MC322 - Programação orientada a objetos
Aula 3.2
Arrays

Prof. Marcos M. Raimundo
Instituto de Computação - UNICAMP

Arrays

Imagine que queremos armazenar a idade de 4 pessoas. Podemos declarar:

```
int idade1;
int idade2;
int idade3;
int idade4;
```

Agora, considere que queremos armazenar a idade de todos os alunos da sala:

```
int idade1;
int idade2;
int idade3;
int idade4;
...
int idade67;
```

Arrays

- Uma forma de solucionar este problema, é através do uso de variáveis indexadas: Arrays
  - A inclusão de índice facilita o tratamento destas variáveis
- Arrays são estruturas de dados que armazenam, usualmente:
  - uma quantidade fixa de dados de um mesmo tipo (tipos primitivos ou referenciados)
  - são conhecidos como estruturas homogêneas de dados
- Em Java, um array é sempre um objeto, portanto, são tipos referenciados e precisam ser instanciados
  - Como qualquer outro objeto, um array pode ser instanciado com a palavra chave new
  - é preciso especificar na instanciação o tipo e o número de posições do array

```
int[] idades; //<- Declara o array de inteiros
idades = new int[50]; // Instancia o array de inteiros com 50 elementos.
```

Arrays

é possível colocar os colchetes ([]) tanto antes quanto depois do nome da variável

```
int[] idades;
// ou
int idades[];
```

O primeiro caso é normalmente usado quando temos diversas variáveis e queremos que todas sejam vetores.

```
int[] idades, vetor2, vetor3;
```

Acessando um elemento

- Para obter um certo elemento de um array, devem ser especificados:
  - o nome da variável que referencia o array
  - a posição relativa do elemento no array (índice)

```
for (int i=0; i<50; ++i){
    idades[i] = 0;
}
```

  - o array é indexado de 0 até (TAM-1)

Acessando um elemento

- Um objeto array possui como atributo público seu próprio tamanho (length)
  o tamanho de um array não pode ser modificado (length é um atributo final)

```
public class Vetores {
  public static void main (String args[]) {
    int[] idades = new int[50];
    System.out.println("Tamanho do array idades: " + idades.lenght);

    for (int i=0; i<50; ++i){
      idades[i] = 0;
    }
  }
}
```

```
> Tamanho do array idades: 50
```

Arrays

Java permite a inicialização de vetores no momento da declaração, por exemplo:

```
public class Vetores{
  public static void main (String args[]){
    String[] meses = {"Janeiro", "Fevereiro", "Marco", "Abril", "Maio", "Junho",
                      "Julho", "Agosto", "Setembro", "Outubro", "Novembro", "Dezembro"};
    int[] diaMes = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}

    System.out.println("Tamanho do array meses: "+meses.length);
    System.out.println("Tamanho do array meses: "+diaMes.lenght);
  }
}
```

```
Tamanho do array meses: 12
Tamanho do array diaMes: 12
```

O tamanho do array será igual ao número de elementos incializados

Arrays

Java possui uma estrutura for aprimorada para percorrer os elementos de um vetor sem utilizar um contador: for (parâmetro: nomeDoVetor)

instrução;

O parâmetro tem que ser compatível com o tipo do Array

```
public class Vetores{
  public static void main (String args[]){
    String[] meses = {"Janeiro", "Fevereiro", "Marco", "Abril", "Maio",
        "Junho",
                      "Julho", "Agosto", "Setembro", "Outubro", "Novembro",
                          "Dezembro"};
    int[] diaMes = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}

    System.out.println("Meses do ano: ");
    for (String i:meses){
      System.out.println("* " + i);
    }
  }
}
```

```
Meses do ano:
* Janeiro
* Fevereiro
* Marco
* Abril
* Maio
* Junho
* Julho
* Agosto
* Setembro
* Outubro
* Novembro
* Dezembro
```

O tamanho do array será igual ao número de elementos incializados

Arrays

A estrutura for aprimorada funciona apenas para um parâmetro

Exemplo com erro de compilação:

```
public class Vetores{
  public static void main (String args[]){
    String[] meses = {"Janeiro", "Fevereiro", "Marco", "Abril", "Maio", "Junho",
                      "Julho", "Agosto", "Setembro", "Outubro", "Novembro", "Dezembro"};
    int[] diaMes = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}

    System.out.println("Meses do ano: ");
    for (String i:meses; int j:diaMes){
      System.out.println("* " + i + "tem " + j + "dias");
    }
  }
}
```

Arrays

Neste caso, a melhor saída seria usar um for tradicional:

```
public class Vetores{
  public static void main (String args[]){
    String[] meses = {"Janeiro", "Fevereiro", "Marco", "Abril", "Maio",
         "Junho",
                      "Julho", "Agosto", "Setembro", "Outubro", "Novembro",
                           "Dezembro"};
    int[] diaMes = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}

    System.out.println("Meses do ano: ");
    for (int i=0; i<meses.lenght; ++i){
      System.out.println("* " + meses[i] + " tem " + diasMes[i] + "dias");
    }
  }
}
```

```
Meses do ano:
* Janeiro tem 31 dias
* Fevereiro tem 28 dias
* Marco tem 31 dias
* Abril tem 30 dias
* Maio tem 31 dias
* Junho 30 dias
* Julho 31 dias
* Agosto 31 dias
* Setembro 30 dias
* Outubro 31 dias
* Novembro 30 dias
* Dezembro 31 dias
```

Arrays - Array como parâmetro

A passagem do Array como parâmetro para uma função altera o seu conteúdo original visto que o array é uma variável referenciada

```
public class Vetores{
  public static void main (String args[]){
    String[] meses = {"Janeiro", "Fevereiro", "Marco", "Abril", "Maio",
        "Junho",
                      "Julho", "Agosto", "Setembro", "Outubro", "Novembro",
                           "Dezembro"};
    int[] diaMes = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}

    System.out.println("Meses do ano: ");
    for (int i=0; i<meses.lenght; ++i){
      System.out.println("* " + meses[i] + " tem " + diasMes[i] + "dias");
    }
    mudaVetor(diaMes);
    System.out.println("Meses do ano: ");
    for (int i=0; i<meses.lenght; ++i){
      System.out.println("* " + meses[i] + " tem " + diasMes[i] + "dias");
    }
  }
  public static void mudaVetor(int []v){
    for (int i=0; i<v.lenght; ++i)
      v[i] = 0;
  }
}
```

```
Meses do ano:
* Janeiro tem 31 dias
* Fevereiro tem 28 dias
* Marco tem 31 dias
* Abril tem 30 dias
* Maio tem 31 dias
* Junho 30 dias
* Julho 31 dias
* Agosto 31 dias
* Setembro 30 dias
* Outubro 31 dias
* Novembro 30 dias
* Dezembro 31 dias
Meses do ano:
* Janeiro tem 0 dias
* Fevereiro tem 0 dias
* Marco tem 0 dias
* Abril tem 0 dias
* Maio tem 0 dias
* Junho 0 dias
* Julho 0 dias
* Agosto 0 dias
* Setembro 0 dias
* Outubro 0 dias
* Novembro 0 dias
* Dezembro 0 dias
```

Cuidado com o for otimizado

```
public class Vetores{
  public static void main (String args[]){
    String[] meses = {"Janeiro", "Fevereiro", "Marco", "Abril", "Maio",
        "Junho",
                      "Julho", "Agosto", "Setembro", "Outubro", "Novembro",
                          "Dezembro"};
    int[] diaMes = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}

    System.out.println("Meses do ano: ");
    for (int i=0; i<meses.lenght; ++i){
      System.out.println("* " + meses[i] + " tem " + diasMes[i] + "dias");
    }
    mudaVetor(diaMes);
    System.out.println("Meses do ano: ");
    for (int i=0; i<meses.lenght; ++i){
      System.out.println("* " + meses[i] + " tem " + diasMes[i] + "dias");
    }
  }
  public static void mudaVetor(int []v){
    for (int i:v)
      i = 0;
  }
}
```

Por que não foi alterado?

```
Meses do ano:
* Janeiro tem 31 dias
* Fevereiro tem 28 dias
* Marco tem 31 dias
* Abril tem 30 dias
* Maio tem 31 dias
* Junho 30 dias
* Julho 31 dias
* Agosto 31 dias
* Setembro 30 dias
* Outubro 31 dias
* Novembro 30 dias
* Dezembro 31 dias
Meses do ano:
* Janeiro tem 31 dias
* Fevereiro tem 28 dias
* Marco tem 31 dias
* Abril tem 30 dias
* Maio tem 31 dias
* Junho 30 dias
* Julho 31 dias
* Agosto 31 dias
* Setembro 30 dias
* Outubro 31 dias
* Novembro 30 dias
* Dezembro 31 dias
```

Array de Referências

Considere agora que queremos criar um Array de objetos do tipo Aluno.

```
public class Aluno{
  private int matricula;
  private String nome;

  public Aluno(int matricula, String nome){
    this.matricula = matricula;
    this.nome = nome;
  }

  public int getMatricula(){
    return matricula;
  }

  public String getNome(){
    return nome;
  }

  @Override
  public String toString(){
    String out = "";
    out += "Aluno "+getNome()+"com matricula"+getMatricula()+"\n";
    return out;
  }
}
```

Array de Referências

Criamos um array para acomodar 5 alunos. Quantos alunos foram criados aqui? Por que o erro quando invocamos o método getNome() para o primeiro aluno do Array?

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

    Aluno alunos[] = new Aluno[5];
    System.out.println("Tamanho do array: "+alunos.length);
    System.out.println(alunos[0].getNome());
  }
}
```

```
> Exception in thread "main"
> Tamanho do array: 5
```

Array de Referências

Na verdade, nenhum aluno foi criado. Foram criados 5 espaços que você pode utilizar para guardar uma referência para um Aluno. Por enquanto, eles referenciam null.

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

    Aluno alunos[] = new Aluno[5];
    System.out.println("Tamanho do array: "+alunos.length);
    System.out.println(alunos[0]);
  }
}
```

```
> Tamanho do array: 5
> null
```

Array de Referências

Neste momento, todas as posições de alunos referenciam null.

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

    Aluno alunos[] = new Aluno[5];
    System.out.println("Tamanho do array: "+alunos.length);
    alunos[0] = new Aluno(123, "Andre");
    alunos[1] = new Aluno(456, "Joana");
    alunos[2] = new Aluno(789, "Pedro");
    System.out.println(alunos[0]);
    System.out.println(alunos[1]);
    System.out.println(alunos[2]);
    System.out.println(alunos[3]);
    System.out.println(alunos[4]);
  }
}
```

```
> Tamanho do array: 5
> Aluno Andre tem matricula #123
> Aluno Joana tem matricula #456
> Aluno Pedro tem matricula #789
> null
> null
```

Arrays copyOf

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

    Aluno alunos[] = new Aluno[5];
    System.out.println("Tamanho do array: "+alunos.length);
    alunos[0] = new Aluno(123, "Andre");
    alunos[1] = new Aluno(456, "Joana");
    System.out.println(alunos[0]);
    System.out.println(alunos[1]);
    System.out.println(alunos[2]);
    alunos = Arrays.copyOf(alunos, 10);
    System.out.println("**********");
    System.out.println("Tamanho do array: "+alunos.length);
    System.out.println(alunos[1]);
    System.out.println(alunos[5]);
  }
}
```

```
> Tamanho do array: 5
> Aluno Andre tem matricula #123
> Aluno Joana tem matricula #456
> null
> **********
> Tamanho do array: 10
> Aluno Joana tem matricula #456
> null
```

Array Multidimensional

Um array multidimensional

- cada elemento do array é uma referência para outro array
- estrutura de dados com pelo menos duas dimensões
- um array bidimensional (matriz) pode ser utilizado para representar valores armazenados em uma tabela organizada em linhas e colunas
- Formato da declaração:

```
<tipo> [][] <nome> = new <tipo> [<#linhas>][<#colunas>];
```

Array Multidimensional

Declaração e inicialização

- Um array bidimensional pode ser declarado e inicializado com valores definidos pelo programador

```
public class Vetores {
  public static void main(String args[]){
    int [][] distancias = {{10, 20, 30}, {30, 40, 50}, {30, 40, 50}}

    System.out.println("Tamanho de distancias: "+distancias.lenght);
  }
}
```

```
Tamanho de distancias: 4
```

Array Multidimensional

A variável length indica o número de linhas

```
public class Vetores {
  public static void main(String args[]){
    int [][] distancias = {{10, 20, 30}, {30, 40, 50}, {30, 40, 50}}

    System.out.println("Tamanho de distancias: "+distancias.lenght);
  }
}
```

```
Tamanho de distancias: 3
```

Array Multidimensional

As linhas não precisam ter o mesmo tamanho

- distancias[0] referencia um array de tamanho 3
- distancias [1] referencia um array de tamanho 2
- distancias [2] referencia um array de tamanho 4

```
public class Vetores {
  public static void main(String args[]){
    int [][] distancias = {{10, 20, 30}, {30, 40}, {30, 40, 50, 60}}

    System.out.println("Tamanho de distancias[0]: "+distancias[0].lenght);
    System.out.println("Tamanho de distancias[1]: "+distancias[1].lenght);
    System.out.println("Tamanho de distancias[2]: "+distancias[2].lenght);
  }
}
```

```
Tamanho de distancias[0]: 3
Tamanho de distancias[1]: 2
Tamanho de distancias[2]: 4
```

Array Multidimensional

Cuidado pois não será possível inicializar desta forma:

```
int[][] distancias = new int[3][];
distancias[0] = new int[3];
distancias[1] = new int[2];
distancias[1] = {30, 40, 50, 60}; // Nao permitido
```

Correto

```
int[][] distancias = new int[3][];
distancias[0] = new int[3];
distancias[1] = new int[2];
distancias[1] = new int[4];

for (int i=0; i<distancias.length; ++i){
  System.out.println("\nLinha " + i + ": ");
  for (int j=0; j<distancias[i].lenght; ++j){
    System.out.println(distancias[i][j] + " ");
  }
}
```

Lista de argumentos variável

Com uma lista de argumentos de tamanho variável é possível criar métodos que recebem um número de argumentos definidos em tempo de execução

- A lista de argumentos deve conter o tipo do argumento seguido de reticências (...) e a identificação da lista de parâmetros
- Um método pode conter no máximo uma lista variável de parâmetros
- O argumento com variável deve ser o último da lista de argumentos

```
public void meuMetodo(String arg1, Object ... args){}
```

Lista de argumentos variável

Exemplo de uso da linguagem:

```
public void meuMetodo(String arg1, Object ... args){}
```

Exemplo:

```
public class VarArg{
  public static void main (String args[]){
    float f1 = 3.6f;
    float f2 = 2.5f;
    float f3 = 9.7f;

    System.out.println("Media de f1 e f2: "+media(f1, f2));
    System.out.println("Media de f1, f2 e f3: "+media(f1, f2, f3));
  }

  public static float media(float...f){
    float media = 0.0f;
    for (float fi:f)
      media += fi;
    media /= f.length;
    return media;
  }
}
```

Por que não foi alterado?

```
Media de f1 e f2: 3.05
Media de f1, f2 e f3: 5.26666
```

Arrays da API Java

Na API Java a classe Arrays encontra-se no pacote java.util e possui diversos métodos estáticos para manipulações típicas de arrays

- Os métodos foram sobrecarregados para arrays de tipos primitivos e de objetos
- Alguns métodos permitem
  ordenação, busca binária, preenchimento, comparação e cópia

Arrays da API Java

Usando os métodos:

```
public class Vetores{
  public static void main (String args[]){
    String[] meses = {"Janeiro", "Fevereiro", "Marco", "Abril", "Maio",
        "Junho",
                      "Julho", "Agosto", "Setembro", "Outubro", "Novembro",
                          "Dezembro"};
    int[] diaMes = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}

    Arrays.sort(meses);

    System.out.println("Meses do ano: ");
    for (String i:meses){
      System.out.println("* " + i);
    }
  }
}
```

```
Meses do ano:
* Abril
* Agosto
* Dezembro
* Janeiro
* Fevereiro
* Junho
* Julho
* Maio
* Marco
* Outubro
* Novembro
* Setembro
```

Arrays da API Java

Usando os métodos:

```
public class Vetores{
  public static void main (String args[]){
    String[] meses = {"Janeiro", "Fevereiro", "Marco", "Abril", "Maio",
          "Junho",
                      "Julho", "Agosto", "Setembro", "Outubro", "Novembro",
                           "Dezembro"};
    int[] diaMes = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}

    System.out.println(Arrays.toString(meses));
    Array.sort(meses);
    System.out.println(Arrays.toString(meses));
    System.out.println("Outubro esta na posicao " +
          (Array.binarySearch(meses, "Outubro")+1) + "do Array");
  }
}
```

```
> Janeiro, Fevereiro, Marco,
      Abril, Maio, Junho, Julho,
      Agosto, Setembro, Outubro,
      Novembro, Dezembro
> Abril, Agosto, Dezembro,
      Janeiro, Fevereiro, Junho,
      Julho, Maio, Marco,
      Outubro, Novembro, Setembro
> Outubro esta na posicao 11 do
      Array
```

ArrayList da API Java

- A classe ArrayList (java.util) consiste em uma estrutura de dados alternativa para armazenar dados de modo sequencial, como um array
- Diferença do ArrayList para Arrays: Tamanho variável
- A classe ArrayList
  - Representa uma Coleção Genérica de tamanho variável
  - API Java
  - A classe ArrayList (java.util) consiste em uma estrutura de dados alternativa para armazenar dados de modo sequencial, como um vetor (array).
  - Entretanto, o tamanho do ArrayList varia dinamicamente, aumentando ou diminuindo de tamanho, para acomodar automaticamente novos dados ou a remoção dos mesmos.
  - ArrayList<Tipo> é uma das coleções genéricas disponíveis para um aplicativo Java
  - Pode armazenar grupos de objetos de um mesmo tipo
  - Como é genérica, pode acomodar qualquer tipo de dados em seus elementos, inclusive tipos não primitivos (tipos básicos)
  - Especificação completa em: http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html

A classe ArrayList (exemplo)

```
public class TesteArray {
  public static void main(String[] args){
    /*Cria uma array de Strings - tamanho default
        inicial de 10 */
    ArrayList<String> v = new ArrayList<String>();

    String nome1 = "Andre";
    String nome2 = "Joana";
    String nome3 = "Pedro";

    /*Adiciona String ao array */
    v.add(nome1);
    v.add(nome2);
    v.add(nome3);
    v.add("Carolina");

    /* Invoca o método toString to ArrayList */
    System.out.println(v);
    System.out.println("Tamanho atual da lista: " +
        v.size());
    String segundo = v.get(1);
    v.remove(1);
    String terceiro = v.get(2);
    System.out.printf("Joana %s esta na lista\n: ",
        v.contains("Joana") ? "" : "nao");
    System.out.printf("Andre %s esta na lista\n: ",
        v.contains("Andre") ? "" : "nao");
    for (int elemento=0; elemento<v.size();
        elemento++){
      System.out.println((elemento+1)+" ->
          "+v.get(elemento));
    }
    String ultimo = (String)v.get(v.size()); //
        Levanta uma excessao, por que?
  }
}
```

```
> [Andre, Joana, Pedro, Carolina]
> Tamanho atual da lista: 4
> Joana não está na lista
> Andre esta na lista
> 1 -> Andre
> 2 -> Pedro
> 3 -> Carolina
```

A classe ArrayList (resumo)

- Métodos:
  - add adiciona um elemento ao array
  - get retorna o elemento na posição passada (como um Object)
  - remove remove o elemento na posição passada do array
  - size retorna o número de elementos contidos no array
- Cuidados especiais:
  - Sempre inicializar a instância de ArrayList e as instâncias das classes que serão armazenadas.
    O construtor de uma classe é um ótimo lugar para criar a instância de um ArrayList!
  - Não acessar elementos fora da faixa de índices válidos.

Classe Random

Gerador de números pseudo-aleatórios

- gera uma sequência de números ou símbolos que não apresentam padrão, ou seja, aparenta ser aleatória
- aplicações que requerem aleatoriedade envolvem simulações de sistemas reais, criptografia, jogos e amostragem estatística
  - Em um jogo: tempo entre um inimigo aparecer no campo de visão, erro no passe
  - Em robótica: simulação do erro de um sensor

Classe Random

Um elemento aleatório pode ser introduzido em um aplicativo java por duas formas:

- O método Math.random() produz valores double no intervalo [0, 1)
- Um objeto da classe Random (no pacote java.util) pode produzir aleatoriedade para os tipos boolean, byte, float, double, int, long e distribuição Gaussiana para o tipo double

Classe Random

A seguinte instrução instancia um gerador de números pseudo-aleatórios:

- Random randomNumbers = new Random();
- Se quisermos garantir que a cada execução do sistema a sequência de número se repita, o objeto Random deve ser criado com uma semente inicial seedValue de tipo long Random randomNumbers = new Random(seedValue);
- Caso contrário, não é necessário passar uma semente inicial e o construtor adota uma semente inicial gerada com base na hora atual do sistema

Classe Random

```
import java.util.Arrays;
import java.util.Random;

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

    int[] lista = new int[10];

    Random randomNumber = new Random();
    for (int i=0; i<lista.lenght; ++i){
        lista[i] = randomNumber.nextInt();
    }

    System.out.println(Arrays.toString(lista));
    Arrays.sort(lista);
    System.out.println(Arrays.toString(lista));
  }
}
```

```
> [-246746270, -1294265705, 660574976, -1668777575, -1328352736, -1178620889, -1481497236, -698717632,
    -1876431999, 196695970]
> [-1876431999, -1668777575, -1481497236, -1328352736, -1294265705, -1178620889, -698717632, -246746270,
    196695970, 660574976]
```

Classe Random

```java
import java.util.Arrays;
import java.util.Random;

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

    int[] lista = new int[10];

    Random randomNumber = new Random();
    for (int i=0; i<lista.lenght; ++i){
        lista[i] = randomNumber.nextInt(100);
    }

    System.out.println(Arrays.toString(lista));
    Arrays.sort(lista);
    System.out.println(Arrays.toString(lista));
  }
}
```

```
> [91, 96, 39, 27, 85, 88, 65, 29, 41, 45]
> [27, 29, 39, 41, 45, 65, 85, 88, 91, 96]
```

Atividades Extra-classe

Exercícios:

- Quais as principais diferenças entre os diversos tipos de estrutura que pode armazenar dados homogêneos em Java?
- Crie um Array de objetos pessoa com nome e idade
  - Instancie elementos neste array
  - Invoque métodos pertencentes a cada objeto
  - Ordene o array pelo nome da pessos
  - Busque os elementos do array com uma certa idade

Referências

- Java: Como Programar, Paul Deitel & Heivey Deitel; Pearson; 7a. Ed
- Object-Oriented Programming with Java: An Introduction, David J. Barnes; Prentice Hall (2000).
- Usberti, F. Notas de Aula de MC322.

MC322 - Programação orientada a objetos
Aula 3.2
Arrays

Prof. Marcos M. Raimundo
Instituto de Computação - UNICAMP