PROGRAMAÇÃO ORIENTADA A OBJETOS

Prof. Milton Palmeira Santana

Anhanguera

Strings em JAVA

» Uma string é uma sequência de caracteres tratada com uma única unidade, que pode incluir letras, dígitos e vários caracteres especiais.

» Especificamente em Java, as strings são tratadas como objetos da classe java.lang.String e, por isso, devem ser declarados e depois instanciados. Lembre-se sempre de que os tipos primitivos byte, short, int, long, float, double, char e boolean não requerem instanciação.

Anhanguera

Strings em JAVA

- Uma sequência de caracteres é sempre apresentada entre aspas.

- Da mesma forma que os arrays, cada elemento de uma string também é referenciado por um índice que indica a posição de determinado caráter na sequência de caracteres.

Métodos da classe String

» **length**(): retorna o tamanho da string expresso em um número inteiro. A formação do tamanho de uma sequência inclui também os caracteres de controle, como espaços e tabulações, por exemplo.

```
Scanner entrada = new Scanner(System.in);
String cpf;
System.out.print("Informe o número do seu CPF: ");
cpf = entrada.nextLine();
if (cpf.length()!=11) System.out.println("CPF inválido");
```

Anhanguera

Métodos da classe String

» **length**(): retorna o tamanho da string expresso em um número inteiro. A formação do tamanho de uma sequência inclui também os caracteres de controle, como espaços e tabulações, por exemplo.

```
String s = "Caracteres";
int c = s.charAt(1);
System.out.printf("%c",c);
```

Métodos da classe String

- **equals (Object umObjeto)**: este método está inserido na categoria dos que realizam comparações entre sequências de caracteres. Ele promove a comparação entre uma string e o argumento umObjeto, que pode ser uma sequência de caracteres ou um objeto da classe String. O resultado será true se – e somente se – o argumento for um objeto String diferente de null que represente a mesma sequência de caracteres.

```
if (!string1.equals(string2))
System.out.println("As senhas não são iguais.");
```

Introdução a coleções de objetos

- O Java contém classes e interfaces que poupam trabalho e oferecem mecanismos para agrupar e processar objetos em conjuntos. Mas, por qual motivo não podemos agrupar elementos afins em uma estrutura de array? Embora a resposta inclua vários itens, é possível resumi-los:

- (i) Via de regra, o tamanho de um array não pode ser modificado após sua criação. Alterações no tamanho devem ser feitas com a criação de outro array, de tamanho diferente, para que os elementos do array original possam ser copiados nele.

Anhanguera

Introdução a coleções de objetos

- (ii) Um array pode apenas conter elementos de um único tipo, exceto se o desenvolvedor lançar mão de recursos complexos para contornar esta limitação.

- (iii) inserções e deleções em um array não são operações simples.

- O Java disponibiliza, como membro do pacote java.util, uma estrutura de coleções. Essa estrutura é formada por interfaces que declaram as operações possíveis de serem realizadas nas várias modalidades de agrupamentos que o Java oferece.

Anhanguera

Introdução a coleções de objetos

Interface	Descrição
Collection	A interface raiz na hierarquia de coleções, a partir da qual as interfaces `Set`, `Queue` e `List` são derivadas.
Set	Uma coleção que não contém duplicatas
List	Uma coleção ordenada que pode conter elementos duplicados.
Map	Associa chaves a valores e não pode conter chaves duplicadas.
Queue	Em geral, uma coleção FIFO (primeiro a entrar, primeiro a sair) que modela uma fila de espera. Outras ordens podem ser especificadas.

Anhanguera

Introdução a coleções de objetos

» Exemplos

```
Scanner entrada = new Scanner(System.in);
 ArrayList<Integer> lista = new ArrayList<Integer>();
 int elemento;
 while (true) {
 System.out.print("Insira um número inteiro positivo
para ser incluído ou -1 para terminar: ");
 elemento = entrada.nextInt();
 if (elemento==-1) {
 break;
}
```

Introdução a coleções de objetos

» Exemplos

```
Scanner entrada = new Scanner(System.in);
 ArrayList<Integer> lista = new ArrayList<Integer>();
 int elemento;
 while (true) {
      System.out.print("Insira um número inteiro positivo para
ser  incluído ou -1 para terminar: ");
      elemento = entrada.nextInt();
      if (elemento==-1) {
          break;
       }
       lista.add(elemento);
 }
```

Anhanguera

Introdução a coleções de objetos

» Exemplos

```
System.out.println("\nA lista que você criou contêm os seguintes elementos: ");
 for (int i: lista) System.out.println(i);

 System.out.print("\nInforme o índice do elemento que você deseja retirar da lista: ");
 entrada.reset();
 elemento = entrada.nextInt();
 lista.remove(elemento);
 System.out.println("Agora sua lista ficou assim: ");
 for (int i: lista) System.out.println(i);
```

Anhanguera

Trabalhando com arquivos

» Exemplo – Classe File


Anhanguera