Department of Computing Curtin University

DATA STRUCTURES AND ALGORITHMS

Assignment Documentation [CryptoGraph]

Semester 2, 2020

Name: Seenarain Bhavyattaa
ID: 19871253

TABLE OF CONTENTS

OVERVIEW

This assignment was developed to analyse crypto-currency trading data. A graph implementation was used to represent symbols, baseAsset ,quoteAsset, bid and volume. Furthermore, a symbol can be broken down into baseAssert and quoteAsset.The source vertex is the baseAsset whereas the destination vertex is the quoteAsset. Both of them are extracted from the JSON asset file.Moreover, the program also make use of trade file which provide edge information between pairs. The mentioned details pertaining to a trade file was put in a TradeDetails class with each characteristic as a class field.

CryptoGraph can be executed in two modes:

1. Report mode - this mode is run at command line whereby the user need to include both the assetFile.json and TradeInfo.json. The values are extracted from the JSON file which are the symbols, baseAsset, quoteAsset, bid and volume.

2. Interactive mode - In this mode the user can manipulate the eleven options that are provided. That is asset details, trade details, display graph, load and save serialised data. Moreover, one cal so opt for asset and trade overview where you can insert the symbol of any asset or trade and thus received its full trade details.

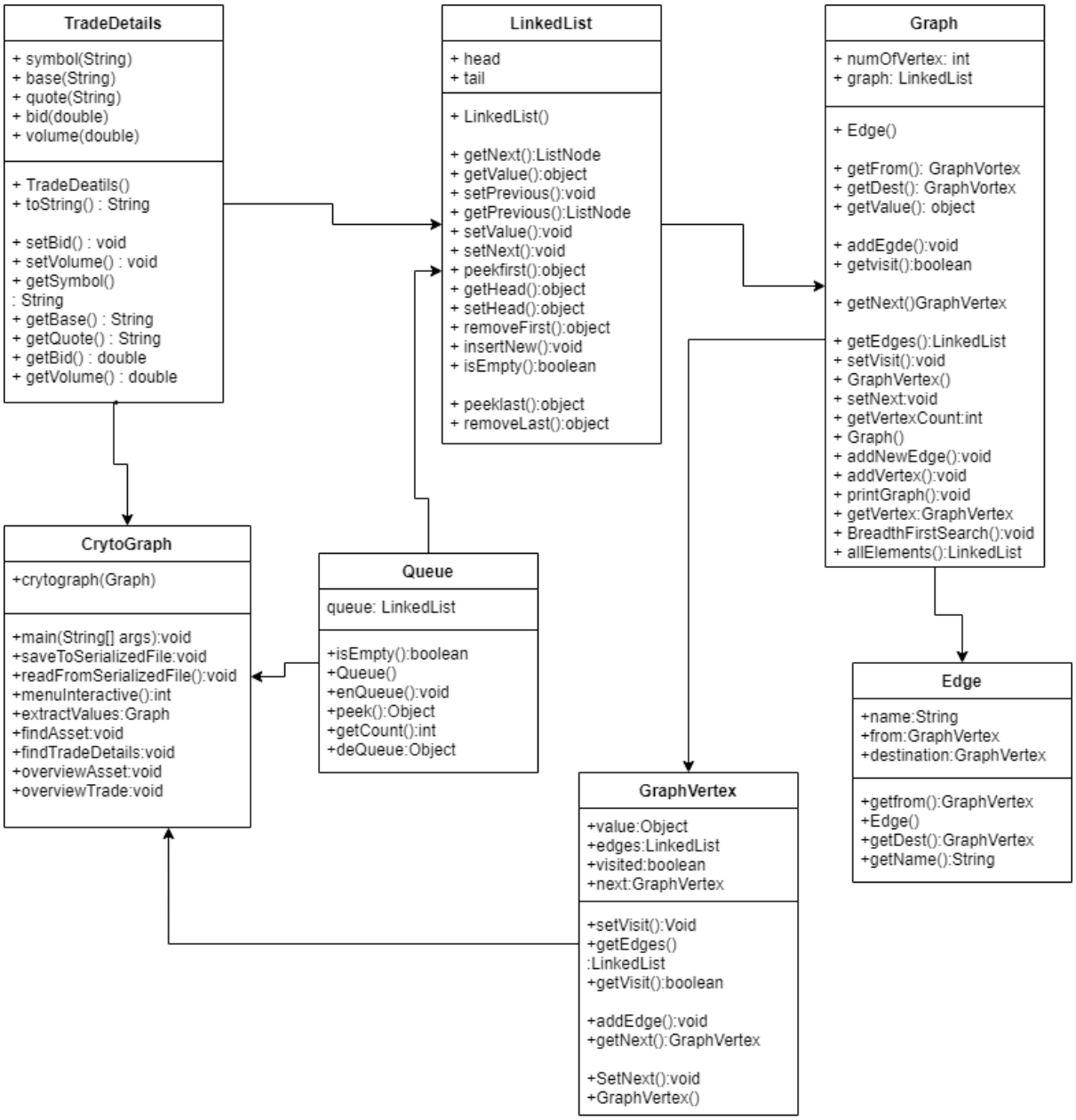
A report of the CryptoGraph was provided to demonstrate the performance of the program where each specific section of the report and interactive mode are explained.

TRACEABILITY MATRIX

		Requirements	Design
0	Menu or Modes	Program display usage information if the command: Java -cp.:json-20200518.jar CryptoGraph is used without argument	CrytoGraph.main()
		Program enters report mode with the command: Java -cp.:json-20200518.jar CryptoGRaph assetFile.json TradeInfo.json	CrytoGraph.main()
		Program enters interactive mode with the command : javac -cp.:json-20200518.jar CryptoGraph -I	CrytoGraph.menuInteractive()
1	Asset Data	This option loads the asset file and enable the user to insert the asset	CryptoGraph.main()

		file ,that is assetFile.json	
2	Trade Data	This option load the trade file and the user will have to insert the trade file - TradeInfo.json	CryptoGraph.main()
3	Display Graph	This option make use of a LinkedList to extract the values (symbols, baseAsset, quoteAsset, bid, volume) from the TradeInfo.json	CryptoGraph.extractValues(assetFileName, tradeDetails,cryptograph)
4	Load Serialized data	This option read object from serialisedObjectFile.txt	CryptoGraph.readFromSerialisedFile()
5	Save Serialized data	This option save object from serialisedObjectFile.txt	CryptoGraph.saveToSerialisedFile(cryptograph)
6	Display Asset	This option asks the user to insert asset symbol, where the function findAsset() search for the asset and display true if found else false	CryptoGraph.findAsset(asset1, assetFileName)
7	Display trade details	This option will allow the user to enter a trade symbol. The method findTradeDetails() finds and displays trade details if correct file is inserted.	CryptoGraph.findTradeDetails(tradeDetails1, tradeDetails)
8	Display potential trade path	This option displays direct and indirect path	Not implemented
9	Asset overview	In this option the user needs to enter a specific asset symbol that he/she wants to receive all its trade details (status, baseAsset and quoteAsset)	CryptoGraph.overviewAsset(asset2, assetFileName)
10	Trade Overview	This option enables the user to insert the trade symbol in order to receive all the tradeDetails such as bid price, volumn and count	CryptoGraph.overviewTrade(tradeDetail2, tradeDetails)
11	Exit	Exit the program	CryptoGraph.main()

UML CLASS DIAGRAM



CLASS DESCRIPTION

Class	Description
TradeDetails	This class is to create a TradeDetail object which are to be considered as a 'node' in linked list class.Moreover, TradeDetails object are used in Graph class as vertices.
LinkedList	LinkedList uses TradeDetails objects as 'nodes. So, a linked list object can be linked list of TradeDetails objects. An extension of practical 4 has been used with some modification according the assignment specification
Graph	Graph uses linked list object to create graph and also implement serialisable
GraphVertex	GraphVertex uses as an object in Graph class and it also implement serialisable
Edge	Edge class methods are used in Graph class and it also provide method getfrom(), getDest() and getNames() to cryptoGraph class Edge class also implements serialisable
Queue	Queue class makes use of linked list, and its functions are used in cryptoGraph class. It also implements serialisable.
CryptoGraph	CryptoGraph is the main framework of the program. It uses the Graph object known as cryptograph where values which have been extracted are save to graph. However, in whole this class uses the method of linked list, Graph, Edge, GraphVertex, TradeDetails and cryptograph class itself.

JUSTIFICATION

LINKED LIST

Linked List is more effective as it holds different TradeDetails object. For instance, in Graph.class the object named graph is used to extract values from the tradefiles:

Using Linked List an abstract datatype is worth it as not only it provides adequate space for any amount of memory but also it can grow and shrink if needed.

However, compare to arrays a fixed size should be used. As it is difficult to do a fixed size estimation according to TradeDetails objects. Hence, Linked List is more appropriate to use in this assignment.

DIRECTED GRAPH

CryptoGraph uses Graph to store the extracted values as:

A graph is simpler as one TradeDetails can be connected to several others

Hence, we can easily showcase TradeDeatils through different edges