


Tipos Enum no Java

Veja neste artigo como criar enumerações em Java, estruturas de dados que armazenam uma coleção de valores fixos predefinidos e imutáveis. Serão apresentadas as características, as declarações e os métodos.



 Gostei (14)  (1)

Vamos apresentar os tipos Enum que, na linguagem Java, são usados para a criação de estruturas de dados organizados, podendo agrupar valores que tenham o mesmo sentido para dentro dessa estrutura.

Enum

São tipos de campos que consistem em um conjunto fixo de constantes (static final), sendo como uma lista de valores pré-definidos. [Na linguagem de programação Java](#), pode ser definido um tipo de enumeração usando a palavra chave **enum**.

Todos os tipos enums implicitamente estendem a classe **java.lang.Enum**, sendo que o Java não suporta herança múltipla, não podendo estender nenhuma outra classe.

Características

Em relação às propriedades é preciso tomar os seguintes cuidados:

- As instâncias dos tipos enum são criadas e nomeadas junto com a declaração da classe, sendo fixas e imutáveis (o valor é fixo).;
- Não é permitido criar novas instâncias com a palavra chave `new`;
- O construtor é declarado `private`, embora não precise de modificador `private` explícito;
- Seguindo a convenção, por serem objetos constantes e imutáveis (`static final`), os nomes declarados recebem todas as letras em MAIÚSCULAS;
- As instâncias dos tipos enum devem obrigatoriamente ter apenas um nome;
- Opcionalmente, a declaração da classe pode incluir variáveis de instância, construtor, métodos de instância, de classe, etc.

Declaração Enum

Na declaração é definida uma classe chamada de tipo enum. O corpo da classe enum pode incluir métodos e outros campos. O compilador automaticamente adiciona alguns métodos especiais quando se cria um enum.

Listagem 1: Declaração Enum (sempre definir como letras maiúsculas).

```
public enum Cartas {  
    A, J, Q, K;  
}
```

Inicializando valores

Para iniciar os valores declarados dentro das variáveis Enum, é preciso declarar um construtor para iniciar os seus atributos que são declarados.

Listagem 2: Declarar valor Enum

```
public enum CartasEnum {  
    J(11),Q(12),K(13),A(14);  
  
    public int valorCarta;  
    CartasEnum(int valor) {  
        valorCarta = valor;  
    }  
}
```

Como mostrado na Listagem 2 inicializamos o construtor com apenas um argumento, o que corresponde ao mesmo número de argumentos declarados nos atributos. Pode ser possível declarar com mais argumentos, basta declarar suas variáveis com seus tipos e inicializar dentro do construtor.

```
public enum CartasEnum {  
    J(11), Q(12), K(13), A(14);  
  
    public int valorCarta;  
  
    CartasEnum(int valor) {  
        valorCarta = valor;  
    }  
}
```

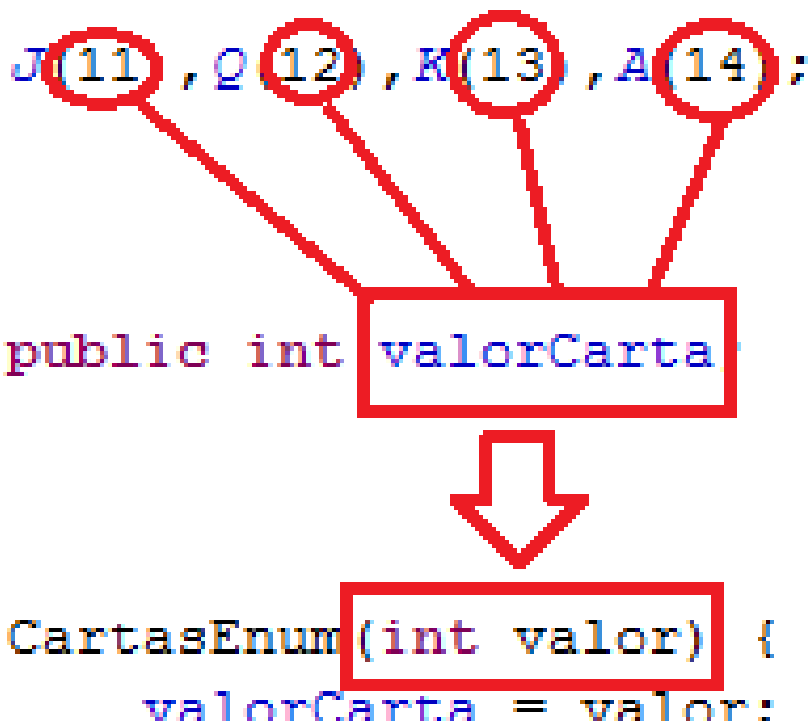


Figura 1: Inicializando Valor Enum

Os tipos enum podem ser usados a qualquer momento em que se precise representar um conjunto fixo de constantes. Nos exemplos das Listagens 3 e 4 é mostrada a criação de um tipo Enum e a invocação dos valores e atributos.

Listagem 3: Enum OpcoesMenu

```
public enum OpcoesMenu {  
    SALVAR(1), IMPRIMIR(2), ABRIR(3), VISUALIZAR(4), FECHAR(5);  
  
    private final int valor;  
    OpcoesMenu(int valorOpcao){  
        valor = valorOpcao;  
    }  
    public int getValor(){  
        return valor;  
    }  
}
```

Declaramos como parâmetro no método `escolheOpcao()` um tipo Enum, onde é feita uma comparação com o valor de entrada supostamente originado pelo usuário.

Listagem 4: Classe Testadora

```
public class TestadoraEnum {  
    public static void escolheOpcao(OpcoesMenu opcao){  
        if(opcao == OpcoesMenu.SALVAR){  
            System.out.println("Salvando o arquivo!");  
        }  
        else if(opcao == OpcoesMenu.ABRIR){  
            System.out.println("Abrindo o arquivo!");  
        }  
    }  
  
    public static void main(String[] args) {  
        escolheOpcao(OpcoesMenu.ABRIR);  
    }  
}
```

Imprimindo valor Enum

Nos exemplos das Listagens 5 e 6 é mostrado como pode ser impresso um valor do tipo enum, através do método **name**.

Listagem 5: Enum MarcasEnum

```
public enum MarcasEnum {  
    AMAZON, DELL, HP, TOSHIBA, LG, SAMSUNG;  
}
```

Veja a saída de um valor através do método name.

Listagem 6: Imprimindo Valor

```
public class TestadoraEnum {  
    public static void main(String[] args) {  
        MarcasEnum hp = MarcasEnum.HP;  
        MarcasEnum samsung = MarcasEnum.SAMSUNG;  
        System.out.println("Nome da Marca = "+hp.name());  
        System.out.println("Nome da Marca = "+samsung.name());  
    }  
}
```

Percorrendo Valores

Os valores Enum tem um método estático chamado **values** que retorna uma matriz contendo todos os valores do enum na ordem em que são declarados. Este método é normalmente usado em combinação com o **for** para construir cada repetição dos valores de um tipo de enumeração.

Neste exemplo é percorrido o tipo Enum feitos na Listagem 3.

Listagem 7: Percorrendo Valor

```
public class TestadoraEnum {  
    public static void main(String[] args) {  
        for(OpcoesMenu op : OpcoesMenu.values()){  
            System.out.print("Menu " + op + " = " + op.getValor()+"\n");  
        }  
    }  
}
```

Comparando Valores Enum

Um Enum pode ser comparado com outro objeto através do método **equals**.

Na Listagem 8 declaramos o método “comparaEnum” do tipo static para ser acessível para toda a classe e fazer referência ao tipo Enum declarado.

Listagem 8: Usando o método equals

```
public class TestadoraEnum {  
    public static void comparaEnum(OpcoesMenu opcao){  
        if(opcao.equals(OpcoesMenu.SALVAR)){  
            System.out.println("Foi escolhido a opção Salvar");  
        }else if(opcao.equals(OpcoesMenu.ABRIR)){  
            System.out.println("Foi escolhido a opção ABRIR");  
        }else if(opcao.equals(OpcoesMenu.FECHAR)){  
            System.out.println("Foi escolhido a opção FECHAR");  
        }  
    }  
  
    public static void main(String[] args) {  
        comparaEnum(OpcoesMenu.SALVAR);  
    }  
}
```

Nos tipos Enum também existem outros métodos descritos abaixo.

- `String toString()` - retorna uma `String` com o nome da instância (em maiúsculas).
- `valueOf(String nome)` – retorna o objeto da classe enum cujo nome é a string do argumento.
- `int ordinal()` - retorna o número de ordem do objeto na enumeração.

Mais informações acesse o link da documentação onde irá encontrar mais características e métodos: <http://docs.oracle.com/javase/6/docs/api/java/lang/Enum.html>

Espero ter ajudado e até a próxima!