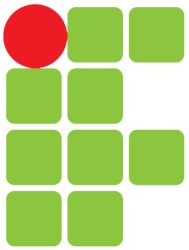

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PIAUÍ

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
MÓDULO II - PROGRAMAÇÃO ORIENTADA A OBJETOS
PROF.: Ely Miranda
ALUNO: Lucas Gomes de Oliveira (20191ADS0185)

Exercício 01

1. Qual a diferença entre objetos e classes? Exemplifique.

Na programação orientada a objetos, uma classe nada mais é do que um molde para a criação de um objeto. Seria uma forma de manipular uma determinada informação a partir de um molde pré-definido. Com uma classe, é possível instanciar um objeto, que seria a representação real dessa mesma classe.

Por exemplo, a classe Caneta é um molde para a criação de futuros objetos. Quando se instancia a classe, é possível criar quantos objetos quisermos, como caneta1, caneta2, caneta3, etc.

2. De forma breve, conceitue atributos e métodos. Pesquise e exemplifique um exemplo de objeto que possua atributos e métodos (notação livre).

Como fora dito anteriormente, uma classe é um molde para a criação de futuros dados. Para que seja possível a concepção desse dado (objetos) é necessário que haja características e ações a serem realizadas. Por exemplo, uma caneta poderá ter como informação a cor, o tamanho, o nome da marca e a espessura. Já as suas ações poderão ser de escrever, desenhar, rabiscar ou estar cheia ou não.

A seguir está um código que representa melhor essa abstração:

```
class Caneta {
    let cor: String;
    let tamanho: number;
    let marca: String;
    let espessura: number;
    let quantidadeTinta: number;
```

```
    constructor(cor, tamanho, marca, espessura, quantidadeTinta) {
        this.cor = cor;
        this.tamanho = tamanho;
        this.marca = marca;
        this.espessura = espessura;
        this.quantidadeTinta = quantidadeTinta;
    }

    public escrever(valor: String): String {
        console.log(`Estou escrevendo => ${valor}`)
    }

   public  desenhar(desenho: String) {
        console.log(`Estou desenhando => ${desenho}`)
    }

   public rabisar(valor: String): String {
        console.log(`Estou rabiscando => ${valor}`)
    }

    public estarCheia(): boolean {
        if (this.quantidadeTinta === 0) {
            return false;
        }
        return true;
    }
}

caneta1 = new Caneta('preta', 2.0, 'Stabilo', 1.5, 78);
caneta2 = new Caneta('azul', 2.5, 'Bic', 1.9, 0);
caneta3 = new Caneta('vermelha', 2.0, 'Faber-Castell', 1.7, 12);
```

3. A abstração visa focar no que é importante para um sistema. Você concorda que um atributo de uma pessoa pode ser importante ou não dependendo do contexto do sistema. Enumere na tabela abaixo contextos/sistemas distintos em que os atributos abaixo seriam ou não relevantes:

Atributo	Sistema em que é importante	Sistema em que não é importante

Peso	Telemedicina	Cadastro escolar
Tipo de CNH	Sistema de aplicação para vagas de emprego	Cadastro clínico
Tipo Sanguíneo	Cadastro clínico	Internet banking
Habilidade destra	Cadastro escolar	Aplicação para o monitoramento da saúde física (ex: Runtastic)
Percentual de gordura	Aplicação para o monitoramento da saúde física (ex: Runtastic)	Cadastro escolar
Saldo em conta	Internet banking	Telemedicina
Etnia	Pesquisa para dados estatísticos (ex: IBGE)	Sistema de aplicação para vagas de emprego

4. Considerando os objetos Pessoa e Conta:

a. Seria interessante em um sistema bancário um objeto "conta" possuir uma "pessoa" como um atributo interno representando o titular da conta?
Sim, pelo fato de facilitar tanto a manutenção do sistema quanto na hora de consultar a situação da conta do titular.

b. Olhando no sentido inverso, seria interessante uma pessoa possuir mais de uma conta como atributo? Que elemento da programação estruturada melhor representaria o conjunto de contas de uma pessoa?
Deveria haver uma implementação maior caso uma única pessoa possuísse inúmeras contas. Ela poderia estar devendo em uma conta e optou por abrir mais outra; para isso deveria haver um algoritmo próprio para verificação dos saldos e da situação das contas. Nesse caso, deveria ser utilizado uma estrutura de array para conter cada conta e relacioná-las com o titular.

5. Identifique pelo menos 5 objetos de um sistema de controle acadêmico. Ex: aluno.
Objeto aluno.
Objeto professor.
Objeto aula/sala.
Objeto matrícula.
Objeto curso.

6. Imagine um jogo qualquer. Identifique o máximo de objetos possíveis e eventuais características (atributos) e comportamentos (métodos) que os mesmos poderiam ter.

Classe Atirador
Atributos: tipo de roupa, arma principal, quantidade de itens, quantidade de munição
Métodos: pular, pulo duplo, atacar, girar, se defender, ataque especial

Classe Médico
Atributos: roupa, quantidade de remédio, tipo de medicação
Métodos: curar, defender

Classe Paladino
Atributos: tipo de roupa, tipo de ataque, quantidade de magia disponível, estamina
Métodos: atacar, ataque mágico, desaparecer, se teletransportar

Classe Antagonista
Atributos: tipo de armadura, arma principal, combate especial
Métodos: atacar, ataque especial, paralisar, se defender

7. Considerando o exemplo da classe Retangulo dos slides, implemente um método adicional chamado que calcule o perímetro do retângulo e altere a classe TestaRetangulo para exibir o cálculo do perímetro.

```
class Retangulo {
    //necessario inicializar os atributos
    l1: number;
    l2: number;

    constructor(l1: number, l2: number) {
        this.l1 = l1;
        this.l2 = l2;
    }

    calculaArea(): void {
        console.log(this.l1 * this.l2);
    }
}

class TestaRetangulo extends Retangulo {
    constructor(l1: number, l2: number) {
```

```
        super(l1, l2);
    }

    calculaPerimetro(): void {
        console.log(2*this.l1 + 2*this.l2);
    }
}

let retangulo: TestaRetangulo;
retangulo = new TestaRetangulo(10, 90);

retangulo.calculaArea();
retangulo.calculaPerimetro();
```

8. Crie uma classe Circulo que possua um atributo raio. Crie dois métodos que calculam a área e o perímetro. Instancie um objeto dessa classe, atribua um valor ao raio e exiba a área e o perímetro chamando os dois métodos definidos.

```
class Circulo {
    //necessario inicializar os atributos
    pi: number = 3.14;
    raio: number = 0;


    calculaArea(): number {
        return Math.pow(this.raio, 2) * this.pi;
    }

    calculaPerimetro(): number {
        return this.pi * this.raio * 2;
    }
}

let circulo: Circulo;
circulo = new Circulo();

circulo.raio = 12;
```

```
console.log(`Valor da area => ${circulo.calculaArea()}`);
console.log(`Valor do perimetro => ${circulo.calculaPerimetro()}`);
```

9. Crie uma classe chamada SituacaoFinanceira com os atributos valorCreditos e valorDebitos. Crie um método chamado saldo() que retorna/calcula a diferença entre crédito e débito. Instancie uma classe SituacaoFinanceira, inicialize os dois atributos e exiba o resultado do método saldo().

```
class SituacaoFianceira {
    valorCreditos: number;
    valorDebitos: number;

    constructor(x: number, y: number) {
        this.valorCreditos = x;
        this.valorDebitos = y
    }

    saldo(): number {
        return this.valorCreditos - this.valorDebitos;
    }
}

const minhaConta = new SituacaoFianceira(323223, 212132);
console.log(`Meu saldo atual => R$ ${minhaConta.saldo()}`);
```