

## ESQUEMA AULA PRÁTICA #3

- **Construir um arquivo (JAR) com classes java, e incluí-lo noutra projecto**
  
- **Objetos e classes de teste**

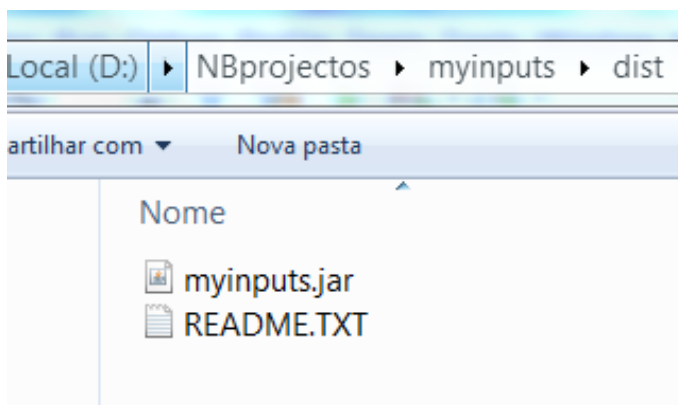
Na folha prática 2, exercício 2, criou, num projeto de nome myinputs, uma classe Ler.

- Verifique se completou e testou a classe com métodos para ler cada um dos tipos primitivos da linguagem Java, (byte, short, int, long, float, double, boolean, char,).

Queremos agora usar a classe Ler, nos projetos a criar daqui adiante.

**1** – Para gerar um arquivo com a classe Ler, fazer o seguinte:

- Apagar do projeto a classe que contém o método main (não precisamos dela noutros projetos).
- Selecione o projeto myinputs e clicando no botão direito do rato selecione Clean and Build.
- Verifique na diretoria do projeto que foi criada uma diretoria de nome **dist**, e que esta diretoria contém um ficheiro com o nome: **myinputs.jar**.

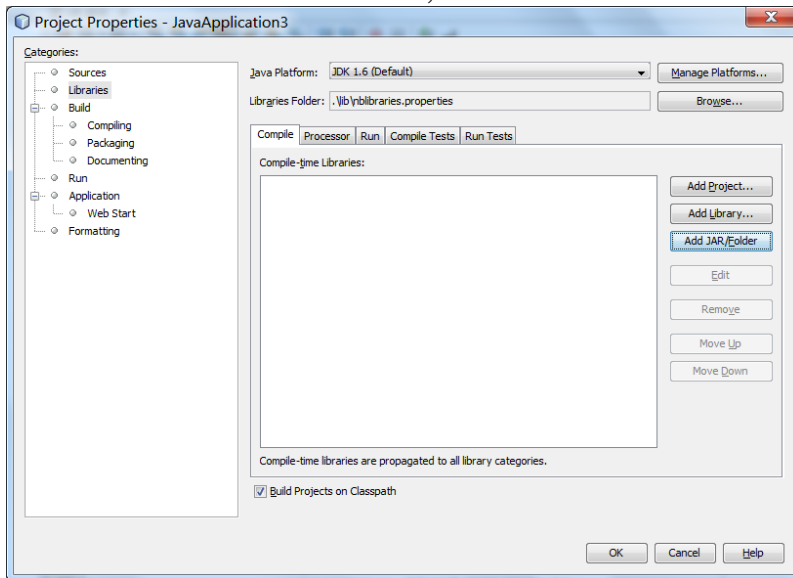


**2** – Usar a classe Ler num outro projeto.

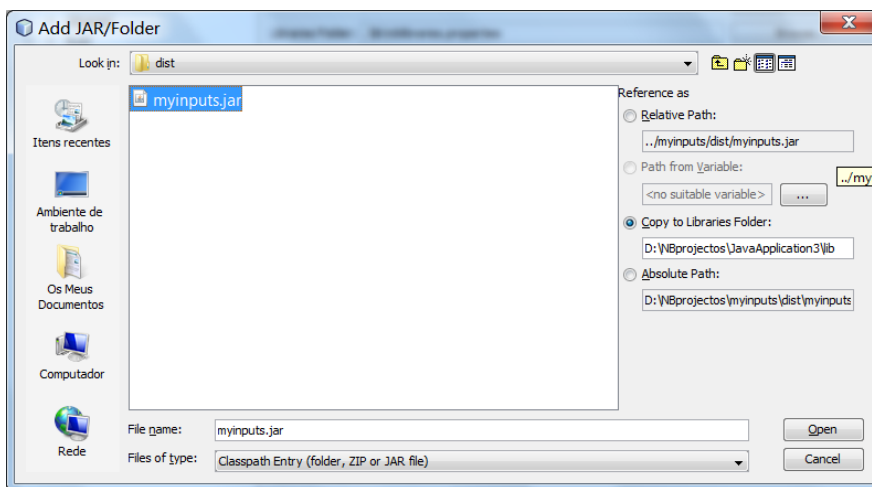
- a) Crie um novo projecto de nome **TestarJar**.
- b) Para adicionar o arquivo criado (ficheiro JAR) ao projecto **TestarJar**, fazer o seguinte:

**NetBeans 8.2:**

- Selecionar o projeto ao qual pretende adicionar o ficheiro JAR. Clicar no botão direito de rato e seleccionar **Properties**.
- No nó Libraries seleccionar, Add JAR/Folder.



- Seleccione o ficheiro **myinputs.jar**, seleccione **Open**, e clique **OK**.

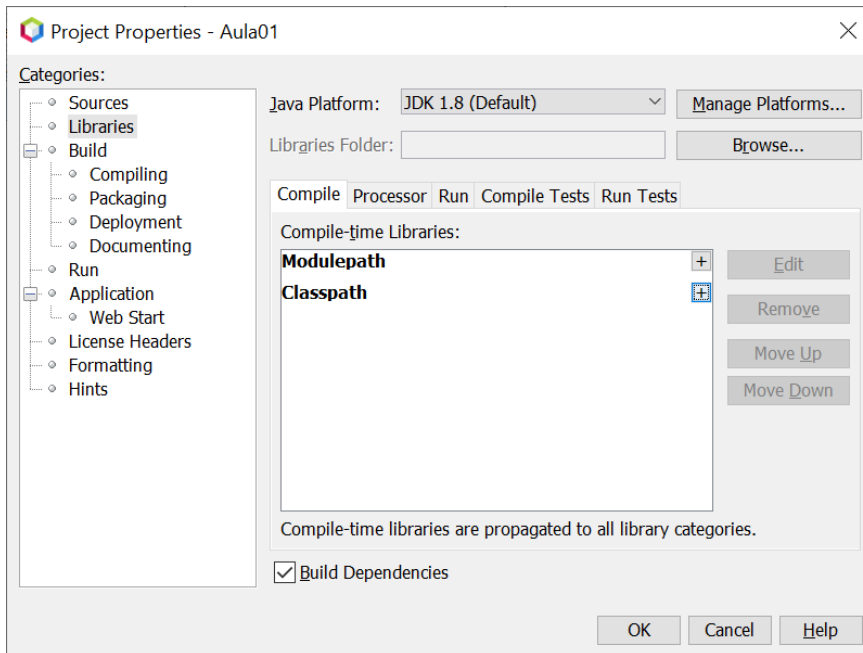


- Pode verificar que a pasta Libraries do seu projeto inclui agora o ficheiro myinputs.jar.

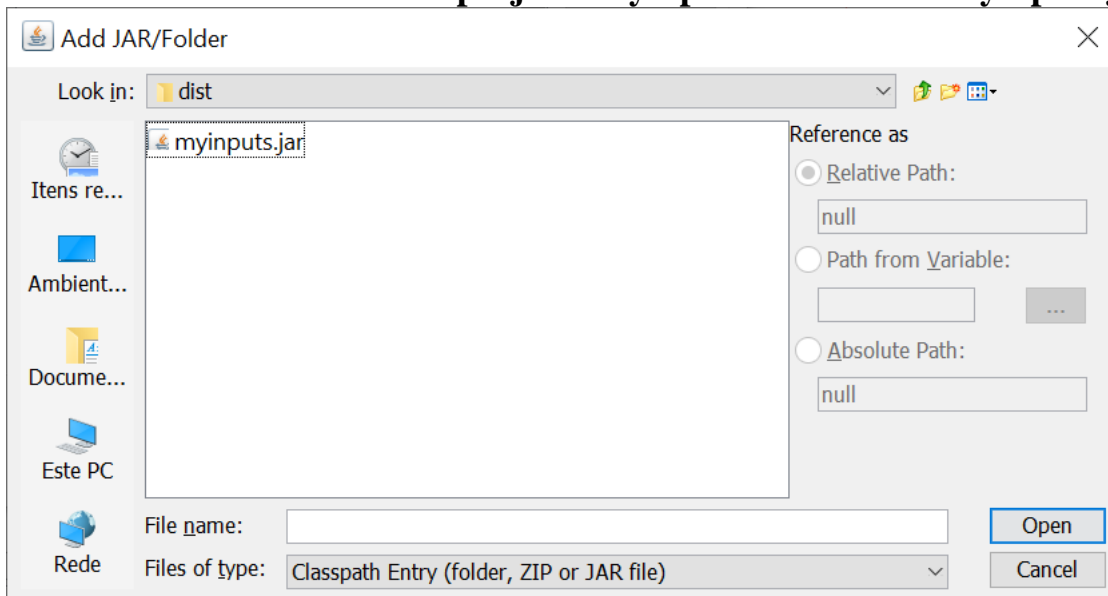


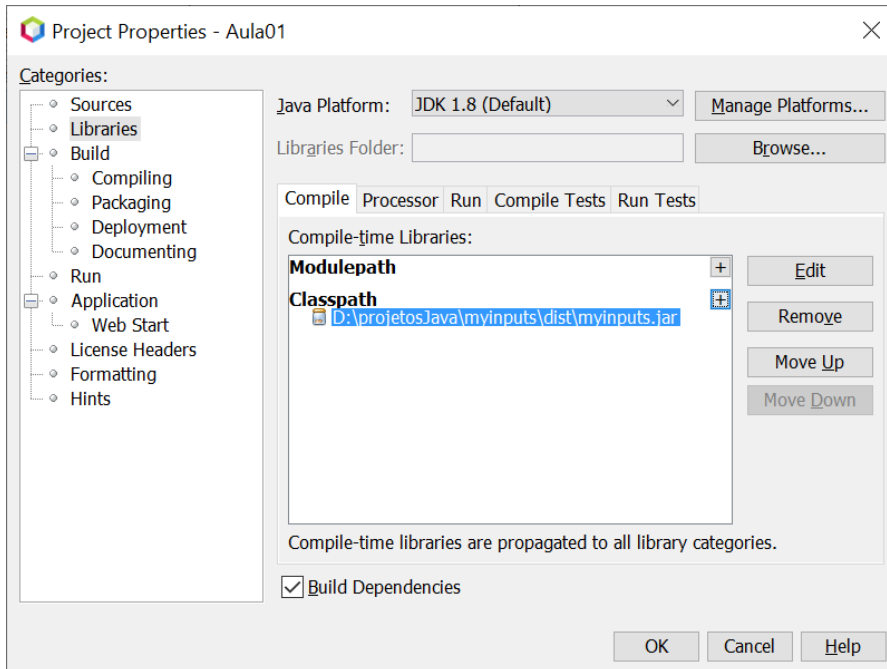
**Apache NetBeans 12.\*:**

- Seleccionar o projeto ao qual pretende adicionar o ficheiro JAR. Clicar no botão direito de rato e seleccionar **Properties**.
- **No nó libraries / compile, abrir Classpath e seleccionar Add JAR/Folder,**

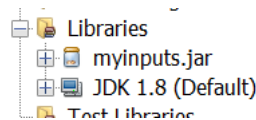


**Posicione-se na diretoria do projecto myinputs e seleccione myinputs.jar:**





Após <ok> poderá observar que o nó **Libraries** do seu projeto inclui agora o ficheiro **myinputs.jar**.



- É agora possível usar a classe **Ler** neste novo projeto.

- Experimentar, no método **main** do seu projecto **TesteJar**, chamar o método **umInt** da classe **Ler**, fazendo:

```
int i;
i = myinputs.Ler.umInt();
```

Pode omitir-se o nome **myinputs** se, antes do cabeçalho da classe, especificar a cláusula de **import**:

```
import myinputs.*; /o asterisco significa, todas as classes do arquivo.
```

Em alternativa podia fazer: **import myinputs.Ler;**

**4** – Escreva um programa que leia 3 inteiros do teclado e mostre o maior.

## Objetos e classes de teste

- ❑ Definição de classes instanciáveis
- ❑ Protecção de variáveis de instância
- ❑ Métodos construtores
- ❑ Métodos de consulta e modificação de variáveis de instância
- ❑ O método toString

### 5 - Pretende-se implementar a classe Contador que foi estudada nas aulas teóricas.

a) As instâncias da classe Contador deverão representar contadores do tipo inteiro capazes de responder a um conjunto de mensagens que implementam as seguintes operações:

- criar um Contador com valor inicial igual a zero;
- criar um Contador com valor inicial igual a uma valor dado pelo utilizador;
- incrementar o Contador de uma unidade;
- incrementar o Contador de uma valor dado pelo utilizador;
- decrementar o Contador de uma unidade;
- decrementar o Contador de uma valor dado pelo utilizador;
- consultar o valor do Contador ;
  
- implemente ainda o método **toString**.

b) Construa uma classe de Teste para verificar a correcção da classe anterior.

c) Depois de testar o método toString estude o que acontece quando numa instrução de escrita coloca apenas o nome do objecto sem lhe enviar a mensagem toString. Finalmente, transforme em comentário o método toString e volte a executar o programa anterior. O que acontece?

d) Construa um programa que gere aleatoriamente N valores inteiros no intervalo de [-100, 100[ sendo o valor de N dado pelo utilizador. Usando objectos da classe Contador o programa deverá contar quantos valores gerados são positivos e quantos são negativos.

6 – Considere a classe Espetaculo. Um espetáculo é um evento que tem um **nome**, tem uma certa **capacidade** (isto é, o número máximo de pessoas que pode assistir ao espetáculo) e ao qual assiste um certo número de **espetadores** que não poderá ser superior à capacidade. Um espectáculo tem ainda um **custo** que corresponde ao preço do bilhete para assistir ao espectáculo.

- a) Defina o cabeçalho e os atributos da classe Espetaculo.
- b) Defina o construtor que recebe como parâmetros o nome, a capacidade e o custo do Espetáculo.
- c) Construa os getters e setters para cada atributo da classe.
- d) Construa o método toString para a classe Espetaculo.
- e) Para a classe Espetaculo construa o método comprarBilhete. O método deverá adicionar uma unidade ao número de espectadores caso ainda haja lugares disponíveis. Nesse caso, o método deve devolver como resultado a String “Tem a pagar €€€”, onde €€€ deve ser substituído pelo custo do bilhete. Se já não houver lugares disponíveis, deverá devolver a mensagem “Espetáculo esgotado”
- f) Construa uma classe para testar a classe Espetaculo.