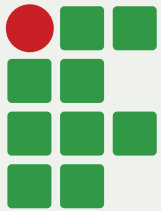
Linguagem de Programação 1

APIs Java: Enum e Tipos Genéricos

Prof. Fábio José Rodrigues Pinheiro

Fevereiro de 2022



INSTITUTO FEDERAL
Catarinense
Campus Videira

Enum

O tipo Enum

- O tipo Enum é usado para **definir um conjunto fixo de constantes**
  - Exemplo: estações do ano, dias da semana, meses do ano, planetas do sistema solar

- Antes da introdução do Enum no JDK1.5, um padrão de projeto era declarar um grupo de constantes inteiras

```
1 public class Teste{
2   public static final int DOMINGO = 0;
3   public static final int SEGUNDA = 1;
4   public static final int TERCA   = 2;
5   // .....
6   public void aulas(){
7     int aulasPoo[] = {SEGUNDA, TERCA};
8   }
9 }
```


Enum para dias da semana

```java
// Arquivo DiasDaSemana.java
public enum DiasDaSemana{
  DOMINGO, SEGUNDA, TERCA, QUARTA, QUINTA, SEXTA, SABADO
}
```

```java
// Arquivo Teste.java
public class Teste{
 public void aulas(){
  DiasDaSemana aulasPoo[] = {DiasDaSemana.SEGUNDA, DiasDaSemana.TERCA};
 }
}
```

Definição

Classes que exportam uma única instância para cada constante enumerada via um atributo público, estático e final.

Enum para cores da Shell

```
1  public enum Cor{
2    PRETO(0), VERMELHO(1), VERDE(2), AMARELO(3),
3    AZUL(4), ROXO(5), CIANO(6), BRANCO(7);
4
5    public final int codigo;
6
7    Cor(int c){
8          this.codigo = c;
9      };
10
11   public static Cor getByCodigo(int c){
12     for (Cor cor: Cor.values()){
13       if (c == cor.codigo){
14         return cor;
15       }
16     }
17     throw new IllegalArgumentException("código inválido");
18   }
19 }
```


Enum para cores da Shell

```
1  public class Teste{
2
3    public static void main(String[] args) {
4      Cor corDoFundo = Cor.getByCodigo(0);
5      Cor corDoTexto = Cor.getByCodigo(7);
6
7      System.out.println("Cor do fundo: " + corDoFundo);
8
9      switch(corDoFundo){
10       case PRETO:
11         switch(corDoTexto){
12           case PRETO:
13             System.out.println("não me parece uma boa combinação");
14             break;
15           case BRANCO:
16             System.out.println("me parece uma boa combinação");
17             break;
18         }
19     }
20 }
```


Representação de Enum em UML

Exercício:

- Fazer um Enum para os planetas do sistema solar
- Para cada planeta é importante guardar qual a sua posição em relação ao sol (e.g. A Terra está na posição 3)

Tipos genéricos em Java

Classe capaz de armazenar instâncias de qualquer classe

```java
public class Caixa{
    private Object dado;

    public void set(Object obj){
        this.dado = obj;
    }

    public Object getDado(){ return this.dado;}
}
```

```java
public class Principal{
    public static void main(String[] args){
        Caixa c = new Caixa();
        String s = "Olá mundo";
        c.set(s);
        String outra = (String) c.getDado(); // typecasting obrigatório
    }
}
```


Classe capaz de armazenar instâncias de qualquer classe

```
Caixa c = new Caixa();
Caixa d = new Caixa();
String s = "Olá mundo";
Pessoa p = new Pessoa("Joao");
c.set(s); // OK
d.set(p); // OK
String nova = (String) c.getDado();
// só apresentará erro durante a execução
String outra = (String) d.getDado(); // ERRO durante execução!!
```

- É **mais fácil detectar** *bugs* em **tempo de compilação** do que em **tempo de execução**.

Classe capaz de armazenar instâncias de qualquer classe

```
Caixa c = new Caixa();
Caixa d = new Caixa();
String s = "Olá mundo";
Pessoa p = new Pessoa("Joao");
c.set(s); // OK
d.set(p); // OK
String nova = (String) c.getDado();
// só apresentará erro durante a execução
String outra = (String) d.getDado(); // ERRO durante execução!!
```

- É **mais fácil detectar** *bugs* em **tempo de compilação** do que em **tempo de execução**.

Tipos genéricos

Possibilitam que bugs possam ser detectados já na fase de compilação

Tipos genéricos em Java

- Permite que tipos (Classes ou Interfaces) sejam passados como parâmetros durante a definição de Classes, Interfaces ou métodos
- **Parâmetros de tipos permite reutilizar a mesma classe** diante de diferentes entradas
- Entradas de parâmetros formais são obrigatoriamente valores

```
public void metodo(int parametro){ .... }
....
objeto.metodo(123); // parâmetro deve ser um valor
```

- **Entradas de parâmetros de tipos** são tipos
  - Exemplo de tipos: String, Pessoa, Carro, etc.

Reescrevendo classe Caixa para usar tipos genéricos

```
public class Caixa<T>{
  private T dado;

  public void set(T obj){
    this.dado = obj;
  }

  public T getDado(){ return this.dado;}
}
```

```
public class Principal{
  public static void main(String[] args){
    Caixa<String> c = new Caixa<>();
    String s = "Olá mundo";
    c.set(s);
    String outra = c.getDado(); // não precisa do typecasting
  }
}
```


Reescrevendo classe Caixa para usar tipos genéricos

```
public class Caixa<T>{
  private T dado;

  public void set(T obj){
    this.dado = obj;
  }

  public T getDado(){ return this.dado;}
}
```

```
Caixa<String> c = new Caixa<>();

Pessoa p = new Pessoa("Joao");

c.set(p); // erro de compilação, tipos errados
```


Parâmetro de tipo limitado

- É possível restringir quais tipos são aceitos
  - Somente as classes ou interfaces da hierarquia seriam permitidas

```
public class Caixa<T extends Personagem>{
  private T personagem;
  .......
}
```

```
public class Principal{
    public static void main(String[] args){

        // ok, pois Aldeao herda de Personagem
        Caixa<Aldeao> c = new Caixa<>();

        Caixa<String> n = new Caixa<>();// erro!
    }
}
```


Convenção de nomes

- Nomes dos parâmetros de tipos devem ser uma única letra maiúscula
  - E – para elementos
  - T – para tipos
  - K – para chaves
  - V – para valores
  - N – para números
- A API Collections do Java faz uso do nome *E*
- Veja mais informações na documentação oficial https://docs.oracle.com/javase/tutorial/java/generics