Atualidades

Tiago Eugenio de Melo
tiago@comunidadesol.org

Comando `for`

- Diretiva `for` avançada
  - A partir da versão 5 do Java, a diretiva `for` passa a contar com uma versão avançada para otimizar o processamento de arranjos e coleções.
  - Sua sintaxe é:

```
for ([TipoElemento] <variável>  : <expressão>) {
    diretiva;
}
```

Comando `for`

- Essa diretiva provê um mecanismo direto de navegação sequencial por todo arranjo ou coleção, sem necessidade de utilizar índices ou iterators.

Comando `for`

- Exemplo:

```
1  public class ExemploComandoFor {
2      public static void main(String[] args) {
3          int vetor[] = {2, 1, 10, 3, 6};
4          int tamanho = vetor.length;
5          for (int i=0;i<tamanho;i++) {
6              System.out.print(vetor[i] + " ");
7          }
8          for(int elemento: vetor) {
9              System.out.print(elemento + " ");
10         }
11     }
12 }
```

Enumerações

- Um tipo enumerado é conjunto fixo de constantes utilizadas com frequência para definir opções de uma classe ou programa, sendo comum este padrão:

```
public class Opcoes {

    public static final int COMPACTAR = 0;

    public static final int DESCOMPACTAR = 0;
}
```

Enumerações

- Essa estrutura apresenta as seguintes desvantagens:
  - Seu código é compilado junto das classes que a utilizam, assim a edição de constantes exige nova compilação.
  - O uso das constantes exige o nome da classe como prefixo.
  - Só podem ser usadas como valores primitivos.
  - Dado que as constantes são nomes associados a inteiros, não permitem a checagem de tipo, pois se confundem com outras constantes definidas do mesmo modo.

Enumerações

- A declaração enum, incluída a partir da versão 5, substitui com vantagens essa estrutura:
- public enum Opcoes {COMPACTAR, DESCOMPACTAR};

```
public class ExemploEnum1 {
    public enum Opcoes {COMPACTAR, DESCOMPACTAR};
    public static void main(String[] args) {
        for (Opcoes opcao: Opcoes.values()) {
            System.out.println(opcao);
        }
    }
}
```

Enumerações

- Vantagens:
  - Provê checagem de tipo durante a compilação.
  - O código é compilado junto das classes clientes, podendo ser alterado livremente.
  - Os nomes das constantes são automaticamente usados como texto informativo.
  - Podem ser utilizadas como objetos.
  - Oferecem um espaço de definição próprio para cada enumeração.

Enumerações

```
public enum Opcoes {
    COMPACTAR("COMPACTACAO", 0), DESCOMPACTAR("DESCOMPACTACAO", 1);
    private final String nome;
    private final int valor;
    Opcoes (String n, int v) {
        nome = n;
        valor = v;
    }
    public int getValor () {return valor;}
    public String getNome() {return nome;}
    public String toString() {
        return nome  + "(" + valor + ")";
    }
}
```

Enumerações

```
public class UsaOpcoes {
    public static void main(String[] args) {
        for (Opcoes constante: Opcoes.values()) {
            System.out.printf("%-25s #%6d : %-20s\n", constante, constante.getValor(), constante.getNome());
            switch(constante) {
            case COMPACTAR:
                System.out.println("Opcao de compactacao");
                break;
            case DESCOMPACTAR:
                System.out.println("Opcao de descompactacao");
                break;
            }
        }
    }
}
```

Genéricos

- Os genéricos são mecanismos para a criação de tipos parametrizados, ou seja, permitem a definição de uma classe ou método para funcionar com uma variedade de tipos, mas que operará com um tipo particular de cada vez.
- Os tipos genéricos foram introduzidos a partir da versão 5 do Java e são considerados como uma das principais características introduzidas nesta versão.

Enumerações

- Considere o trecho a seguir:

```
import java.util.ArrayList;

public class TesteGenerico {
    public static void main (String[] args) {
        ArrayList lista = new ArrayList();
        lista.add(new Integer(0));
        lista.add("1");
        lista.add(new Integer(2));
        Integer i1 = (Integer) lista.get(0); //uso de coerção (downcasting)
        Integer i2 = (Integer) lista.get(1); // erro
    }
}
```

Enumerações

- Dois problemas:
  - Objetos de tipo diferente de Integer podem ser livremente adicionados à lista.
  - Uso de operações de coerção (downcasting) para manipular os objetos contidos na lista.

Enumerações

- O uso de genéricos pode resolver esse problema:
- ArrayList<Integer> lista = new ArrayLista<Integer>( );
- Com essa declaração o erro apareceria na compilação.