Решение реализовано на языке Java (локально тестировалось на Open JDK 1.8.0\_252) и состоит из следующих классов:

Main – точка входа  
DataParser – класс, осуществляющий преобразование входящих строковых данных в целочисленные массивы  
DataProcessor – реализация алгоритма сортировки и подготовка набора выходных данных  
InputReader – чтение данных из входного файла  
InputValidator – валидация входных данных  
OutputWriter – подготовка выходных данных и запись в файл  
Stack – реализация стэка для хранения данных о вагонах на станции  
ValidationResult – тип данных для хранения результатов валидации

В соответствии с указанием в гайдлайне тестового задания оптимизация кода не проводилась, поэтому в коде встречаются «дорогие» операции (строковые объекты в цикле) и повторы.

В директории testData содержатся специально подготовленные входные данные для тест-кейсов, описанных ниже.

Некоторое количество комментариев содержится в коде. В этом документе – расширенные пояснения.

**Запуск программы**

**Алгоритм сортировки вагонов**

Будем рассматривать всю процедуру как набор шагов. Один минимальный шаг – это перемещение одного вагона с пути A на станцию либо со станции на путь B. Шаги продолжаются до тех пор, пока на пути A остаются вагоны или ожидаемый вагон доступен для перемещения со станции на путь B (если недоступен, перемещаем следующий вагон с пути A на станцию). Если ни то, ни другое невозможно на текущем шаге, то для заданной последовательности вагонов ответ «No». Если число вагонов на пути B достигло N, - ответ «Yes».

По условию задачи не требуется сохранять последовательность действий, приведших к успеху, а вагоны поезда на пути A отсортированы по возрастанию. Таким образом, в каждый момент необходимо знать только номер вагона, ожидаемого на пути B, и номер вагона на станции, ближайшего к пути B.

Для хранения данных о вагонах на станции используется тип данных стэк, реализованный в отдельном классе

**Валидатор**

Валидация осуществляется с помощью отдельного класса InputValidator с методом validate(), проверяющим соответствие входных данных ожиданиям по ряду критериев. В случае несоответствия возвращается объект ValidationResult с флагом isValid==false и содержательным сообщением об ошибке.

Валидация происходит последовательно и таким образом, что после первой встреченной ошибки дальнейшие проверки не проводятся, а в результирующем объекте возвращается единственное сообщение об ошибке.

Перечень проверок валидатора:

* длина последовательности не превышает 1000
* все значения являются положительными целыми числами, пустые строки отсутствуют
* в последних двух строчках файла с данными содержится «0»
* более нигде в данных не содержатся две последовательные строчки с «0»
* При вырожденном частном случае, когда длина поезда равна 1, ожидаемая последовательность тоже состоит из одного числа 1
* длина ожидаемой последовательности вагонов равна переданной в заголовке блока
* в конце блока данных «0»
* любое число ожидаемой последовательности не превышает значения длины последовательности (следствие того, что вагоны входящего поезда пронумерованы натуральным рядом чисел от 1 до N)
* в ожидаемой последовательности нет повторяющихся чисел

Валидатор можно развивать. Разумными видятся следующие улучшения:

* сохранять все сообщения об ошибках с локализацией их в наборе пользовательских данных
* оптимизировать код

Для тестирования приложения выбран подход «черный ящик». Т.е. заключение о корректности работы приложения делается только на основании анализа отклика на входные данные. Поэтому некоторые проверки могут являться повторяющимися и избыточными с точки зрения «белого ящика».

**Тест-план**

1. Объект тестирования

Тестируется программа оценки возможности сортировки вагонов поезда

1. Предмет тестирования

Тестируется корректность вычислений (алгоритма), обработки входных данных и формирования выходных данных, реакции на некорректные входные данные

1. Виды и способы тестирования

Проводится функциональное тестирование путем передачи на вход программе данных с заранее известным ожидаемым результатом

1. Уровни тестирования

Приемочное тестирование на ограниченном наборе данных

1. Порядок тестирования

Тестирование осуществляется после окончания разработки приложения путем его многократного запуска с различным набором тестовых данных

1. Анализ результатов тестирования и критерии оценки

Критерий успешного прохождения тестирования – соответствие фактических результатов запуска приложения ожидаемым. В том числе получение ожидаемых сообщений об ошибках при передаче на вход некорректных данных.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип проверки | Вид сценария (позитивный/ негативный) | Входные данные | Тест-кейс | Ожидаемый результат | Имя файла с вх. данными |
| Тип данных | + | Целые положительные значения | Перестановка в указанной последовательности возможна | “Yes” в выходом файле | 01positiveInt |
| - | Перестановка в указанной последовательности невозможна | “No” в выходном файле | 02positiveInt |
| - | Ноль | Длина поезда (N) равна 0, в ожидаемой последовательности – целые положительные значения | Exception «Train Marshalling Error: a train length doesn't look like a valid integer number» | 03zero |
| - | Длина поезда – целое положительное значение, в ожидаемой последовательности встречается 0 | Exception «Train Marshalling Error: a coach number equals zero» | 04zero |
| - | Целые отрицательные значения | Длина поезда – отрицательное значение | Exception «Train Marshalling Error: an unexpected negative value occurred» | 05negativeInt |
| - | В ожидаемой последовательности встречается отрицательное значение | Exception «Train Marshalling Error: an unexpected negative value occurred» | 06negativeInt |
| - | Дробные положительные значения | Длина поезда – дробное число | Exception «Train Marshalling Error: unable to parse an integer number» | 07positiveFrac |
| - | В ожидаемой последовательности дробное число | Exception «Train Marshalling Error: unable to parse an integer number» | 08positiveFrac |
| - | Дробные отрицательные значения | Длина поезда – дробное отрицательное число | Exception «Train Marshalling Error: unable to parse an integer number» | 09negativeFrac |
| - | В ожидаемой последовательности дробное отрицательное число | Exception «Train Marshalling Error: unable to parse an integer number» | 10negativeFrac |
| - | Непустая строка с текстовым символом | В длине поезда присутствует текстовый символ | Exception «Train Marshalling Error: unable to parse an integer number» | 11textChar |
| - | В ожидаемой последовательности присутствует текстовый символ | Exception «Train Marshalling Error: unable to parse an integer number» | 12textChar |
| - | Непустая строка с нетекстовым специальным символом | В длине поезда присутствует «\» | Exception «Train Marshalling Error: unable to parse an integer number» | 13specialChar |
| - | В ожидаемой последовательности присутствует «\» | Exception «Train Marshalling Error: unable to parse an integer number» | 14specialChar |
| - | Пустая строка | Вместо длины поезда пустая строка | Exception «Train Marshalling Error: unexpected empty string» | 15emptyString |
| - | Вместо ожидаемой последовательности пустая строка | Exception «Train Marshalling Error: unexpected empty string» | 16emptyString |
| Структура данных | - | Корректные данные за исключением маркера конца файла | Отсутствует две строчки с «0» в конце файла | Exception «Train Marshalling Error: Invalid input file structure: concluding '0' expected» | 17eofMark |
| - | Между двумя блоками данных дополнительная строчка с «0» | Exception «Train Marshalling Error: End of file marker occurred while the end of the data set hasn't been reached» | 18extraZero |
| - | Корректные данные за исключением маркера конца блока | Два блока данных идут слитно, без «0» между ними | Exception «Train Marshalling Error: The length of the expected sequence is inconsistent with the passed amount» (поскольку строчка с числом вагонов из следующего блока будет интерпретирована как принадлежащая к текущему блоку) | 19eobMark |
| - | Корректные данные за исключением заголовка одного из блоков (числа вагонов) | Отсутствует число вагонов в одном из блоков | Exception «Train Marshalling Error: a train length doesn't look like a valid integer number» | 20amountMissing |
| + | Двойные пробелы и символы табуляции | Во всех строчках входных данных встречаются дополнительные пробелы и символы табуляции | Поскольку в требованиях нет указания на разделитель данных, то такие данные должны обрабатываться.  В выходном файле «Yes», если ожидаемая последовательность возможна, и «No» - если нет | 21extraWhitespace |
| Консистентность данных | - | В блоке данных отсутствует ожидаемая последовательность | Во входных данных один из блоков пуст, т.е. после числа вагонов идет маркер окончания блока | Exception «Train Marshalling Error: Invalid sequence length: the train is supposed to have at least one coach» | 22emptyBlock |
| - | Число N, указанное в начале блока не равно числу вагонов в последовательности | Указанное в начале блока число вагонов больше количества чисел в ожидаемой последовательности | Exception «Train Marshalling Error: The length of the expected sequence is inconsistent with the passed amount» | 23inconsistentAmount |
| - | Указанное в начале блока число вагонов меньше количества чисел в ожидаемой последовательности | Exception «Train Marshalling Error: The length of the expected sequence is inconsistent with the passed amount» | 24inconsistentAmount |
| - | Каждое число в последовательности не превышает ее длину | В ожидаемой последовательности есть число, большее, чем число вагонов | Exception «Train Marshalling Error: a coach number occurred that exceeds its maximum possible number» | 25largerNum |
| - | Последовательность не содержит повторяющихся чисел | В ожидаемой последовательности есть дубликаты | Exception «Train Marshalling Error: duplicate value occurred» | 26duplicates |
|  | - | Единица | Один блок с одним числом (вырожденная последовательность), отличным от единицы | Exception «Train Marshalling Error: Invalid number: expected 1 but found <n>» | 27one |
| Тестирование алгоритма. Пограничные случаи | + | Единица | Один блок с единицей (вырожденная последовательность) | «Yes» | 28algoTestOne |
|  | + | Корректные данные. Приближение к максимальному значению снизу | Длина последовательности 999 | В выходном файле «Yes», если ожидаемая последовательность возможна, и «No» - если нет | 29algoTestPremax |
|  | + | Длина последовательности 1000 | В выходном файле «Yes», если ожидаемая последовательность возможна, и «No» - если нет | 30algoTestMax |
|  | - | Превышено максимальное (согласно требованиям) число вагонов | Длина последовательности 1001 | Exception «Train Marshalling Error: maximum train length exceeded» | 31algoTestExceeded |
|  | + | Корректные данные | Две последовательности из 5 вагонов – возможная и невозможная | «Yes» «No» | 32algoTest |
| Прочее | - | Неверное имя входного файла в командной строке |  | Выбрасывается исключение FileNotFoundException |  |