

Tipos Enumerados

UA.DETI.POO

Motivação

❖ Como podemos restringir os valores possíveis de uma variável?

– Exemplo 1

```
public class SomeClass {  
    void someFunction(int diaDaSemana) {  
        // diaDaSemana deve estar entre 1 (domingo) e 7 (sábado)  
    }  
}
```

– Exemplo 2

```
public class SomeOtherData {  
    void someFunction(int estacaoDoAno) {  
        // estacaoDoAno deve estar entre 1 (spring) e 4 (winter)  
    }  
}
```

Solução

❖ Tipos enumerados (`enum`)

- Criamos um novo tipo de dados com valores predefinidos e constantes
- Exemplo 1

```
public enum Dia {  
    domingo, segunda, terca, quarta, quinta, sexta, sabado  
}
```

```
public class SomeClass {  
    void someFunction(Dia diaDaSemana) {  
        // diaDaSemana varia entre Dia.domingo e Dia.sabado  
    }  
}
```

Solução

❖ Tipos enumerados (enum)

– Exemplo 2

```
public enum Season {  
    SPRING, SUMMER, FALL, WINTER  
}
```

Convenção: devemos usar preferencialmente maiúsculas

```
public class SomeOtherData {  
    void someFunction(Season estacaoDoAno) {  
        // estacaoDoAno varia entre Season.SPRING e Season.WINTER  
    }  
}
```

Tipos Enumerados

- ❖ Mais Valia Importante: “compile-time type safety”
i.e. “void someFunction(Season estacaoDoAno)”

- ❖ Declaração na forma geral

- Criamos um ficheiro java
- i.e. “Color.java”

```
public enum Color {  
    WHITE, BLACK, RED, YELLOW, BLUE  
}
```

- Como usar?

```
Color cor = Color.WHITE;  
...  
if (cor == Color.WHITE)  
...  
cor = Color.BLUE;
```

Tipos Enumerados

- ❖ Declaração do enum dentro de uma Classe já existente.

```
public class Painter {  
    ...  
    public enum Color {  
        WHITE, BLACK, RED, YELLOW, BLUE  
    }  
    ... // Painter body  
  
}
```

– Como usar?

```
Painter.Color cor = Painter.Color.WHITE;  
...  
if (cor == Painter.Color.WHITE)  
...  
cor = Painter.Color.BLUE;
```

Tipos Enumerados – outro exemplo

- ❖ Definimos um enum para os Meses do Ano

```
public enum Mes {  
    JANEIRO, FEVEREIRO, MARCO, ABRIL, MAIO,  
    JUNHO, JULHO, AGOSTO, SETEMBRO, OUTUBRO,  
    NOVENBRO, DEZEMBRO ;  
}
```

- ❖ Podemos utilizarmos na classe Data

```
// Construtor de Data  
public Data (int iDia, Mes iMes, int iAno){  
    //...  
}  
  
// Main  
Data d1 = new Data(11, Mes.MAIO, 1900);
```

- ❖ Limitação importante

- Não podemos ler um enumerado através de um Scanner

Enum – Métodos Disponíveis

```
public enum Season {  
    SPRING, SUMMER, FALL, WINTER  
}
```

❖ Fornecem alguns métodos úteis:

- `valueOf(String val)`: converte a String (elemento do conjunto) para um valor
- `ordinal()`: posição (int) do valor na lista de elementos
- `values()`: devolve a lista de elementos

```
Season s1 = Season.valueOf("WINTER");  
System.out.println(s1);  
System.out.println(s1.ordinal());  
for (Season s2: Season.values())  
    System.out.println(s2);
```

```
WINTER  
3  
SPRING  
SUMMER  
FALL  
WINTER
```


Enum – Switch Statement

- ❖ A instrução switch funciona com enumerados

```
Color myColor = Color.BLACK;
```

```
switch(myColor){  
    case WHITE: ...;  
    case BLACK: ...;  
    ...  
    case BLUE: ...;  
    default: ...;  
}
```

Tipos Enumerados em Java

- ❖ enum é uma "classe", não um tipo primitivo.
 - Pode implementar interfaces
 - Os valores são objetos
 - Suportam comparação (== ou equals()).
- ❖ Os tipos enumerados não são inteiros.
- ❖ Só têm construtores privados.
- ❖ Os valores enumerados são constantes
 - São automaticamente public, static, final.

Enum – uma Classe

- ❖ Podemos ter tipos Enumerados com dados e operações associadas

```
public enum Color {  
    WHITE(21), BLACK(22), RED(23), YELLOW(24), BLUE(25);  
  
    // Dados  
    private int code;  
  
    // Construtor  
    private Color(int c) {    code = c; }  
  
    // Método  
    public int getCode() {    return code; }  
}
```

Enum – implements Interface

- ❖ Os tipos enum podem implementar interfaces

```
public enum Color implements Runnable {  
    WHITE, BLACK, RED, YELLOW, BLUE;  
  
    public void run() {  
        System.out.println("name()=" + name() +  
            ", toString()=" + toString());  
    }  
}
```

- Utilização:

```
for (Color c : Color.values()) { c.run();}
```

- Ou

```
for (Runnable r : Color.values()) { r.run();}
```