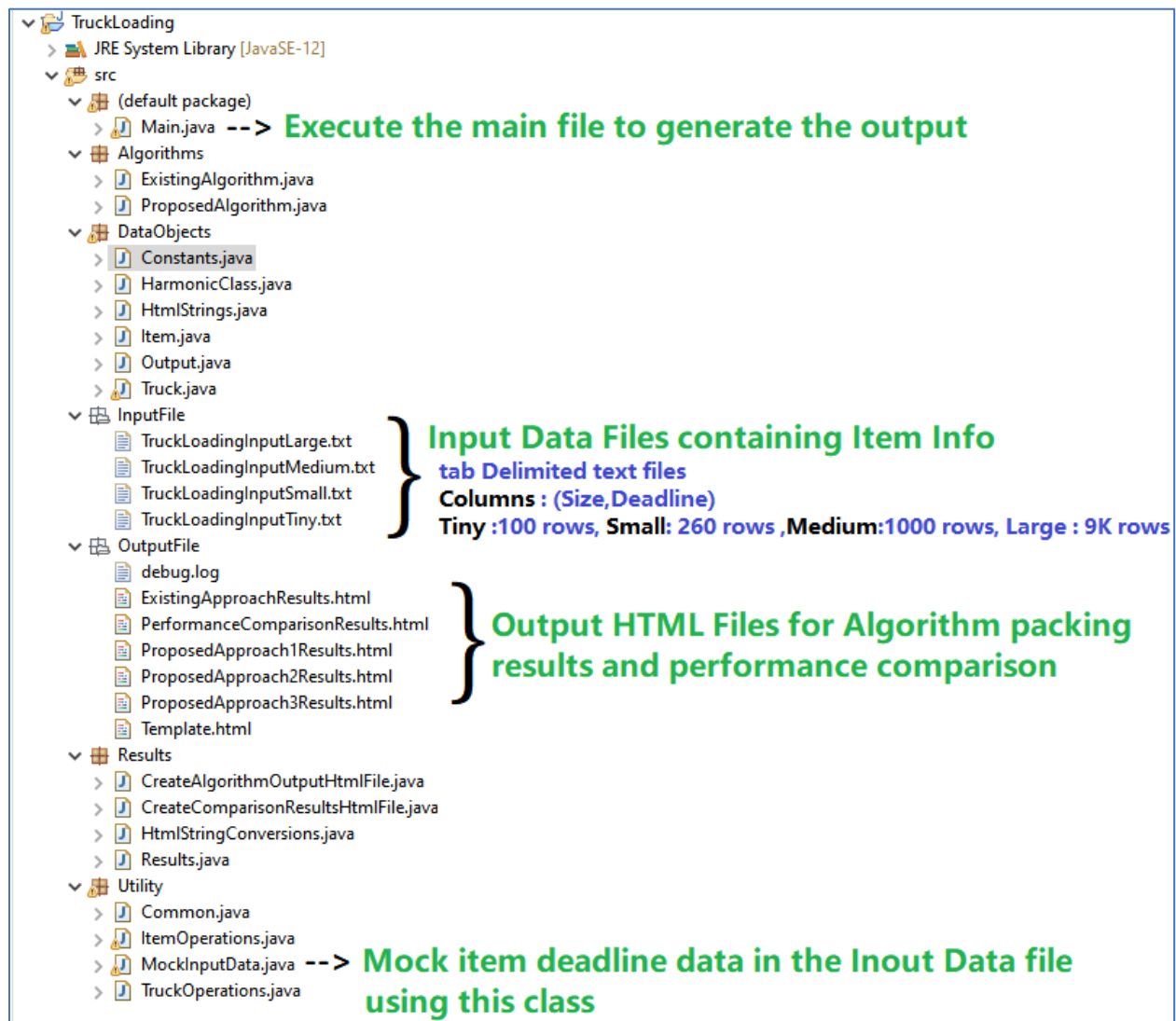


Read Me File for Truck Loading java Application

Steps and Output Execution

- 1) Software Requirements for running Truck Loading Java application
 - a. Java JDK min version 12.0 (<https://www.oracle.com/java/technologies/javase-downloads.html>)
 - b. Eclipse IDE for Enterprise Java Developers (<https://www.eclipse.org/downloads/packages/release/2018-12/r/eclipse-ide-enterprise-java-developers>)
- 2) Download project from below GitHub Link:
<https://github.com/JasmineRandhawa/TruckLoading>
- 3) Open the project in the Eclipse IDE
- 4) Run the Project executing the (src/Main.java) file
- 5) All Output HTML files are generated in folder (/src/OutputFile)

Project Structure



The screenshot shows the project structure of 'TruckLoading' in Eclipse. The project is located under 'JRE System Library [JavaSE-12]'. The 'src' folder contains the following structure:

- (default package)
 - Main.java --> **Execute the main file to generate the output**
- Algorithms
 - ExistingAlgorithm.java
 - ProposedAlgorithm.java
- DataObjects
 - Constants.java
 - HarmonicClass.java
 - HtmlStrings.java
 - Item.java
 - Output.java
 - Truck.java
- InputFile
 - TruckLoadingInputLarge.txt
 - TruckLoadingInputMedium.txt
 - TruckLoadingInputSmall.txt
 - TruckLoadingInputTiny.txt

Input Data Files containing Item Info
tab Delimited text files
Columns : (Size,Deadline)
Tiny :100 rows, Small: 260 rows ,Medium:1000 rows, Large : 9K rows
- OutputFile
 - debug.log
 - ExistingApproachResults.html
 - PerformanceComparisonResults.html
 - ProposedApproach1Results.html
 - ProposedApproach2Results.html
 - ProposedApproach3Results.html
 - Template.html

Output HTML Files for Algorithm packing results and performance comparison
- Results
 - CreateAlgorithmOutputHtmlFile.java
 - CreateComparisonResultsHtmlFile.java
 - HtmlStringConversions.java
 - Results.java
- Utility
 - Common.java
 - ItemOperations.java
 - MockInputData.java --> **Mock item deadline data in the Inout Data file using this class**
 - TruckOperations.java

- 1) Assumption parameters can be modified by changing value in below file in Constants.java file:

```
// Assumptions
public static final int OpenTrucksThreshold = 2;
public static final double TruckCapacity = 1.0;
```

- 2) Before running the project ,user needs to select the input files to be processed and set the input file path to “dataFilePath” variable in “/src/Main.java” file as below :

```
// input data file path
String dataFilePath = Constants.TinyInputDataFilePath;
```

- 3) Input Files in Project folder (/src/InputFile). The paths to these files are configure in “/src/DataObjects/Constants.java” file as below

```
// Input File paths
public static final String TinyInputDataFilePath = "src/InputFile/TruckLoadingInputTiny.txt";
public static final String SmallInputDataFilePath = "src/InputFile/TruckLoadingInputSmall.txt";
public static final String MediumInputDataFilePath = "src/InputFile/TruckLoadingInputMedium.txt";
public static final String LargeInputDataFilePath = "src/InputFile/TruckLoadingInputLarge.txt";
```

- 4) To mock Item deadline data , uncomment below line in the “/src/Main.java”. The below mock function automatically randomizes the deadline data in the input file. Specify the path of input data file to be mocked in parameter of this function.

```
public static void main(String[] args) throws IOException
{
    try
    {
        /* mock deadline data by copying item size from source file, mock random deadline data and
           and writing (size, deadline) back to source file */
        //MockInputData.MockInputItemDeadlineData(Constants.TinyInputDataFilePath);
    }
}
```