ETE
ESCOLA TÉCNICA ESTADUAL
PORTO DIGITAL

GOVERNO DO ESTADO
Pernambuco
Secretaria de Educação

Desenvolvimento Aplicado a Desktop

Pedro Albino

Array

Um Array é uma estrutura de dados que armazena **uma coleção sequencial de tamanho fixo de elementos do mesmo tipo.**

Os arrays são usados para armazenar uma coleção de dados, porém é interessante pensar neles como sendo coleções de variáveis do mesmo tipo.

Array

Algumas Características dos Arrays em Java

- **Uma variável array em Java pode ser declarada como qualquer outra variável, acrescentando colchetes [ ]** após o tipo de dados declarado.
- **Arrays são objetos em Java.** Assim, podemos descobrir seu tamanho usando o atributo length do objeto.
- Os elementos em um array **são indexados a partir de zero**.
- Um array pode ter **uma ou mais dimensões.**
- O **tamanho de um array sempre deve ser um valor int,** e nunca um long, short ou byte.

Array

Criando um array unidimensional

Declaramos um array de uma dimensão da seguinte forma:

- **tipo nomeArray[]** ou **tipo[] nomeArray**

O **tipo** determina qual será o tipo de dados de todos os elementos que serão armazenados nas posições do array.

Exemplo de declaração de array:

- **double[] salarios;**
- **double salarios[];** // Ambas as formas são válidas.

Array

Onde tamanho se refere ao número de elementos do array, e nomeArray é o nome da variável que será ligada ao array físico. Quando usamos o operador new devemos especificar o número de elementos que serão alocados no array.

Portanto, criar um array em Java é um processo em duas etapas: Declaramos uma variável do tipo desejado para o array, e então alocamos memória para efetivamente armazenar o array, atribuindo a variável criada.

Exemplo:

```
double[] salarios; // Declarar o array

salarios = new double[50]; // Alocar memória para o array
```

Array

Podemos ainda declarar a variável de referência, criar o array, e atribuir a referência do array à variável de uma só vez, como segue:

tipoDado[] variávelReferência = new tipoDado[tamanhoArray];

Em nosso exemplo:

double[] salarios = new double[50];

Assim, declaramos e criamos um array de 50 posições para armazenar dados do tipo double.

Array

Array literal

Quando sabemos o tamanho do array e quais serão os valores que serão armazenados nas posições do array, podemos usar array literais podemos atribuir esses valores diretamente na declaração do array, usando uma lista de inicialização. Para isso podemos usar a seguinte sintaxe:

tipoDado[] variavelReferencia = {valor1, valor2, …, valorN};

Array

Por exemplo, **suponha um array de 5 posições do tipo inteiro**, do qual sabemos que será necessário armazenar os números 5, 9, 12, 3 e 4. podemos declarar esse array da seguinte forma:

int[] numeros = { 5, 9, 12, 3, 4 };

Desta forma o array é criado, e suas cinco posições já serão preenchidas com os dados desejados.

Array

Como acessar os elementos de um array em Java

Os elementos em um array são sempre acessados por meio de seu número de índice (que é a posição do elemento). Esse índice sempre se inicia em zero, e termina em tamanhoArray – 1.

Por exemplo, um array de dez posições tem a sua primeira posição com índice zero e sua última posição com 10 -1 = índice 9. Portanto, se trata de um array cujas posições vão de 0 a 9.

Array

Podemos acessar um elemento individual em um array simplesmente indicando o número de índice da posição desejada (entre colchetes), junto ao nome do array em si. Veja o exemplo:

Array

Arra.java

```
1
2  public class Arra {
3
4      public static void main(String[] args) {
5          // TODO Auto-generated method stub
6          // Criar o array e atribuir-lhe valores a partir de uma lista de inicialização
7          double[] valores = {4.5, 5.9, 4.1, 2.0, 8.9, 6.3, 7.8, 5.3, 1.2, 0.8 };
8          // Acessando seu quinto elemento (número de posição 4)
9          System.out.println(valores[4]);
10
11     }
12
13 }
```

Console

```
<terminated> Arra [Java Application] C:\Users\Pedro Albino\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20200815-0932\jre\bin\javaw.exe (
8.9
```

Array

O mesmo vale para a atribuição de valores ao array. Por exemplo, vamos modificar os valores das posições de índices 4 e 6 para 2.3 e 7.1, respectivamente:

```
valores[4] = 2.3;

valores[6] = 7.1;
```

E então acessamos os elementos para verificar as mudanças realizadas:

Array

```
public class Arra {

	public static void main(String[] args) {

		// TODO Auto-generated method stub

		// Criar o array e atribuir-lhe valores a partir de uma lista de inicialização

		double[] valores = {4.5, 5.9, 4.1, 2.0, 2.3, 6.3, 7.1, 5.3, 1.2, 0.8 };

		// Acessando seu quinto elemento (número de posição 4)

		System.out.println(valores[4]);

		System.out.println(valores[6]);

	}

}
```

Array

Acessando todas as posições de uma vez via laço for

Podemos acessar todas as posições de um array usando um simples laço for e a propriedade length do array, como segue:

```
for (int i = 0; i < nomeArray.length; i++) {

  Código a executar para cada elemento

}
```

Array

Exemplo: Vamos criar um array de números double e acessar seus elementos usando um laço for.

Array

```
public class Arra1 {

	public static void main(String[] args) {

		// TODO Auto-generated method stub

		double[] valores = { 4.5, 5.9, 4.1, 2.0, 8.9, 6.3, 7.8, 5.3, 1.2, 0.8 };

		for (int i = 0; i < valores.length; i++) {

			System.out.println("Elemento " + i + " = " + valores[i]);

		}

	}

}
```

Array

```
Elemento 0 = 4.5
Elemento 1 = 5.9
Elemento 2 = 4.1
Elemento 3 = 2.0
Elemento 4 = 8.9
Elemento 5 = 6.3
Elemento 6 = 7.8
Elemento 7 = 5.3
Elemento 8 = 1.2
Elemento 9 = 0.8
```

Listas

A **List** em Java é uma interface que se comporta de forma muito semelhante a um array.

É uma coleção ordenada (também conhecida como sequência).

O usuário desta interface tem controle preciso sobre onde cada item é inserido na lista.

O usuário pode acessar os itens por seu índice inteiro (posição na lista).

O usuário pode pesquisar itens na lista percorrendo os itens nela.

Na verdade, a classe mais comum que implementa a interface **List** usa um array internamente e é até chamado **ArrayList**.

Listas

ArrayList

É uma classe que implementa a interface **List**. Fornece métodos realmente poderosos que tornam o tratamento do array muito mais simples.

Um item em um ArrayList é conhecido como um elemento.

Listas

Vamos dar uma olhada em alguns dos métodos do ArrayList:

- **add(elemento E):** acrescenta o elemento especificado ao final desta lista.
- **add(int index, elemento E):** Acrescenta o elemento especificado ao índice especificado desta lista.
- **get (int index):** Retorna o elemento na posição especificada nesta lista.
- **contains(Object o):** Retorna verdadeiro se esta lista contiver o elemento especificado.
- **remove(int index):** Remove o elemento na posição especificada nesta lista.
- **size():** Retorna o número de elementos na lista.

Listas

```
import java.util.ArrayList;

public class Lista {

        public static void main(String[] args) {

        String aula1 = "Modelando a classe Aula";    String aula2 = "Conhecendo mais de listas";          String
aula3="Trabalhando com Cursos e Sets";

                ArrayList<String> aulas = new ArrayList<String>();

                aulas.add(aula1);

                aulas.add(aula2);

                aulas.add(aula3);

                System.out.println(aulas);

        }
```

Listas

[Modelando a classe Aula, Conhecendo mais de listas, Trabalhando com Cursos e Sets]

Listas

Para criar e inicializar um ArrayList:

ArrayList grades = new ArrayList();

Em seguida, você pode adicionar elementos chamando o add()método:

grades.add(100);

grades.add(97);

grades.add(85);

Para acessar o primeiro item da lista:

grades.get(0);

Isso retornará o valor armazenado no índice 0 (neste caso: "100");

Listas

```
import java.util.ArrayList;

public class Lista1 {

	public static void main(String[] args) {

		ArrayList grades = new ArrayList();

		grades.add(100);

		grades.add(97);

		grades.add(85);

		System.out.println(grades.get(0));

	}

}
```

Listas

A qualquer momento, você pode verificar o tamanho de uma matriz usando o **size()** método:

System.out.println(grades.size());

Isso imprime o número de elementos na lista (neste caso: "3")

Você também pode remover itens por índice:

grades.remove(0);

O que removerá o primeiro item da lista com índice = 0 ("100") e, em seguida, deslocará os outros 2 itens ("97" e "85") para ter os índices (0 e 1) respectivamente;

Você pode até limpar a lista inteira chamando o **clear()** método:

grades.clear();

Listas

Uma solução melhor e recomendada é utilizar uma **ArrayList** em vez de uma array, uma vez que é redimensionável. Não há um tamanho fixo de **ArrayList**, portanto sempre que houver a necessidade de adicionar um novo elemento, você pode simplesmente adicionar executando **testList.add(element)**.

Listas

```
import java.util.ArrayList;

public class Lista1 {

	public static void main(String[] args) {

	ArrayList<String>testList= new ArrayList<String>();

	testList.add("1");	testList.add("2");	testList.add("3");	testList.add("4");	testList.add("5");

	System.out.println(testList);

	testList.add("6");

	testList.add("7");

	System.out.println(testList);

   }

}
```

Listas

[1, 2, 3, 4, 5]

[1, 2, 3, 4, 5, 6, 7]

Listas

Ordenando a lista

A classe **java.util.Collections** é um conjunto de métodos estáticos auxiliares as coleções. Dentro dela há o método sort:

Collections.sort(cadastro);

Listas

Salvar na lista =

```
import java.util.ArrayList;
import java.util.Scanner;

public class ListaNova {

    public static void main(String[] args) {

        String nome = null;
        int i =0;

        Scanner s = new Scanner(System.in);

        ArrayList<String> cadastro = new ArrayList<>();

        while(i == 0 ) {
            System.out.println("Informe o nome para cadastro");
            nome = s.next();

            cadastro.add(nome);

            System.out.println("Digite 1 para sair ou 0 para continuar");
            i = s.nextInt();
        }

        System.out.println(cadastro);
    }

}
```

Listas

Lista ordenada =

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

public class Lista {
    public static void main(String[] args) {

        String nome = null;
        int i = 0;
        Scanner s = new Scanner(System.in);

        ArrayList<String> cadastro = new ArrayList<String>();

        while(i == 0) {
            System.out.println("Informe o nome para cadastro");
            nome = s.next();

            cadastro.add(nome);
            System.out.println(nome);

            System.out.println("Digite 1 para sair ou 0 para continuar");
            i = s.nextInt();
        }
            System.out.println(cadastro);

            Collections.sort(cadastro);

            System.out.println(cadastro);

    }
}
```

Listas

Lista tamanho =

```
import java.util.ArrayList;
import java.util.Scanner;

public class ListaNova {

    public static void main(String[] args) {

        String nome = null;
        int i =0;

        Scanner s = new Scanner(System.in);

        ArrayList<String> cadastro = new ArrayList<>();

        while(i == 0 ) {
            System.out.println("Informe o nome para cadastro");
            nome = s.next();

            cadastro.add(nome);

            System.out.println("Digite 1 para sair ou 0 para continuar");
            i = s.nextInt();
        }

        System.out.println(cadastro);
        System.out.println("O tamanho é : " + cadastro.size());
    }

}
```

Listas

Lista remover =

```
import java.util.ArrayList;
import java.util.Scanner;

public class ListaNova {

    public static void main(String[] args) {

        String nome = null;
        int i =0;

        Scanner s = new Scanner(System.in);

        ArrayList<String> cadastro = new ArrayList<>();

        while(i == 0 ) {
            System.out.println("Informe o nome para cadastro");
            nome = s.next();
            cadastro.add(nome);
            System.out.println("Digite 1 para sair ou 0 para continuar");
            i = s.nextInt();
        }

        System.out.println(cadastro);
        System.out.println("Remover o primeiro elemento : " + cadastro.remove(0));
        System.out.println(cadastro);
    }

}
```

Listas

Lista contem =

```java
import java.util.Scanner;

public class ListaContem {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String nome = null;
        int i =0;

        Scanner s = new Scanner(System.in);

        ArrayList<String> cadastro = new ArrayList<>();

        while(i == 0 ) {
            System.out.println("Informe o nome para cadastro");
            nome = s.next();
            cadastro.add(nome);
            System.out.println("Digite 1 para sair ou 0 para continuar");
            i = s.nextInt();
        }

        System.out.println(cadastro);
        System.out.println(cadastro.contains("Pedro"));
        System.out.println(cadastro);

    }

}
```

Listas

Lista adicionar na primeira posição =

```
import java.util.Scanner;

public class ListaContem {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String nome = null;
        int i =0;

        Scanner s = new Scanner(System.in);

        ArrayList<String> cadastro = new ArrayList<>();

        while(i == 0 ) {
            System.out.println("Informe o nome para cadastro");
            nome = s.next();
            cadastro.add(nome);
            System.out.println("Digite 1 para sair ou 0 para continuar");
            i = s.nextInt();
        }

        System.out.println(cadastro);
        cadastro.add(0,"Teste");
        System.out.println(cadastro);

    }

}
```

Listas

Exercício

1 - Criar uma lista de estudantes e exibir em seguida os nomes;

Listas

Exercício

2 - Criar uma lista de estudantes e em seguida os nomes de forma ordenada;

Listas

Exercício

3 - Criar uma lista de estudantes e em seguida remover o primeiro nome;