REFATORAÇÃO E QUALIDADE DE CÓDIGO

OI!

Me chamo Elma Santos

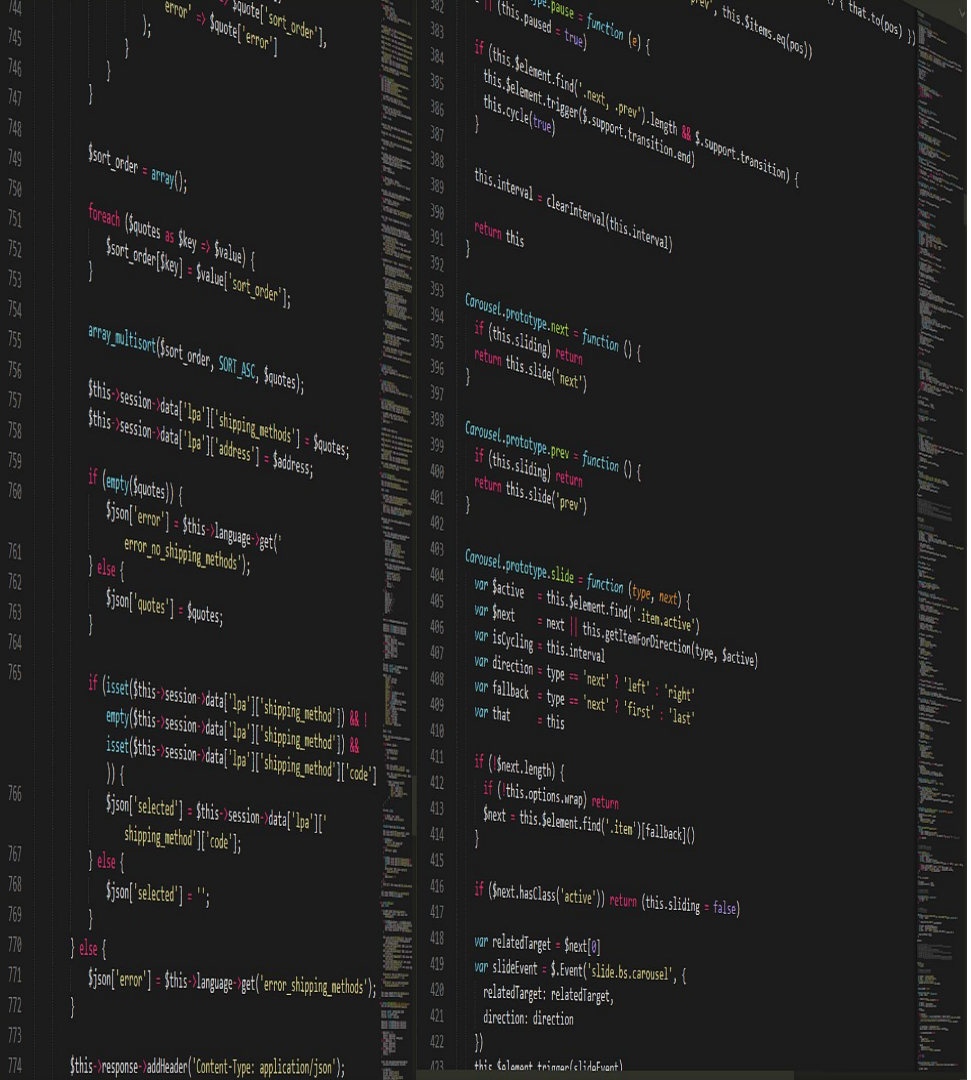
Técnica em informática (**IFRN**). Graduanda em tecnologia da informação (**UFRN**)
Desenvolvedora de Software na **Codeminer 42**
Participo: **COMTI** e **WTM NATAL**
Startup **MANAS**

@elmasantos

@elmasnts

“

É o processo de alterar o código fonte de uma maneira que não altere seu comportamento externo e ainda melhore a sua estrutura interna. É uma técnica disciplinada de limpar e organizar o código, e por consequência minimizar a chance de introduzir novos bugs
– Martin Fowler.



Por que refatorar?

- Melhora a qualidade do código
  - Mais fácil de ler
  - Mais fácil de entender
  - Mais fácil de manter
- Bugs e linhas de código desnecessárias são mais fáceis de ser encontrados
- Facilita a implementação de testes
- Essencial para desenvolvimento em times

{ Refatorando }

Técnicas a serem seguidas para escrever código de qualidade

l = 10 

credit_limit = 10 

info() 

customer_information() 

Use nomes significativos

Nomes de variáveis e funções devem revelar seu propósito.

Lembre-se de seguir as convenções da linguagem!

```
get_address_of_user()
get_address_of_users()
```

Evite o uso de nomes parecidos

Pode causar confusão e má interpretação de seu código..

```python
def concat(n1, n2):
    return n1 + " " + n2

def full_name(first_name, last_name):
    return first_name + " " + last_name
```

Evite números sequenciais em nomes de variáveis

Busque sempre deixar claro o que a variável, função ou classe representam.

```
a = genymdhms()
```

Use nomes pronunciáveis e passíveis de busca

Como você poderia se referir a uma função chamada genymdhms?

Uma busca em seu projeto pela variável “a” traria diversos resultados indesejados.

```python
class User:
    def __init__(self, name):
        self.name = name
```

Nomes de classes

Classes devem ser nomeadas com substantivos.

Não utilize verbos.

Funções devem ser pequenas

Além de ter nomes bem descritivos, você precisa evitar ao máximo que uma função grande e com muitas estruturas aninhadas.

```
def undo_point(self):
        point = self.mainscreen.point_comboBox.currentIndex()
        if self.mainscreen.lane_comboBox.currentText() != "Ladder":
            if len(self.mainscreen.analysisLine.points) > 0:
                if len(self.mainscreen.analysisLine.points) == 1:
                    self.remove_line(False, self.mainscreen.analysisLine)
                else:
                    del self.mainscreen.analysisLine.points[-1]
                    self.mainscreen.point_comboBox.removeItem(point)

                self.mainscreen.generate_widgets()
                self.mainscreen.update_after_changes()
                self.mainscreen.update_plot(self.mainscreen.verticalSlider.value())
```

```
LADDER_LANE = "Ladder"

def undo_point(self):
    if self.mainscreen.lane_comboBox.currentText() != LADDER_LANE:
        undo_for_not_ladder()

def undo_for_not_ladder():
    if self.mainscreen.analysisLine.points:
        remove_line_or_item()

def remove_line_or_item():
    point = self.mainscreen.point_comboBox.currentIndex()
    if len(self.mainscreen.analysisLine.points) == 1:
        self.remove_line(False, self.mainscreen.analysisLine)
    else:
        del self.mainscreen.analysisLine.points[-1]
        self.mainscreen.point_comboBox.removeItem(point)
```

```
def show_game_results(user):
    if user.points >= POINTS_TO_WIN:
        show_victory_results()
    else:
        show_defeat_results()
```

Blocos e identação

O ideal é que blocos de código dentros de if, else, for e outros tenham apenas 1 linha.

O nível de identação dentro de uma função deve ser no máximo 1 ou 2.

```
def show_game_results(user):
    if user.points >= POINTS_TO_WIN:
        show_victory_results()
    else:
        show_defeat_results()
```

A função deve fazer apenas uma coisa

Cada função deve desempenhar seu papel sem realizar tarefas extras.

Se sua função se chama “game_results” significa que a responsabilidade dela é apenas mostrar os resultados do jogo.

```python
#print message about age limit
def pmal(a):
    if a < 15:
        print(a, 'is below the limit.')
    else:
        print(a, 'is old enough.')
    print('Minimum age is ', 15)

AGE_LIMIT = 15
def age_limit_output(age):
    if age < AGE_LIMIT:
        print(age, 'is below the limit.')
    else:
        print(age, 'is old enough.')
    print('Minimum age is ', AGE_LIMIT)
```

Evite magic numbers ou magic strings

É difícil entender o que significa tal coisa ser menor ou maior do que 15 sem saber previamente o contexto.

Utilize constantes para evitar magic numbers ou magic strings.

```
def show_victory_results(user):
    print(user.rounds)
    print(user.points)
    print("Winner!")

def show_defeat_results(user):
    print(user.rounds)
    print(user.points)
    print("Loser!")
```

Don't repeat yourself (DRY)

Atente-se sempre à linhas de códigos que se repetem ao longo de seu projeto. É uma boa prática evitar a repetição dessas linhas movendo-as para uma função.

```
def show_victory_results(user):
    game_details(user)
    print(“Winner!”)

def show_defeat_results(user):
    game_details(user)
    print(“Loser!”)

def game_details(user):
    print(user.rounds)
    print(user.points)
```

Don’t repeat yourself (DRY)

Atente-se sempre à linhas de códigos que se repetem ao longo de seu projeto. É uma boa prática evitar a repetição dessas linhas movendo-as para uma função.

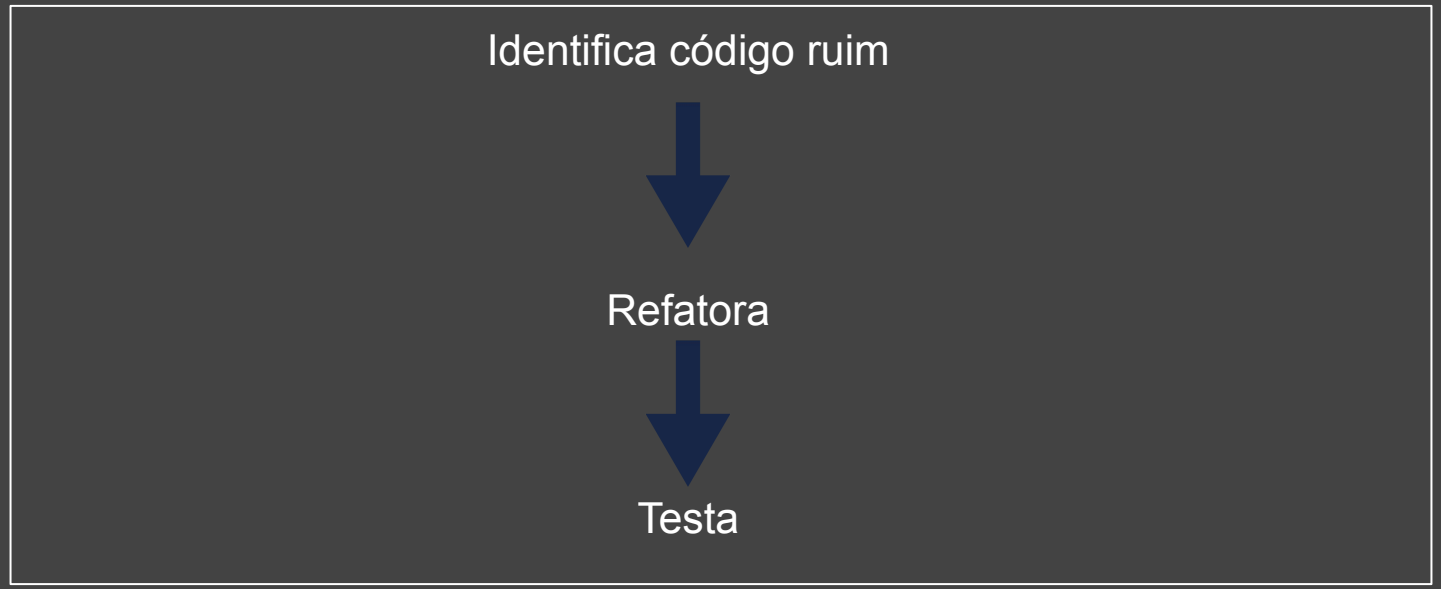
```
def show_victory_results(user):
    game_details(user)#detalhes do jogo
    print(“Winner!”)#mensagem

def game_details(user):
    print(user.rounds)#jogadas
    print(user.points)#pontos
```

Comentários

Com um código bem escrito, comentários não são necessários.

Refatoração constante



Obrigada! :)

Perguntas?

elmasantos94@hotmail.com