TABLE OF CONTENTS

TEST & BUILD THE APPLICATION

```
$ sbt test assembly doc
```

ScalaDoc generated documentation will be available in the following file:

```
./target/scala-2.11/api/index.html
```

This application requires SBT and Scala 2.11

EXECUTE THE JAR IN THE COMMAND LINE

```
java -jar java -jar target/scala-2.11/PaintShop-assembly-1.0.jar
    {<input file with test cases>}
    [-o <output file with batches solutions>]
```

If `-o` (or `--output-file`) isn't provided, the output will be printed to the console.

Examples:

```
java -jar target/scala-2.11/PaintShop-assembly-1.0.jar
./src/test/resources/inputs/success_from_specification.txt

java -jar target/scala-2.11/PaintShop-assembly-1.0.jar
./src/test/resources/inputs/success_from_specification.txt
-o output.txt

java -jar target/scala-2.11/PaintShop-assembly-1.0.jar
./src/test/resources/inputs/performance/small_dataset.txt
-o small_dataset_output.txt

java -jar target/scala-2.11/PaintShop-assembly-1.0.jar
./src/test/resources/inputs/performance/large_dataset.txt
-o large_dataset_output.txt
```

For help execute the following command:

```
java -jar target/scala-2.11/PaintShop-assembly-1.0.jar --help
```

Which will print the following:

```
Usage: com.zalando.paintshop.app.PaintShop$ [options] <input file>

  <input file>
        Name of the input file containing test cases.
  -o <file> | --output-file <file>
        Name of the output file. If not provided will print output
        to console.
  --help
        prints this help text
```

ON THE DESIGN CHOICES

This project mirrors in basic lines the Java one available in the package regarding architecture and algorithm. Please refer to the Java project for details on the design, as well as algorithm analysis.

Aside from obvious differences between Java's and Scala's syntax, the most appealing is that Scala supports mixin inheritance with traits:

```
object PaintShop
    extends PlainTextInputParser
    with TestCaseProcessor
    with SimpleOutputFormatter {
   ...
}
```

`PlainTextInputParser`, `TestCaseProcessor` and `SimpleOutputFormatter` are the equivalent components to the ones available in the Java version, but using traits is nicer than composition (and much nicer than classical inheritance, which should be avoided).

There is a significant reduction in the number of lines in the Scala code when compared to java. In general, the scala classes and methods are much smaller.

I tried to be functional as much as I could. I'm only using vars for the `PlanTextFileInputIterator` (line counter), `PlainTextInputParser` (Arrays) and `TestCaseProcessor` (muttable BitSet).

I'm using mutable Arrays and BitSet, as they are faster due to the somehow large number of iterations over them.

I hope to achieve 100% functional code in the future (years of imperative programming are hard to shake off!).

For unit tests, I'm using ScalaTest. For parsing CLI input arguments I'm using Scopt.

PERFORMANCE

There is a Python script in the Java project, which has been used to generate the "large data set" and "small data set". Please refer to the Java project for details.

Running on my computer, a Lenovo Yoga 2 laptop running Ubuntu 12.04, I got the following results:

Large data set:

```
10:33:08 {master} ~/workspace/IdeaProjects/Zalando/Scala/PaintShop$
java -jar target/scala-2.11/PaintShop-assembly-1.0.jar
./src/test/resources/inputs/performance/large_dataset.txt
-o large_dataset_output.txt

Total processing time: 39 ms
```

Small data set:

```
10:32:19 {master} ~/workspace/IdeaProjects/Zalando/Scala/PaintShop$
java -jar target/scala-2.11/PaintShop-assembly-1.0.jar
./src/test/resources/inputs/performance/small_dataset.txt
-o small_dataset_output.txt

Total processing time: 37 ms
```

HOW TO VISUALIZE THIS DOCUMENT

This document is better visualized using IntelliJ's Markdown Plugin. In case it isn't available, there is a PDF version of this document in the same directory.

For my own reference, to convert markdown to PDF use the following command:

```
pandoc README.md -f markdown -t latex -s -o README.pdf
```