

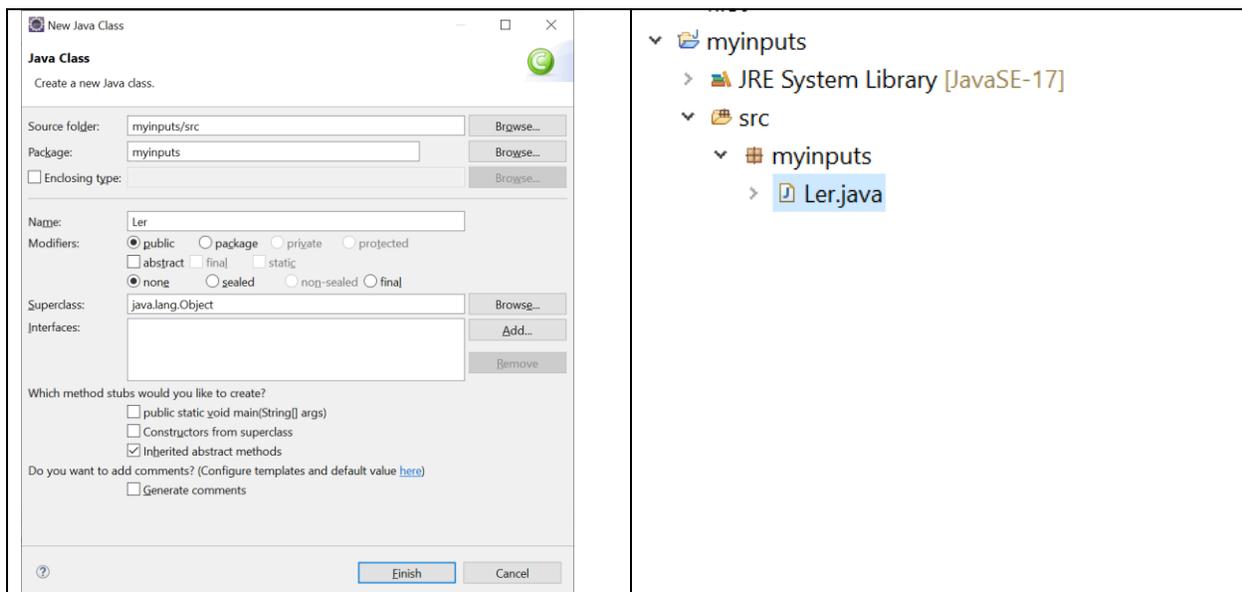
ESQUEMA AULA PRÁTICA 2

- ❑ **Introdução à linguagem Java (continuação)**
- ❑ **Leitura de dados do teclado com tratamento de erros**

1 – Na aula prática anterior aprendemos a ler valores do teclado através de uma caixa de diálogo mas não foi feito qualquer tratamento de erros. Se o programa espera um inteiro e o utilizador introduz um valor real o programa termina assinalando um erro. Vamos implementar uma classe, **Ler**, que nos permita ler os principais tipos primitivos da linguagem de uma forma mais robusta. Mais tarde aprenderemos os conceitos que nos permitirão compreender esta classe na totalidade.

a) Crie um projeto com o nome **myinputs**.

b) Nesse projeto, crie agora uma nova classe, **Ler**, no **package myinputs**. Desta vez não selecione a opção de criar o método main (ver figura). Pode observar que no seu editor (à esquerda) aparece agora a classe Ler dentro da pasta myinputs.



c) Para a classe Ler, copie o código listado abaixo:

A classe contém um método para ler uma cadeia de caracteres (isto é, uma String) e um método para ler cada um dos tipos primitivos da linguagem java (char, byte, short, int, long, float, double e boolean). Acima do cabeçalho da classe foi inserida a linha: `import java.io.*;`

Ficando:

```

package myinputs;

import java.io.*;
public class Ler {
// Método para ler uma String:
public static String umaString (){
    String s = "";
    try{
        BufferedReader in = new BufferedReader ( new InputStreamReader (System.in));
        s= in.readLine();
    }
    catch (IOException e){
        System.out.println("Erro ao ler fluxo de entrada.");
    }
    return s;
}
// Método para ler um int:
public static int umInt(){
    while(true){
        try{
            return Integer.parseInt(umaString().trim());
        }
        catch(NumberFormatException e){
            System.out.println("Não é um inteiro válido!!!");
        }
    }
}
// Método para ler um byte:
public static byte umByte(){
    while(true){
        try{
            return Byte.parseByte(umaString().trim());
        }
        catch(NumberFormatException e){
            System.out.println("Não é um byte válido!!!");
        }
    }
}
}

```

```
// Método para ler um short:
public static short umShort(){
    while(true){
        try{
            return Short.parseShort(umaString().trim());
        }
        catch(NumberFormatException e){
            System.out.println("Não é um short válido!!!");
        }
    }
}

// Método para ler um long:
public static long umLong(){
    while(true){
        try{
            return Long.parseLong(umaString().trim());
        }
        catch(NumberFormatException e){
            System.out.println("Não é um long válido!!!");
        }
    }
}

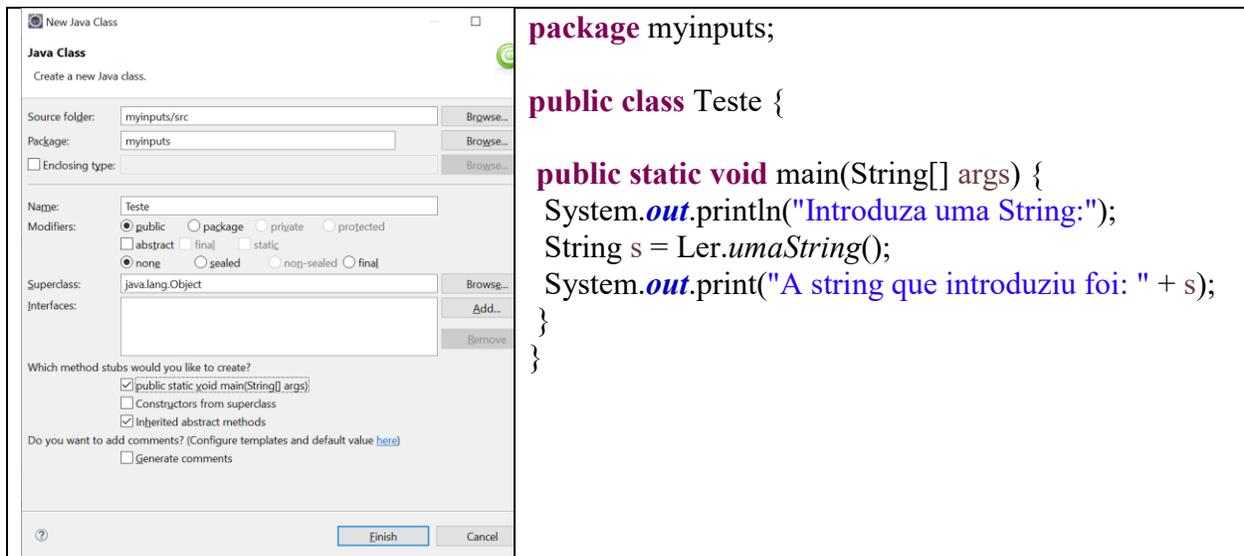
//// Método para ler um float;
public static float umFloat(){
    while(true){
        try{
            return Float.parseFloat(umaString().trim());
        }
        catch(NumberFormatException e){
            System.out.println("Não é um float válido!!!");
        }
    }
}

// Método para ler um double:
public static double umDouble(){
    while(true){
        try{
            return Double.valueOf(umaString().trim());
        }
        catch(NumberFormatException e){
            System.out.println("Não é um double válido!!!");
        }
    }
}
```

```
// Método para ler um char:
public static char umChar(){
    while(true){
        try{
            return umaString().charAt(0);
        }
        catch(Exception e){
            System.out.println("Não é um char válido!!!");
        }
    }
}

// Método para ler um boolean:
public static boolean umBoolean(){
    while(true){
        try{
            return Boolean.parseBoolean(umaString().trim());
        }
        catch(Exception e){
            System.out.println("Não é um boolean válido!!!");
        }
    }
}
}
```

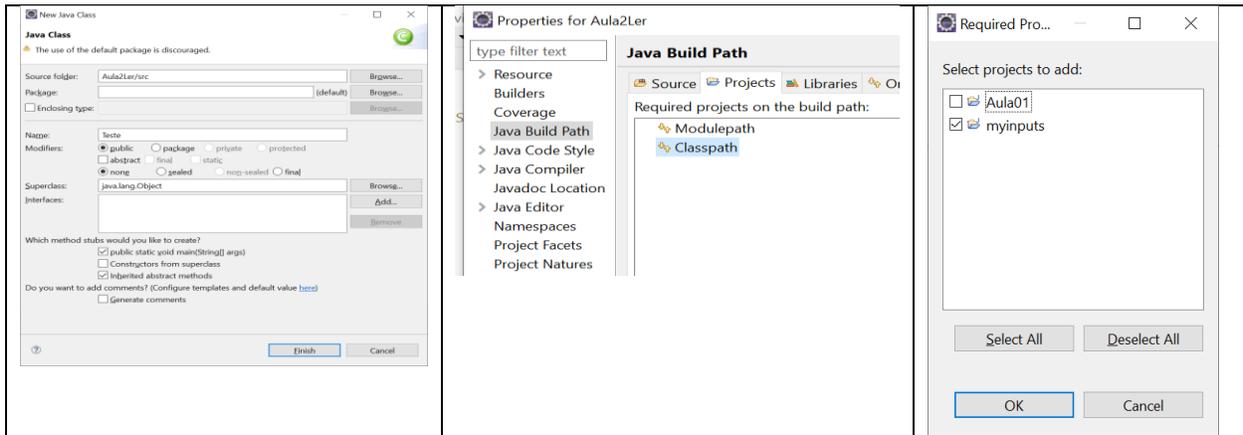
d) Para testar a leitura de dados vamos criar, no projeto anterior, uma nova classe, **Teste**, que desta vez deve ter o método main (ver figura à esquerda) e o conteúdo listado na figura (à direita) - Execute o programa que acabou de criar.



- e) Por analogia, teste agora o método para ler um valor do tipo int.
 - Ainda para testar o método umInt, experimente introduzir um valor do tipo char. O que acontece?
 - E se introduzir um valor do tipo double. O que acontece?
- f) Teste o funcionamento de cada um dos outros métodos da classe Ler.
- g) Escreva um programa que leia 3 inteiros do teclado, usando a classe Ler, e determine qual o maior dos valores usando o operador ternário.

❑ Usar classes de um projeto noutro projeto

- A partir de agora podemos usar a classe Ler noutros projetos. Crie um novo projeto, **Aula2Ler**, ou outro nome à sua escolha. Colocando o rato sobre o nome do projeto, com o botão direito selecione **Properties**, depois em **Java Build Path** na aba **Projects** selecione **Classpath**. Clique no botão **Add** e selecione o projeto myinputs (ver figuras). Após **ok** e **apply and close**, pode aceder à classe Ler nas classes do seu novo projeto.



- Crie uma classe Teste com o cabeçalho do método main. Acima do cabeçalho da classe, deve fazer o import do package myinputs. Teste o programa abaixo:

```
import myinputs.Ler;
public class Teste {
    public static void main(String[] args) {
        System.out.println ("Escreva um inteiro:");
        int x = Ler.umInt();
        System.out.println ("Introduziu o inteiro:" + x );
    }
}
```

Para treinar:

Exercícios de programação para revisão da linguagem C

Resolva os exercícios que se seguem utilizando os métodos disponibilizados pela classe `Ler` para ler valores do teclado.

1 – Escreva um programa que leia do teclado dois números inteiros, p e u , tais que $p \leq u$, e mostre o resultado de

$$\sum_{i=p}^u i \quad (= p + p+1 + p+2 + \dots + u-1 + u)$$

- a) usando um ciclo `for`;
- b) usando um ciclo `while`;
- c) usando um ciclo `do-while`.

2 – Escreva um programa que leia uma palavra do teclado e mostre a letra com menor valor de código ASCII correspondente.

3 – Faça um programa para “inverter” os dígitos um número inteiro positivo (e.g. 123 passa a 321).

4 – Faça um programa que peça ao utilizador uma chave do Totoloto (6 números inteiros distintos entre 1 e 49).

5 – Modifique o programa anterior para que seja capaz de gerar de forma automática uma chave do Totoloto (6 números inteiros distintos entre 1 e 49). Utilize o gerador de números pseudo-aleatórios da classe `Math`: `Math.random()`. Não se esqueça de que uma chave não pode ter valores duplicados.

*O método `Math.random()` devolve um valor do tipo `double` pertencente ao intervalo $[0, 1[$. (É possível converter um dado tipo num outro compatível usando o operador unário de coerção (*casting*)).*

6 – Modifique o programa do exercício anterior de modo a que o utilizador indique o número de chaves que pretende.

7 – Apresente o histograma dos números gerados pelo programa anterior.