



## AVALIAÇÃO 2º BIMESTRE

Disciplinas: POO / Linguagem de Programação  
Professora: Adriana Nakahara Miquiline

Aluno: Vinicius Gustavo da Silva

RA: 207035

Aluno: Josue Lara

RA: 207455

Curso : GTI/ADS

Data: 16/06/2020

NOTA:

LEIA COM ATENÇÃO AS INSTRUÇÕES GERAIS

- Identifique corretamente esta prova.
- Valor da prova: 7 pontos

### QUESTÕES

1. (0,5 ponto) Considere o seguinte diagrama de classe:

Conta Corrente
numero da conta nome do correntista saldo
construtor (numero, nome, saldo) alterarNome(nome) deposito(valor) saque(valor) print()

Considerando este diagrama responda:

A. Qual é o nome da classe?

O nome da classe é Conta Corrente.

B. Quais são os atributos?

Os atributos são:

numero da conta, nome do correntista e saldo.

C. Relacione os métodos da classe.

Os métodos são:

construtor, alterarNome, deposito, saque e print.

2. (2 pontos) **Classe Conta Corrente:** Crie uma classe para implementar uma conta corrente. A classe deve possuir os seguintes atributos: número da conta, nome do correntista e saldo.

Os métodos são os seguintes: alterarNome, depósito (só faz esta operação se o valor do depósito for maior que zero) , saque (verifica se possui saldo) e print; No construtor, saldo é opcional, com valor default zero e os demais atributos são obrigatórios.

Conta Corrente
numero da conta nome do correntista saldo
construtor (numero, nome, saldo) alterarNome(nome) deposito(valor) saque(valor) print()

```
class ContaCorrente{

    constructor (conta, nome, saldo){

        this.conta = conta;

        this.nome = nome;

        this.saldo = saldo;

    }

    altera(nome){

        this.nome = nome;

    }

    deposito(valor) {

        if (valor > 0) {

            this.saldo = this.saldo + valor;

        } else {alert("Deposito precisa ser maior que zero!");}

    }

    saque(valor){

        if (valor > 0) {
```

```
        this.saldo = this.saldo - valor;

    } else { alert("Saldo zerado"); }

}

print(){

    console.log("Conta: " + this.conta);

    console.log("Cliente: " + this.nome);

    console.log("Saldo = "+ this.saldo);

}

}
```

3. (0,5 ponto) Após a implementação da classe Conta Corrente, instancie um objeto e teste todos os métodos.

```
let x = new ContaCorrente(01, "Vinicius && Josué", 1000)
```

```
x.print()
```

```
> x.print()
Conta: 1
Cliente: Vinicius && Josue
Saldo = 1000
< undefined
> |
```

```
x.deposito(20)
```

```
> x.deposito(20)
< undefined
> x.print()
Conta: 1
Cliente: Vinicius && Josue
Saldo = 1020
< undefined
>
```

x.saque(500)

```
> x.saque(500)
< undefined
> x.print()
Conta: 1
Cliente: Vinicius && Josue
Saldo = 520
< undefined
>
```

x.altera("Vini && Josué Lara");

```
> x.altera("Vini && Josué Lara");
< undefined
> x.print();
Conta: 1
Cliente: Vini && Josué Lara
Saldo = 1000
< undefined
>
```

4. (1 ponto) Acrescente os métodos getter e setter.

```
set _conta(conta){
```

```
    this.conta = conta;
```

```
}
```

```
get _conta(){
```

```
    return this.conta;
```

```
}
```

```
set _nome(nome){
```

```
    this.nome = nome;
```

```
}
```

```
get _nome(){
```

```
    return this.nome;
```

```
}
```

```
set _saldo(saldo){
```

```
this.saldo = saldo;  
  
}  
  
get _saldo(){  
  
    return this.saldo;  
  
}
```

5. (0,5 ponto) Uma dupla construiu o getter e setter para saldo da seguinte forma:

```
19  set saldo(saldo){  
20      if (saldo>0)  
21          this.saldo = saldo;  
22  }  
23  get saldo(){  
24      return this.saldo;  
25  }
```

```
78  
79  let c = new Cliente (1, "Andre", 5);  
80  c.saldo = -1;  
81  c.print();  
82  let f = new Fisica (2, "Jeni", 5000, 156789);  
83  f.print();
```

Não funcionou. Explique o motivo.

É necessário que haja o `_` ao definir os setters e getters para referenciar o objeto.  
Exemplo: se o código estivesse conforme abaixo, o mesmo funcionaria:

```
set _saldo(saldo){  
  
    this.saldo = saldo;  
  
}  
  
get _saldo(){  
  
    return this.saldo;  
  
}
```

6. (2 pontos) Construa a Classe Fisica que é filha da classe conta corrente. A classe deve acrescentar o CPF. Os métodos são os seguintes: getter, setter e print e construtor.  
Lembrete: reutilize o que vem de herança.

```
class Fisica extends ContaCorrente{  
  
    constructor (conta, nome, saldo, cpf){  
  
        super (conta, nome, saldo);  
  
        this.cpf = cpf;  
  
    }
```

```
set _cpf(cpf){  
    this.cpf = cpf;  
}  
  
get _cpf(){  
    return this.cpf;  
}  
  
print (){  
    super.print();  
    console.log ("CPF: "+ this.cpf);  
}  
}
```

7. (0,5 ponto) Após a implementação da classe Física, instancie um objeto e teste todos os métodos.

```
let f = new Fisica(01, "Vinicius && Josue", 5000, 12332151250)
```

```
f.print();
```

```
> let f = new Fisica(01, "Vinicius && Josue", 5000, 12332151250)  
< undefined  
> f.print()  
Conta: 1  
Cliente: Vinicius && Josue  
Saldo = 5000  
CPF: 12332151250  
< undefined  
>
```

```
f.deposito(2500);
```

```
> f.deposito(2500)
```

```
< undefined
```

```
> f.print()
```

```
Conta: 1
```

```
Cliente: Vinicius && Josue
```

```
Saldo = 7500
```

```
CPF: 12332151250
```

```
< undefined
```

f.saque(6000);

```
> f.saque(6000);
```

```
< undefined
```

```
> f.print()
```

```
Conta: 1
```

```
Cliente: Vinicius && Josue
```

```
Saldo = 1500
```

```
CPF: 12332151250
```

```
< undefined
```

```
>
```

f.altera("Vini && Josue Lara");

```
> f.altera("Vini && Josue Lara");
```

```
< undefined
```

```
> f.print()
```

```
Conta: 1
```

```
Cliente: Vini && Josue Lara
```

```
Saldo = 1500
```

```
CPF: 12332151250
```

```
< undefined
```

```
>
```