04

Interface em ação

Olga estudou Java durante um bom tempo, inclusive C#. Ela decidiu replicar um código que havia escrito nessas linguagem em TypeScript.

Ela criou duas classes que representam formas geométricas:

```
class Retangulo {

    constructor(private altura: number, private lado: number) {}

    calculaArea(): number {

        return this.altura * this.lado;
    }
}

class Quadrado {

    constructor(private lado: number) {}

    calculaArea(): number {

        return this.lado * this.lado;
    }
}
```

Em seguida, criou a classe `CalculadorDeArea` . Ela é a responsável em receber N formas geométricas e no final retornar o total da àrea de todas elas. Assim como no ECMASCRIPT 2016, é possível criar métodos estáticos em TypeScript:

```
class CalculadorDeArea {

    static calcula(area1: Quadrado, area2: Retangulo): number {

        return area1.calculaArea() + area2.calculaArea();
    }
}
```

Para testar seu código ela fez:

```
const quadrado = new Quadrado(30);
const retangulo = new Retangulo(50, 30);

const total = CalculadorDeArea.calcula(quadrado, retangulo);
console.log(total);
```

Excelente! Funcionou como esperado. Contudo, ela precisou criar mais uma área calculável:

```
class Circulo {

    constructor(private raio: number) {}

    calculaArea() {

        return Math.PI * this.raio * this.raio;
    }
}
```

O problema agora é que ela precisará alterar o método `CalculadorDeArea.calcula` para receber agora um objeto do tipo `Circulo`, inclusive alterar sua lógica para que seja capaz de levar em consideração a nova area. **É uma situação na qual herança não ajudaria, pois não faria sentido ela herdar um código já pronto, pois o cálculo da área é sempre diferente para as formas geométricas**. Foi então que ela lembrou que o uso de interfaces pode ajudá-la.

Ela criou a interface `AreaCalculavel`:

```
interface AreaCalculavel {

    calculaArea(): number;
}
```

Em seguida, fez com que as três classes Quadrado, Retangulo e Circulo a implementassem:

```
class Retangulo implements AreaCalculavel {

    constructor(private altura: number, private lado: number) {}

    calculaArea(): number {

        return this.altura * this.lado;
    }
}

class Quadrado implements AreaCalculavel  {

    constructor(private lado: number) {}

    calculaArea(): number {

        return this.lado * this.lado;
    }
}

class Circulo implements AreaCalculavel {

    constructor(private raio: number) {}

    calculaArea() {

        return Math.PI * this.raio * this.raio;
```

```
    }
}
```

Ao implementar a interface `AreaCalculavel`, todas as classes ganham a obrigação de definir o método definido pela interface. Como elas já definiam esse métodos antes mesmo de implementá-la, o código compilou sem problema algum.

Marque a opção que altera corretamente a classe `CalculadorDeArea` para que possa lidar com objetos que implementem a interface `AreaCalculavel`:

Selecione uma alternativa

A

```
class CalculadorDeArea {

    static calcula(...areas: AreaCalculavel[]): number {

        return areas.reduce((total, area) => total + area.calculaArea(), 0);
    }
}


const quadrado = new Quadrado(30);
const retangulo = new Retangulo(50, 30);
const circulo = new Circulo(20);

const total = CalculadorDeArea.calcula(quadrado, retangulo, circulo);
console.log(total);
```

B

```
class CalculadorDeArea {

    static calcula(...areas: AreaCalculavel): number {

        return areas.reduce((total, area) => total + area.calculaArea(), 0);
    }
}


const quadrado = new Quadrado(30);
const retangulo = new Retangulo(50, 30);
const circulo = new Circulo(20);

const total = CalculadorDeArea.calcula(quadrado, retangulo, circulo);
console.log(total);
```

C

```
class CalculadorDeArea {

    static calcula(areas: AreaCalculavel[]): number {

        return areas.reduce((total, area) => total + area.calculaArea(), 0);
    }
}


const quadrado = new Quadrado(30);
```

```
const retangulo = new Retangulo(50, 30);
const circulo = new Circulo(20);

const total = CalculadorDeArea.calcula(quadrado, retangulo, circulo);
console.log(total);
```