UNIP
UNIVERSIDADE PAULISTA

Linguagem de Programação Orientada a Objeto

Coleções

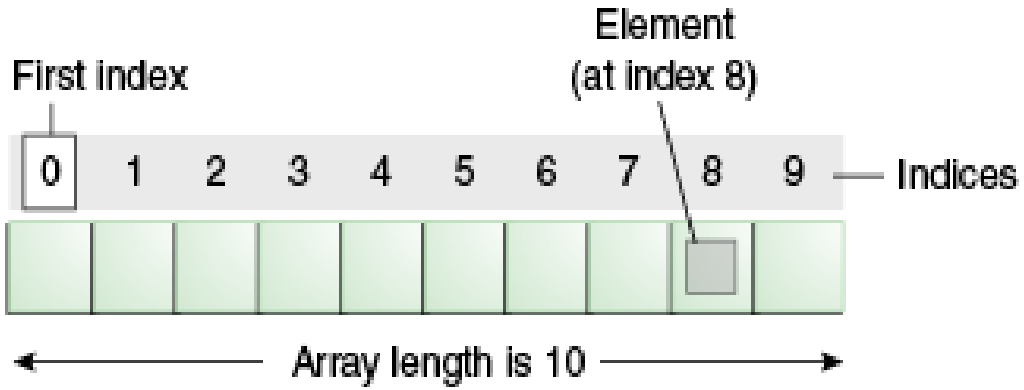
Professora Sheila Cáceres

Coleções

- Uma coleção é uma estrutura de dados que permite armazenar vários objetos
- As operações que podem ser feitas em coleções variam mas normalmente incluem:
  - Adição de elementos
  - Remoção de elementos
  - Acesso aos elementos
  - Pesquisa de elementos
  - Indagar sobre atributos

Arrays

- São contenedores que armazenam um número fixo de dados de um único tipo.
- Seu comprimento é estabelecido quando é criado e tal comprimento não pode variar.
- Seus elementos podem ser tipos primitivos de java ou objetos.
- São criados assim: Object[] objArray = new Object[10];
  objArray[1] = new Object();
- Cada elemento do array pode ser acessado pelo seu índice numerico começando em 0.



Arrays (exemplo com inteiros)

```
package lpoo.JavaBasics;

class ArrayDemo {
    public static void main(String[] args) {
        // declares um array de inteiros
        int[] anArray;
        // aloca memoria para 10 inteiros.
        anArray = new int[10];
        // initializar os elementos
        anArray[0] = 0;
        anArray[1] = 100;
        anArray[2] = 200;

        for (int i=3;i<10;i++)
            anArray[i]=i*100;

        for (int i = 0; i < anArray.length; i++)
            System.out.println("Elemento no indice " + i + " é : " + anArray[i]);
    }
}
```

Saida:

```
Elemento no indice 0 é : 0
Elemento no indice 1 é : 100
Elemento no indice 2 é : 200
Elemento no indice 3 é : 300
Elemento no indice 4 é : 400
Elemento no indice 5 é : 500
Elemento no indice 6 é : 600
Elemento no indice 7 é : 700
Elemento no indice 8 é : 800
Elemento no indice 9 é : 900
```

Lembrando da classe Empregado

```
class Empregado {
   private String nome;
   private double salario;

   public Empregado(String nome, double salario) {
      this.nome = nome;
      this.salario = salario;
   }
   public Empregado(String nome) {
      this.nome = nome;
      this.salario = 5000;
   }
   public void trabalhar(){
       System.out.println("Eu trabalho muito");
   }
   public double calculaBonificacao(){    return 0.2*salario;   }

   public double calculaBonificacao(double percentagem){return percentagem*salario; }

   public String getNome() {         return nome;     }
   public void setNome(String nome) {     this.nome = nome;     }
   public double getSalario() {          return salario;     }
   public void setSalario(double salario) {         this.salario = salario;     }
}
```

Array com Objetos

```
class ArrayDemo2 {
    public static void main(String[] args) {
        // declares um array de inteiros
        Empregado[] arrayEmpregados;
        // aloca memoria para 10 inteiros.
        arrayEmpregados = new Empregado[10];
        // initializar os elementos

        for (int i=0;i<arrayEmpregados.length;i++)
            arrayEmpregados[i]=new Empregado("empregado"+i);

        for (int i = 0; i < arrayEmpregados.length; i++)
            System.out.println("arrayEmpregados[ " + i + " ] : "
                    + arrayEmpregados[i].getNome());

        System.out.println("Somando os salarios de todos os empregados");
        int soma=0;
        for (Empregado empregado : arrayEmpregados) {
            soma +=empregado.getSalario();
        }
        System.out.println("A soma dos salarios é : "+soma);
    }
}
```

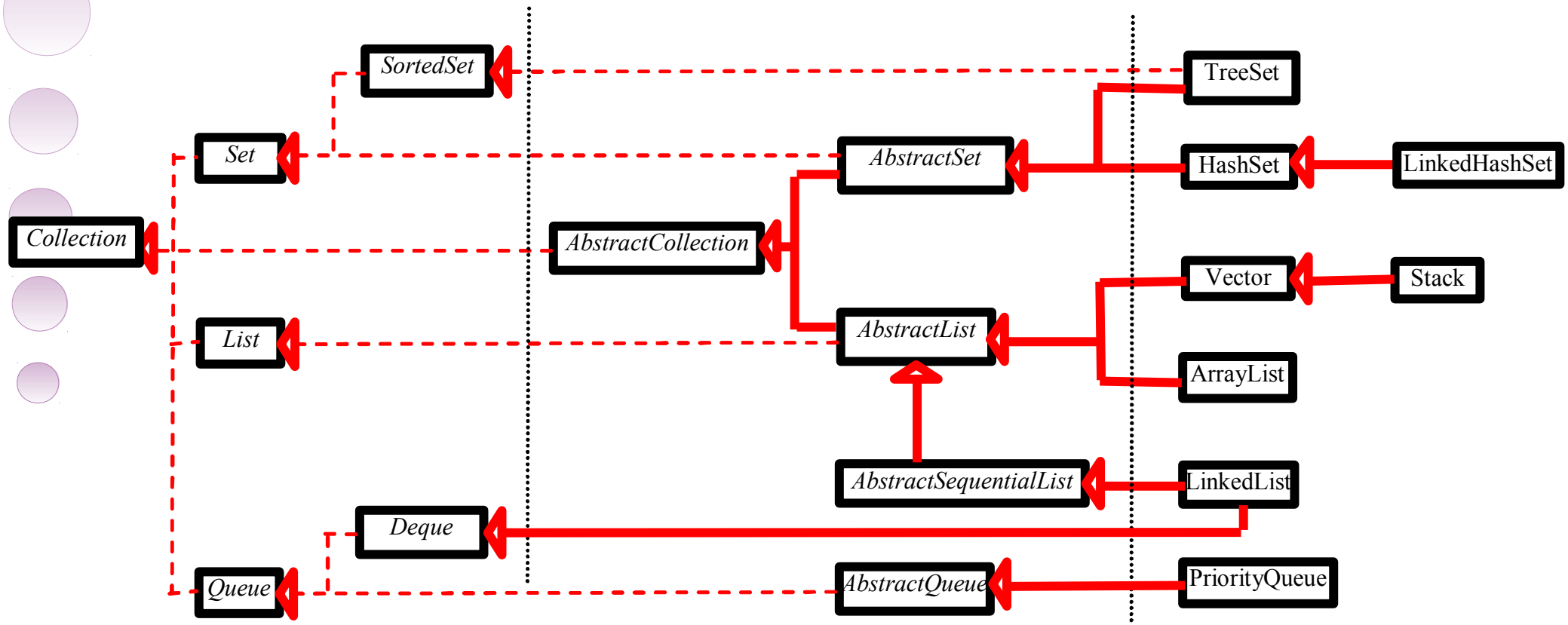
Saida:

```
arrayEmpregados[ 0 ] : empregado0
arrayEmpregados[ 1 ] : empregado1
arrayEmpregados[ 2 ] : empregado2
arrayEmpregados[ 3 ] : empregado3
arrayEmpregados[ 4 ] : empregado4
arrayEmpregados[ 5 ] : empregado5
arrayEmpregados[ 6 ] : empregado6
arrayEmpregados[ 7 ] : empregado7
arrayEmpregados[ 8 ] : empregado8
arrayEmpregados[ 9 ] : empregado9
Somando os salarios de todos os empregados
A soma dos salarios é : 50000
```

Coleções em pacotes Java

- Bibliotecas de classes java (chamadas *pacotes*) contêm classes de coleção, experimentadas e testadas.
- Utilizamos a classe `ArrayList` do pacote `java.util`.
- Itens podem ser adicionados e removidos.
- Todo item tem um índice.
- Valores de índice podem mudar se os itens forem removidos (ou, então, se outros itens forem adicionados).
- Os principais métodos `ArrayList` são `add`, `get`, `remove` e `size`.

- Existem diversos tipos de coleções em Java:



ArrayList

- Em geral, tem a mesma funcionalidade de um array mas o seu comprimento pode variar (são dinámicos) enquanto o programa roda (tempo de execução).
- São criados como objetos de qualquer classe adicionando o tipo de seus elementos.

```
ArrayList<BaseType> aList = new ArrayList<>();
```

ArrayList (Exemplo com Strings)

```
import java.util.*;
public class ArrayListDemo {
    public static void main(String args[]) {
      ArrayList<String> al = new ArrayList<>();
      System.out.println("Tamanho inicial de al: " + al.size());

      // Adicionando elementos
      al.add("A");
      al.add("C");
      al.add("E");
      al.add(1, "A2");
      System.out.println("Novo tamanho depois de insersoes: " + al.size());

      // Mostrando os elementos
      System.out.println("Elementos de al: " + al);

      // deletando elementos
      al.remove(2);
      System.out.println("Tamanho depos de deletar alguns elementos: " + al.size());
      System.out.println("Elementos de al: " + al);
   }
}
```

Saida

```
Tamanho inicial de al: 0
Novo tamanho depois de insersoes: 4
Elementos de al: [A, A2, C, E]
Tamanho depos de deletar alguns elementos: 3
Elementos de al: [A, A2, E]
```

ArrayList (Exemplo com Empregados)

```
public class ArrayListDemoEmpregado {
   public static void main(String args[]) {
      // Criando um ArrayList de objetos de tipo Empregado
      ArrayList<Empregado> al = new ArrayList<>();
      System.out.println("Tamanho inicial de al: " + al.size());

      // Adicionando elementos
      al.add(new Empregado("Joao",2000));
      al.add(new Empregado("Maria",5000));
      al.add(new Empregado("Pedro",3000));
      al.add(2, new Empregado("Arnaldo", 4000));
      Empregado carlos=new Empregado("Carlos",3000);
      al.add(carlos);
      al.add(new Empregado("Beto",3000));

      al.remove(carlos);//Removendo um empregado.
      System.out.println("Elementos de al: " + al); // Precisa-se percorre-los de outra forma.

      System.out.println("Percorrendo al: ");
      for (Empregado empre : al) {
          System.out.println("*Empregado com nome: "+empre.getNome() + " e salario "
          + empre.getSalario() +" e bonificacao "+empre.calculaBonificacao(0.1));
      }
   }
}
```

```
Tamanho inicial de al: 0
Elementos de al: [lpoo.Empregado@25d35bf2,
lpoo.Empregado@57398044, lpoo.Empregado@141d19,
lpoo.Empregado@28825459, lpoo.Empregado@46fb3d6]
Percorrendo al:
*Empregado com nome: Joao e salario 2000.0 e bonificacao 200.0
*Empregado com nome: Maria e salario 5000.0 e bonificacao 500.0
*Empregado com nome: Arnaldo e salario 4000.0 e bonificacao 400.0
*Empregado com nome: Pedro e salario 3000.0 e bonificacao 300.0
*Empregado com nome: Beto e salario 3000.0 e bonificacao 300.0
```

Coleções dentro de classes

- Podemos criar objetos que precisam de coleções de outros objetos para existir.
- Por exemplo:
  - Aula de classes tem Alunos
  - Sindicato tem Empregados
  - Biblioteca tem Livros
  - Bloco de Notas tem notas

- A seguir mostraremos a classe Sindicato que esta formada por um conjunto de Empregados.

```
public class Sindicato {
    private ArrayList<Empregado> lstEmpregados;
    private String nomeSindicato;
    private Empregado presidente;

    public Sindicato(){
        lstEmpregados = new ArrayList<>();
    }
    public Sindicato(String nomeSindicato, Empregado presidente){
        lstEmpregados = new ArrayList<>();
        this.presidente=presidente;
        this.nomeSindicato=nomeSindicato;
        lstEmpregados.add(presidente);
    }
    public void incluirEmpregado(Empregado emp){    lstEmpregados.add(emp);     }

    public void excluirEmpregado(Empregado emp){       lstEmpregados.remove(emp);    }

    public void excluirTodosOsEmpregados(Empregado emp)
    {  lstEmpregados.removeAll(lstEmpregados);    }

    public int getTotalEmpregados(){       return lstEmpregados.size();    }

    public void imprimirEmpregados(){
         for (Empregado empregado : lstEmpregados) {
             System.out.println("- " + empregado.getNome());
         }
    }
}
```

```
public class SindicatoTeste {

    public static void main(String [] args){
        Empregado presidente = new Empregado("Joao Ferreira",35);
        Sindicato sindicato = new Sindicato("Os empregados felices", presidente);
        sindicato.incluirEmpregado(new Empregado("Marcio"));
        sindicato.incluirEmpregado(new Empregado("Leandro"));
        sindicato.incluirEmpregado(new Empregado("Fulgencio"));
        System.out.println("O número de empregados é:"+sindicato.getTotalEmpregados());
        sindicato.imprimirEmpregados();
    }
}
```

```
O número de empregados é:4
- Joao Ferreira
- Marcio
- Leandro
- Fulgencio
```

Um bloco de notas

- Notas podem ser armazenadas.
- Notas individuais podem ser visualizadas.
- Não há um limite para o número de notas.
- Informará-se quantas notas estão armazenadas.

```
import java.util.ArrayList;
public class BlocoNotas
{
    // Campo para um número arbitrário de notas.
    private ArrayList<String> notas;

    /*Realiza qualquer inicialização que seja necessária */
    public Notebook(){
        notas = new ArrayList<>();
    }

   public void salvarNota(String nota){
        notas.add(nota);
    }

    public int getTotalNotas()
       return notas.size();
    }
}
```

Adicionando uma nova nota

Retornando o número de notas (*delegação*).

Bibliografia

- Programação orientada a objetos com Java.
  Autores David J. Barnes e Michael Kolling.
- Java: Como programar.
  Autores: H. M. Deitel e P. J. Deitel
  Editora: Pearson – 8a Edição
- Nota: Alguns exemplos desta apresentação foram extraídos das fontes aqui apresentadas.