Conceitos e paradigmas da programação orientada a objetos

Pâmela Cristina Sperafico[1], Hylson Vescovi Netto[1]

[1]Instituto Federal Catarinense - Campus Blumenau - Blumenau, SC - Brasil

ppanmeli@gmail.com, hylson.vescovi@ifc.edu.br

1. Introdução

O presente trabalho tem por objetivo apresentar comparações entre bibliotecas que são implementadas em paradigmas estrutural, que trata de funções que são aplicadas de modo integral no código, e orientado a objetos, que trata da reusabilidade. Assim, o termo programação orientada a objetos (POO) tornou-se muito popular e hoje muitas linguagens seguem o padrão de desenvolvimento orientado a objetos, que tem por ideia fundamental o reaproveitamento de código.

Utilizarei uma abordagem direta durante o desenvolvimento do trabalho, tratando dos conceitos gerais e também da aplicação desses conceitos por meio de códigos que estarão disponíveis em repositório do GitHub[2].

2. Estudo do artigo

O artigo apresenta como fomento para o desenvolvimento do trabalho uma discussão sobre a programação estrutural versus a POO.

O autor discorre acerca de conceitos como classes, objetos, heranças, bibliotecas orientadas a objetos e procedurais, reusabilidade e programação orientada a objetos e estrutural. Com isso, o autor apresenta várias concepções e ideias que são fundamentais para basear os conceitos e paradigmas da programação orientada a objetos.

3. Conceitos e ideias acerca de paradigmas, bibliotecas e reusabilidade

O paradigma de programação estrutural, é facilmente confundido com a POO devido a seus métodos e utilizações. Contudo, diferentemente da POO, a programação estrutural consiste em um paradigma sequencial, na qual o programa é definido por etapas. Já o paradigma de programação orientada a objetos baseia-se em um conceito de classes e objetos, estruturando códigos para que ele possa ser reutilizável.

Assim, a reusabilidade é um recurso valioso, pois amplia a eficiência e produtividade, tornando um mesmo código moldável a diferentes situações . Com isso, as bibliotecas são a ferramenta ideal pois utilizam-se dessa reusabilidade, fazendo com que diferentes contextos possam ser abordados tendo como partida um ponto em comum, já que essas são repositórios de classes reutilizáveis para aplicação dentro do

---

[2]Link do repositório: https://bit.ly/2TUYNA0

desenvolvimento de código, sendo incorporadas e utilizadas pelo código.

Por fim, pode-se afirmar que a principal diferença entre a programação orientada a objetos e a estrutural é a sequência em que o código é escrito e também processado. A programação estrutural conta métodos escritos de maneira global e executados sequencialmente, tendo um código constante, que é o contrário do que ocorre na POO.

4. Implementações comparativas em código

A implementação dos códigos foi realizada em Python, tanto em programação estrutural quanto em orientada a objetos. O código realiza a transformação de strings em código morse e mostra a string original em tela. A biblioteca implementada e utilizada em ambos os casos é a mesma, composta pelos alfanuméricos e sua codificação para código morse.

Percebe-se que, ao realizar uma programação estrutural, o código pertence à biblioteca e é definido por etapa, indo de um bloco de código para outro. A programação estrutural utiliza-se de algumas rotinas, como condicionais (if, switch, laços for e while) para construir seu código. Contudo, devido a essa construção por etapas, o programa foca nas tarefas que devem ser feitas, deixando de lado o que deve ser feito dessas tarefas. Abaixo, segue um recorde de código, que pode ser visualizado e deve ser testado em sua completude pelo repositório do GitHub. Pelo recorte de código, pode-se analisar a questão das condicionais e como tudo está interligado, indo de um bloco para outro, da parte superior para a inferior.

Figura 1. Implementação do código de modo estrutural

```
def decodificar_morse(morse):
    morse += " "
    texto_plano = ""
    caracter_plano = ""
    for caracter in morse:
        if caracter != " ":
            i = 0
            caracter_plano += caracter
        else:
            i += 1
            if i == 2:
                texto_plano += " "
            else:
                texto_plano += list(codigo_morse.keys())[
                    list(codigo_morse.values()).index(caracter_plano)
                ]
                caracter_plano = ""
    return texto_plano
```

Com isso, já na programação orientada a objetos, o código é definido por diferentes classes tendo em vista a possibilidade de reusabilidade das mesmas em outros

casos. A POO tem como principais características a herança, o polimorfismo, a abstração e o encapsulamento. Dessa forma, as classes dentro do código podem ser facilmente reutilizadas, trazendo uma maleabilidade para o código. Abaixo, segue um recorde do código utilizando orientação a objetos, que pode ser visualizado e deve ser testado em sua completude pelo repositório do GitHub. Com isso, pelo recorte de código, a questão da possibilidade de reutilização é bem mais visível, juntamente com a questão das classes.

Figura 2. Implementação do código de modo orientado a objetos

```
def morse_para_caracteres(morse):
    for caracter in codigo_morse:
        if codigo_morse[caracter] == morse:
            return caracter
    return ""


def decodificar_morse(morse):
    texto_plano = ""
    for caracter_morse in morse.split(" "):
        caracter_plano = morse_para_caracteres(caracter_morse)
        texto_plano += caracter_plano
    return texto_plano
```

5. Considerações finais

O presente trabalho teve por objetivo trazer na prática conceitos que foram introduzidos por meio do artigo e discutidos em conjunto. A aplicação dos conceitos por meio de código se torna essencial para a visualização crua das diferenças entre a programação estrutural e a orientada a objetos. Ainda assim, a sugestão de utilizar uma mesma implementação para ambos os paradigmas deixa mais explícito ainda suas diferenças.

Por fim, a questão das bibliotecas também pode ser vista, tendo um conceito e realizando na prática a implementação de uma biblioteca, que foi utilizada em ambos os exemplos de paradigmas, trazendo também a questão da reusabilidade.

6. Referências Bibliográficas

Wegner, P. (1990). Concepts and Paradigms of Object-Oriented Programming. OOPS Messenger, 1.