# 1. Information for Use

## 1.1. Introduction

CryptoGraph is a program designed and implemented to facilitate the analysis of crypto-currency trading data.

## 1.2. Installation

CryptoGraph requires the usage of the org.json Java library. The library enables the usage of JSON encoders and decoders in Java, as these functionalities are not native to the Java programming language. Fortunately, no you do not need to install this library yourself as it has been included for you.

## 1.3. Walkthrough

Interactive: Preparation

I will first provide a walkthrough for running the program in interactive.

Before we begin, navigate to the CryptoGraph directory and compile CryptoGraph by executing java -cp json-2020518.jar:. *.java.

Then, run CryptoGraph by executing java -cp json-2020518.jar:. CryptoGraph -i

Once you execute the above command, you should see the Interactive Menu.

Interactive: Asset Details

At the Selection: prompt, enter 1. This will launch Asset Details. You should see the Asset Details Menu

At the Selection: prompt, enter 2.

Afterwards, you will be returned to the Asset Details Menu. At the Selection: prompt, enter 1.

At the Symbol: prompt, enter the symbol of your choice of cryptocurrency. For the sake of demonstration enter BTC. Afterwards, you should see various details in regards to BTC.

Afterwards, you will be returned to the Asset Details Menu.

At the Selection: prompt, enter 4 and at the following Filename (without .json extension): prompt, enter the filename of your choice. For the sake of demonstration enter exchangeInfo.

Afterwards, you will be returned to the Asset Details Menu.

From there, at the Selection: prompt, enter 3 and at the following Filename: prompt, enter exchangeInfo.json, the file we created in the previous step.

Afterwards, you will be returned to the Asset Details Menu.

At the Selection: prompt, enter 1 and and at the following Symbol: prompt, enter the symbol of your choice of cryptocurrency. Like before, enter BTC. Again, you should see various details in regards to BTC.

Afterwards, you will be returned to the Asset Details Menu.

At the Selection: prompt, enter 0. You will be returned to the Interactive Menu.

<u>Interactive: Trade Details</u>

At the Selection: prompt, enter 2. This will launch Trade Details. You should see the Trade Details Menu.

At the Selection: prompt, enter 2.

Afterwards, at the Base Asset Symbol: prompt, enter the symbol of your choice of base cryptocurrency. For the sake of demonstration, enter ETH and at the following Quote Asset Symbol: prompt, enter the symbol of your quote cryptocurrency. For the sake of demonstration, enter BTC.

Thereafter, you will be returned to the Trade Details Menu.

At the Selection: prompt, enter 1 and you should see various details regarding ETHBTC.

Thereafter, you will be returned to the Trade Details Menu.

At the Selection: prompt, enter 4 and at the following Filename (without .json extension): prompt, enter a filename of your choice. For the sake of demonstration enter ETHBTCpriceChangeInfo.

Afterwards, you will be returned to the Trade Details Menu.

At the Selection: prompt, enter 3 and at the following Filename: prompt, enter ETHBTCpriceChangeInfo.json, the file we created in the previous step.

Afterwards, you will be returned to the Trade Details Menu.

At the Selection: prompt, enter 1 and you should see various details regarding ETHBTC.

Afterwards, you will be returned to the Trade Details Menu.

At the Selection: prompt, enter 0. You should be returned to the Interactive Menu.

<u>Interactive: Trade Paths</u>

At the Selection: prompt, enter 3. This will launch Trade Paths. You should see the Trade Paths Menu.

At the Selection: prompt, enter 2.

Thereafter, you will be returned to the Trade Details Menu.

At the Selection: prompt, enter 1.

At the Base Asset Symbol: prompt, enter the symbol of your choice of base cryptocurrency. For the sake of demonstration, enter ETH and at the following Quote Asset Symbol: prompt, enter the symbol of your choice of quote cryptocurrency. For the sake of demonstration, enter BTC.

Afterwards, you should see trade paths from ETH to BTC.

Thereafter, you will be returned to the Trade Details Menu.

At the Selection: prompt, enter 4 and at the following Filename (without .txt extension): prompt, enter a filename of your choice. For the sake of demonstration enter assetGraph.

Afterwards, you will be returned to the Trade Details Menu.

At the Selection: prompt, enter 3 and at the following Filename: prompt, enter assetGraph.txt, the file we created in the previous step.

Afterwards, you will be returned to the Trade Details Menu.

At the Selection: prompt, enter 1.

At the Base Asset Symbol: prompt, enter the symbol of your choice of base cryptocurrency. For the sake of demonstration, enter ETH and at the following Quote Asset Symbol: prompt, enter the symbol of your choice of quote cryptocurrency. For the sake of demonstration, enter BTC.

Afterwards, you should see trade paths from ETH to BTC.

Afterwards, you will be returned to the Trade Details Menu.

At the Selection: prompt, enter 0. You should be returned to the Interactive Menu.

## <u>Interactive: Asset Filter 1</u>

At the Selection: prompt, enter 4. This will launch Asset Filter. You should see the Asset Filters Menu.

At the Selection: prompt, enter 1.

At the Asset: prompt, enter the symbol of your choice of cryptocurrency. For the sake of demonstration, enter BTC. You should see a response that reads "BTC" will be excluded from future analyses.

Afterwards, you will be returned to the Asset Filter Menu.

At the Selection: prompt, enter 3 and you will see the excluded asset, BTC.

Afterwards, you will be returned to the Asset Filter Menu.
At the Selection: prompt, enter 0. You should be returned to the Interactive Menu.

## <u>Interactive: Asset Overview 1</u>

At the Selection: prompt, enter 5. This will launch Asset Overview. You should see the Asset Overview Menu

At the Selection: prompt, enter 2.

Afterwards, you will be returned to the Asset Details Menu.

At the Selection: prompt, enter 1.

Afterwards, you should see various details in regards to all trade pairs excluding those where a base or quote asset is an excluded asset. In this case, you will see all trade pairs which do not include BTC.

Thereafter, you will be returned to the Asset Overview Menu

At the Selection: prompt, enter 4 and at the following Filename (without .json extension): prompt, enter the filename of your choice. For the sake of demonstration enter exchangeInfo.

Thereafter, you will be returned to the Asset Overview Menu.

From there, at the Selection: prompt, enter 3 and at the following Filename: prompt, enter exchangeInfo.json, the file we created in the previous step.

Afterwards you will be returned to the Asset Overview Menu.

At the Selection: prompt, enter 1.

Again, you should see various details in regards to trade pairs.

Thereafter, you will be returned to the Asset Overview Menu.

At the Selection: prompt, enter 0. You should be returned to the Interactive Menu.

## Interactive: Asset Filter 2

At the Selection: prompt, enter 4. This will launch Asset Filter. You should see the Asset Filters Menu.

At the Selection: prompt, enter 2.

At the Asset: prompt, enter the symbol of your choice of cryptocurrency. For the sake of demonstration, enter BTC, the asset we excluded before. You should see a response that reads “BTC” will be included in future analyses.

Afterwards, you will be returned to the Asset Filter Menu.

At the Selection: prompt, enter 3 and you will see no excluded assets.

Afterwards, you will be returned to the Asset Filter Menu.
At the Selection: prompt, enter 0. You should be returned to the Interactive Menu.

Interactive: Asset Overview 2

At the Selection: prompt, enter 5. This will launch Asset Overview. You should see the Asset Overview Menu

At the Selection: prompt, enter 2.

Afterwards, you will be returned to the Asset Details Menu.

At the Selection: prompt, enter 1.

Afterwards, you should see various details in regards to all trade pairs excluding those where a base or quote asset is an excluded asset. In this case, you will see all trade pairs as there are no excluded assets.

Thereafter, you will be returned to the Asset Overview Menu

At the Selection: prompt, enter 4 and at the following Filename (without .json extension): prompt, enter the filename of your choice. For the sake of demonstration enter exchangeInfo.

Thereafter, you will be returned to the Asset Overview Menu.

From there, at the Selection: prompt, enter 3 and at the following Filename: prompt, enter exchangeInfo.json, the file we created in the previous step.

Afterwards you will be returned to the Asset Overview Menu.

At the Selection: prompt, enter 1.

Again, you should see various details in regards to trade pairs.

Thereafter, you will be returned to the Asset Overview Menu.

At the Selection: prompt, enter 0. You should be returned to the Interactive Menu.

<u>Interactive: Trade Overview</u>

At the Selection: prompt, enter 6. This will launch Trade Overview. You should see the Trade Overview Menu

At the Selection: prompt, enter 2.

At the Base Asset Symbol: prompt, enter the symbol of your choice of base cryptocurrency. For the sake of demonstration, enter ETH and at the following Quote Asset Symbol: prompt, enter the symbol of your choice of quote cryptocurrency. For the sake of demonstration, enter BTC. Lastly, at the Number of results to return: prompt, enter the number of your choice. For the sake of demonstration, enter 10.

Afterwards, you will be returned to the Trade Overview Menu.

At the Selection: prompt, enter 1.

Afterwards, you should details in regards to the top 10 ETHBTC trades in terms of price, quantity and quote.

Afterwards, you will be returned to the Trade Overview Menu

At the Selection: prompt, enter 4 and at the following Filename (without .json extension): prompt, enter the filename of your choice. For the sake of demonstration enter ETHBTCTradeDetails.

Thereafter, you will be returned to the Trade Overview Menu.

From there, at the Selection: prompt, enter 3 and at the following Filename: prompt, enter ETHBTCTradeDetails.json, the file we created in the previous step.

Afterwards you will be returned to the Asset Overview Menu.

At the Selection: prompt, enter 1.

Again, you should details in regards to the top 10 ETHBTC trades in terms of price, quantity and quote.

Thereafter, you will be returned to the Trade Overview Menu.

At the Selection: prompt, enter 0. You should be returned to the Interactive Menu.

At the subsequent Selection: prompt, enter 0. CryptoGraph execution should be stopped.

<u>Report: Preparation</u>
Before we begin, navigate to the CryptoGraph directory and compile CryptoGraph by executing java -cp json-2020518.jar:. *.java.

Then, run CryptoGraph by executing java -cp json-2020518.jar:. CryptoGraph -r exchangeInfo.json ETHBTCTradeDetails.json

Once you execute the above command, you should see the Report Menu.

<u>Report: Asset Filter 1</u>

At the Selection: prompt, enter 1. This will launch Asset Filter. You should see the Asset Filter Menu.

At the Selection: prompt, enter 1.

At the Asset: prompt, enter the symbol of your choice of cryptocurrency. For the sake of demonstration, enter BTC. You should see a response that reads “BTC” will be excluded from future analyses.

Afterwards, you will be returned to the Asset Filter Menu.

At the Selection: prompt, enter 3 and you will see the excluded asset, BTC.

Afterwards, you will be returned to the Asset Filter Menu.

At the Selection: prompt, enter 0. You should be returned to the Report Menu.

<u>Report: Asset Overview 1</u>

At the Selection: prompt, enter 2. This will launch Asset Overview.

Afterwards, you should see various details in regards to all trade pairs excluding those where a base or quote asset is an excluded asset. In this case, you will see all trade pairs which do not include BTC.

Thereafter, you will be returned to the Report Menu

<u>Report: Asset Filter 2</u>

At the Selection: prompt, enter 1. This will launch Asset Filter. You should see the Asset Filter Menu.

At the Selection: prompt, enter 2.

At the Asset: prompt, enter the symbol of your choice of cryptocurrency. For the sake of demonstration, enter BTC, the asset we excluded before. You should see a response that reads “BTC” will be included in future analyses.

Afterwards, you will be returned to the Asset Filter Menu.

At the Selection: prompt, enter 3 and you will see no excluded assets.

Afterwards, you will be returned to the Asset Filter Menu.

At the Selection: prompt, enter 0. You should be returned to the Report Menu.

<u>Report: Asset Overview 2</u>

At the Selection: prompt, enter 2. This will launch Asset Overview.

Afterwards, you should see various details in regards to all trade pairs excluding those where a base or quote asset is an excluded asset. In this case, you will see all trade pairs as there are no excluded assets.

Thereafter, you will be returned to the Report Menu

<u>Report: Trade Overview</u>

At the Selection: prompt, enter 3. This will launch Trade Overview.

Afterwards, you should details in regards to the top 10 ETHBTC trades in terms of price, quantity and quote.

Afterwards, you will be returned to the Report Menu

At the Selection: prompt, enter 0. CryptoGraph execution should be stopped.

## 1.4. Future Work

Dating CryptoGraph Responses

Currently, responses from CryptoGraph are undated and as such, if a user does not manually date information when they save it, they will have no way of knowing when particular information was obtained. Dating responses would allow users to, for instance, track the performance of a given trade pair over time–a feature which I believe could be useful. This would involve parsing the date parameters from JSONObject and JSONArray responses to a DateObject in Java. I would have implemented this feature, however, I did not deem it to be as important as the other features I implemented and time constraints made implementing additional features more difficult.

## 2. Traceability Matrix

The UI for my implementation of CryptoGraph differs from the Assignment Specification.

When the program starts in Interactive, it displays the following options:

| | |
|---|---|
| (1) Asset Details<br>(1) Display<br>(2) Make Live Request<br>(3) Load from File<br>(4) Save to File<br>(5) Exit | Given a user-specified asset, displays trade pair statistics including total number of trade pairs, trade pairs where the user-specified asset is the base asset and those where it is the quote asset. |
| (2) Trade Details<br>(1) Display<br>(2) Make Live Request<br>(3) Load from File<br>(4) Save to File<br>(5) Exit | Given a user-specified trade pair, displays price change statistics including price, price change ($) and price change (%). |
| (3) Trade Paths<br>(1) Display<br>(2) Make Live Request<br>(3) Load from File<br>(4) Save to File<br>(5) Exit | Given a user-specified trade pair, displays all direct and indirect trade paths from the base asset to the quote asset |
| (4) Asset Filter<br>(1) Exclude<br>(2) Include<br>(3) Display<br>(4) Exit | Allows user to exclude or include certain assets for asset overviews |
| (5) Asset Overview<br>(1) Display<br>(2) Make Live Request<br>(3) Load from File<br>(4) Save to File<br>(5) Exit | Displays all trade pairs involving assets which have not been added to the asset filter |
| (6) Trade Overview<br>(1) Display<br>(2) Make Live Request<br>(3) Load from File<br>(4) Save to File | Given a user-specified trade pair, displays top n trades for price, quantity and quote sorted in descending order. |
| (7) Exit | |

When the program starts in Report, it displays the following options:

| | |
|---|---|
| (1) Asset Filter<br>(1) Exclude<br>(2) Include<br>(3) Display<br>(4) Exit | Allows user to exclude or include certain assets for asset overviews |
| (2) Display Asset Overview | Displays all trade pairs involving assets which have not been added to the asset filter |
| (3) Display Trade Overview | Given a user-specified trade pair, displays top n trades for price, quantity and quote sorted in descending order. |
| (4) Exit | |

As such, the traceability matrix(es) are as follows:

**Interactive**

| | | **Requirement** | **Code** | **Test** |
|---|---|---|---|---|
| 0 | Mode | 1. Launches with -i argument | CryptoGraph.main() | Walkthrough: Interactive / Preparation |
| | | 2. Program responds when user enters commands | InteractiveEntry.entry() | Walkthrough: Interactive |
| 1 | Asset Details | 1. Displays asset details if user calls Display | InteractiveDisplayer.displayAssetTradePairs() | Walkthrough: Interactive / Asset Details |
| | | 2. Displays error message if user calls Display before Make Live Request or Load from File | InteractiveAnalyser.getAssetTradePairs() | |
| | | 3. Makes live request if user calls Make Live Request | IOJSONObject.readFromURL() | |
| | | 4. Saves to file if user calls Save to File | IOJSONArray.writeToFile() | |
| | | 5. Loads from file if user calls Load from | IOJSONArray.readFromFile() | |

| | | | | |
|---|---|---|---|---|
| | | File and provides valid filename | | |
| | | 6. Displays error message if user calls Load from File and provides invalid filename | InteractiveAnalyser.getAssetTradePairs() | |
| | | 7. Returns to Interactive Menu if user calls Exit | InteractiveAnalyser.getAssetTradePairs() | |
| 2 | Trade Details | 1. Displays trade details if user calls Display | InteractiveDisplayer.displayPriceChangeInfo() | Walkthrough: Interactive / Trade Details |
| | | 2. Displays error message if user calls Display before Make Live Request or Load from File | InteractiveAnalyser.getTradePriceChangeInfo() | |
| | | 3. Makes live request if user calls Make Live Request with valid trade symbol | IOJSONObject.readFromURL() | |
| | | 4. Displays error message if user calls Make Live Request | InteractiveAnalyser.getTradePriceChangeInfo() | |

| | | | | |
|---|---|---|---|---|
| | | with invalid trade symbol | | |
| | | 5. Saves to file if user calls Save to File | InteractiveAnalyser.getTradePriceChangeInfo()<br>IOJSONObject.writeToFile() | |
| | | 6. Loads from file if user calls Load from File with a valid filename | IOJSONObject.readFromFile() | |
| | | 7. Displays error message if user calls Load from File with an invalid filename | IOJSONObject.readFromFile() | |
| | | 8. Returns to Interactive Menu if user calls Exit | InteractiveAnalyser.getTradePriceChangeInfo() | |
| 3 | Trade Paths | 1. Displays trade paths if user calls Display with valid trade symbol | InteractiveDisplayer.displayTradePaths() | Walkthrough: Interactive / Trade Paths |
| | | 2. Displays error message if user calls Display with invalid trade symbol | InteractiveAnalyser.getAssetGraph() | |
| | | 3. Displays error message if user calls Display before Make | InteractiveAnalyser.getAssetGraph() | |

| | | | | |
|---|---|---|---|---|
| | | Live Request or Load from File | | |
| | | 4. Makes live request if user calls Make Live Request | InteractiveDisplayer.displayTradePaths() | |
| | | 5. Saves to file if user calls Save to File | IODSAGraph.write() | |
| | | 6. Loads from file if user calls Load from File with a valid filename | IODSAGraph.read() | |
| | | 7. Displays error message if user calls Load from file with an invalid filename | InteractiveAnalyser.getAssetGraph() | |
| | | 8. Returns to Interactive Menu if user calls Exit | InteractiveAnalyser.getAssetGraph() | |
| 4 | Asset Filter | 1. Excludes asset if user calls Exclude with the symbol of an asset that was not previously excluded | excludedAssets.add() | Walkthrough: Interactive / Asset Filter |
| | | 2. Displays error message if user calls | InteractiveAnalyser.addToExcludedAssets() | |

| | | | | |
|---|---|---|---|---|
| | | Exclude with the symbol of an asset that was previously excluded | | |
| | | 3. Includes asset if calls Include with the symbol of an asset that was previously excluded | excludedAssets.remove() | |
| | | 4. Displays error message if user calls Exclude with the symbol of an asset that was not previously excluded | InteractiveAnalyser.addToExcludedAssets() | |
| | | 5. Displays asset filter if user calls Display | InteractiveAnalyser.addToExcludedAssets() | |
| | | 6. Returns to Interactive Menu if user calls Exit | InteractiveAnalyser.addToExcludedAssets() | |
| 5 | Asset Overview | 1. Displays asset overview if user calls Display after calling Make Live Request*3 | InteractiveAndReportDisplayer.displayAssetOverview() | Walkthrough: Interactive / Asset Overview |

| | | | | |
|---|---|---|---|---|
| | | 2. Displays error message if user calls Display before calling Make Live Request*4 | InteractiveAnalyser.getAllTradePairs() | |
| | | 3. Displays asset overview if user calls Display after calling Load from File*1 | InteractiveAndReportDisplayer.displayAssetOverview() | |
| | | 4. Displays error message if user calls Display before calling Load from File with the filename of valid file*2 | InteractiveAnalyser.getAllTradePairs() | |
| | | 5. Makes live request if user calls Make Live Request | IOJSONObject.readFromURL() | |
| | | 6. Saves to file if user calls Save to File | IOJSONArray.writeToFile() | |
| | | 7. Loads from file if user calls Load from File with filename of valid file | IOJSONArray.readFromFile() | |

| | | | | |
|---|---|---|---|---|
| | | 8. Displays error message if user calls Load from File with filename of invalid file | IOJSONArray.readFromFile() | |
| | | 9. Returns to Interactive Menu if user calls Exit | InteractiveAnalyser.getAllTradePairs() | |
| 6 | Trade Overview | 1. Displays trade overview if user calls Display after calling Make Live Request | InteractiveAndReportDisplayer.displayRecentTradePrices()<br>InteractiveAndReportDisplayer.displayRecentTradeQtys()<br>InteractiveAndReportDisplayer.displayRecentTradeQuotes() | Walkthrough: Interactive / Trade Overview |
| | | 2. Displays error message if user calls Display before calling Make Live Request | InteractiveAnalyser.getRecentTrades() | |
| | | 3. Displays trade overview if user calls Display after calling Load from File | InteractiveAndReportDisplayer.displayRecentTradePrices()<br>InteractiveAndReportDisplayer.displayRecentTradeQtys()<br>InteractiveAndReportDisplayer.displayRecentTradeQuotes() | |
| | | 4. Displays error message if user calls Display before | InteractiveAnalyser.getRecentTrades() | |

| | | | | |
|---|---|---|---|---|
| | | calling Load from File with the filename of a valid file | | |
| | | 5. Makes live request if calls Make Live Request with symbols of valid base and quote assets and number of results | IOJSONArray.readFromURL() | |
| | | 6. Saves to file if user calls Save to File | IOJSONArray.writeToFile() | |
| | | 7. Loads from File if user calls Load from File with filename of valid file | IOJSONArray.readFromFile() | |
| | | 8. Displays error message if user calls Load from File with filename of invalid file | InteractiveAnalyser.getRecentTrades() | |
| | | 9. Returns to Interactive Menu if user calls Exit | InteractiveAnalyser.getRecentTrades() | |

**Report**

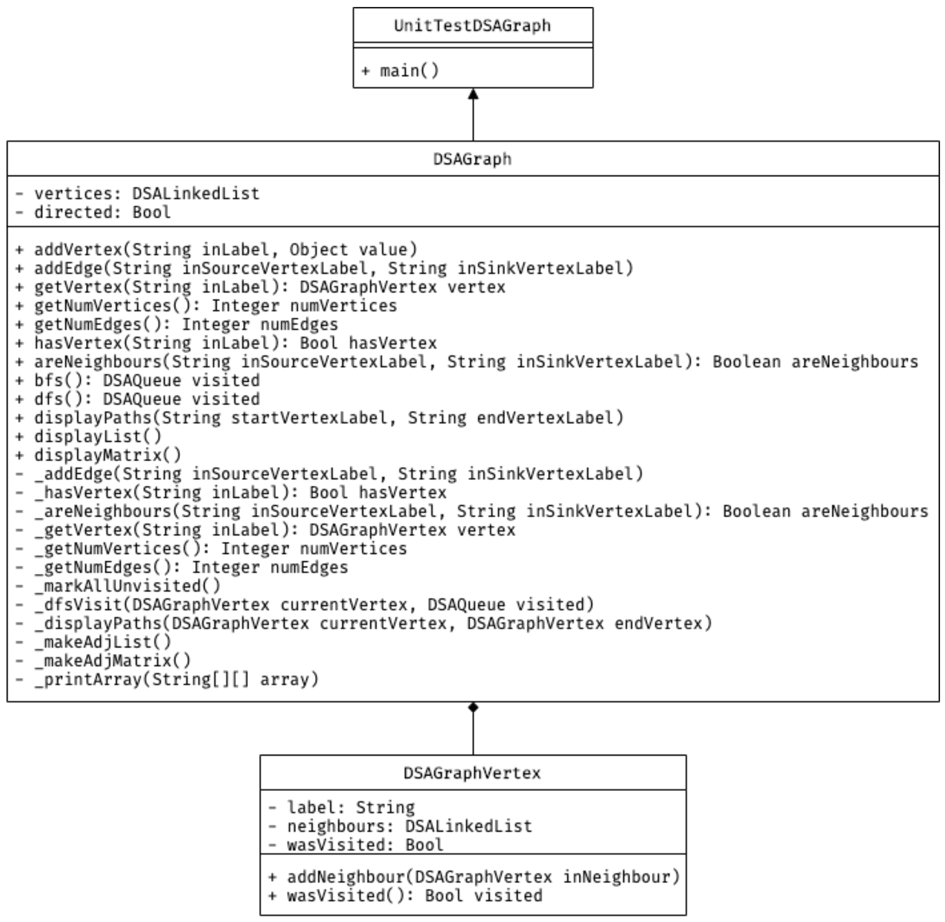
| | | **Requirement** | **Code** | **Test** |
|---|---|---|---|---|
| 0 | Modes | 1. Displays usage if called with incorrect arguments | CryptoGraph.main() | Trying to run CryptoGraph in Report in a manner other than specified in README.txt |
| | | 2. Launches with -r, <asset filename>.json and <trade filename>.json arguments | CryptoGraph.main() | Walkthrough: Report / Preparation |
| | | 3. Responds when user enters commands | ReportEntry.entry() | Walkthrough: Report |
| 1 | Asset Filter | 1. Excludes asset if user calls Exclude with the symbol of an asset that was not previously excluded | excludedAssets.add() | Walkthrough: Report / Asset Filter |
| | | 2. Displays error message if | InteractiveAnalyser.changeExcludedAssets() | |

| | | | | |
|---|---|---|---|---|
| | | user calls Exclude with the symbol of an asset that was previously excluded | | |
| | | 3. Includes asset if calls Include with the symbol of an asset that was previously excluded | excludedAssets.remove() | |
| | | 4. Displays error message if user calls Exclude with the symbol of an asset that was not previously excluded | InteractiveAnalyser.changeExcludedAssets() | |
| | | 5. Displays asset filter if user calls Display | InteractiveAnalyser.changeExcludedAssets() | |

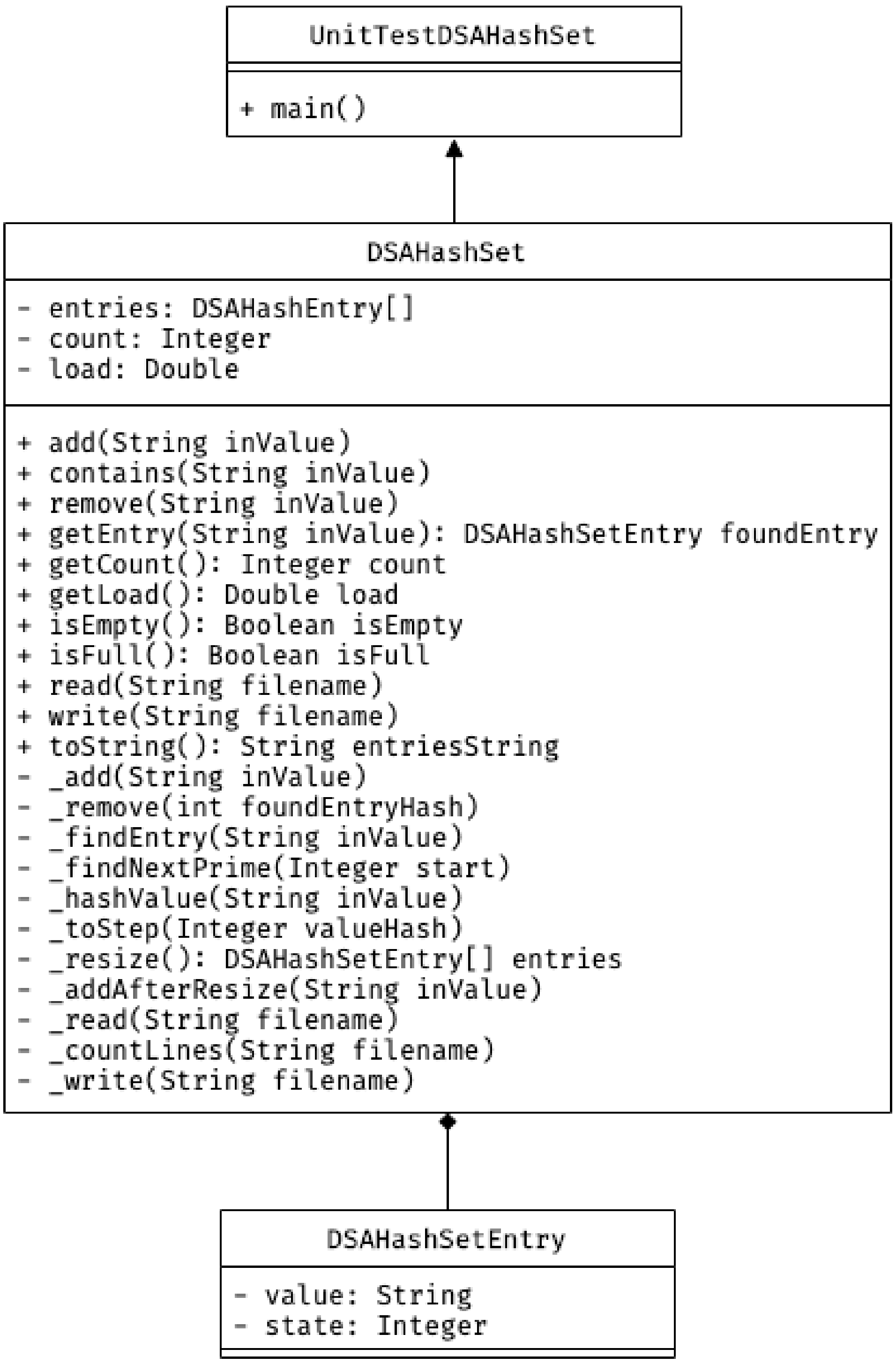
| | | | | |
|---|---|---|---|---|
| | | 6. Returns to Interactive Menu if user calls Exit | InteractiveAnalyser.changeExcludedAssets() | |
| 2 | Asset Overview | 1. Displays asset overview | InteractiveAndReportDisplayer.displayAssetOverview() | Walkthrough: Report / Asset Overview |
| 3 | Trade Overview | 1. Displays trade overview | InteractiveAndReportDisplayer.displayRecentTradePrices()<br>InteractiveAndReportDisplayer.displayRecentTradeQtys()<br>InteractiveAndReportDisplayer.displayRecentTradeQuotes() | Walkthrough: Report / Trade Overview |

## 3. UML Class Diagram

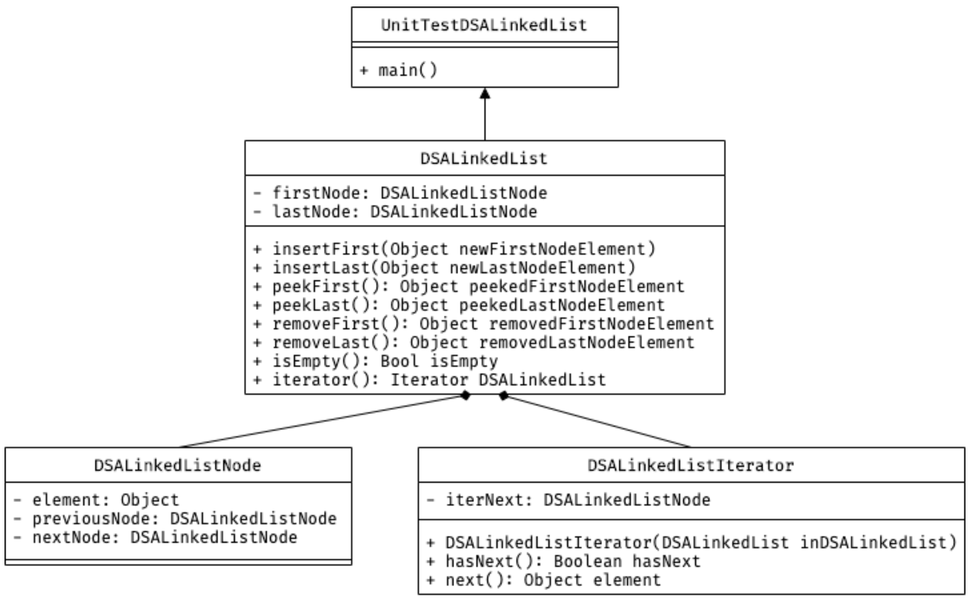
### 3.1. DSAGraph


UnitTestDSAGraph
+ main()
DSAGraph
- vertices: DSALinkedList
- directed: Bool
+ addVertex(String inLabel, Object value)
+ addEdge(String inSourceVertexLabel, String inSinkVertexLabel)
+ getVertex(String inLabel): DSAGraphVertex vertex
+ getNumVertices(): Integer numVertices
+ getNumEdges(): Integer numEdges
+ hasVertex(String inLabel): Bool hasVertex
+ areNeighbours(String inSourceVertexLabel, String inSinkVertexLabel): Boolean areNeighbours
+ bfs(): DSAQueue visited
+ dfs(): DSAQueue visited
+ displayPaths(String startVertexLabel, String endVertexLabel)
+ displayList()
+ displayMatrix()
- _addEdge(String inSourceVertexLabel, String inSinkVertexLabel)
- _hasVertex(String inLabel): Bool hasVertex
- _areNeighbours(String inSourceVertexLabel, String inSinkVertexLabel): Boolean areNeighbours
- _getVertex(String inLabel): DSAGraphVertex vertex
- _getNumVertices(): Integer numVertices
- _getNumEdges(): Integer numEdges
- _markAllUnvisited()
- _dfsVisit(DSAGraphVertex currentVertex, DSAQueue visited)
- _displayPaths(DSAGraphVertex currentVertex, DSAGraphVertex endVertex)
- _makeAdjList()
- _makeAdjMatrix()
- _printArray(String[][] array)
DSAGraphVertex
- label: String
- neighbours: DSALinkedList
- wasVisited: Bool
+ addNeighbour(DSAGraphVertex inNeighbour)
+ wasVisited(): Bool visited

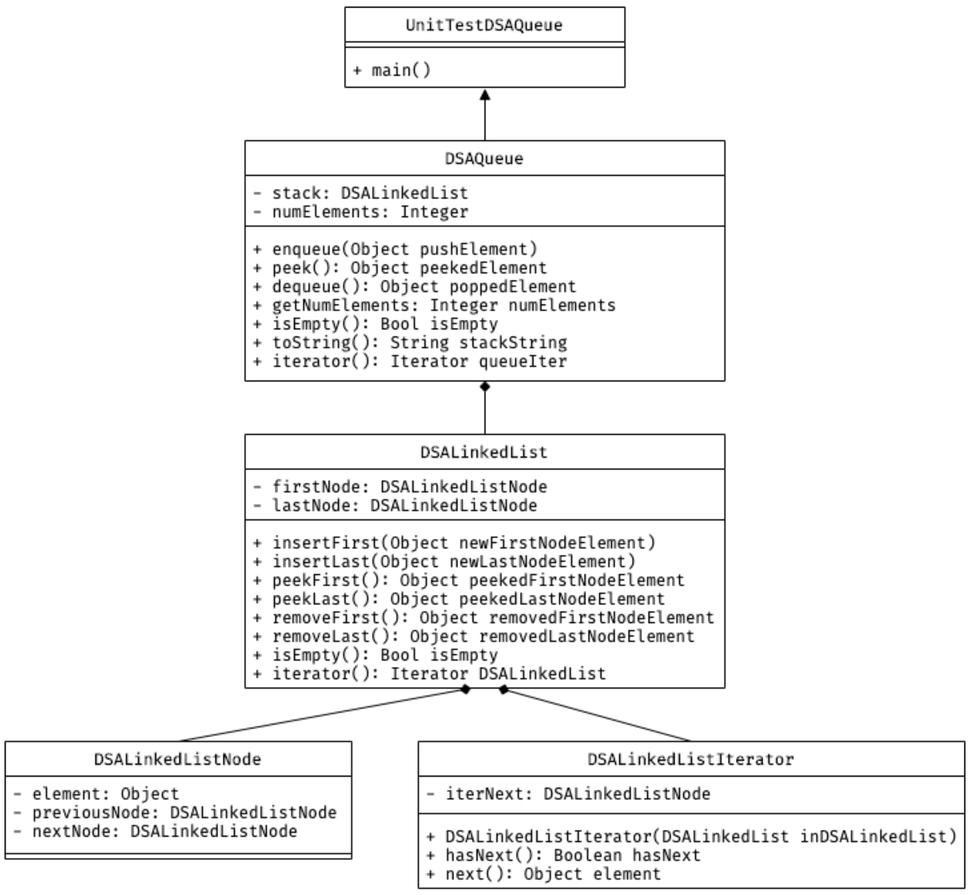
## 3.2. DSAHashSet


UnitTestDSAHashSet
+ main()
DSAHashSet
- entries: DSAHashEntry[]
- count: Integer
- load: Double
+ add(String inValue)
+ contains(String inValue)
+ remove(String inValue)
+ getEntry(String inValue): DSAHashSetEntry foundEntry
+ getCount(): Integer count
+ getLoad(): Double load
+ isEmpty(): Boolean isEmpty
+ isFull(): Boolean isFull
+ read(String filename)
+ write(String filename)
+ toString(): String entriesString
- _add(String inValue)
- _remove(int foundEntryHash)
- _findEntry(String inValue)
- _findNextPrime(Integer start)
- _hashValue(String inValue)
- _toStep(Integer valueHash)
- _resize(): DSAHashSetEntry[] entries
- _addAfterResize(String inValue)
- _read(String filename)
- _countLines(String filename)
- _write(String filename)
DSAHashSetEntry
- value: String
- state: Integer

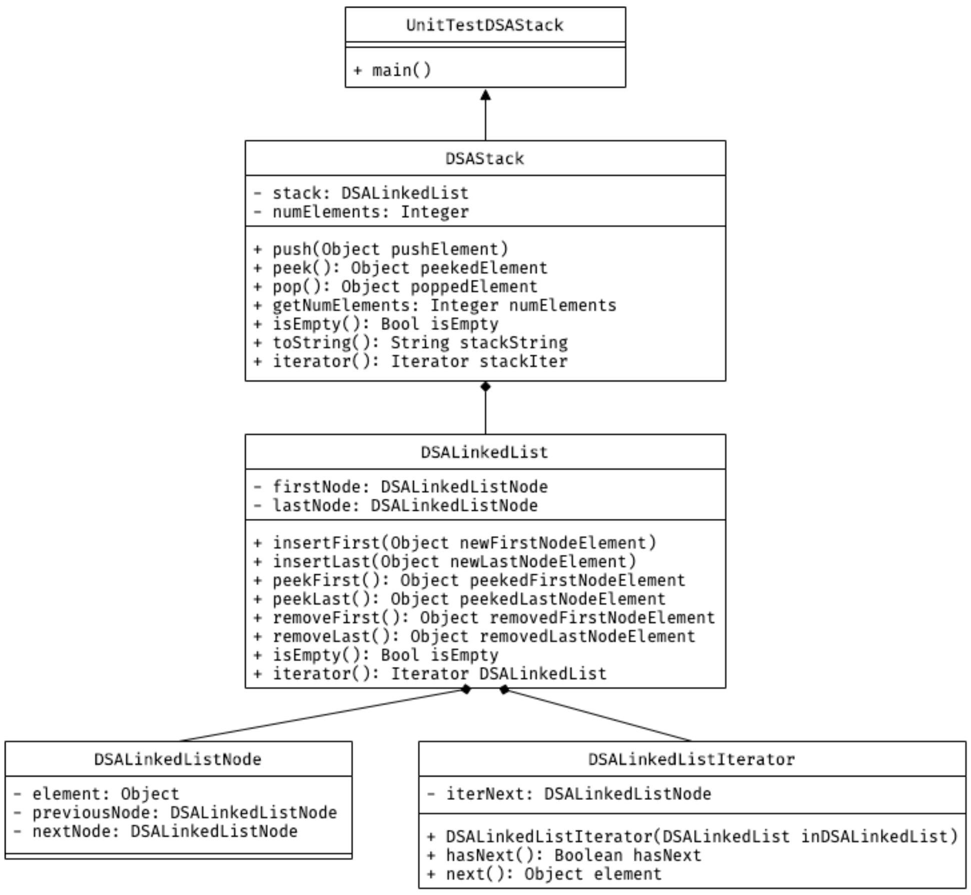
## 3.3. DSALinkedList


UnitTestDSALinkedList
+ main()
DSALinkedList
- firstNode: DSALinkedListNode
- lastNode: DSALinkedListNode
+ insertFirst(Object newFirstNodeElement)
+ insertLast(Object newLastNodeElement)
+ peekFirst(): Object peekedFirstNodeElement
+ peekLast(): Object peekedLastNodeElement
+ removeFirst(): Object removedFirstNodeElement
+ removeLast(): Object removedLastNodeElement
+ isEmpty(): Bool isEmpty
+ iterator(): Iterator DSALinkedList
DSALinkedListNode
- element: Object
- previousNode: DSALinkedListNode
- nextNode: DSALinkedListNode
DSALinkedListIterator
- iterNext: DSALinkedListNode
+ DSALinkedListIterator(DSALinkedList inDSALinkedList)
+ hasNext(): Boolean hasNext
+ next(): Object element

## 3.4. DSAQueue


UnitTestDSAQueue
+ main()
DSAQueue
- stack: DSALinkedList
- numElements: Integer
+ enqueue(Object pushElement)
+ peek(): Object peekedElement
+ dequeue(): Object poppedElement
+ getNumElements: Integer numElements
+ isEmpty(): Bool isEmpty
+ toString(): String stackString
+ iterator(): Iterator queueIter
DSALinkedList
- firstNode: DSALinkedListNode
- lastNode: DSALinkedListNode
+ insertFirst(Object newFirstNodeElement)
+ insertLast(Object newLastNodeElement)
+ peekFirst(): Object peekedFirstNodeElement
+ peekLast(): Object peekedLastNodeElement
+ removeFirst(): Object removedFirstNodeElement
+ removeLast(): Object removedLastNodeElement
+ isEmpty(): Bool isEmpty
+ iterator(): Iterator DSALinkedList
DSALinkedListNode
- element: Object
- previousNode: DSALinkedListNode
- nextNode: DSALinkedListNode
DSALinkedListIterator
- iterNext: DSALinkedListNode
+ DSALinkedListIterator(DSALinkedList inDSALinkedList)
+ hasNext(): Boolean hasNext
+ next(): Object element

## 3.5. DSAStack


UnitTestDSAStack
+ main()
DSAStack
- stack: DSALinkedList
- numElements: Integer
+ push(Object pushElement)
+ peek(): Object peekedElement
+ pop(): Object poppedElement
+ getNumElements: Integer numElements
+ isEmpty(): Bool isEmpty
+ toString(): String stackString
+ iterator(): Iterator stackIter
DSALinkedList
- firstNode: DSALinkedListNode
- lastNode: DSALinkedListNode
+ insertFirst(Object newFirstNodeElement)
+ insertLast(Object newLastNodeElement)
+ peekFirst(): Object peekedFirstNodeElement
+ peekLast(): Object peekedLastNodeElement
+ removeFirst(): Object removedFirstNodeElement
+ removeLast(): Object removedLastNodeElement
+ isEmpty(): Bool isEmpty
+ iterator(): Iterator DSALinkedList
DSALinkedListNode
- element: Object
- previousNode: DSALinkedListNode
- nextNode: DSALinkedListNode
DSALinkedListIterator
- iterNext: DSALinkedListNode
+ DSALinkedListIterator(DSALinkedList inDSALinkedList)
+ hasNext(): Boolean hasNext
+ next(): Object element

# 4. Class Descriptions & Justification

## 4.1. Abstract Data Types

As for the ADTs made in past practicals–DSAGraph, DSALinkedList, DSAStack and DSAQueue–the point of interest is why I chose to implement these classes in the manner in which I did. For classes which involved the implementation of a sub class (e.g. DSALinkedList and DSALinkedListNode), I chose to implement the sub-class as a private inner class as opposed to one housed in a separate file.

The most notable reasons for making this decision include:

1. **Grouping**
   Nested classes enable the grouping of classes that are only used in one place. Specifically, if a particular class is only used by one other class (e.g. the usage DSALinkedListNode by DSALinkedList), it would make logical sense to embed that particular class within the class it is used by, keeping both classes together. Doing so, nesting "helper" classes, makes packages in their entirety, more cohesive.
2. **Encapsulation**
   Consider DSALinkedListNode and DSALinkedList, where DSALinkedListNode needs to access the members of DSALinkedList that would otherwise be declared private. Nesting DSALinkedListNode within DSALinkedList allows DSALinkedListNode to be hidden from the end user, as they are typically only concerned with interacting with the interface of a particular class (in this case, DSALinkedList) and are not concerned with the underlying details of how this class is implemented.
3. **Readability and Maintainability**
   By nesting helper classes within their top-level classes, the code contained within is placed closer to where it is used.
   (Oracle, n.d.)

## 4.2. DSAHashSet

The purpose of this class was to provide an efficient means to store assets to exclude from analyses. Seeing as a simple asset filter would involve lookups, I decided to implement the most well-suited ADT for such a purpose, a hash set. Seeing as the underlying data structure used to implement a hash set is a hash table, the average time complexity for add, search and remove operations is O(1).

## 4.3. CryptoGraph

The purpose of this class was to serve as the entry point to the CryptoGraph cryptocurrency analysis suite. It is from here that Interactive and Report are launched.

### 4.4. InteractiveAnalyser

This class serves as the data “getter” for various Interactive operations. That is to say, information such as price change info, asset trade pairs and recent trades are obtained here and then returned to the calling function(s) for later displaying.

### 4.5. InteractiveAndReportDisplayer

The purpose of this class is to facilitate the displaying of data which is applicable to both Interactive and Report. Specifically, Asset Overview and Trade Overview are both available operations in Interactive and Report so rather than having two different classes with essentially duplicate code, I created a single class and had both modes utilise this class.

### 4.6. InteractiveAndReportSorter

The purpose of this class is to sort recent trades by price, quantity and quote and this relates to the Trade Overview operation. The operation is used in both Interactive and Report modes so, again, rather than have duplicate code in different classes, I created just a single class and had Trade Overview from Interactive and Report just call this class instead.

### 4.7. InteractiveDisplayer

This class serves to display data obtained in Interactive which is only used in this mode (i.e. everything else apart from Asset Overview and Trade Overview).

### 4.8. InteractiveEntry

This class serves as the entry point for the Interactive mode operations of CryptoGraph.

### 4.9. InteractiveGrapher

The purpose of this class is to create asset graphs for later finding trade paths between two given assets.

### 4.10. IODSAGraph

This class serves to facilitate Input and Output operations–read and write–in regards to DSAGraphs. This class is used to read and write asset graphs for use in finding trade paths between two assets.

### 4.11. IOJSONArray

The purpose of this class is to enable to the Input and Output of JSONArrays. It allows the reading of JSONArrays from URLs and files along with the writing of

JSONArrays to files too. This is mainly used for obtaining recent trades for a given trade pair, displaying those trades and saving them to files as well.

## 4.12. IOJSONObject

This class operates almost identically to IOJSONArray, however, it facilitates IO operations in regards to JSONObjects. It allows the same reading from URLs and files and the writing of JSONObjects to files. This is the main IO class for JSON encoding and decoding and is used throughout CryptoGraph.

## 4.13. ReportEntry

This class serves as the entry point for the Interactive mode operations of CryptoGraph.

## 4.14. UserInterface

I created this class in PDI in Semester 1, 2020 and it serves as the interface with which user input–integers and strings–is retrieved. Seeing as the retrieval and usage of user input is a common feature in programs having a class dedicated to such a task is incredibly convenient and allows one to avoid having multiple scanners scattered throughout their programs as this is just “boiler-plate” code which just takes up space and distracts from what it is that a submodule is meant to do.

Excluding the ADTs which I explained and justified previously, the classes mentioned in 4.3–4.14 were created for essentially the same reason. Specifically, modularity (i.e. cohesion and coupling).

### Cohesion

Definition: “*cohesion* is the extent to which a single module does one well-defined task.” (EECMS 2020).

Moreover, when engineering software our goal should be to *maximise* cohesion. In order to maximise cohesion, *well-defined* tasks should have their own submodules in their relevant classes. As a result, I have a variety of different submodules and classes to uphold this very principle (EECMS 2020).

For instance, I could have just created a single IO.java class, however, it would not be well-defined enough (in my personal opinion) as there are a variety of different data which I would require IO for (in this case, DSAGraphs, JSONObjects and JSONArrays). The aforementioned are all very distinct data types and as such, they warrant their own IO classes. In turn, they have IO classes that are made specifically for their given datatype, maximising cohesion (EECMS 2020).

Furthermore, the higher cohesion awarded from separating distinct tasks across various submodules/classes is more efficient from a mental standpoint. That is to say, it is easier to understand submodules which have a singular, well-defined purpose as your mind will only be concerned with the implementation of a *single* problem as opposed to many (which occurs when submodules have low cohesion). If code is easier to understand, it is therefore faster to write, test, inspect and in addition, modify (EECMS 2020).

**Coupling**

The coupling exhibited in the various submodules/classes I implemented was almost exclusively limited to *data coupling*, *data-structured* coupling and *control coupling*– the three loosest forms of coupling (GeeksforGeeks 2020).

<u>Data Coupling</u>
The relativity between a good number of submodules I implemented was limited to the sharing of elementary data between submodules (i.e. passing a string defining a filename to a function that reads a file) (GeeksforGeeks 2020).

<u>Data-Structured Coupling</u>
Many of the submodules I implemented shared ADTs and used elements of those ADTs (e.g. passing a DSAGraph to a submodule which only needs to use its vertices field). Alternatively, I could have opted for data coupling here as well, however, the current implementation allowed for more flexibility in how I chose to construct certain submodules (GeeksforGeeks 2020).

<u>Control Coupling</u>
The last and probably most necessary form of coupling present in my code was control coupling. That is to say, I had submodules which would control the flow of another / others, much like when a user makes a particular selection and then CryptoGraph would have the associated action occur (GeeksforGeeks 2020).

The only other form of coupling which was present is *external coupling*–coupling where modules using externally obtained data–as I needed to use the org.json library was JSON encoding and decoding is not native to Java.

## 5. Implementation of Additional Data File

I chose not to implement the additional information stored in the asset_info.csv file as time constraints would have made doing so impractical. However, if I were to have used it, I would have implemented it in Asset Details as Binance doesn't offer any information in regards to a particular asset's market cap, price or 24hr volume. This information would have allowed me to provide a more fully featured Asset Details implementation.

## References

School of Electrical Engineering, Computing and Mathematical Sciences (EECMS). n.d. "Example of a Traceability Matrix". Curtin University.

Oracle. n.d. "Nested Classes". Oracle. https://docs.oracle.com/javase/tutorial/java/javaOO/nested.html

EECMS. 2020. "Introduction to Software Engineering (ISAD) – Lecture 7: Modularity". Slides. Curtin University.

GeeksforGeeks. "Software Engineering | Coupling and Cohesion". 2020. GeeksforGeeks. https://www.geeksforgeeks.org/software-engineering-coupling-and-cohesion/