
JAVA™

Garbage Collector

Objetos nascem e objetos morrem. Você é quem manda no ciclo de vida de um objeto. Decide quando e como construí-los. Também decide quando destruí-los. Exceto pelo fato de não ser você próprio que destruirá realmente o objeto, apenas o abandonará.

Mas quando ele for abandonado, o Coletor de Lixo (garbage Collector) poderá eliminá-lo, solicitando a memória que o objeto estava usando.

Garbage Collector

A vida de um objeto depende inteiramente da vida das referências que apontam para ele. Se a referência for considerada “ativa”, o objeto continuará ativo. Se ela for eliminada, o objeto também será.

Há três maneiras de eliminar a referência de um objeto:

1 - a referência for eliminada permanentemente:

```
void go(){
    Vida z = new Vida();
}
```

a referência z será eliminada ao fim da execução do método

2 - a referência é atribuida a outro objeto:

```
Vida z = new Vida();
z = new Vida();
```

o primeiro objeto é abandonado quando z é “reprogramada” para um novo objeto

3 - a referência é configurada explicitamente como nula:

```
Vida z = new Vida();
z = null;
```

o primeiro objeto é abandonado quando z é “desprogramada”

Usando a biblioteca Java

O Java vem com centenas de classes pré-definidas. Você não terá que reinventar a roda se souber como encontrar o que precisa na biblioteca Java, normalmente conhecida como API Java.

Você pode escrever somente as partes que forem exclusivas de seu aplicativo.

Usando a biblioteca Java

1 - uma matriz comum tem de saber seu tamanho na hora que é criada.

2 - para inserir um objeto em uma matriz comum, você terá de atribuí-lo a um local especifico (índice).

3 - as matrizes usam uma sintaxe própria que não é empregada em nenhum outro local em Java.

Usando a biblioteca Java

E se existisse uma matriz que pudesse ser reduzida quando removêssemos algo. Uma que permitisse excluirmos um elemento sem a necessidade de sabermos a posição dele.

Na biblioteca do Java existe algo assim. Mas não é uma matriz, é uma ArrayList.

Usando a biblioteca Java

ArrayList

add(Object elem) - adicionará o elemento a lista.

remove(int index) - removerá o elemento com base no índice.

isEmpty() - retornará "verdadeiro" se a lista não tiver elementos.

indexOf(Object element) - retornará o índice do parâmetro especificado.

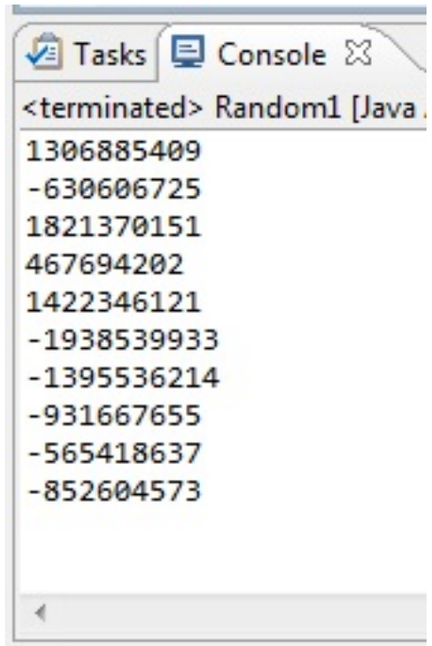
size() - retornará a quantidade de elementos existentes na lista atualmente.

get(int index) - retornará o objeto que se encontra atualmente na posição passada pelo parâmetro index.

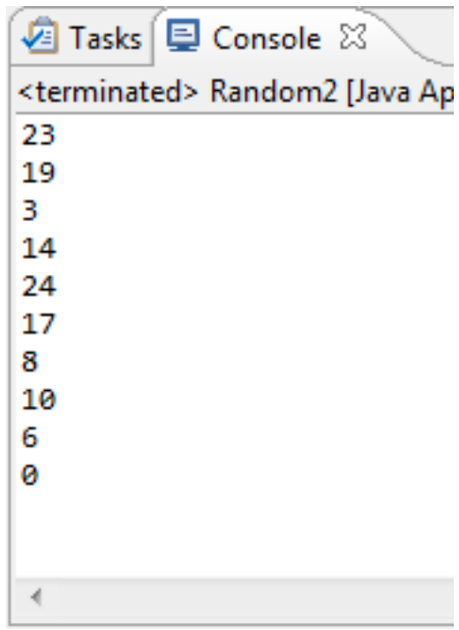
Isso é apenas um exemplo de ALGUNS dos métodos de ArrayList

Usando a biblioteca Java Random

```
import java.util.Random;

public class Random1 {
   public static void main(String[] args) {
      // instância um objeto da classe Random

      Random gerador = new Random();

      // imprime sequência de 10 números inteiros aleatórios
      for (int i = 0; i < 10; i++){
         System.out.println(gerador.nextInt());
      }
   }
}
```



Usando a biblioteca Java Random




```
import java.util.Random;

public class Random2 {
   public static void main(String[] args) {
      //instância um objeto da classe Random
      Random gerador = new Random();

      //imprime sequência de 10 números inteiros aleatórios entre 0 e 25

      for (int i = 0; i < 10; i++) {
         System.out.println(gerador.nextInt(26));
      }
   }
}
```

Usando a biblioteca Java Scanner

```
import java.util.Scanner;

class Ler{

    public static void main(String args[]){
        int num;

        Scanner dados = new Scanner(System.in);

        System.out.println("Digite um numero ");
        num = dados.nextInt();

        System.out.println(num);
    }
}
```

Métodos estáticos e métodos não estáticos

A diferença entre métodos comuns (não-estáticos) e estáticos

O Java é orientado a objetos, mas em algumas situações você pode ver um caso especial, normalmente um método utilitário (como os métodos de Math), onde não é necessária a existência de uma instância da classe. A palavra-chave static permite que um método seja executado sem qualquer instância da classe. Um método ser estático significa “que o comportamento não depende de uma variável de instância, portanto não são necessárias instâncias/objetos. Apenas a classe”.

Chame um método não estático usando o nome de uma **VARIÁVEL** de **REFERÊNCIA**

```
Som t = new Som();
t.tocar();
```

Chame um método estático usando o nome de uma **CLASSE**

```
Math.min(88, 86);
```

Métodos estáticos e métodos não estáticos

Variável estática: o valor é o mesmo para TODAS as instâncias da classe

private static int contador = 0;

As variáveis estáticas finais são constantes

public static final double PI = 3.141592;

A palavra-chave “final” não é usada apenas para variáveis estáticas...

Uma variável ser *final* significa que você não poderá *alterar* seu valor.

Um método ser *final* significa que você não poderá *sobrepô-lo*.

Uma classe ser *final* significa que você não poderá *estendê-la* (isto é, não poderá criar uma subclasse).

Atividade

Faça um programa que gere um código "captcha" aleatório com 6 caracteres. Este código deve ser formado por letras e números. O código deve ser exibido e o usuário deve digitá-lo. Em seguida o captcha deve ser validado.