Linguagem de Programação II

Classes úteis

Universidade do Estado do Rio de Janeiro-UERJ
Instituto de Matemática e Estatística-IME
Ciência da Computação
Professor: Alexandre Sztajnberg

Wrapper Class

- Uma Wrapper Class é uma classe cujo objeto envolve ou contém tipos de dados primitivos.
- Esta é uma outra opção para declarar tipos, na qual a Wrapper Class cria um objeto que envolve o tipo primitivo.
- As Wrapper Class possuem métodos que ajudam a manipular os dados .
- Cada tipo primitivo possui uma Wrapper Class associado a ele.

Wrapper Class

- Benefícios da Wrapper Class:
  - Sua passagem é por referência e não por valor como os tipos primitivos,
  - São compatíveis com Listas que armazenam a]somente referencias (objetos) e não tipos primitivos,
  - Podem ser usadas para dar suporte à sincronização em multithreading,
  - Usadas para conversão de tipos primitivos

Tipo Primitivo /Wrapper Class

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

Métodos das Classes Wrappers

- valorprimitivoValue() – Esse método não contém argumentos e é utilizado para realizar uma conversão do valor de um objeto do tipo Wrapper para um tipo primitivo.

  Ex: `inteiro.intValue();`
- parse*ClasseWrapper*(String s) – Método do tipo primitivo usado para converter um objeto String para um tipo primitivo, sendo que retorna um primitivo nomeado.

  Ex: `double a = Double.parseDouble("4.12");`
- toString() – Retorna um objeto do tipo String

Métodos das Classes Wrappers

- Ex:

```
import java.io.*;

public class lerInteiro{

  public static void main (String[] args) throws IOException{
     System.out.printf("Digite um numero: ");
     BufferedReader s = new BufferedReader(new InputStreamReader(System.in));
     String ler;
     ler = new String(s.readLine());
     int valor = Integer.parseInt(ler);
     System.out.println("Valor: "+valor);
  }
}
```

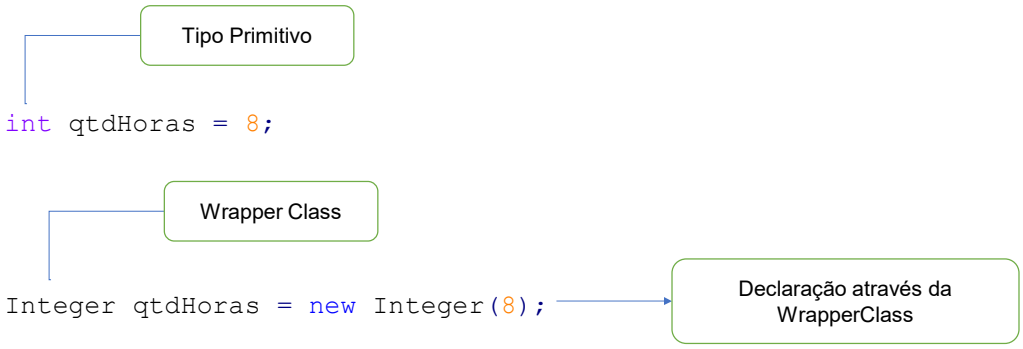
Métodos das Classes Wrappers

- Ex:

```
import java.io.*;

public class divisaoFloat{
  public static void main (String[] args) throws IOException{
     BufferedReader s = new BufferedReader(new InputStreamReader(System.in));
     String ler;
     System.out.printf("Digite um numero: ");
     String valorA = new String(s.readLine());
     System.out.printf("Digite um numero: ");
     String valorB = new String(s.readLine());
     float a = Float.parseFloat(valorA);
     float b = Float.parseFloat(valorB);
     System.out.println("Resultado: "+(a/b));
  }
}
```

Primitivo / Wrapper Class



Autoboxing e Unboxing

- Autoboxing é o recurso de conversão automática de tipos de dados primitivos em seu objeto equivalente.
  - Ex: `Integer n = 9;`
- Unboxing é o processo de conversão de objetos em tipos primitivos de dados correspondentes, ou seja, contrario ao autoboxing.
  - Ex: `int n = 0;`
    `n = new Integer(9);`

Autoboxing

- Autoboxing é aplicada pelo compilador quando:
  - Quando um valor primitivo é passado como um parâmetro para um método que espera um objeto da classe Wrapper correspondente.
  - Quando um valor primitivo é atribuído a uma variável da classe Wrapper correspondente.

Autoboxing

- Ex:

```
public int somaPar(List<Integer> listaInteiro ) {
    int soma = 0;
    for (Integer i: listaInteiro )
        if ( i % 2 == 0 )
            soma += i; //Integer é transformado em int em tempo de execução
     return soma;
}
```

Unboxing

- Unboxing é aplicada pelo compilador quando:
  - Quando um objeto é passado como um parâmetro para um método que espera um valor primitivo correspondente.
  - Quando um objeto é atribuído a uma variável do tipo primitivo correspondente.

Unboxing

- Ex:

```
public class somaInt{
  public static void main (String[] args){
     int a = 5;
     Integer b = new Integer(5);
     int c = a+b;
     System.out.println(c);
  }
}
```

ArrayList

- Graças ao uso da classe Wrapper e os recursos de autoboxing e unboxing, podemos criar um ArrayList de um tipo primitivo usando como parâmetro o seu correspondente Wrapper.

ArrayList

```
import java.util.*;

public class ArrayListInteger{

  public static void main(String args[]){
     int n = 100;
     ArrayList<Integer> a= new ArrayList<Integer>();
     for(int i=1;i<=n;i++){
          a.add(i);
     }
     System.out.println(a);
     n=0;
     for(Integer i: a){
          a.set(n,i*5);
          n++;
     }
     System.out.println(a);
  }
}
```