Lonsec Coding Test -- Terry Flander
Installation, Configuration and Execution

The application **OutperformanceReport** is designed to take as its input fund and benchmark return series for selected funds over a period of months and produce a report comparing the actual return of each against a selected benchmark. From this comparison a performance excess is calculated. Additionally, funds are ranked by their actual returns on a monthly basis.

Source information from the input files is persisted in **FundReturnSeries** which serves as the source for the output report. Additional calculations could be added to this class which would make them available to **OutperformanceReport** or other similar reports.

The data source has also been abstracted into **FundDataSource** which in addition to providing access to the external CSV files, provides an alternative source of internal data. This class could be extended to source data from external resources such as data bases, application APIs , or Soap/REST requests.

Installation

The Coding Test application is stored in the GitHub internet source repository under the account of Terry Flander. This is a public repository so can be shared with others if required.

1) Go to the GitHub site using https://github.com/terry-flander/LonsecTest

2) From the **Clone or download** button, choose **Download ZIP**

3) Move the downloaded file **LonsecTest-master.zip** to a working directory and unpack.

   This will create a directory structure containing the Java source for the application, tests, configuration file, and default directories and files used by the application when run.

Compiling and initial testing

Compiling, testing and running the application is supported by a simple Maven POM. In a development environment this would be modified to take advantage of Maven's goals to separate the various activities required in application lifecycle.

4) From an OS shell **cd** into the root directory **<install>/LonsecTest-master/Lonsec**

5) Compile and initial test with the command **mvn clean package**

The application will compile and the tests will run automatically. Success is indicated by tests running without error.

```
...
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ my-app ---
[INFO] Building jar: ./LonsecTest-master/Lonsec/target/my-app-1.0-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 10.017 s
```

```
[INFO] Finished at: 2017-01-12T08:47:29+11:00
[INFO] Final Memory: 17M/151M
[INFO] ------------------------------------------------------------------------
```

The final test is an integration test of the full report using test data extracted from the document. "JavaCodingTest_V2.2.pdf". This data is built into the **FundDataSource** class and is used by unit tests as well. No external data is required for these tests to run.

Configuration

The application configuration is completely external to the application and is contained in the file **fund.properties** found in **src/main/resources**. This configuration file is used by the **FundProperties** class and allows configuration of the properties listed below. These properties and their use are defined in the file itself for reference while making changes.

Input file controls

`inputDirectory` – the directory under which all input files will be located. This directory name will be prepended to all the file names values below.
`hasHeader` – if set to true then first line of input files contains a header row which will be ignored. Otherwise will process all input lines.
`fundFileName` – the file name of the CSV file containing the Fund data. Logical name of this file is **fund**.
`benchmarkFileName` – the file name of the CSV file containing the Benchmark Data. Logical name of this file is **benchmark**.
`fundReturnSeriesFileName` – the file name of the CSV file containing the Fund Return Series data. Logical name of the file is **fundReturnSeries**.
`benchmarkReturnSeriesFileName` – the file name of the CSV file containing the Benchmark Return Series data. Logical name of this file is **benchmarkReturnSeries**.

Output file controls

`outputDirectory` – the directory into which the output file will be created. This directory name will be prepended to the file name value below.
`outputFileName` – the file which will contain the output from running this project. Logical name of this file is **output**.

Excess Lookup

`excessLookup` – a specially formatted string defining the values of the Excess Text lookup class used for formatting output. The definition is as follows:

- Semicolon separated (;) list of comma separated (,) limit/description pairs
- Must be in sequential order from low to high
- Limit value of 0 is turning point for significance of limit and must be included

  i.e. before 0, the limit is taken to mean less than (<) the limit value and after 0, the limit is taken to mean greater than (>) the limit value. Therefore the limits

are exclusive of their value. A limit of -1 means < -1 and a limit of 1 means > 1. The limits are assumed to be double values and so may contain decimal values.

NOTE

After the Installation, the copy of **fund.properties** in the directory **target/classes** controls the application. If step 5) above is repeated, this file will be overwritten with the version in **src/main/resources**. Make changes to both files to prevent the loss of settings.

The same caution applies to the contents of **target/classes/TestData** which is the default location for both input and output CSV files used by the application. To prevent this from being a problem during testing, copy the directory and its contest to another location and modify both **fund.properties** to refer to the new location.

Testing

The suite of test programs are designed using JUnit. The entire suite of unit tests as well as the final integration test are run automatically when the application is installed. The simplest way to re-run the tests is to use the Maven command **mvn package**. By omitting the **clean** option any changes made to **fund.properties** or the contents of **TestDatat** in the **target** directories will remain unchanged.

Running

The Main class is **OutperformanceReport** which is executed when invoked with the following command.

mvn exec:java

This will produce the desired output in the target directory using the provided test files. All test files and the output file will be located in **target/classes/TestData.**

At any time application can be run with the **test** argument cause an integration report to be generated using the internal data source with output to the console. To run the application to product this output type the following command:

mvn exec:java -Dexec:args="test"

The output should be as follows:

```
Fund Code,Fund Name,Date,Excess,Out Performance,Return,Rank
fund1,fund 1,30/11/2016,-0.01,,-1.47,1
fund1,fund 1,31/10/2016,-0.02,,-1.32,1
fund1,fund 1,30/09/2016,-0.01,,-0.23,1
fund1,fund 1,31/08/2016,0.00,,0.42,1
fund1,fund 1,31/07/2016,-0.02,,0.71,1
fund1,fund 1,30/06/2016,-0.20,,1.11,1
```