Programação Orientada a Objetos

Arrays de objetos e cadastros

Ely – ely.miranda@ifpi.edu.br

1

Array de Objetos

- Relacionamentos “1 para N” podem ser representados como array de objetos;
- Todo array de objetos é na verdade um array de referências:

```
let contas : Conta[] = [];
```


2

Array de Objetos

- Os elementos podem ser acessados como em arrays de tipos primitivos:

```
let contas : Conta[] = [];

let c1 : Conta = new Conta("1", 10);
contas[0] = c1;
console.log(contas[0].saldo); // 10
```


Array de Objetos

- Criar um array de objetos cria apenas o array, não seus objetos internos;

```
let contas : Conta[] = [];
console.log(contas[0]); // undefined
```

Array de Objetos

- Os objetos devem ser inicialmente instanciados e só depois atribuídos às referências do array:

```
let contas: Conta[] = [];
let c1 : Conta = new Conta("1",1000);
contas[0] = c1;
// ou
//contas[0] = new Conta("1", 1000);
```


Trabalhando com cadastros

- Tipicamente:
  - há classes **básicas** de um sistema que representam entidades do mundo real;
  - classes que **gerenciam** as classes básicas;

Trabalhando com cadastros

- Cadastros ou repositórios têm como funções meios de salvar/persistir de objetos:
  - Meios não persistentes: arrays, listas e pilhas, árvores;
  - Meios persistentes: arquivos ou bancos de dados;
- Representam um relacionamento “contém”.

7

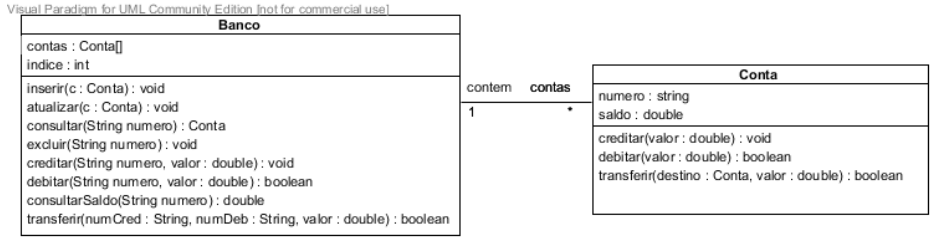
Trabalhando com cadastros

- Operações comuns são: consultar, adicionar, alterar, excluir;
- Como atributos, têm sempre:
  - Uma “coleção” de objetos representadas por arrays ou outra de estrutura de dados;
  - Outros como: índices, classes de conexão;
- Dizemos que um cadastro **contém** várias classes básicas.

8

Um banco de contas

- Um banco tem um array de contas;
- Possui um índice para manter a posição atual do array;
- Possui métodos de “cadastro” e métodos aplicáveis a contas como depositar, debitar...



Construtores omitidos no diagrama UML

"Esqueleto" de um cadastro

```
class Banco {
    inserir(c : Conta): void {...}
    alterar(c : Conta): void {...}
    consultar(numero: String): Conta  {...}
    excluir(numero: String): void {...}
    depositar(numero: String, valor: number):  void  {...}
    sacar(numero: String, valor: number):  void  {...}
    consultarSaldo(numero: String): number  {...}
    transferir(numeroCredito: String,
               numeroDebito:String,
               valor: number): void {...}
    constructor() {...}
 }
```

Banco

```
class Banco {
    contas: Conta[] = [];
}
```


"Esqueleto" de um cadastro

- Inserir:
  - Insere um objeto no array;
  - Testa se já **não** existe um repetido;
  - Testa se o objeto a ser inserido possui os requisitos mínimos:
    - atributos preenchidos e valores aceitáveis e consistentes (opcional, até o momento);
  - Sinônimos: cadastrar, persistir, salvar...

Classe Banco: método inserir

```
class Banco {
    contas: Conta[] = [];

    inserir(c: Conta): void {
        this.contas.push(c);
    }
}
```


Classe Banco: método consultar

- Consultar:
  - Consulta um objeto pelo número percorrendo o array;
  - Retorna o objeto se encontrado, do contrário, retorna nulo;
  - É utilizado em métodos como inserir, alterar, excluir e outros;
  - Sinônimos: pesquisar, localizar...

Classe Banco: método consultar

```
consultar(numero: String): Conta {
    let contaProcurada!: Conta;
    for (let c of this.contas) {
        if (c.numero == numero) {
            contaProcurada = c;
            break;
        }
    }

    return contaProcurada;
}
```


Classe Banco: método consultar índice

```
consultarIndice(numero: String): number {
    let indice: number = -1;
    for (let i: number = 0; i < this.contas.length; i++) {
        if (this.contas[i].numero == numero) {
            indice = i;
            break;
        }
    }

    return indice;
}
```

Classe Banco: método alterar

- Alterar:
  - Altera ou substitui um elemento já cadastrado;
  - Testa se o objeto **já** existe usando o consultar
  - Testa se o objeto a ser alterado possui os requisitos mínimos:
    - atributos preenchidos;
    - valores aceitáveis e consistentes (opcional, por enquanto);
  - Sinônimos: persistir, salvar, atualizar...

Classe Banco: método alterar

```
alterar(c: Conta): void {
    let indice =
                this.consultarIndice(c.numero);

    if (indice != -1) {
        this.contas[indice] = c;
    }
}
```

Classe Banco: método excluir

- Excluir:
  - Remove um objeto de uma coleção
  - Testa se o objeto existe
  - Sinônimos: deletar, remover, apagar...
  - No caso específico de um estático:
    - Descobrir o índice da Conta a ser excluída;
    - Deve-se preocupar em sobrepor as contas após a conta excluída.

Classe Banco: método excluir

```
excluir(numero: String): void {
   let indice: number = this.consultarIndice(numero);
   if (indice != -1) {
      for (let i: number = indice; i < this.contas.length; i++) {
          this.contas[i] = this.contas[i + 1];
      }
      this.contas.pop();
   }
}
```

Classe Banco: Depositar

```
depositar(numero: String, valor: number) {
      let conta: Conta = this.consultar(numero);

      if (conta != null) {
          conta.depositar(valor);
      }
 }
```


21

Utilizando a classe Banco

```
let b: Banco = new Banco();
b.inserir(new Conta("1", 100));
b.inserir(new Conta("2", 200));
console.log(b.consultar("2").saldo); //200
console.log(b.consultarIndice("1")); //0

let c2 = new Conta("2", 300);
b.alterar(c2);
console.log(b.consultar("2").saldo); //300
b.inserir(new Conta("3", 300));
b.excluir("1");
console.log(b.consultarIndice("3")); //1
b.depositar("3", 100);
console.log(b.consultar("3").saldo); //400
```


22

Indo além de um cadastro...

- Regras de negócio:
  - Métodos especiais que realizam operações desde que determinadas condições sejam rigorosamente cumpridas:
    - Ex: creditar, debitar, transferir...
    - Emprestar apenas se:
      - Um livro estiver disponível
      - Se o usuário não estiver com um mesmo exemplar alugado
      - Não houver reservas
      - O usuário não tiver mais que N livros já emprestados
      - Etc

Indo além de um cadastro...

- Regras de validação:
  - Métodos que formatos e preenchimento:
    - Campos obrigatórios preenchidos
    - Formatos válidos como em uma placa de carro (LLL-NNNN)
    - CPF preenchidos com dígitos verificadores...

Programação Orientada a Objetos

Arrays de objetos e cadastros

Ely – ely.miranda@ifpi.edu.br

25