本次我选择的是题目一：Trains

题意

简单说明题意，如下：
给定一个有向图，然后回答下面一些问题：

- 计算指定路线的总距离
- 计算指定最大停顿次数的方案数量
- 计算指定精确停顿次数的方案数量
- 计算起点与终点不同时的最短路径
- 计算起点与终点相同时的最短路径（注意，不能直接输出 0 而是需要至少走一圈）
- 计算小于最大距离的方案数量（样例中是起点与终点相同，但其实也可以不同）

思路

整体解决思路如下：

- 使用矩阵来存储图信息，不存在的边用-1 表示
- 对于样例 1-5，直接求加法即可；需要注意的是当某相邻两个点的距离不存在即等于-1 时，最终结果就是”NO SUCH ROUTE”
- 对于样例 6 和 7，都使用了 BFS 搜索算法，唯一区别是判断是否合法的条件不同
- 对于样例 8，直接使用了图论中的 Dijastra 算法来解决
- 对于问题 9，则是通过变相和加强版本的 Dijastra 算法来解决的
- 对于样例 10，还是使用 BFS 搜索解决的

代码实现细节

- 程序会一次性读取 txt 文件作为输入
- 将 txt 文件的第一行作为图信息进行存储
  - 具体实现时，是将 A、B、C、D、E 这些大写字母表示的图顶点，转换为 0，1，2，3，4 表示，并通过矩阵的方式进行存储
  - 具体可参见 com.cyh.data.structure.Graph
- 其余的每一行作为一个请求，分别进行计算
- 计算之前，采用了工厂模式来调用不同的计算策略
  - 具体可参见 com.cyh.factory.CalculatorFactory#calculate
- 在分析具体需要使用哪种计算策略时，使用了正则表达式来匹配，并通过分组直接得到输入参数，避免了对字符串的解析
  - 具体实现时，采用枚举来保存正则表达式的编译结果，同时保存了对应

计算策略的实现类的实例，从而简单实现了类似 SpringIOC 的单实例容器

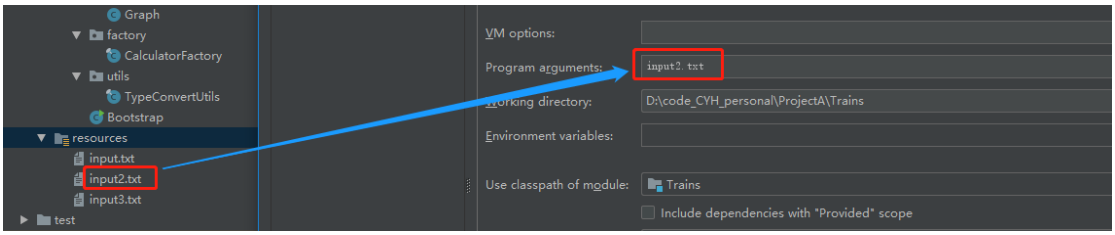
- 具体可参见 com.cyh.factory.CalculatorFactory.CalculatorTypeEnum

➢ 然后调用不同的计算策略分别进行计算

- 具体可参见 com.cyh.calculate 包中的实现类
- 实现思路上面已有描述，此处不再赘述

代码运行

直接运行 com.cyh.Bootstrap 主类即可。
此启动类默认会读取 resources 目录下的 input.txt 文件作为程序输入，如果需要运行其它文件，将文件置于此目录下，并将文件名设置为程序参数即可。像下面这样：



运行效果

```
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

Process finished with exit code 0
```