labWeek8 - atv1

Aluno: Arthur Calciolari

Grupo: Masterclass A, Tutoria A

> Começaremos com o primeiro princípio, o KISS (Keep It Simple, Stupid):

```
public class SimpleAverageCalculator {

    public static double calculateAverage(int[] numbers) {
        if (numbers.length == 0) {
            return 0;
        }

        int sum = 0;
        for (int num : numbers) {
            sum += num;
        }

        return (double) sum / numbers.length;
    }

    public static void main(String[] args) {
        int[] values = { 10, 20, 30, 40, 50 };
        double average = calculateAverage(values);
        System.out.println("Average: " + average);
    }
}
```

> Ao invés de fazermos métodos complicados que utilizam fórmulas matemáticas que não se encaixam em nossas necessidades, e

> evitar um overhead no código. Manter as coisas simples, ajuda na hora de manter, atualizar e expandir o código.

> Partindo para o segundo princípio, o DRY (Dont Repeat Yourself):

```
public class Product {
    private String name;
    private double price;

    public Product(String name, double price) {
        this.name = name;
        this.price = price;
    }

    public double calculateTotalPrice(int quantity) {
        return price * quantity;
    }
}

public class ShoppingCart {
    public static void main(String[] args) {
        Product apple = new Product("Apple", 0.5);
        Product banana = new Product("Banana", 0.3);

        int appleQuantity = 5;
        int bananaQuantity = 3;

        double appleTotalPrice = apple.calculateTotalPrice(appleQuantity);
        double bananaTotalPrice = banana.calculateTotalPrice(bananaQuantity);

        System.out.println("Apple Total Price: " + appleTotalPrice);
        System.out.println("Banana Total Price: " + bananaTotalPrice);
    }
}
```

> Ao usar o mesmo método para cálcular o preço total das bananas e das maçãs, evitamos escrever a mesma função para dois objetos diferentes, tirando proveito do princípio DRY.

> Partindo agora para o último princípio, mas não menos importante, temos o YAGNI (You ain’t Gonna Need It):

```
class Task {
    private String description;
    private boolean isCompleted;

    public Task(String description) {
        this.description = description;
        this.isCompleted = false;
    }

    public void markAsCompleted() {
        this.isCompleted = true;
    }

    @Override
    public String toString() {
        return (isCompleted ? "[X] " : "[ ] ") + description;
    }
}

public class TaskManager {
    public static void main(String[] args) {
        Task task1 = new Task("Implement the login functionality");
        Task task2 = new Task("Design the user profile page");

        System.out.println("Tasks:");
        System.out.println(task1);
        System.out.println(task2);

    }
}
```

> Nesse exemplo, o código segue o princípio YAGNI ao não implementar imediatamente a funcionalidade de exclusão de tarefas. Neste estágio inicial, apenas a criação e marcação de tarefas como concluídas são necessárias. A implementação da exclusão de tarefas é adiada até que seja realmente necessária, evitando a introdução prematura de complexidade no sistema.