Programação orientada a objetos em Java
Classes Wrappers

Profª. Heloisa Moura

Programação orientada a objetos em Java
Classes Wrappers

Os Wrapper são conhecidos na linguagem Java como classes especiais que possuem métodos capazes de fazer conversões em variáveis primitivas e também de encapsular tipos primitivos para serem trabalhados como.

Sendo assim, existe uma classe Wrapper para cada tipo primitivo identificado pelo mesmo nome do tipo que possui e tendo a primeira letra maiúscula. Essa regra de declaração é aplicada a todos os tipos, exceto aos que são char classificados como Character e boolean como Boolean.

Programação orientada a objetos em Java
Classes Wrappers

As Classes Wrappers do Java são:

Boolean	boolean
Byte	byte
Character	char
Short	short
Integer	int
Long	long
Float	float
Double	double

Programação orientada a objetos em Java
Classes Wrappers

Tipos primitivos e Classes wrappers

	Tipo primitivo	Classe Wrapper	Subclasse
Lógico	Boolean	Boolean	Object
Caractere	Char	Caracter	
Integral	byte	Byte	Number
	short	Short	
	int	Integer	
	long	Long	
Ponto Flutante	float	Float	
	Double	Double	

Programação orientada a objetos em Java
Construtores das Classes Wrappers

Normalmente, as Classes Wrapper admitem dois construtores, um que recebe o tipo primitivo correspondente como argumento e outro que recebe uma String como argumento (Character é uma exceção).

Exemplo:

- Double d1 = new Double(1.0);
- Double d2 = new Double(“1.0”);

Programação orientada a objetos em Java
Instanciando uma Classe Wrapper

Formas de instanciar uma Classe Wrapper:

```
boolean b1 = true;
Boolean b2 = new Boolean(b1);
Boolean b3 = new Boolean(false);
short s1 = 41;
Short s2 = new Short(s1); Não dá para passar 41 direto,
Short s3 = new Short((short)41); porque esse valor é considerado int
Integer i1 = new Integer(1000);
Integer i2 = new Integer(“1000”);
Float f1 = new Float(1.0f);
Float f2 = new Float("1.0"); Se a String passada como argumento
não for um número ou não pertencer ao intervalo correspondente ao
float, então a exceção NumberFormatException é lançada
```

Programação orientada a objetos em Java
Classes Wrappers

Classes wrappers e seus construtores

Primitivo	Classe Wrapper	Argumentos do construtor
boolean	Boolean	boolean ou String
Byte	Byte	byte ou String
Char	Character	char
double	Double	double ou String
Float	Float	float, double ou String
Int	Integer	int ou String
Long	Long	long ou String
short	Short	short ou String

Programação orientada a objetos em Java
Convertendo String para Tipo Primitivo

As Classes Wrappers fornecem métodos para a conversão de uma String em um tipo primitivo correspondente.

- Exemplos desses métodos são:
- byte b = Byte.parseByte("1");
- long l = Long.parseLong("1");
- int i = Integer.parseInt("1");
- double d = Double.parseDouble("1");

Programação orientada a objetos em Java
Convertendo Tipo Primitivo em Wrapper

Também nas Classes Wrappers existem métodos para a conversão de um tipo primitivo em sua classe wrapper correspondente, sem precisar de utilizar o construtor da classe wrapper (forma de instancia).

- Exemplos desses métodos são:
- Byte b = Byte.valueOf(1);
- Integer i = Integer.valueOf(10);
- Long l = Long. valueOf(100);
- Double d = Double.valueOf(10.0);

Programação orientada a objetos em Java
Recuperando o Tipo Primitivo da classe wrapper

Cada Classe Wrapper também fornece um método que retorna o tipo primitivo por ela encapsulado.
As assinaturas desses métodos são:

```
public byte byteValue( );

public short shortValue( );

public int intValue( );
```

Programação orientada a objetos em Java
Classes Wrappers

Exemplo pratico

TesteClassesWrappers

ExemplosWrappers

Programação orientada a objetos em Java
Autoboxing e Auto-unboxing com Wrappers

Autoboxing e Auto-unboxing são recursos que vieram na versão do Java 5 para simplificar a conversão entre tipos primitivos e objetos wrappers, sem codificação adicional por parte do programador.

Autoboxing

Uma conversão boxing como é chamada converte automático um valor de tipos primitivos em um o objeto de suas classes de wrapper correspondentes.

Programação orientada a objetos em Java
Autoboxing e Auto-unboxing com Wrappers

Auto-unboxing

Uma conversão unboxing como é chamada converte automático um objeto de suas classes de wrapper em um valor de tipos primitivos correspondentes.

Exemplo pratico

ExemploAutoBoxingUnboxing

Programação orientada a objetos em Java
Classes Wrappers e Collections

- Os métodos para adição de elementos em uma coleção, normalmente, admitem um objeto como argumento e não um tipo primitivo.
- Veja o exemplo do método add da classe ArrayList:
- public boolean add(Object o)

Programação orientada a objetos em Java
Classes Wrappers e Collections

Sendo assim, para armazenar um tipo primitivo em uma coleção é necessário encapsulá-lo usando a Classe Wrapper correspondente para então, adicioná-lo à coleção.

Exemplo:

```
boolean b1 = true;

Boolean b2 = new Boolean(b1);

ArrayList list = new ArrayList();

list.add(b2);
```

Programação orientada a objetos em Java
Classes Wrappers e Collections

Exemplo pratico

ExemploWrapperCollection

Programação orientada a objetos em Java
Classes Wrappers e Collections

Quais são as vantagens de tipos primitivos?

- Tipos primitivos são bem rápidos
- Consomem pouca memória
- Além de permitirem operações mais complexas
- São bastantes eficientes em laços e expressões

Programação orientada a objetos em Java
String e Tipos Primitivos

A classe String oferece os seguintes métodos estáticos (isto é, métodos que independem de uma instância) para obter uma cadeia de caracteres (String) a partir de um dado tipo primitivo:

```
public static String valueOf(boolean b)
public static String valueOf(char c)
public static String valueOf(int i)
public static String valueOf(long l)
public static String valueOf(float f)
public static String valueOf(double d)
```

Programação orientada a objetos em Java
String e Tipos Primitivos

Um método estático é associado à classe e não a uma instância particular da mesma. É fácil entender por que o método em questão deve ser estático, pois, quando resolvemos expressar um número na forma de caracteres, ainda não temos uma String. O método a ser chamado é que irá criá-la.

Exemplo:

```
double p = 3.14;
int i = 123;
String s1 = String.valueOf(p);
String s2 = String.valueOf(i);
```

Programação orientada a objetos em Java
Classes Wrappers

EXERCÍCIO