Fundamentos de Java

Enums, Classes Wrappers e Autoboxing

Softblue
cursos online

Tópicos Abordados

- Enums
  - Atributos constantes X Enums
  - Elementos dos enums
- Classes wrappers
  - Tipos primitivos
  - Criando objetos wrappers
  - Utilitários de conversão
- Autoboxing

Enums

- Os enums (enumerations) permitem restringir variáveis a apenas valores previamente determinados
- Ajudam a reduzir bugs no código
- Podem ser declarados dentro de uma classe (da mesma forma que atributos)
- Podem ser declarados num arquivo próprio (como se fossem uma classe)

Atributos Constantes X Enums

- Constantes sem usar enums

```
public static final int VOLUME_ALTO  = 0;
public static final int VOLUME_MEDIO = 1;
public static final int VOLUME_BAIXO = 2;
```

- Usando enums

```
enum Volume {
    ALTO,
    MEDIO,
    BAIXO
}
```

Usando Enums no Código

`Volume v1 = Volume.ALTO;`	OK
`Volume v2 = 100;`	Não compila
`if (v1 == Volume.BAIXO) { ... }`	OK. É possível utilizar o operador "==" na comparação
`Volume[] v = Volume.values();`	Retorna um array com os elementos do enum

Outros Elementos de um Enum

```
public enum Prioridade {
    ALTA(10),
    MEDIA(5),
    BAIXA(1);

    private int valor;

    Prioridade(int valor) {
        this.valor = valor;
    }

    public int getValor() {
        return valor;
    }
}
```

Valores do enum

Atributo

Construtor privado

Método

```
int v = Prioridade.ALTA.getValor();
```

Classes Wrappers

- São as classes que representam os tipos primitivos
  - Desta forma os tipos primitivos podem ser adicionados a coleções ou retornados por um método que retorna objetos
- Possuem diversos métodos utilitários para os tipos primitivos
  - Converter tipos primitivos para strings e vice-versa

Classes Wrappers x Tipos Primitivos

Tipo Primitivo	Classe Wrapper
boolean	*Boolean*
byte	*Byte*
short	*Short*
char	*Character*
int	*Integer*
float	*Float*
long	*Long*
double	*Double*

Criando Objetos Wrappers

```
Integer i1 = new Integer(10);
Integer i2 = new Integer("10");
Double d1 = new Double(30.3);
Double d2 = new Double("30.3");
Character c1 = new Character('a');
```

Wrappers do valor inteiro **10**

Wrappers do valor decimal **30.3**

Wrapper do caractere **'a'**

Objetos wrappers são imutáveis: seu valor não pode ser alterado

Criando Objetos Wrappers

```
Integer i1 = Integer.valueOf("10");
```

Wrapper do valor inteiro **10**

```
Integer i2 = Integer.valueOf("1010", 2);
```

Wrapper do valor inteiro **10**, fornecido em base binária

```
Double d1 = Double.valueOf("30.3");
```

Wrapper do valor decimal **30.3**

Utilitários de Conversão

```
Integer i1 = new Integer(10);
int i = i1.intValue();
float f = i1.floatValue();
```

Converte um **Integer** para os tipos primitivos **int** e **float**

```
int i2 = Integer.parseInt("10");
```

Converte uma **String** para um **int**

```
double d1 = Double.parseDouble("4.5");
```

Converte uma **String** para um **double**

```
int i3 = Integer.parseInt("abc");
```

Lança uma exceção

Autoboxing

- Faz as conversões entre os tipos primitivos e seus wrappers de forma automática

Sem autoboxing

Integer i = 20 Integer i = 22

```
int t = i.intValue();
t = t + 2;
i = new Integer(t);
```

Com autoboxing

Integer i = 20 Integer i = 22

```
i = i + 2;
```

Autoboxing

- Os wrappers continuam sendo imutáveis
- O autoboxing é apenas uma facilidade para o programador
  - Internamente, os tipos primitivos continuam sendo "embrulhados" e "desembrulhados"

```
i = i + 2;
```

compilador

```
int t = i.intValue();
t = t + 2;
i = new Integer(t);
```