JAVA BÁSICO

LABORATÓRIO

1. Abra o **Eclipse** e selecione para “File > New > Java Project”; em **Project Name** coloque “Lab.Excecoes” e então confirme.

2. Busque a **pasta** com o nome “src” no lado esquerdo no **Package Explorer**

   (a) Clique com o botão direito na pasta “src” e selecione “New > Class”

   (b) No campo nome digite “Main” e confirme pressionando **Finish**

3. Caso o novo código não tenha sido aberto automaticamente, busque no **Package Explorer** dentro da pasta “src” o arquivo de classe recém criado “Main.java”

4. Crie o método **main** para a classe Main como mostrado abaixo

```
public static void main(String[] args) {
        String s = null;

        char ch = s.charAt(0);

        System.out.println(ch);
}
```

5. Execute o código e observe o resultado

   (a) A exceção abaixo foi mostrada no console

```
Exception in thread "main" java.lang.NullPointerException
at Main.main(Main.java:11)
```

6. Isso indica que um objeto nulo foi acessado, a máquina virtual Java emite uma exceção para indicar que a operação foi invalida

7. Para tratar uma exceção devemos criar um bloco **try-catch** e capturar o tipo da exceção emitida

   (a) Execute e teste o código abaixo

```
Try {
        String s = null;

        char ch = s.charAt(0);
        System.out.println(ch);
} catch (NullPointerException ex) {
        System.out.println("Exceção de ponteiro nulo capturada");
}
```

8. É importante notar que a exceção desfaz a pilha de execução até encontrar um **try-catch**, ou seja, o que aconteceria no exemplo abaixo?

```
public static void test()

{
        String s = null;
        s.charAt(0);
}

public static void main(String[] args) {

        try {
                Main.test();
        } catch (NullPointerException ex) {
                System.out.println("Exceção de ponteiro nulo capturada");
        }
}
```

9. Realize uma pequena modificação no código original para que se torne igual ao mostrado abaixo

```
try {

	String s = "Teste";
	char ch = s.charAt(5);

	System.out.println(ch);
} catch (NullPointerException ex) {
	System.out.println("Ponteiro nulo");
}
```

10. Qual a exceção emitida? O **catch** conseguiu capturar a exceção?

```
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String
index out of range: 5

	at java.lang.String.charAt(String.java:658)
	at Main.main(Main.java:18)
```

11. Percebemos que o **try-catch** so captura as exceções que especificamos, para adicionar também tratamento para exceções de StringIndexOutOfBoundsException podemos utilizar

```
try {

	String s = "Teste";
	char ch = s.charAt(5);

	System.out.println(ch);
} catch (NullPointerException ex) {
	System.out.println("Ponteiro nulo");
} catch (StringIndexOutOfBoundsException ex) {
	System.out.println("String fora dos limites");
}
```

12. Podemos também utilizar a classe base comum para tratar

qualquer tipo de exceção

(a) Tome cuidado, tratar exceções de maneira genérica pode esconder erros

(b) Um exemplo comum disso é encontrado abaixo:

```
        try {

              String s = "Teste";
              char ch = s.charAt(5);

              System.out.println(ch);
        } catch (Throwable ex) {
              System.out.println("Erro na utilização do sistema,
                                  por favor consulte o manual");
        }
```

13. Também podemos utilizar o comando **finally** para garantir que um determinado comando será executado independente de uma exceção ser emitida ou não.

```
  public static void main(String[] args) {


        try {

              System.out.println("Digite uma palavra: ");
              Scanner scanner = new Scanner(System.in);
              String s = scanner.nextLine();
              System.out.println("CharAt(3) = " + s.charAt(3));

        } catch (StringIndexOutOfBoundsException ex) {
              System.out.println("ERROR: String out of bounds");
        } finally {
              System.out.println("Finally rodou");
        }
  }
```

14. Algumas exceções obrigam que o usuário as avalie. Por exemplo **FileNotFoundException**

(a) Caso o usuário não lide com a exceção o programa não compilará

(b) Essas são chamadas de **Checked Exceptions**; todas as exceções que herdam de **Exception** devem ser tratadas

(c) Um exemplo:

```
└ java.lang.Throwable
   └ java.lang.Exception
      └ java.io.IOException
         └ java.io.FileNotFoundException
```

(d) Do outro lado, exceções que herdam de RuntimeException não necessariamente devem ser tratadas.

i. Caso um programa não trate uma exceção ele será abortado

---

15. Para disparar uma exceção podemos utilizar o comando **throw**

```
public static void main(String[] args) {

      throw new RuntimeException("Esse programa não foi feito");

}
```