

Evidencias Desafio Java - Vinícius Maffioli

[Repositorio GitHub](#)

Meu objetivo:

Criar o retorno dos cálculos solicitados;

Sobre:

JRE 1.8 (ou compatível com List/ArrayList)

Sem bibliotecas externas ou executáveis

Único arquivo externo envolvido é o .txt para inserir parâmetros alternativos para os cálculos.

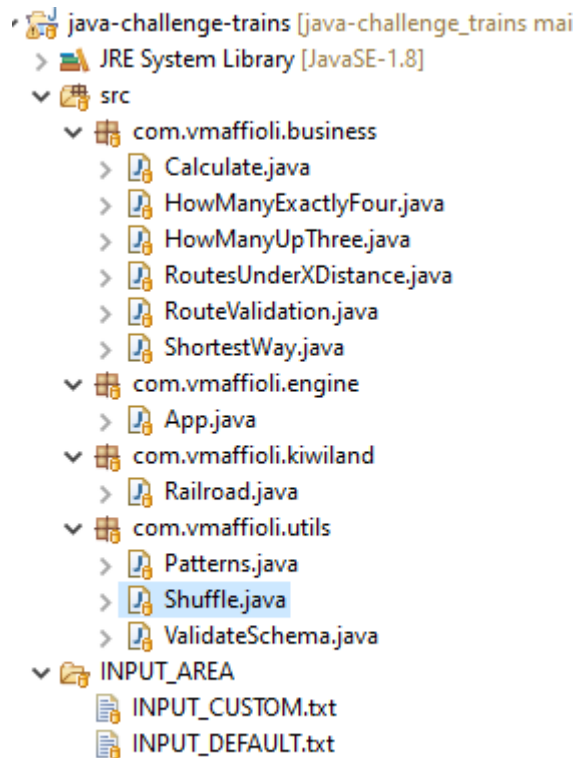
O projeto foi dividido em quatro pacotes:

Business – Onde concentrei os cálculos e a lógica para montagem dos outputs

Engine - Localização do método “main” e chamada dos cálculos. (App.java)

Kiwiland – Contém as classes relacionadas ao ambiente do case, no caso, a montagem e validação do esquema ferroviário algumas constantes.

Util – Contem formatadores e validadores;



Fluxo

Primeiro a aplicação carrega o arquivo “INPUT_CUSTOM.txt” localizado na pasta “INPUT_AREA” na raiz do projeto que pode ou não conter um esquema ferroviário válido (aplicando a mesma regra a demais parâmetros, caso seja necessário testar rotas com distancias e disponibilidades diferentes)

Caso o carregamento falhe, o arquivo “INPUT_DEFAULT.txt” localizado na mesma pasta será carregado (e por precaução, caso o arquivo default não esteja disponível, gravei uma constante com os parâmetros originais do teste em com.vmaffoli.kiwiland.Railroad e submeti o conteúdo carregado a uma segunda validação, localizada em com.vmaffoli.utils.ValidateSchema)

```

3
4
5 public class App {
6
7
8 @SuppressWarnings("resource") // Reason: BufferedReader
9 public static void main (String[] args) throws IOException {
10
11     List<String> results = new ArrayList<String>();
12     FileReader read = null;
13     try {
14         read = new FileReader("INPUT_AREA/INPUT_CUSTOM.txt");
15
16     } catch (Exception e) {
17         System.out.println("!ERROR! INPUT_CUSTOM.txt not found! Loading INPUT_DEFAULT.txt");
18         read = new FileReader("INPUT_AREA/INPUT_DEFAULT.txt");
19     }
20     String line;
21
22     while((line = new BufferedReader(read).readLine()) != null ){
23         Railroad.getBuilder(line, ValidateSchema.getValidateSchema(line));
24         results.add(Calculate.RouteValidation("A-B-C"));
25         results.add(Calculate.RouteValidation("A-D"));
26         results.add(Calculate.RouteValidation("A-D-C"));
27         results.add(Calculate.RouteValidation("A-E-B-C-D"));
28         results.add(Calculate.RouteValidation("A-E-D"));
29         results.add(Calculate.HowManyUpThree("C", "C"));
30         results.add(Calculate.HowManyExactlyFour("A", "C"));
31         results.add(Calculate.ShortestWay("A", "C"));
32         results.add(Calculate.ShortestWay("B", "B"));
33         results.add(Calculate.RoutesUnderXDistance("C", "C", 30));
34
35     }
36     for (int i=0;i<results.size();i++) {
37         System.out.println("Output #"+(i+1)+" : "+results.get(i));
38     }
39 }
40
41 }

```

Encapsulei as chamadas dos cálculos e centralizei suas distribuições através do com.vmafflioli.business.Calculate

```

1 package com.vmafflioli.business;
2
3
4 public class Calculate {
5
6
7     public static String RouteValidation(String route) {
8         return RouteValidation.getRouteValidation(route);
9     }
10
11     public static String HowManyUpThree(String start, String end) {
12         return HowManyUpThree.getHowManyUpThree(start, end);
13     }
14
15     public static String HowManyExactlyFour(String start, String end) {
16         return HowManyExactlyFour.getHowManyExactlyFour(start, end);
17     }
18
19     public static String ShortestWay(String start, String end) {
20         return ShortestWay.getShortestWay(start, end);
21     }
22
23     public static String RoutesUnderXDistance(String start, String end, int distance) {
24         return RoutesUnderXDistance.getRoutesUnderXDistance(start, end, distance);
25     }
26
27
28

```

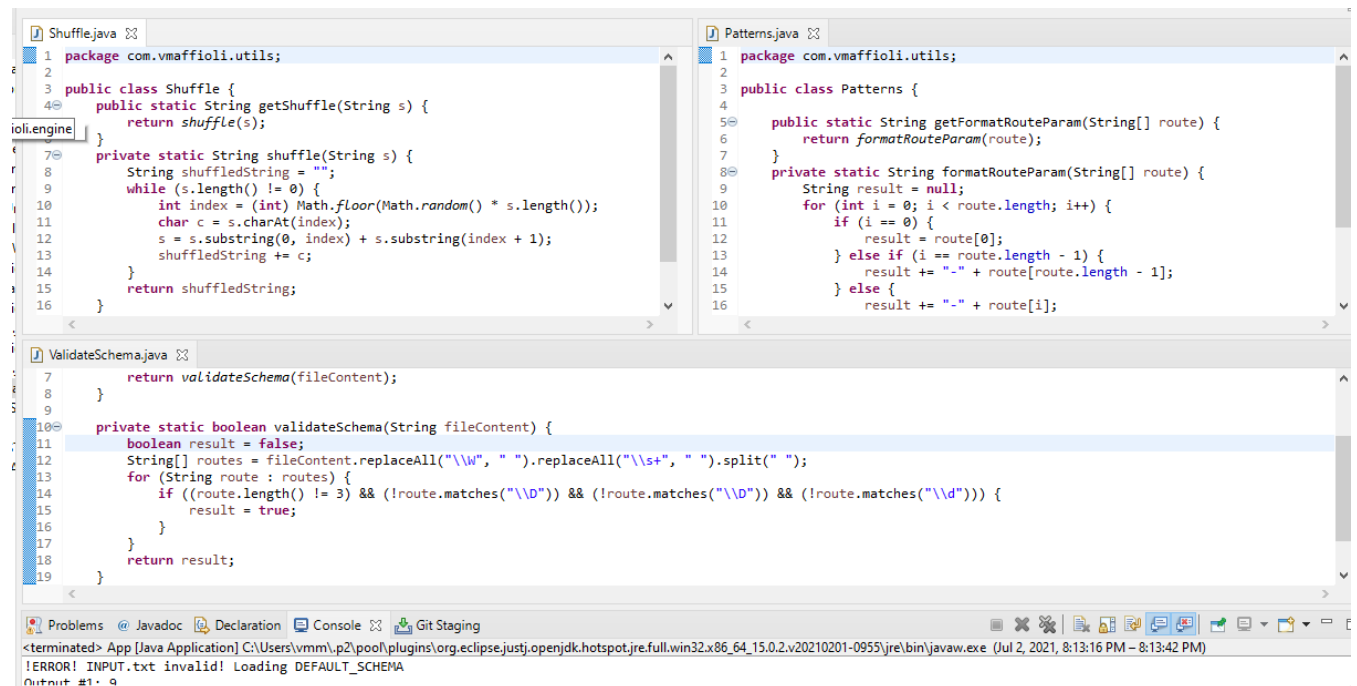
Com.vmafflioli.business.RouteValidation é a validação mais básica que criei e é utilizada pelos outros cálculos existentes em com.vmafflioli.business para validar se existe rota acessível entre os dois parâmetros inseridos (chamo-os de stations) retornando "NO SUCH ROUTE" para rota inacessível ou sua distância total caso seja uma rota válida.

```

1 package com.vmafflioli.business;
2
3 import com.vmafflioli.kiwiland.Railroad;
4
5 public class RouteValidation {
6
7     public static String getRouteValidation(String param) {
8         return routeValidation(param);
9     }
10
11     private static String routeValidation(String param) {
12         String[] route;
13         String result;
14         int[] calcTemp = { 0, 0 };
15         route = param.replaceAll("-", " ").split(" ");
16         for (int i = 0; i < route.length - 1; i++) {
17             for (String routeOnSchema : Railroad.getSchema()) {
18                 try {
19                     String external = route[i] + route[i + 1],
20                         internal = routeOnSchema.split(" ")[0] + routeOnSchema.split(" ")[1];
21                     if (external.equals(internal)) {
22                         calcTemp[0] += Integer.parseInt(routeOnSchema.split(" ")[routeOnSchema.split(" ").length - 1]);
23                         calcTemp[1]++;
24                     }
25                 } catch (Exception e) {
26                     System.out.println("!ERROR! com.vmafflioli.business.Calculate.RouteDistance -> " + e);
27                 }
28             }
29         }
30         if (calcTemp[1] == route.length - 1) {
31             result = String.valueOf(calcTemp[0]);
32         } else {
33             result = Railroad.getROUTE404MESSAGE();
34         }
35         return result;
36     }
37 }

```

E criei essas funções de apoio (em ordem de exibição) para embaralhar as listas de caracteres usadas para gerar Anagramas (simulando rotas para serem avaliadas), outra para aplicar padrões para leitura de strings efetuadas pelo com.vmafflioli.business e a ultima para validar o schema ferroviário recebido do usuario ou proprio sistema. (Acabei não adicionando um validador para delimitar as letras utilizadas (ABCDE).



```
1 package com.vmafflioli.utils;
2
3 public class Shuffle {
4     public static String getShuffle(String s) {
5         return shuffle(s);
6     }
7     private static String shuffle(String s) {
8         String shuffledString = "";
9         while (s.length() != 0) {
10             int index = (int) Math.floor(Math.random() * s.length());
11             char c = s.charAt(index);
12             s = s.substring(0, index) + s.substring(index + 1);
13             shuffledString += c;
14         }
15         return shuffledString;
16     }
17 }

1 package com.vmafflioli.utils;
2
3 public class Patterns {
4
5     public static String getFormatRouteParam(String[] route) {
6         return formatRouteParam(route);
7     }
8     private static String formatRouteParam(String[] route) {
9         String result = null;
10         for (int i = 0; i < route.length; i++) {
11             if (i == 0) {
12                 result = route[0];
13             } else if (i == route.length - 1) {
14                 result += "-" + route[route.length - 1];
15             } else {
16                 result += "-" + route[i];
17             }
18         }
19     }
20 }

7     return validateSchema(fileContent);
8 }
9
10 private static boolean validateSchema(String fileContent) {
11     boolean result = false;
12     String[] routes = fileContent.replaceAll("\\W", " ").replaceAll("\\s+", " ").split(" ");
13     for (String route : routes) {
14         if ((route.length() != 3) && (!route.matches("\\D")) && (!route.matches("\\d"))) {
15             result = true;
16         }
17     }
18     return result;
19 }
```

Problems | Javadoc | Declaration | Console | Git Staging

<terminated> App [Java Application] C:\Users\vmml\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0955\jre\bin\javaw.exe (Jul 2, 2021, 8:13:16 PM - 8:13:42 PM)

!ERROR! INPUT.txt invalid! Loading DEFAULT_SCHEMA

Output #1: 9

Saídas:

1) AB5, BC4, CD8, DC8, DE6, AD5, CE2, EB3, AE7

(entrada padrão proposta pelo desafio)

```
<terminated> App [Java Application] C:\Users\mmm\p2\pool\plugins\org.eclipse.just
Loading DEFAULT_SCHEMA
Output #1: 9
Output #2: 5
Output #3: 13
Output #4: 22
Output #5: NO SUCH ROUTE
Output #6: 2
Output #7: 3
Output #8: 9
Output #9: 9
Output #10: 7
```

2) AC2, BA8, CE5, BC8, DA3, AD2, CD1, EB9, AB1

(entrada de dados customizada, seguindo o padrão de dados aplicando a mesma regra de cálculo)

```
Loading CUSTOM_SCHEMA
Output #1: 9
Output #2: 2
Output #3: NO SUCH ROUTE
Output #4: NO SUCH ROUTE
Output #5: NO SUCH ROUTE
Output #6: 2
Output #7: 4
Output #8: 7
Output #9: 9
Output #10: 34
```

3) AAA, 2AS. @#\$@, asdasd, 1234qwd, aa2, bc3, d4

(ao tentar executar parâmetros inválidos no "INPUT_CUSTOM.txt", a aplicação ignora o arquivo incompatível e carrega o "INPUT_DEFAULT.txt");

```
Loading DEFAULT_SCHEMA
Output #1: 9
Output #2: 5
Output #3: 13
Output #4: 22
Output #5: NO SUCH ROUTE
Output #6: 2
Output #7: 3
Output #8: 9
Output #9: 9
Output #10: 7
```

E por fim, se o arquivo customizado não existe, ele também identifica e carrega o default, essa regra também se aplica ao default se estiver invalido, carregando então a constante localizada em `com.vmafflioli.kiwiland.Railroad (String[] schema);`

