Question

Question 3

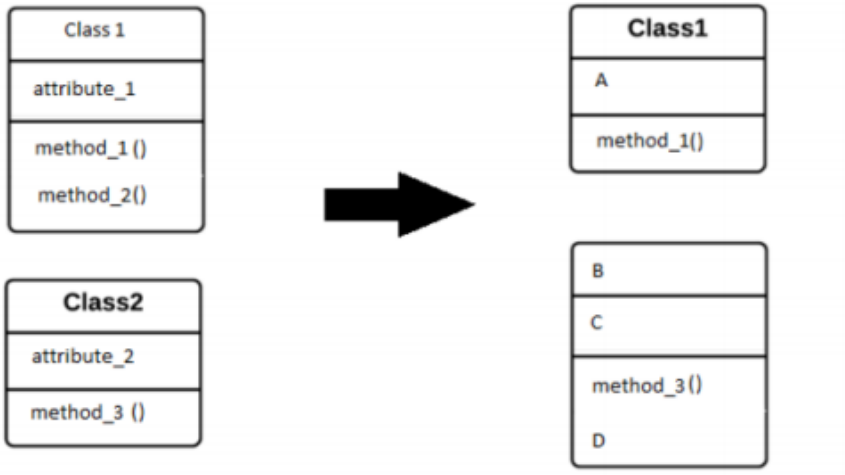
Select a solution that best illustrates move method being applied to the structure on the left.

A:

B:

C:

D:



Answer:

A: attribute_1

B: Class2

C: attribute_2

D: method_2()

Edit Question