REVISÃO DE CONCEITOS

⚠ Não seguiremos a UML ao pé da letra ⚠

Você não precisa saber UML.

Você não precisará se preocupar com UML e padrões de projeto ao mesmo tempo.

Utilizaremos UML, mais especificamente DIAGRAMA DE CLASSES, apenas para tornar os padrões de projeto mais VISUAIS.

ORIENTAÇÃO A OBJETOS

&

DIAGRAMA DE CLASSES

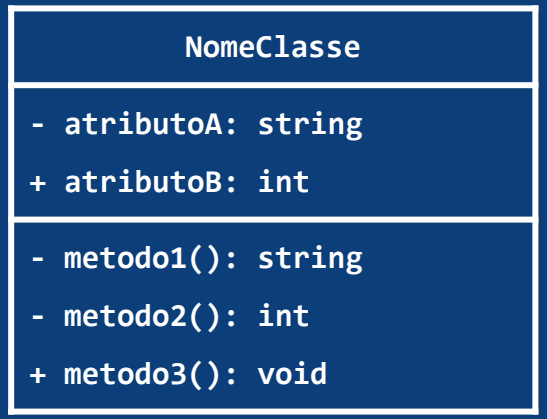
CLASSE

```
class NomeClasse
{
	private string atributoA;
	public int atributoB;

	private function metodo1(): string
	{
		//corpo do método
	}

	protected function metodo2(): int
	{
		//corpo do método
	}

	public function metodo3(): void
	{
		//corpo do método
	}
}
```



private

protected

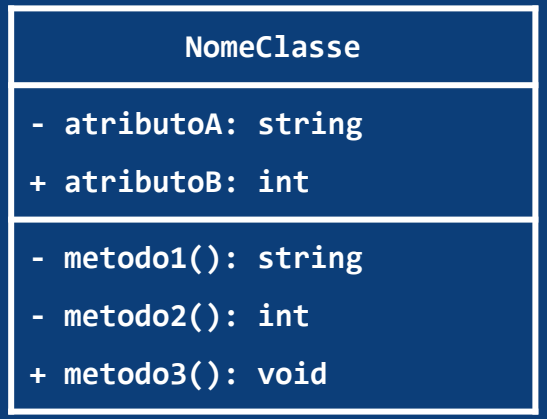
public

OBJETO

Um objeto é o resultado do processo de instanciação de uma classe

```
objeto1 = new NomeClasse();
objeto1->metodo3();
objeto1->metodo2();


objeto2 = new NomeClasse();
objeto2->metodo3();
```



private

protected

public

MÉTODOS DE CLASSE (ESTÁTICOS)

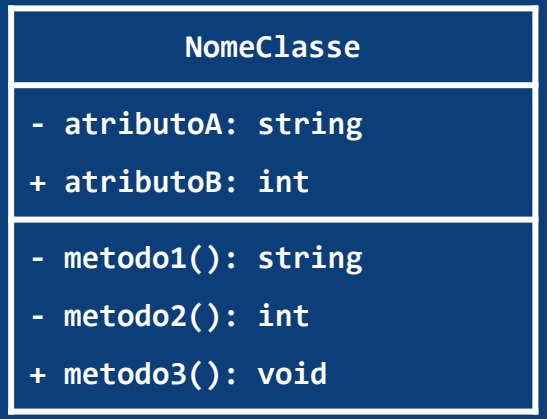
Métodos estáticos podem ser invocados sem que exista um objeto da classe

```
class NomeClasse
{
	private string atributoA;
	public int atributoB;

	private function metodo1(): string
	{
		//corpo do método
	}

	protected function metodo2(): int
	{
		//corpo do método
	}

	public static function metodo3(): void
	{
		//corpo do método
	}
}
```




```
NomeClasse::metodo3();
```

private

protected

public

static

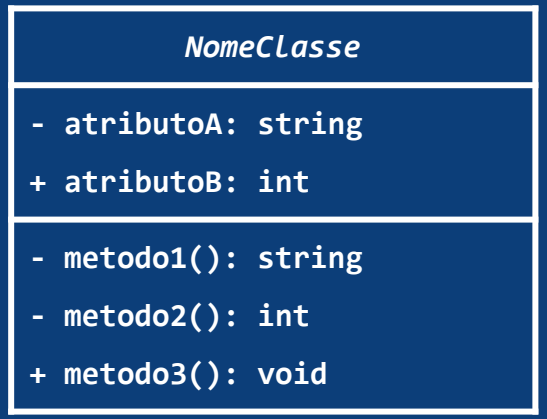
CLASSE ABSTRATA

```
abstract class NomeClasse
{
	private string atributoA;
	public int atributoB;

	private function metodo1(): string
	{
		//corpo do método
	}

	protected function metodo2(): int
	{
		//corpo do método
	}

	public function metodo3(): void
	{
		//corpo do método
	}
}
```



private

protected

public

static

Não pode ser instanciada
Não gera objetos

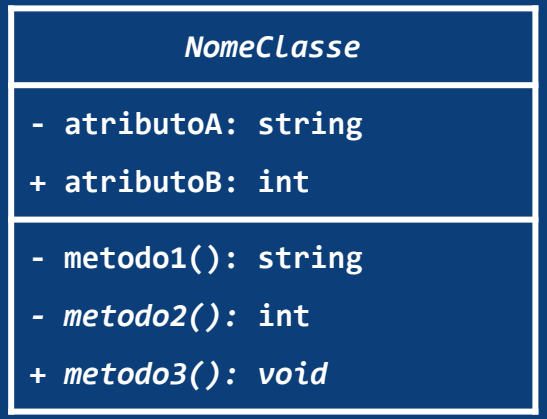
MÉTODOS ABSTRATOS

```
abstract class NomeClasse
{
      private string atributoA;
      public int atributoB;

      private function metodo1(): string
      {
            //corpo do método
      }

      abstract protected function metodo2(): int;

      abstract public function metodo3(): void;
}
```



private

protected

public

static

Alguma subclasse deve implementar os métodos abstratos

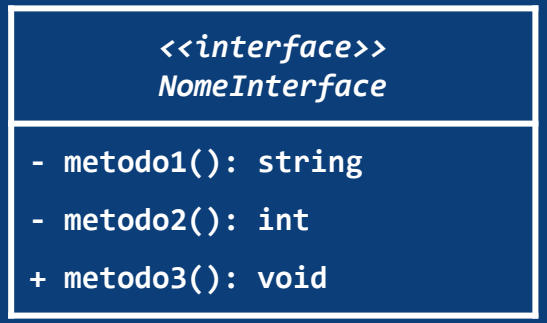
Define um **SUPERTIPO / TIPO ABSTRATO**

INTERFACES

```
interface NomeInterface
{
	private function metodo1(): string;

	protected function metodo2(): int;

	public function metodo3(): void;
}
```



private

protected

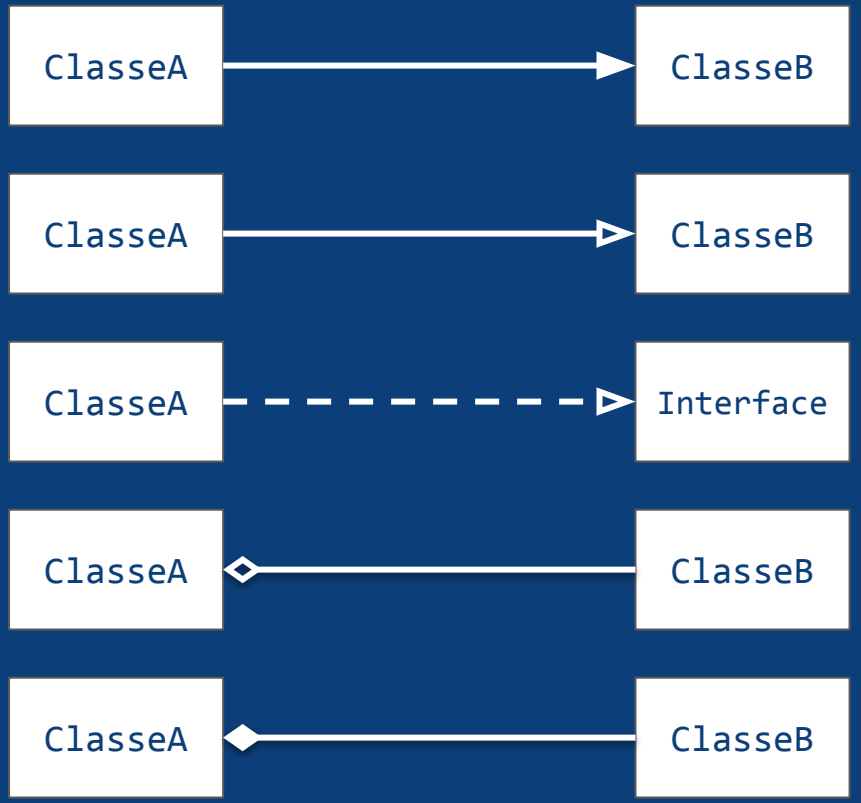
public

As classes que implementam uma interface devem implementar todos os métodos definidos nela

Define um **SUPERTIPO / TIPO ABSTRATO**

RELACIONAMENTOS

- ASSOCIAÇÃO
- GENERALIZAÇÃO
- REALIZAÇÃO
- AGREGAÇÃO
- COMPOSIÇÃO



ASSOCIAÇÃO


```
class ClasseA
{
      //Início do corpo da classe
      ...
            variavel = new classeB();
      ...
      //Fim do corpo da classe
}
```

A ClasseA depende da ClasseB

GENERALIZAÇÃO


```
class ClasseA extends ClasseB
{
      //Corpo da classe
}
```

A ClasseA herda a ClasseB

A ClasseA é uma subclasse da ClasseB

REALIZAÇÃO


```
class ClasseA implements Interface
{
      //Corpo da classe
}
```

A ClasseA implementa a Interface

AGREGAÇÃO


```
class ClasseA
{
      public function metodo(ClasseB b)
      {
            //Corpo do método
      }

}
```

O objeto da ClasseB é utilizado/faz parte da ClasseA

A ClasseA não é fortemente ligada a ClasseB (podem existir separadamente)

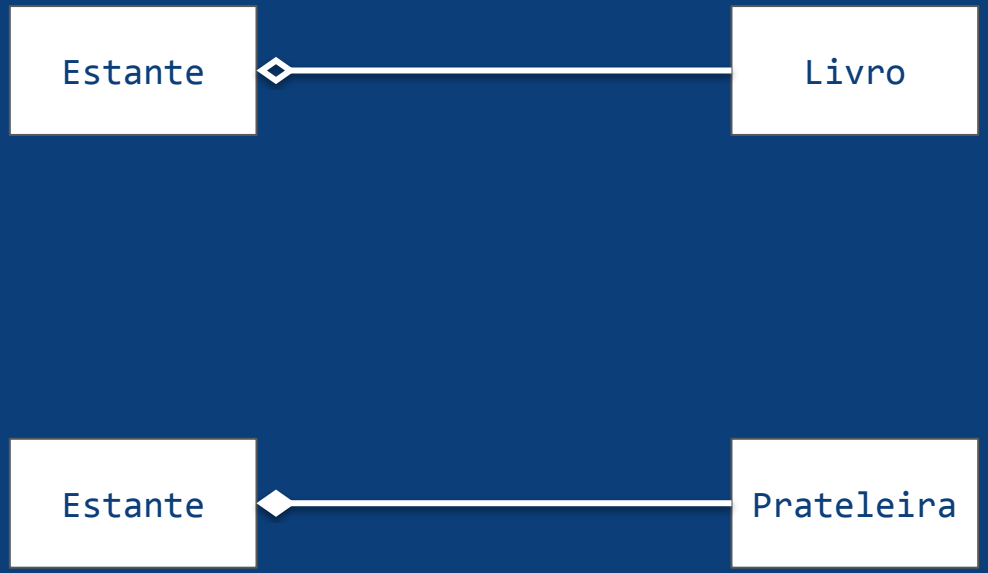
COMPOSIÇÃO


```
class ClasseA
{
	public function metodo(ClasseB b)
	{
		//Corpo do método
	}

}
```

O objeto da ClasseB é utilizado/faz parte da ClasseA

Quando a ClasseA deixar de existir a ClasseB também deixará de existir



INJEÇÃO DE DEPENDÊNCIA (Composição)


```
class ClasseA
{
	public function construtor(ClasseB b)
	{
		//Corpo do construtor
	}
}
```

OUTROS RELACIONAMENTOS

EXISTEM OUTROS RELACIONAMENTOS

Mas nosso foco está nos

PADRÕES DE PROJETO

Então vamos simplificar tudo que não for nosso foco

PILARES DA ORIENTAÇÃO A OBJETOS

- ENCAPSULAMENTO
- HERANÇA
- ABSTRAÇÃO
- POLIMORFISMO

ENCAPSULAMENTO


```
conta = new Conta();
conta->calculaSaldo();

conta->sacar();
conta->calculaSaldo();
```

private

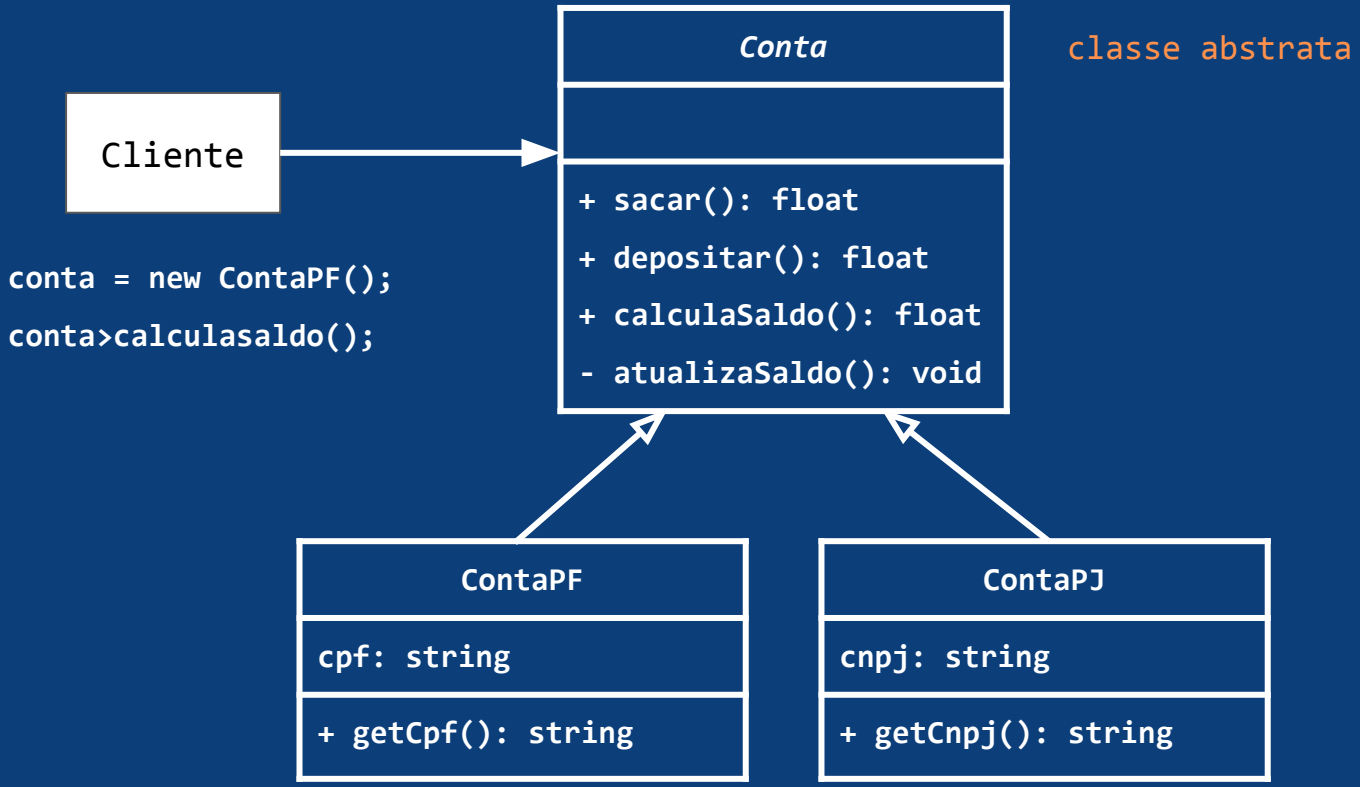
protected

HERANÇA

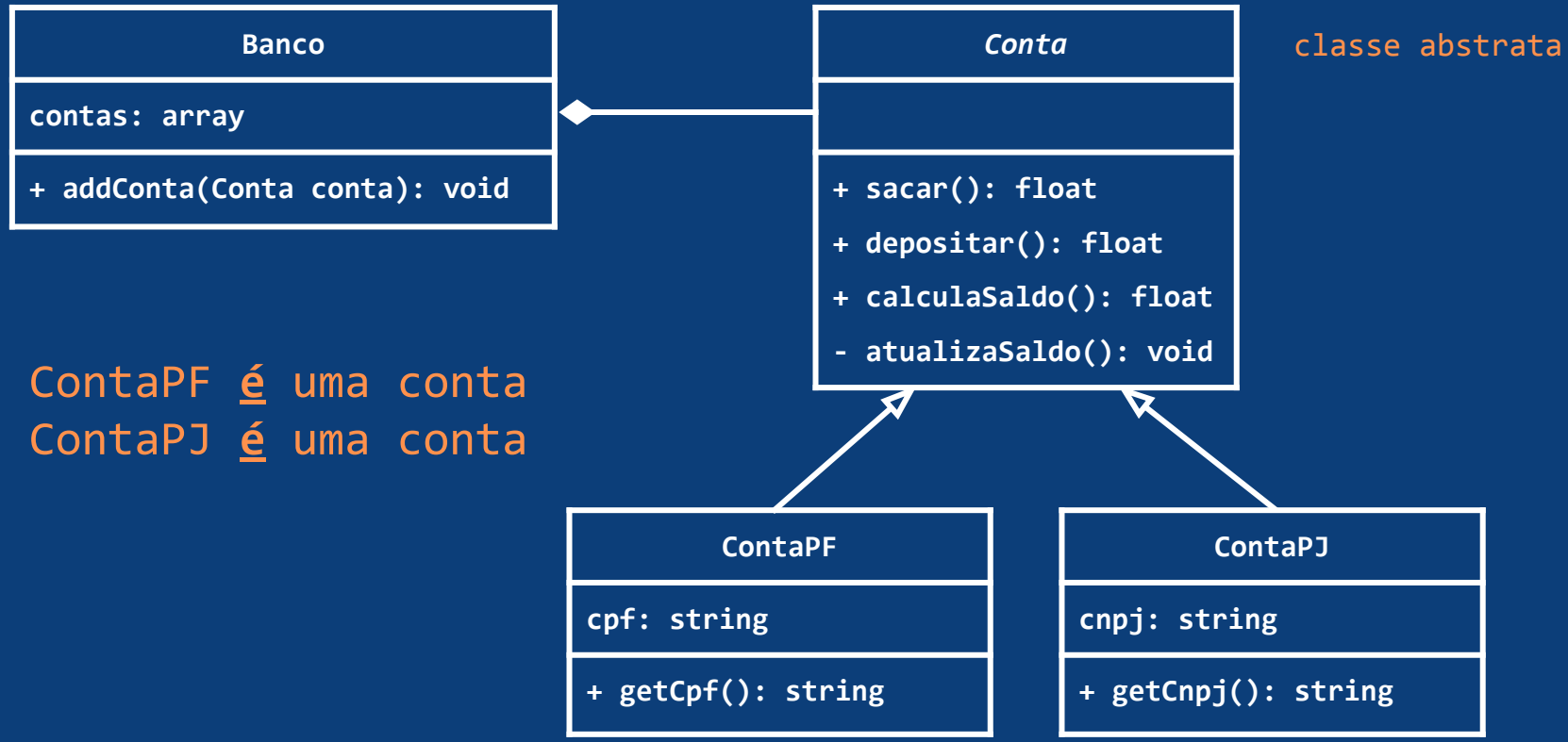
private

protected

public



ABSTRAÇÃO



POLIMORFISMO