



# EducaCiência FastCode

Fala Galera,

Neste artigo, abordaremos um tema muito interessante.

- Artigo: 10/2020 Data: Fevereiro/2020
- Público Alvo: Desenvolvedores – Iniciantes ao Avançado
- Tecnologia: Exceção NullPointerException Java
- Tema: Tratamento NullPointerException (If/Else, Optional, OrElse)
- Link: <https://github.com/perucello/DevFP>

Desta vez , escolhi um tema interessante, vamos entrar um pouco no que mais atormenta o Desenvolvedor – eu mesmo sofro muito com **NullPointerException** mais conhecido como **NPE**. O tema que escolhi, para este artigo é como tratarmos o **NPE** !

Primeiramente, vamos entender o que é NPE e o porquê que ele ocorre !

O famoso **NPE** mais conhecido como **NullPointerException** é uma exceção lançada pelo Java quando teoricamente seu programa tenta acessar um determinado objeto na memória em que não foi inicializado ou não há um retorno válido, ocorre normalmente quando:

- ✓ *Ao acessar métodos de objetos que estão nulos.*
- ✓ *Alterar ou visualizar atributos de objetos nulos.*
- ✓ *Verificação de tamanho (length) de um array ainda nulo*
- ✓ *Modificação de campos de um array nulo*
- ✓ *Lançar uma exceção como se ela fosse throwable (“lançável”).*

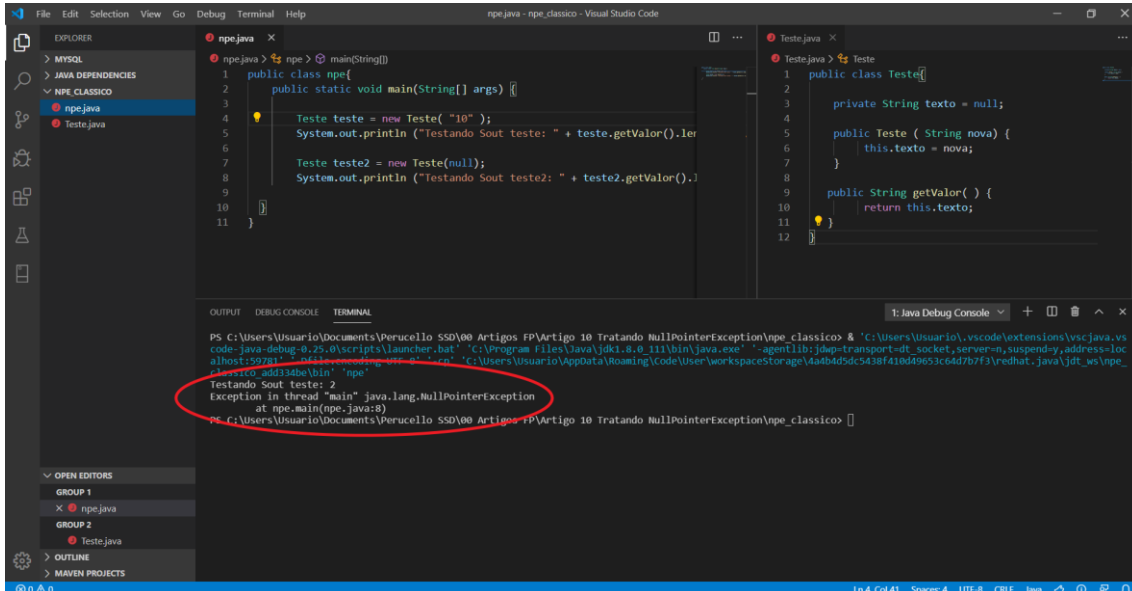
Fonte: <https://docs.oracle.com/javase/8/docs/api/java/lang/NullPointerException.html>  
<https://www.devmedia.com.br/java-lang-nullableexception-dica/28677>

Vale lembrar que para o Java, tudo basicamente subentende-se que é uma “**classe**” e se porventura, você for utilizar de qualquer de um dos objetos desta “classe” terá que ter em mente que:

- a) *Deve declarar sempre as variáveis*
- b) *Deve sempre iniciá-las*



Vamos para um exemplo clássico que nos retornará **NPE**, note que para o objeto **teste2**, nós o referenciamos para um objeto do tipo “**null**”, porém, deveria existir uma classe **String** mas neste exemplo, havia apenas a referência como “**null**”.



```

1 public class npe {
2     public static void main(String[] args) {
3
4         Teste teste = new Teste( "10" );
5         System.out.println ("Testando Sout teste: " + teste.getValor().length());
6
7         Teste teste2 = new Teste(null);
8         System.out.println ("Testando Sout teste2: " + teste2.getValor().length());
9
10    }
11 }

```

```

1 public class Teste {
2
3     private String texto = null;
4
5     public Teste ( String nova ) {
6         this.texto = nova;
7     }
8
9     public String getValor() {
10        return this.texto;
11    }
12 }

```

Terminal Output:

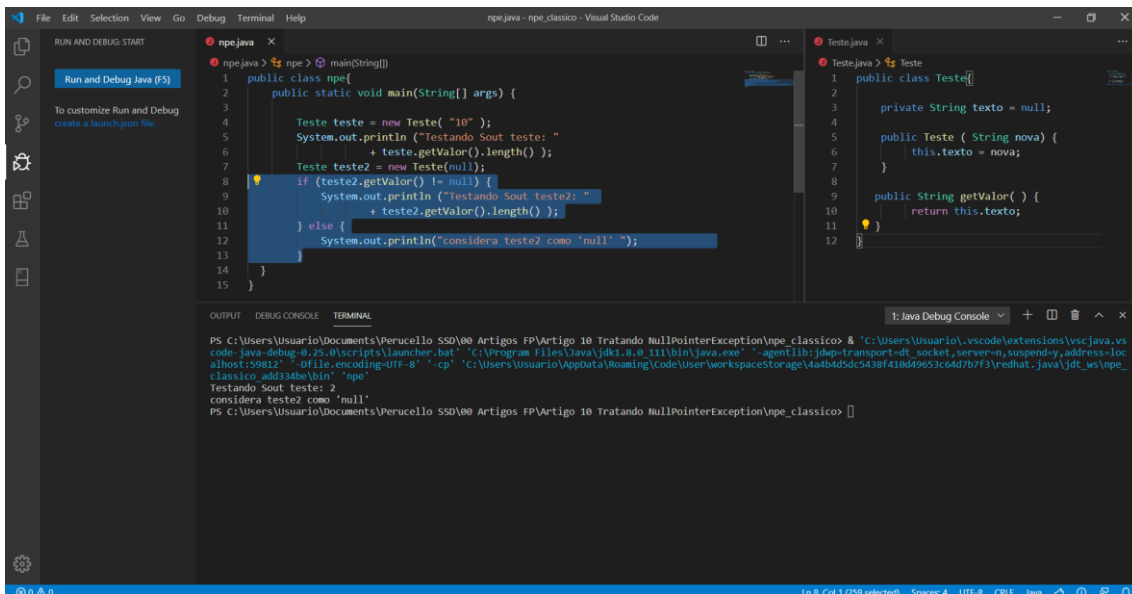
```

PS C:\Users\Usuario\Documents\Perucello SSD\00 Artigos FP\Artigo 10 Tratando NullPointerException\npe_classico & "C:\Users\Usuario\vscode\extensions\vscjava.vscode-java-debug-0.25.0\scripts\launcher.bat" "C:\Program Files\Java\jdk1.8.0_111\bin\java.exe" "-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=5005" -cp "C:\Users\Usuario\AppData\Roaming\Code\User\workspaceStorage\4a4b4d5dc5438f410d49653c6dd7b7f3\redhat.java\jdk_ws\npe_classico_add344be\bin" npe
Testando Sout teste: 2
Exception in thread "main" java.lang.NullPointerException
    at npe.main(npe.java:8)
PS C:\Users\Usuario\Documents\Perucello SSD\00 Artigos FP\Artigo 10 Tratando NullPointerException\npe_classico &

```

No caso acima citado, sanamos o problema com a condição **If/Else** onde para o caso de **teste2** tratamos a condição se “**for igual**” a null como “**se for diferente**” de null.

Ficou da seguinte maneira:



```

1 public class npe {
2     public static void main(String[] args) {
3
4         Teste teste = new Teste( "10" );
5         System.out.println ("Testando Sout teste: "
6             + teste.getValor().length() );
7
8         Teste teste2 = new Teste(null);
9
10        if (teste2.getValor() != null) {
11            System.out.println ("Testando Sout teste2: "
12                + teste2.getValor().length() );
13        }
14        else {
15            System.out.println("considera teste2 como 'null' ");
16        }
17    }
18 }

```

```

1 public class Teste {
2
3     private String texto = null;
4
5     public Teste ( String nova ) {
6         this.texto = nova;
7     }
8
9     public String getValor() {
10        return this.texto;
11    }
12 }

```

Terminal Output:

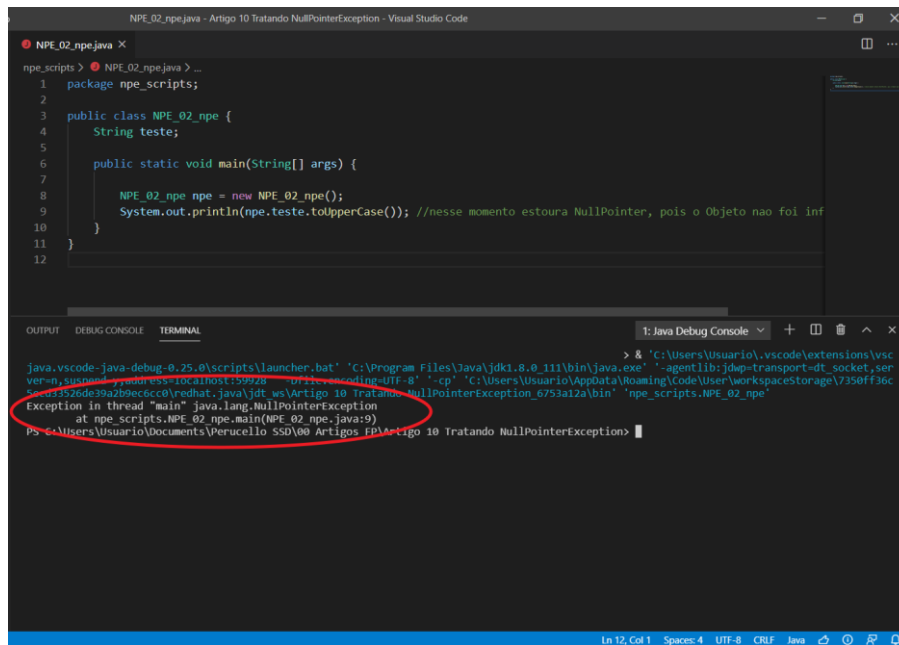
```

PS C:\Users\Usuario\Documents\Perucello SSD\00 Artigos FP\Artigo 10 Tratando NullPointerException\npe_classico & "C:\Users\Usuario\vscode\extensions\vscjava.vscode-java-debug-0.25.0\scripts\launcher.bat" "C:\Program Files\Java\jdk1.8.0_111\bin\java.exe" "-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=5005" -cp "C:\Users\Usuario\AppData\Roaming\Code\User\workspaceStorage\4a4b4d5dc5438f410d49653c6dd7b7f3\redhat.java\jdk_ws\npe_classico_add344be\bin" npe
Testando Sout teste: 2
considera teste2 como 'null'
PS C:\Users\Usuario\Documents\Perucello SSD\00 Artigos FP\Artigo 10 Tratando NullPointerException\npe_classico &

```



Já em uma outra classe “NPE\_02\_npe” que criamos, estamos forçando o Sistema a nos retornar um **NullPointerException** (npe), justamente pelo fato de pedirmos para que a variável seja convertida para **UpperCase**, como não temos nenhum valor a ser convertido, vai estourar um “npe”.



```

1 package npe_scripts;
2
3 public class NPE_02_npe {
4     String teste;
5
6     public static void main(String[] args) {
7
8         NPE_02_npe npe = new NPE_02_npe();
9         System.out.println(npe.teste.toUpperCase()); //nesse momento estoura NullPointerException, pois o Objeto nao foi inf
10     }
11 }
12

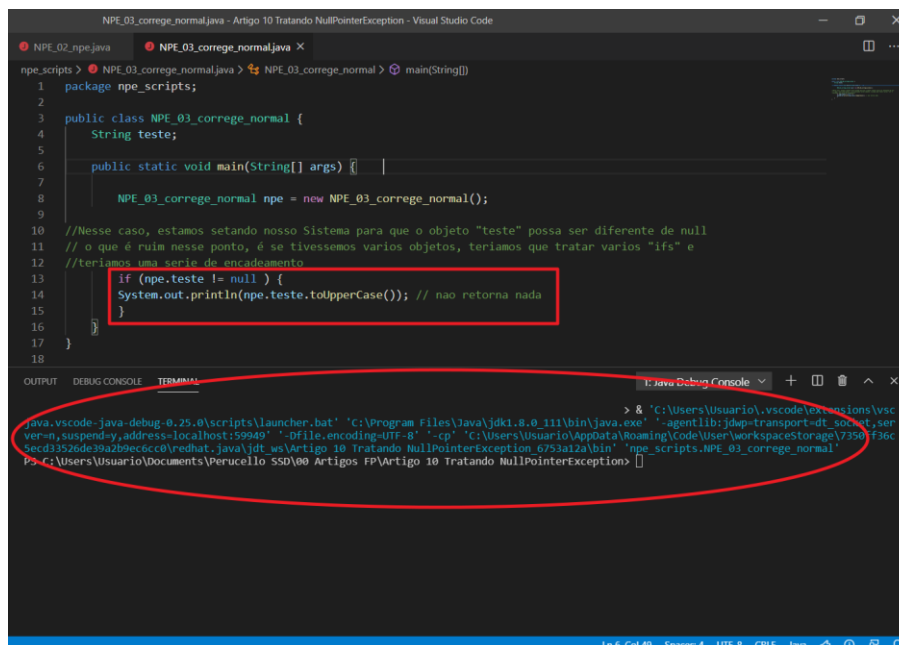
```

```

> & 'C:\Users\Usuario\.vscode\extensions\vsc
java:vscode-java-debug-0.25.0\scripts\launcher.bat' 'C:\Program Files\Java\jdk1.8.0_111\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,ser
ver=0,suspend=y,address=localhost:59949' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\Usuario\AppData\Roaming\Code\User\workspaceStorage\7350f736c
5ecd33526de39a2b9ec6cc0\redhat.java\jdk1.8.0_111\bin\java.exe' 'npe_scripts.NPE_02_npe'
Exception in thread "main" java.lang.NullPointerException
    at npe_scripts.NPE_02_npe.main(NPE_02_npe.java:9)
PS C:\Users\Usuario\Documents\Perucello SSD\00 Artigos FP\Artigo 10 Tratando NullPointerException>

```

Nesse ponto, fizemos um “tratamento” em que o Sistema irá “corrigir o problema” de maneira que o objeto “teste” não apresente o erro, porém, fizemos um tratamento com “If” condicionando ele a ser diferente de null, pode ser uma boa pratica de momento, porém, imagina se você tivesse vários objetos, neste momento, teria que ter “muitos If’s” para condicionar seu sistema a não retornar um “npe”, vejamos o código abaixo:



```

1 package npe_scripts;
2
3 public class NPE_03_correge_normal {
4     String teste;
5
6     public static void main(String[] args) {
7
8         NPE_03_correge_normal npe = new NPE_03_correge_normal();
9
10        //Nesse caso, estamos setando nosso Sistema para que o objeto "teste" possa ser diferente de null
11        // o que é ruim nesse ponto, é se tivessemos varios objetos, teriamos que tratar varios "ifs" e
12        //teriamos uma serie de encadeamento
13        if (npe.teste != null) {
14            System.out.println(npe.teste.toUpperCase()); // nao retorna nada
15        }
16    }
17 }
18

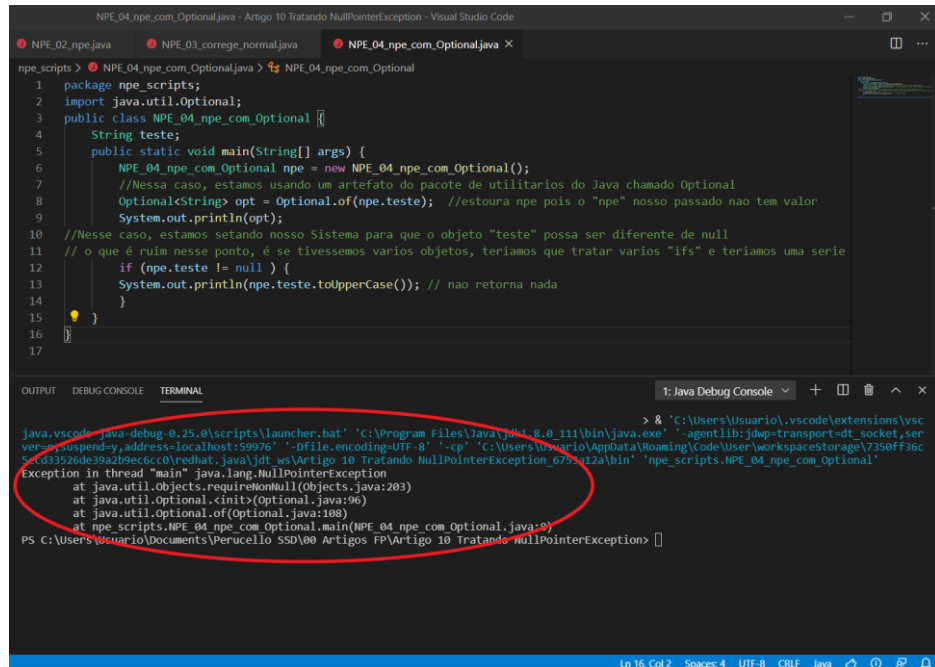
```

```

> & 'C:\Users\Usuario\.vscode\extensions\vsc
java:vscode-java-debug-0.25.0\scripts\launcher.bat' 'C:\Program Files\Java\jdk1.8.0_111\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,ser
ver=0,suspend=y,address=localhost:59949' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\Usuario\AppData\Roaming\Code\User\workspaceStorage\7350f736c
5ecd33526de39a2b9ec6cc0\redhat.java\jdk1.8.0_111\bin\java.exe' 'npe_scripts.NPE_03_correge_normal'
PS C:\Users\Usuario\Documents\Perucello SSD\00 Artigos FP\Artigo 10 Tratando NullPointerException>

```

Uma boa prática que o Java8 nos proporciona, é trabalhar com a importação do pacote de utilitário chamado “**OPTIONAL**”, mas, ele também deve ser trabalhado com cautela, pois se escrito de maneira incorreta, também vai nos levar a um **NullPointerException**, vejamos !



```

1 package npe_scripts;
2 import java.util.Optional;
3 public class NPE_04_npe_com_Optional {
4     String teste;
5     public static void main(String[] args) {
6         NPE_04_npe_com_Optional npe = new NPE_04_npe_com_Optional();
7         //Nessa caso, estamos usando um artefato do pacote de utilitarios do Java chamado Optional
8         Optional<String> opt = Optional.of(npe.teste); //estoura npe pois o "npe" nosso passado nao tem valor
9         System.out.println(opt);
10    }
11    //Nesse caso, estamos setando nosso Sistema para que o objeto "teste" possa ser diferente de null
12    // o que é ruim nesse ponto, é se tivéssemos varios objetos, teriamos que tratar varios "ifs" e teriamos uma serie
13    if (npe.teste != null) {
14        System.out.println(npe.teste.toUpperCase()); // nao retorna nada
15    }
16 }
17

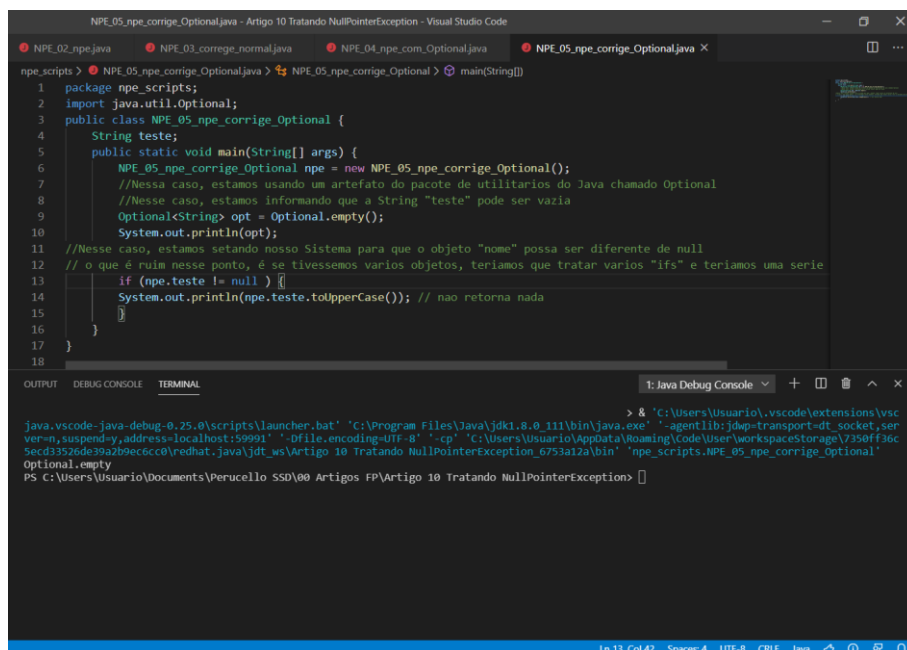
```

```

java.lang.NullPointerException
    at java.util.Objects.requireNonNull(Objects.java:203)
    at java.util.Optional.<init>(Optional.java:96)
    at java.util.Optional.of(Optional.java:108)
    at npe_scripts.NPE_04_npe_com_Optional.main(NPE_04_npe_com_Optional.java:8)
PS C:\Users\Usuario\Documents\Perucello SSD\00 Artigos FP\Artigo 10 Tratando NullPointerException>

```

Novamente, podemos setar nosso código para “passar” por este problema, e teríamos uma exorbitante quantidade de If’s, no caso abaixo, vamos ver o código para que nosso Optional está retornando “empty”.



```

1 package npe_scripts;
2 import java.util.Optional;
3 public class NPE_05_npe_corrige_Optional {
4     String teste;
5     public static void main(String[] args) {
6         NPE_05_npe_corrige_Optional npe = new NPE_05_npe_corrige_Optional();
7         //Nessa caso, estamos usando um artefato do pacote de utilitarios do Java chamado Optional
8         //Nessa caso, estamos informando que a String "teste" pode ser vazia
9         Optional<String> opt = Optional.empty();
10        System.out.println(opt);
11    }
12    //Nesse caso, estamos setando nosso Sistema para que o objeto "nome" possa ser diferente de null
13    // o que é ruim nesse ponto, é se tivéssemos varios objetos, teriamos que tratar varios "ifs" e teriamos uma serie
14    if (npe.teste != null) {
15        System.out.println(npe.teste.toUpperCase()); // nao retorna nada
16    }
17 }
18

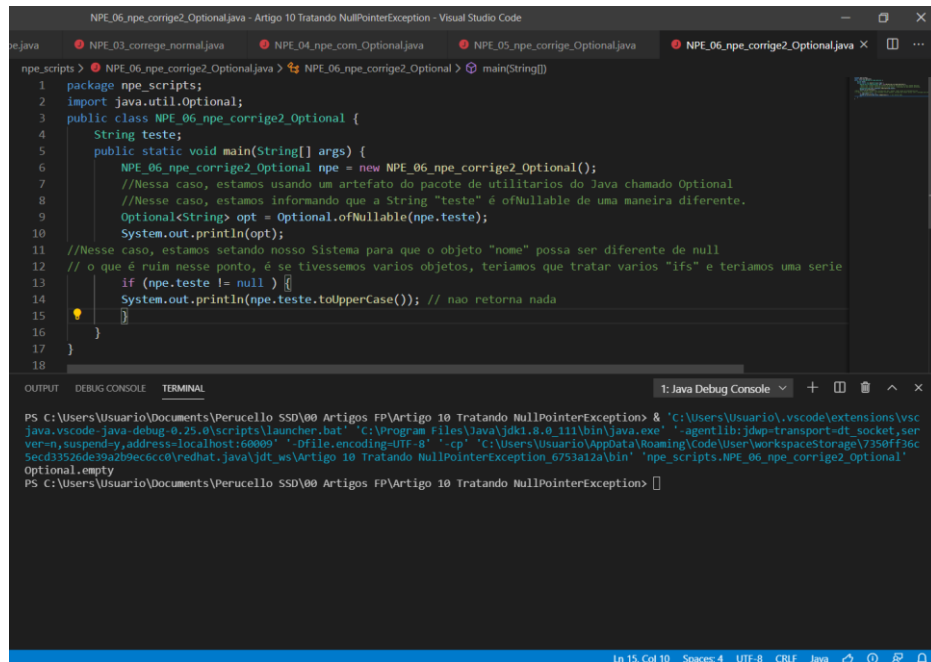
```

```

java.lang.NullPointerException
    at java.util.Objects.requireNonNull(Objects.java:203)
    at java.util.Optional.<init>(Optional.java:96)
    at java.util.Optional.of(Optional.java:108)
    at npe_scripts.NPE_05_npe_corrige_Optional.main(NPE_05_npe_corrige_Optional.java:8)
PS C:\Users\Usuario\Documents\Perucello SSD\00 Artigos FP\Artigo 10 Tratando NullPointerException>

```

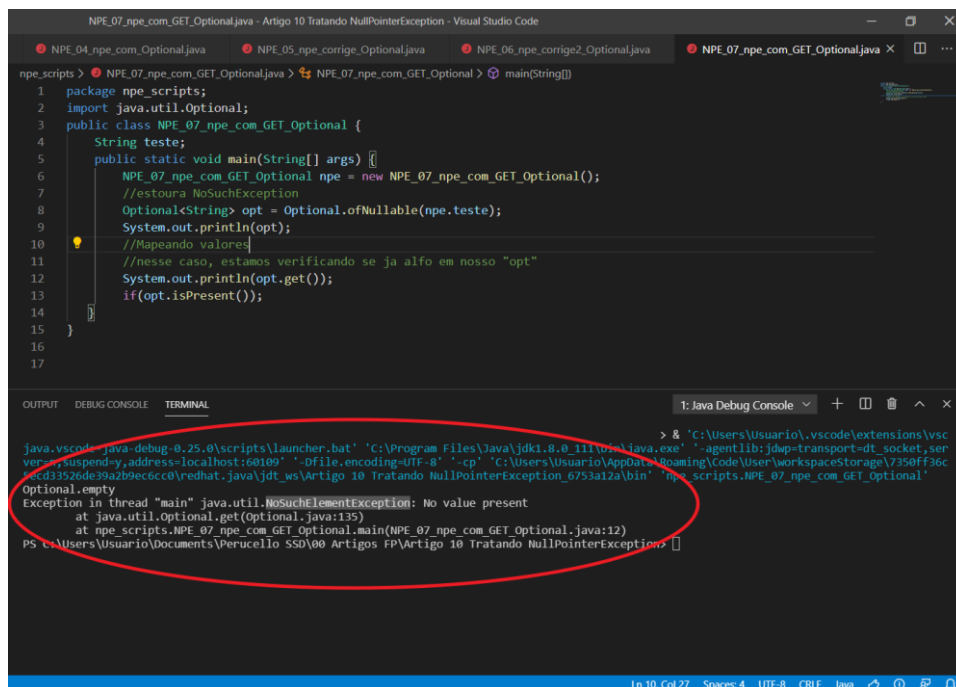
Agora, estamos declarando nosso Optional instanciando-o como “**ofNullable**” , este é um recurso do Java8 e pode ser utilizado a partir desta versão!



```
1 package npe_scripts;
2 import java.util.Optional;
3 public class NPE_06_npe_corrige2_Optional {
4     String teste;
5     public static void main(String[] args) {
6         NPE_06_npe_corrige2_Optional npe = new NPE_06_npe_corrige2_Optional();
7         //Nessa caso, estamos usando um artefato do pacote de utilitarios do Java chamado Optional
8         //Nesse caso, estamos informando que a String "teste" é ofNullable de uma maneira diferente.
9         Optional<String> opt = Optional.ofNullable(npe.teste);
10        System.out.println(opt);
11    }
12    //Nesse caso, estamos setando nosso Sistema para que o objeto "nome" possa ser diferente de null
13    // o que é ruim nesse ponto, é se tivessemos varios objetos, teriamos que tratar varios "ifs" e teriamos uma serie
14    if (npe.teste != null ) {
15        System.out.println(npe.teste.toUpperCase()); // nao retorna nada
16    }
17 }
18
```

```
PS C:\Users\Usuario\Documents\Perucello SSD\00 Artigos FP\Artigo 10 Tratando NullPointerException> & 'C:\Users\Usuario\.vscode\extensions\vsc
java.vscod
java -debug-0.25.0\scripts\launcher.bat' 'C:\Program Files\Java\jdk1.8.0_111\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,ser
ver=,suspend=y,address=localhost:60009' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\Usuario\AppData\Roaming\Code\User\workspaceStorage\7350ff36c
5ecd33526de39a2b9ec6cc0\redhat.java\jdt_ws\Artigo 10 Tratando NullPointerException_6753a12a\bin' 'npe_scripts.NPE_06_npe_corrige2_Optional'
Optional.empty
PS C:\Users\Usuario\Documents\Perucello SSD\00 Artigos FP\Artigo 10 Tratando NullPointerException>
```

Mas ele também é um recurso que devemos tomar cuidado, pois podemos receber o retorno de “**NoSuchElementException**” , porém, esta exceção nos informará que “Não haverá um valor presente” o que já irá ajudar e muito no nosso código !



```
1 package npe_scripts;
2 import java.util.Optional;
3 public class NPE_07_npe_com_GET_Optional {
4     String teste;
5     public static void main(String[] args) {
6         NPE_07_npe_com_GET_Optional npe = new NPE_07_npe_com_GET_Optional();
7         //estoura NoSuchElementException
8         Optional<String> opt = Optional.ofNullable(npe.teste);
9         System.out.println(opt);
10        //Mapeando valores
11        //nesse caso, estamos verificando se ja alfo em nosso "opt"
12        System.out.println(opt.get());
13        if(opt.isPresent());
14    }
15 }
16
17
```

```
java.vscod
java -debug-0.25.0\scripts\launcher.bat' 'C:\Program Files\Java\jdk1.8.0_111\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,ser
ver=,suspend=y,address=localhost:60109' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\Usuario\AppData\Roaming\Code\User\workspaceStorage\7350ff36c
5ecd33526de39a2b9ec6cc0\redhat.java\jdt_ws\Artigo 10 Tratando NullPointerException_6753a12a\bin' 'npe_scripts.NPE_07_npe_com_GET_Optional'
Optional.empty
Exception in thread "main" java.util.NoSuchElementException: No value present
    at java.util.Optional.get(Optional.java:135)
    at npe_scripts.NPE_07_npe_com_GET_Optional.main(NPE_07_npe_com_GET_Optional.java:12)
PS C:\Users\Usuario\Documents\Perucello SSD\00 Artigos FP\Artigo 10 Tratando NullPointerException>
```



Mas não só de **If/Else** e **Optional** pode-se tratar um npe, também podemos usar de outro recurso do Java8 que é **OrElse** !

Nesse caso, faria um pouco mais sentido nosso código, pois receberíamos o “**Optional.empty**” como esperado, e ainda seria possível mapear os valores , onde no nosso código, pedimos para nos retornar a mensagem de “vazio” para que possamos entender melhor.

Vamos ver como ficaria logo mais abaixo:

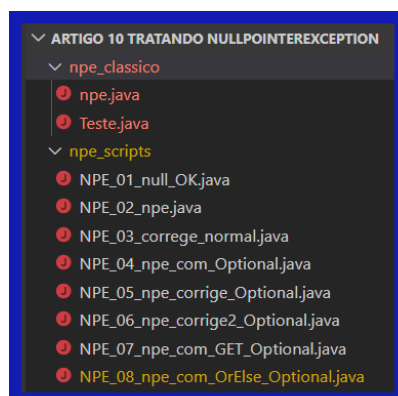
```
1 package npe_scripts;
2 import java.util.Optional;
3 public class NPE_08_npe_com_OrElse_Optional {
4     String teste;
5     public static void main(String[] args) {
6         NPE_08_npe_com_OrElse_Optional npe = new NPE_08_npe_com_OrElse_Optional();
7         Optional<String> opt = Optional.empty();
8         System.out.println(opt);
9         //Mapeando valores
10        System.out.println(opt.orElse("vazio"));
11    }
12 }
13
14
```

OUTPUT DEBUG CONSOLE TERMINAL

1: Java Debug Console

```
java.vscode-java-debug-0.25.0\scripts\launcher.bat 'C:\Program Files\Java\jdk1.8.0_111\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,ser
ver=0.0.0.0,address=localhost:60136' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\Usuario\AppData\Roaming\Code\User\workspaceStorage\7350ff36c
5ecd33526de39a20dec6cc0\redhat.java\jdt_ws\Artigo 10 Tratando NullPointerException_6753a12a\bin' 'npe_scripts.NPE_08_npe_com_OrElse_Optional'
Optional.empty
vazio
PS C:\Users\Usuario\Documents\Perucello SSD\00 Artigos FP\Artigo 10 Tratando NullPointerException>
```

Os códigos estarão disponíveis para que possam baixá-los !



Até mais !

Espero ter ajudado !

