Universidade Federal do Ceará - Campus de Quixadá

Programação para Design

Classes Abstratas e Interfaces

Prof. Anderson Lemos

Introdução

- Exemplo:
  - Na UFC temos, pelo menos, dois tipo de servidor:
    - Professor.
    - STA (Servidor técnico administrativo).
  - Ambos ganham bonificação no final do ano. Mas a regra da bonificação é diferente para os dois:
    - Professor recebe 1 real por hora de aula ministrada como bonificação.
    - STA recebe 10% do salário como bonificação.

Introdução

- Exemplo - Servidor:

```
// arquivo Servidor.ts
// pacote: br.com.ufc.dd.poo.model
export class Servidor{
    private nome : string;
    private cpf : string;
    private salario : number;
    public constructor(nome : string, cpf : string,
                       salario : number){
        this.nome = nome;
        this.cpf = cpf;
        this.salario = salario;
    }
    public getBonificacao() : number {
        return this.salario*0.1;
    }
}
```

Introdução

- Exemplo - STA:

```
// arquivo STA.ts
// pacote: br.com.ufc.dd.poo.model
import {Servidor} from './Servidor';

export class STA extends Servidor{
    public constructor(nome : string, cpf : string,
                       salario : number){
        super(nome, cpf, salario);
    }
}
```

Introdução

- Exemplo - Professor:

```
// arquivo Professor.ts
// pacote: br.com.ufc.dd.poo.model
import {Servidor} from './Servidor';

export class Professor extends Servidor{
    private horasMinistradas : number;
    public constructor(nome : string, cpf : string,
                       salario : number, horasMinistradas : number){
        super(nome, cpf, salario);
        this.horasMinistradas = horasMinistradas;
    }

    // abaixo temos uma sobrescrita do método
    public getBonificacao() : number {
        return this.horasMinistradas;
    }
}
```

Introdução

- Exemplo
  - Faz sentido ter instâncias do tipo Servidor?
    - `let s : Servidor = new Servidor(); // isso faz sentido?`
  - Existe um servidor sem tipo específico?

Classes Abstratas

- Classes abstratas servem apenas como modelos para outras classes.
- Classes abstratas não podem ser instanciadas.
- É interessante quando não queremos permitir criar instâncias a partir de um classe.
- Uma classe que não é abstrata é uma classe concreta.
- TypeScript tem classes abstratas nativamente.
  - Uso da palavra reservada ***abstract***.

Classes Abstratas

- Exemplo - Servidor:

```
// arquivo Servidor.ts
// pacote: br.com.ufc.dd.poo.model
export abstract class Servidor{
    private nome : string;
    private cpf : string;
    private salario : number;
    public constructor(nome : string, cpf : string,
                       salario : number){
        this.nome = nome;
        this.cpf = cpf;
        this.salario = salario;
    }
    public getBonificacao() : number {
        return this.salario*0.1;
    }
}
```

Classes Abstratas

- Entendendo o exemplo:
  - Agora, queremos adicionar um novo tipo de servidor: **Psicólogo**. Esse cargo tem uma nova regra de bonificação: Psicólogo recebe 20 reais por cada consulta realizada.
  - Assim, cada cargo tem um modo de bonificação diferente. Vale a pena implementar *getBonificação* em Servidor?
  - Não existe regra geral de bonificação. Logo, a resposta é **não**.
    - Jogamos fora o método *getBonificação* de Servidor?
    - A resposta é: jogamos somente a implementação do método fora.
    - Isso porque ele passará a ser um **método abstrato**.

Métodos Abstratos

- Métodos abstratos não possuem implementação.
- Métodos abstratos só podem aparecer em classes abstratas ou interfaces.
- Não podem ser chamados diretamente.
- Sua implementação é feita através de sobrescrita.
- Obriga alguma classe filha a implementar o método.
- TypeScript tem métodos abstratos nativamente.
    - Também utilizando a palavra reservada ***abstract***.

Métodos Abstratos

- Exemplo - Servidor:

```
// arquivo Servidor.ts
// pacote: br.com.ufc.dd.poo.model
export abstract class Servidor{
    private nome : string;
    private cpf : string;
    private salario : number;
    public constructor(nome : string, cpf : string,
                       salario : number){
        this.nome = nome;
        this.cpf = cpf;
        this.salario = salario;
    }
    public abstract getBonificacao() : number;
}
```

Métodos Abstratos

- Assim, é necessário implementar o método *getBonificacao* na classe STA e também na classe Psicólogo.
- Se uma classe abstrata ***A*** tem um método abstrato ***M***, todas as classes concretas “descendentes” de ***A*** tem que implementar o método ***M***, se não tiver nenhum antecessor que implemente ***M***.

Interfaces

- Interface define as operações que uma classe deve ter.
- Só contém métodos sem implementação que as classes “filhas” devem implementar.
  - É como uma classe abstrata somente com métodos abstratos.
  - Filhas entre aspas porque uma Interface não é uma classe, então nenhuma outra pode herdar dela.
  - As classes não herdam de interfaces, elas implementam interfaces.
    - No entanto, o relacionamento também é ***is-a*** (é um).
- Em TypeScript temos Interfaces nativamente.
  - Uso das palavras reservadas ***interface*** e ***implements***.
- Interface vs. Classe abstrata:
  - Classe abstrata pode conter atributos e implementação de alguns métodos.

Interfaces

- Exemplo:

```
// arquivo Calculadora.ts
// pacote: br.com.ufc.dd.poo.model

export interface Calculadora{
    soma(a : number, b : number) : number;
    sub(a : number, b : number) : number;
    mult(a : number, b : number) : number;
    div(a : number, b : number) : number;
}
```

Interfaces

- Exemplo:

```
// arquivo CalculadoraImpl.ts
// pacote: br.com.ufc.dd.poo.model
import {Calculadora} from './Calculadora';

export class CalculadoraImpl implements Calculadora{
    public constructor(){}
    public soma(a : number, b : number) : number{
        return a+b;
    }
    public sub(a : number, b : number) : number{
        return a-b;
    }
    public mult(a : number, b : number) : number{
        return a*b;
    }
    public div(a : number, b : number) : number{
        return a/b;
    }
}
```

Interfaces

- Exemplo:

```
// arquivo Principal.ts
// pacote: br.com.ufc.dd.poo.exec
import {Calculadora} from '../model/Calculadora';
import {CalculadoraImpl} from '../model/CalculadoraImpl';

let calculadora : Calculadora = new CalculadoraImpl();
console.log(calculadora.soma(15,3)); // imprime 18
console.log(calculadora.sub(15,3)); // imprime 12
console.log(calculadora.mult(15,3)); // imprime 45
console.log(calculadora.div(15,3)); // imprime 5
```

Exercício

- Sistema de banco
  - Nosso banco tem dois tipos de conta: **Conta Corrente** e **Conta Poupança**. Ambos tem *saldo* e *número de conta*.
  - Todo mês, o dinheiro na conta é corrigido segundo a inflação. Mas as duas contas valorizam de modo diferente:
  - Conta poupança valoriza 6,5%.
  - Conta corrente valoriza apenas 1,3%.
    - Implemente em TypeScript as classes que representam as contas desse sistema bancário.

Perguntas?