

Programação Orientada a Objetos

Arrays, Strings, arrays de objetos, cadastros

Ely – ely.miranda@ifpi.edu.br

Cuidado!

Tente fazer os códigos sem copiar e colar, pois dependendo das configurações, podem ir junto do texto caracteres especiais que irão gerar erros no código.

Arrays

- Conjunto de tipos primitivos ou de objetos indexáveis da posição 0 a (tamanho-1)
- Para utilizar arrays em Java deve-se:
 - declarar uma variável que conterá o array;
 - Instanciar um novo objeto array;
 - referenciar a variável ao objeto;
 - armazenar dados no array;

ely.miranda@ifpi.edu.br

3

Declarando um array

- Sintaxe:
`<tipo>[] <nome>; //ou`
`<tipo> <nome>[];`
- Onde:
 - **tipo**: classe ou tipo primitivo da linguagem;
 - **nome**: identificador válido;
 - Ex:
`int[] notas;`

Obs:

- Um array é sempre um objeto.
- A variável **notas** é uma referência.

ely.miranda@ifpi.edu.br

4

Criando um array

- Instanciando:

```
int[] notas = new int[10];
```

- Inicializando o array de forma estática:

```
int[] notas = {1,3,5,7};
```

```
String[] strArr = {"teste", "str1",  
"str2"};
```

ely.miranda@ifpi.edu.br

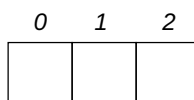
5

Acessando os dados

- Os índices vão de 0 a (tamanho – 1)
- São verificados em tempo de compilação para que não ultrapassem os limites;
- O acesso a um índice inválido do array gera um erro chamado `IndexOutOfBoundsException`

Ex: 3 posições: índices de 0 a 2

```
int[] numeros = new int[3];
```



ely.miranda@ifpi.edu.br

6

Arrays

- Para obter o número de elementos em um array acesse a propriedade ***length***;
System.out.println(numeros.length);
- Um array não pode mudar de tamanho.
- Para usar mais posições deve-se:
 - criar um novo array com um novo tamanho
 - preenchê-lo com mais elementos do antigo array

ely.miranda@ifpi.edu.br

7

Percorrendo Arrays

```
public class PercorrerArrays {  
    public static void main(String args[]) {  
        int[] numeros = new int[10];  
        for (int i = 0; i < numeros.length; i++) {  
            numeros[i] = i * 10;  
        }  
        for (int i = 0; i < numeros.length; i++) {  
            System.out.println(numeros[i]);  
        }  
    }  
}
```

ely.miranda@ifpi.edu.br

8

Percorrendo Arrays

```
public class PercorrerArrays {  
    public static void main(String args[]) {  
        int[] numeros = new int[10];  
        for (int i = 0; i < numeros.length; i++) {  
            numeros[i] = i * 10;  
        }  
        // Usando enhanced-for  
        for (int numero: numeros) {  
            System.out.println(numero);  
        }  
    }  
}
```

ely.miranda@ifpi.edu.br

9

Arrays multidimensionais

- A colocação de [] adicionais na declaração do array permite a declaração de arrays multidimensionais:

```
int[][] duasDim = new int[120][16];
```

- Arrays multidimensionais não precisam ser retangulares:

```
int[][] duasDim = {{1,2},{3,4,5},{6,7,8,9}};
```

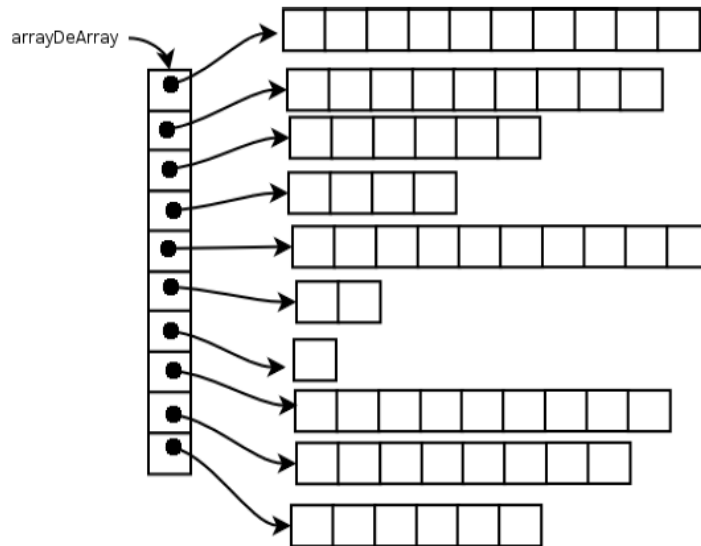
- Para acessar elementos de um array use o nome da variável seguido de “[”, o índice do elemento e “]”

```
int tmp = duasDim[1][2];
```

ely.miranda@ifpi.edu.br

10

Arrays multidimensionais



ely.miranda@ifpi.edu.br

11

Strings

- São instâncias da classe String;
- Não precisam ser instanciadas ;
- Tipo de dados especial em Java:
 - São tratadas como se fossem tipos primitivos;
 - São mantidas internamente como arrays de caracteres;
 - Possuem métodos utilitários para operações com texto;
- O operador "+" é sobrecarregado:
 - Usado para concatenar Strings;

```
String s = "Bom";  
s = s + " dia";
```

ely.miranda@ifpi.edu.br

12

Coomparação entre Objetos

- **equals()**: compara duas Strings caractere a caractere
 - A priori, objetos não são comparados com == ou !=;
 - A partir do Java 7.0 as Strings podem ser comparadas com == ;
 - Existe um método especial chamado equals que retorna verdadeiro ou false;
 - Ex:

```
String nome = "João";  
if (nome.equals("José")) {  
    ...  
}
```

ely.miranda@ifpi.edu.br

13

Alguns métodos úteis

- **valueOf()**: converte valores de vários tipos para a representação em String
- **toUpperCase()**, **toLowerCase()**: retorna a String em caixa alta (maiúsculas) e em caixa baixa (minúsculas);
- **charAt()**: retorna o caractere em uma determinada posição da String;
- **split(char c)**: quebra uma String em um array a cada ocorrência do caractere c
- **indexOf(char c)**: retorna o índice do caractere c
- **length()**: retorna o tamanho da String.
- **substring()**: retorna uma substring da String original.
- **trim()**: elimina espaços em branco no início e no fim de uma String.

ely.miranda@ifpi.edu.br

14

Exemplo

```
public class TesteStrings {
    public static void main(String[] args) {
        String s = "Instituto Fedearal do Piaui";
        System.out.println(s.toUpperCase());
        System.out.println(s.toLowerCase());
        System.out.println(s.charAt(2));
        System.out.println(s.length());
        System.out.println(s.substring(5));
        System.out.println(s.substring(0,4));
        String[] stringDividida = "Instituto Federal do
        Piaui".split(" ");
        System.out.println(stringDividida[0] + "-" +
            stringDividida[1] + "-" +
            stringDividida[2]);

        System.out.println("-" + " a ".trim() + "-");
    }
}
```

ely.miranda@ifpi.edu.br

15

Array de Objetos

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

```
Conta[] contas;
contas = new Conta[10];
```

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

```
Conta c1 = new Conta(1,1000);
contas[0] = c1;
double saldo = contas[0].saldo;
```

ely.miranda@ifpi.edu.br

16

Array de Objetos

- Instanciar um array de objetos cria apenas o array, não seus objetos internos;
- O código abaixo cria um espaço para armazenar 10 contas, ou seja, referências para Contas:
 - Atualmente elas não referenciam nada (null);
 - Caso acessemos, um dos itens, teremos `NullPointerException`:

```
Conta[] contas;  
contas = new Conta[10];  
contas[0].saldo = 100;
```

ely.miranda@ifpi.edu.br

17

Array de Objetos

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

```
Conta[] contas;  
contas = new Conta[10];  
Conta c1 = new Conta(1, 1000);  
contas[0] = c1;  
  
// ou  
contas[0] = new Conta(1, 1000);
```

ely.miranda@ifpi.edu.br

18

Trabalhando com cadastros

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

ely.miranda@ifpi.edu.br

19

Trabalhando com cadastros

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

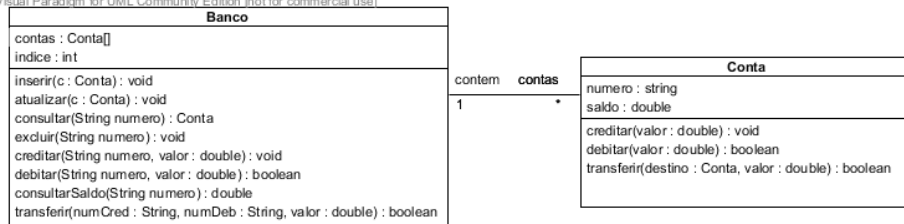
ely.miranda@ifpi.edu.br

20

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 creditar, debitar...

Visual Paradigm for UML Community Edition [not for commercial use]



Construtores omitidos no diagrama UML

ely.miranda@ifpi.edu.br

21

"Esqueleto" de um cadastro

```



class Banco {
    void inserir(Conta c) {...}
    void alterar(Conta c) {...}
    Conta consultar(String numero) {...}
    void excluir(numero) {...}
    void creditar(String numero, double valor) {...}
    void debitar(String numero, double valor) {...}
    double consultarSaldo(String numero) {}
    void transferir(String numCred,
                   String numDeb,
                   double valor) {...}

    Banco () {...}
    //...
}
    
```

ely.miranda@ifpi.edu.br

22

Banco: atributos e construtor

```
class Banco {  
    Conta[] contas;  
    int indice;  indice é inicializado com 0  
  
    Banco(int tamanho)  O tamanho do array é configurado no construtor {  
        contas = new Conta[tamanho];  
    }  
}
```

ely.miranda@ifpi.edu.br

23

Testando a classe Banco

```
public class TestaBanco {  
    public static void main(String[] args) {  
        Banco banco = new Banco(10);  
        //...  
    }  
}
```

ely.miranda@ifpi.edu.br

24

"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...

ely.miranda@ifpi.edu.br

25

Classe Banco: método inserir

```
public class Banco {  
    //demais campos omitidos  
    void inserir(Conta c) {  
        contas[indice] = c;  
        indice++;  
    }  
}
```

ely.miranda@ifpi.edu.br

26

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...

ely.miranda@ifpi.edu.br

27

Classe Banco: método consultar

```
Conta consultar(String numero) {  
    Conta c = null;  
    for (int i=0; i < indice; i++) {  
        if (contas[i].numero.equals(numero)) {  
            c = contas[i];  
            break;  
        }  
    }  
    return c;  
}
```

ely.miranda@ifpi.edu.br

28

Classe Banco: método consultar índice

```
int consultarIndice(String numero){  
    int pos = -1;  
    for(int i = 0; i < indice; i++){  
        if(contas[i].numero.equals(numero)){  
            pos = i;  
            break;  
        }  
    }  
    return pos;  
}
```

ely.miranda@ifpi.edu.br

29

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...

ely.miranda@ifpi.edu.br

30

Classe Banco: método alterar

```
void alterar(Conta c) {  
    int indice = consultarIndice(c.numero);  
    if (indice != -1) {  
        contas[indice] = c;  
    }  
}
```

ely.miranda@ifpi.edu.br

31

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 array estático:
 - Descobrir o índice da Conta a ser excluída;
 - Fazer com que essa conta seja nula;
 - Deve-se preocupar em sobrepor as contas após a conta excluída;
 - Decrementar o índice;

ely.miranda@ifpi.edu.br

32

Classe Banco: método excluir

```
void excluir(String numero){
    int posicao = consultarIndice(numero);
    if( posicao != -1){
        for(int i = posicao; i < indice; i++){
            contas[i] = contas[i+1];
        }
        indice--;
    }
}
```

ely.miranda@ifpi.edu.br

33

Debitar

```
void debitar(String numero,
              double valor) {
    Conta c = consultar(numero);
    if (c != null)
        c.debitar(valor);
    else
        System.out.println("Conta
        Inexistente.");
}
```

ely.miranda@ifpi.edu.br

34

Utilizando a classe Banco

```
public class ExecutaBanco {  
    public static void main(String args[]) {  
        ...  
        Banco b = new Banco();  
        Conta c1 = new Conta("111", 50.00);  
        b.inserir(c1);  
        b.debitar(c1.numero, 10.00);  
        ....  
        Conta c2 = b.consultar("111");  
        ...  
    }  
}
```

ely.miranda@ifpi.edu.br

35

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
- 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...

ely.miranda@ifpi.edu.br

36

Programação Orientada a Objetos

Arrays, Strings, arrays de objetos, cadastros

Ely – ely.miranda@ifpi.edu.br