Решение реализовано на языке Java (Open JDK 1.8.0\_41) 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.

**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

**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.

Выполняется следующий **набор проверок**

Тип данных

* целое положительное
* целое отрицательное
* дробное положительное
* непустая строка с текстовым символом
* непустая строка с произвольным нетекстовым символом
* пустая строка

Структура данных

* есть маркер конца файла
* маркер конца файла в середине данных
* есть маркера конца блока
* есть заголовок блока (число вагонов)
* двойной пробел и табуляция

Консистентность данных

* блок содержит хотя бы одну последовательность
* длина последовательности соответствует заявленной
* каждое число в последовательности не превышает ее длину
* последовательность не содержит повторяющихся чисел

Пограничные случаи

* один блок с одним числом, отличным от единицы
* один блок с единицей
* длина последовательности 999
* длина последовательности 1000 (позитивный и негативный сценарии)
* длина последовательности превышает 1000

Прочее

* файл с входными данными существует

Тестирование алгоритма (сравнение результата с ожидаемым)

* подача корректных данных приводит к ожидаемому результату