CRANFIELD UNIVERSITY

WIKTOR BEDNAREK

APPLICATION SOFTWARE FOR STOCK MARKET PREDICTION

CRANFIELD UNIVERSITY

Computational & Software Techniques In Engineering

MSc

Academic Year: 2016 - 2017

Supervisor: Dr Irene Moulitsas

April 2017

CRANFIELD UNIVERSITY

CRANFIELD UNIVERSITY

Computational & Software Techniques In Engineering

MSc

Academic Year 2016 - 2017

WIKTOR BEDNAREK

APPLICATION SOFTWARE FOR STOCK MARKET PREDICTION

Supervisor:  Dr Irene Moulitsas

April 2017

This thesis is submitted in partial fulfilment of the requirements for the degree of MSc

***(NB. This section can be removed if the award of the degree is based solely on examination of the thesis)***

© Cranfield University 2017. All rights reserved. No part of this publication may be reproduced without the written permission of the copyright owner.

ABSTRACT

The topic of this thesis is to create an application that predicts stock market behaviour. The user will be asked to indicate the company from the stock exchange that interests him. Our application will analyse the data from the past (prices), recommendations, etc. of the chosen stock-offering company and according to them will generate one possible output: buy, hold, sell. The application will download current and past data from the stock market API. To solve this problem commonly known economic theory will be used such as Technical analysis or Black-Scholes model. The technologies that we will consider to solve the problem are Java, Android and other external libraries.

Keywords:

Technical analysis, Black-Scholes model, Stock market prediction application, Stock exchange prediction software

ACKNOWLEDGEMENTS

TABLE OF CONTENTS

[ABSTRACT i](#_Toc490820582)

[ACKNOWLEDGEMENTS ii](#_Toc490820583)

[LIST OF FIGURES v](#_Toc490820584)

[LIST OF TABLES vii](#_Toc490820585)

[LIST OF EQUATIONS ix](#_Toc490820586)

[LIST OF ABBREVIATIONS xi](#_Toc490820587)

[NOMENCLATURE xii](#_Toc490820588)

[1 INTRODUCTION 1](#_Toc490820589)

[1.1 Forecasting methods 1](#_Toc490820590)

[1.1.1 Technical analysis 1](#_Toc490820591)

[1.1.2 Black-Scholes Formula 2](#_Toc490820592)

[2 Literature review 5](#_Toc490820593)

[2.1 Other Solutions 5](#_Toc490820594)

[2.1.1 TinyCast 5](#_Toc490820595)

[2.1.2 CrowdWorx 6](#_Toc490820596)

[2.1.3 Our solution 7](#_Toc490820597)

[3 Methodology 8](#_Toc490820598)

[3.1 Work progress 8](#_Toc490820599)

[3.1.1 Understanding options 8](#_Toc490820600)

[3.1.2 Understanding Black-Scholes formula 15](#_Toc490820601)

[3.2 The Technical Analysis 22](#_Toc490820602)

[3.2.1 Data collection 23](#_Toc490820603)

[3.2.2 Regression line 25](#_Toc490820604)

[3.2.3 Decision making 27](#_Toc490820605)

[3.2.4 Strategy evaluation approach – periods to consider 29](#_Toc490820606)

[4 Results 31](#_Toc490820607)

[4.1 Used technology 31](#_Toc490820608)

[4.1.1 Hardware used the application 31](#_Toc490820609)

[4.1.2 Software used for the application 31](#_Toc490820610)

[4.1.3 External libraries 32](#_Toc490820611)

[4.2 Design pattern to use 33](#_Toc490820612)

[4.3 Project mock-ups 33](#_Toc490820613)

[4.4 Validation of results 36](#_Toc490820614)

[4.4.1 Comparing results with the other external source 36](#_Toc490820615)

[4.4.2 Unit tests 38](#_Toc490820616)

[4.5 Software opinion results 53](#_Toc490820617)

[5 Conclusions and further work 62](#_Toc490820618)

[5.1.1 Further work 62](#_Toc490820619)

[5.1.2 Final conclusions 63](#_Toc490820620)

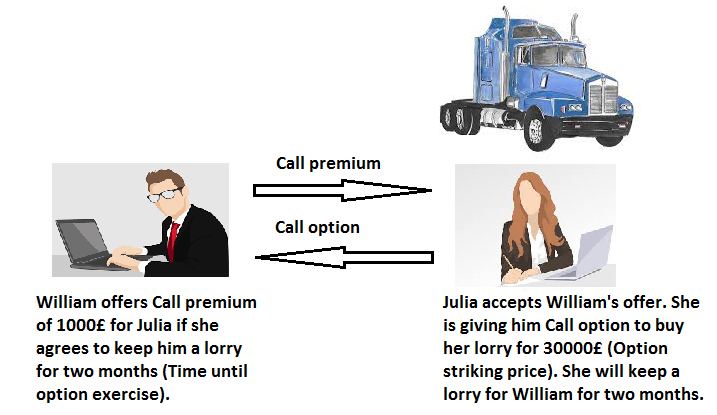
[REFERENCES 64](#_Toc490820621)

[APPENDICES 68](#_Toc490820622)

# LIST OF FIGURES

[Figure 1 Tinycast features from the prodecer website. 6](#_Toc490820623)

[Figure 1 CrowdWorx features from prodecer website. 7](#_Toc490820624)

[Figure 2 Image shows how example Call option works 10](#_Toc490820625)

[Figure 3 Apple current stock price 11](#_Toc490820626)

[Figure 4 Apple Inc. option example illustration 12](#_Toc490820627)

[Figure 5 Put option possible situations 14](#_Toc490820628)

[Figure 6 Example chart of stock price of two companies - Company A and Company B 17](#_Toc490820629)

[Figure 8 Company with a high volatility 22](#_Toc490820630)

[Figure 8 Microsoft share prices from 10.10.2016 to 10.03.2017 with a daily interval of data collection applied. The orange line presents actual stock prices. 23](#_Toc490820631)

[Figure 9 Microsoft stock prices from 10.08.2014 to 10.03.2017 with a monthly interval of data collection applied. The orange line presents actual stock prices. 24](#_Toc490820632)

[Figure 10 Apple stock prices from 11.01.2015 to 10.08.2017 with the monthly interval. The orange line presents actual stock prices, and the yellow line shows the applied regression line. 27](#_Toc490820633)

[Figure 11 Example of decision-making process. The whole prices times series of Microsoft are divided into five chunks (parts). The yellow line presents the General Regression line, and the green line is an example of the local Regression line for the last fifth data prices part. 29](#_Toc490820634)

[Figure 12 The main application menu 34](#_Toc490820635)

[Figure 13 The application mock-up in the Black-Scholes model 35](#_Toc490820636)

[Figure 14 The application mock-up in the Technical analysis mode 36](#_Toc490820637)

[Figure 15 Our application output showing Min-max value indicator applied for the Apple stocks for the period from 11.01.2015 to 10.08.2017 with a monthly interval of prices. 45](#_Toc490820638)

[Figure 16 Our application output showing calculated derivatives applied for the Apple stocks for the period from 11.01.2015 to 10.08.2017 with a monthly interval of prices. 48](#_Toc490820639)

[Figure 17 Our application output showing calculated moving average applied to the Apple stocks for the period from 11.01.2015 to 10.08.2017 with a monthly interval of prices. The Orange line is actual Apple share prices; yellow line is smoothed prices series (moving average). 51](#_Toc490820640)

LIST OF TABLES

[Table 2 PC hardware utilised in the business application project 31](#_Toc490820641)

[Table 3 Software utilised in the business application solution 31](#_Toc490820642)

[Table 4 Architecture diagram of JavaFX 32](#_Toc490820643)

[Table 5 Unit test of Probability density function. Comparing values calculated by our own using MATLAB R2015a and Excel 2016 with our software output 40](#_Toc490820644)

[Table 6 Unit test of the Cumulative distribution function. Comparing values calculated by our software with Cumulative distribution function values from the Standard Normal Distribution Table. 43](#_Toc490820645)

[Table 7 Unit test of the Cumulative distribution function. Comparing values calculated by our software with the Microsoft Excel 2016 NORM.S.DIST() function with the true Cumulative flag. 43](#_Toc490820646)

[Table 8 Unit test of Call premium. Comparing values calculated by our software with the Microsoft Excel 2016 output computed using Excel functions such as SQRT(), LN() or NORM.S.DIST(), function with the true Cumulative flag. 44](#_Toc490820647)

[Table 9 Unit test of the minimum value in the selected vector function. Comparing index number of minimum value evaluated by our software with the minimum index assessed by a human (author). 46](#_Toc490820648)

[Table 10 Unit test of the minimum value in the selected vector function. Comparing index number of minimum value evaluated by our software with the minimum index assessed by a human (author). 47](#_Toc490820649)

[Table 11 Unit test of the first derivative function. Comparing values calculated by our software with the results from Microsoft Excel 2016 with applied (4‑5) equation. 49](#_Toc490820650)

[Table 12 Unit test of the second derivative function. Comparing values calculated by our software with the results from Microsoft Excel 2016 with applied (4‑7) equation. 50](#_Toc490820651)

[Table 13 Values calculated using three-period Moving average method (4‑9). 52](#_Toc490820652)

[Table 14 The application outputs considering the set of 51 (49, since 2 of them are not considered - NC) U.S. Companies. The table shows Potential revenue, Opinion of about potential market movement (BUY, HOLD, SELL), and Opinion accuracy calculated by the thesis application. Considered Short-term period on which the analysis was performed from 10.10.2016 to 10.03.2017 and prices data are loaded in the daily interval. 54](#_Toc490820653)

[Table 15 The application outputs considering the set of 53 U.S. Companies (50, since 3 of them are not considered - NC). The table shows Potential revenue, Opinion about potential market movement (BUY, HOLD, SELL), and opinion accuracy calculated by the thesis application. Considered period on which the analysis was performed from 10.08.2014 to 10.08.2016 and prices data are loaded in the monthly interval. 57](#_Toc490820654)

[Table 16 The application outputs considering the set of 53 U.S. Companies (50, since 3 of them are not considered - NC). The table shows Potential revenue, Opinion about potential market movement (BUY, HOLD, SELL), and opinion accuracy calculated by the thesis application. Considered period on which the analysis was performed from 10.08.2011 to 10.08.2016 and prices data are loaded in the monthly interval. 59](#_Toc490820655)

LIST OF EQUATIONS

[(1‑1) 3](#_Toc490820656)

[(1‑2) 3](#_Toc490820657)

[(1‑3) 3](#_Toc490820658)

[(3‑1) 15](#_Toc490820659)

[(3‑2) 15](#_Toc490820660)

[(3‑3) 15](#_Toc490820661)

[(3‑4) 18](#_Toc490820662)

[(3‑5) 19](#_Toc490820663)

[(3‑6) 19](#_Toc490820664)

[(3‑7) 19](#_Toc490820665)

[(3‑8) 19](#_Toc490820666)

[(3‑9) 25](#_Toc490820667)

[(3‑10) 25](#_Toc490820668)

[(3‑11) 26](#_Toc490820669)

[(4‑1) 39](#_Toc490820670)

[(4‑2) 39](#_Toc490820671)

[(4‑3) 40](#_Toc490820672)

[(4‑4) 41](#_Toc490820673)

[(4‑5) 48](#_Toc490820674)

[(4‑6) 49](#_Toc490820675)

[(4‑7) 49](#_Toc490820676)

[(4‑8) 51](#_Toc490820677)

[(4‑9) 51](#_Toc490820678)

[(4‑10) 52](#_Toc490820679)

[(4‑10) 52](#_Toc490820680)

[(4‑12) 54](#_Toc490820681)

LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| IT | Information Technology |
|  |  |
|  |  |
|  |  |

NOMENCLATURE

= Call premium (European call option)

= Current stock price

= Time until option exercise

= Option striking price (Exercise price)

= Risk-free interest rate

= Cumulative standard normal distribution

= Exponential term

= Standard deviation (Volatility)

= Natural logarithm

– Dependent variable

– Slope of the line

–Constant term of y-intercept. Point where the y-axis is intersected with the line

– Independent variable

– Slope of the line

– The mean of

– The mean of

Correlation between and

Standard deviation of

Standard deviation of

# INTRODUCTION

Determining the future value of the financial instrument or the company price is challenging, often impossible task. However, to simplify that process many economic, mathematical and statistic tools have been developed [9].

Techniques of financial forecasting are still under development, and the fast growth of the computer industry facilitates their efficient implementation [2]. Financial institutions have been using this kind of tools for decades in order to forecast asset prices [1]. Techniques such as:

* Technical analysis [1]
* Fundamental analysis [14]

are commonly used to evaluate future asset prices in the industry [9].

Option pricing issue has drastically changed since *The Pricing of Options and Corporate Liabilities* [1] thatwork was introduced by Fisher Black & Myron in 1973. However, that work was later improved by Robert *C.* Merton [6, 7] in order to be more usable in real life. This was the Thanks to their paper option pricing has reached entirely new level. This was the first time when option pricing was formalised through the means of the mathematical model [12] until then much decisions were based on intuition and experience.

For decades people were trading options, they were buying and selling them, but there wasn’t exists accurate mathematical model to estimate option price. Black-Scholes formula puts information how to value options into mathematical model [5, 6, 7].

## Forecasting methods

### Technical analysis

In the late 1800s, Charles Dow introduced a new approach to financial instruments price prediction, but not many professional investors believed in the effectiveness of that method [3].

The technical analysis is using statistical analysis of market activity such as volume and price [10]. It takes into account internal stock market data. Using technical analysis, we can evaluate possible future stock market evolution. It is used for short-term forecasting of the share price[9]. Using past prices to predict future prices may be valuable. That analysis is based on three fundamental assumptions [10]:

* The market discounts everything
* Price moves in trends
* History trends repeat itself.

For a long time, there has been a disagreement between academia and industry of the value of technical analysis. The industry knew empirically that this technique was working. However, academics were not convinced. Some circles even named the technical analysis as “voodoo finance” [2].

The technical analysis can be used as well for evaluation of Foreign Exchange market profitability using a Genetic programming approach [3]. It was also shown that particular application of that method could give statistically significant forecast power for Dow Jones Industrial Average changes over a long period of time [4].

### Black-Scholes Formula

Probably the most famous formula in the field of finance is the also called Black–Scholes–Merton formula. Myron Scholes and Bob Merton were awarded Nobel prize for their work [5]; Fisher Black died before the award was given [11]. Robert C. Merton in his work [6, 7] improved the Black-Scholes Formula by reformulating it in the way that it is more usable. Nowadays the current interpretations are based on Merton’s work.

Thanks to that revolutionary formula option pricing prediction problem has significantly changed, and a simplified approach to that process was introduced in Cox, J. C., Ross, S. A., & Rubinstein work [8].

Despite model restricted assumptions like: Dividends are not paid out during the option life, no transaction cost or option has to be European and must be exercised at expiration, that approach is still widely used [13]. Thanks to that formula theoretical value of the European-style options can be forecasted.

Black-Scholes formula is expressed by the following equation [12]:

|  |  |
| --- | --- |
|  | (1‑1) |
|  | (1‑2) |
|  | (1‑3) |

where,

= Call premium (European call option)

= Current stock price

= Time until option exercise

= Option striking price (Exercise price)

= Risk-free interest rate

= Cumulative standard normal distribution

= Exponential term

= Standard deviation (Volatility)

= Natural logarithm

# Literature review

It was crucial to understanding a stock market behaviour especially the part related to financial instruments price prediction. We have chosen a set of high-quality articles especially related to the Black-Scholes model and the Technical analysis.

Reading articles was invaluable hence understanding that fascinating financial part of initial research and getting acquainted with the topic of our thesis.

## Other Solutions

All other solutions are not easily accessible, the ones we found were only paid. We can only provide the producer's descriptions.

### TinyCast

TinyCast is one of the premium software. We were not be able to test it in the real life we can only provide information from producer website.

In general Tinycast appears to be based more on the crowd (other users) opinion than on the mathematical algorithms.

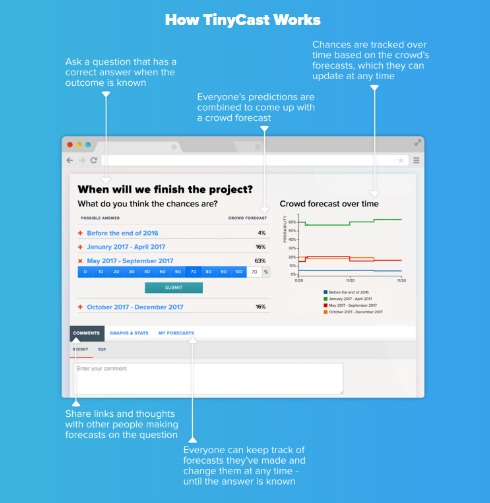


Figure 1 Tinycast features from the prodecer website.

### CrowdWorx

The second software like the previous one is not explicitly described. We can only suppose that like in the first case decision making is based on crowd – users decisions.



Figure 2 CrowdWorx features from prodecer website.

### Our solution

We exclude human factor totally in our software. We are focused only on the data from the past. We are evaluating program output (opinion) downloading price times series and we are performing Mathematical and Statistical methods on those data in order to make some decision in the output (BUY, HOLD, SELL).

# Methodology

## Work progress

In this section, we will discuss the progress of our solution. We will give detailed information about the thesis and the application creation process.

### Understanding options

Thinking about Options is different from the standard way of considering typical market financial instruments. The way stocks, bonds or deposits work is entirely different from Options concept. We are not a financial specialist, so most of our explanations will be simplified in order to understand how the option works and later on use that knowledge into our software.

To start with Options, we will cover basic description and explain some nomenclature. At this point we should look at simplified option definition:

*The option is a* ***right*** *to buy or sell some underlying asset for specified price on or before specific date [18]. What is important to claryfy is that with options as the word suggests, we are given the opportunity to buy them without being obliged to do so.*

What is an underlying asset? Is just a financial instrument on which option is based, It could be stock, currency or futures [17].

To sum up when we talk about options we need to be aware that we are buying a right to further buying or selling the underlying asset.

The next concept that we need to know are types of options. We distinguish two types of options:

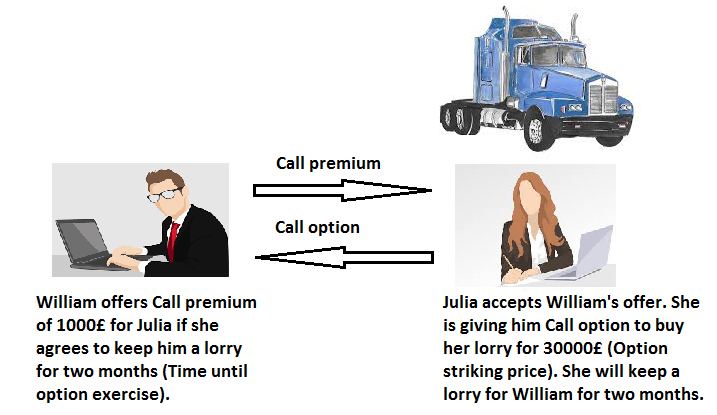
* **Call option** – is the right to buy some underlying asset for specified price
* **Put option** - is the right to sell some underlying asset for specified price

Those definitions as it was mentioned before are simplified in order to explain that issue as easy as possible [18, 19]. We will present some examples to show how options work. Our explanation is based on straightforward explanation presented on video [20]. As symbols we use were defined in nomenclature

Firstly we will explain Call options.

In our scenario, we have two people (parties). One is William who wants to buy a lorry and Julia who tries to sell it. Julia put the advisement on the internet for selling the truck for 28000£, that price is called the current stock price . William works in Construction industry and he needs more transport equipment for his investment, for that reason he really needs Julia’s lorry. He believes that price of that kind of equipment will go up within next months, but he doesn’t have money at the specific moment – 1st of June. In that case he asks Julia if she can keep for him that truck and take the advertisement of Internet for two months. During that period he believes he would be able to gain needed amount of money to buy lorry. He suggest to Julia that he will buy her lorry for 30000£ after two months.

Using the terminology that refers to options, the price of 30000£ that Julia agreed to sell the truck after two months is called Option striking price or Exercise price . William offers Julia 1000£ immediately for keeping the lorry for him for period of 2 months, that period is called the time to expiration . Julia agrees for William’s offer, she gave him the Call option, that situation is described on Figure 2.

Figure 3 Image shows how example Call option works[[1]](#footnote-1)

However, Construction industry market is unpredictable, and price of the lorry can drastically change within a month. If there will be demand for such kind of equipment, the price of the lorry could rise to 40000£. In a scenario in which almost no one is interested in buying lorries, its price could drop down to 20000£. The price of the truck can remain stable, and after two months I could cost the initial price 30000£. In all those scenarios Julia will earn 1000£ of call premium.

However, what could happen in above the scenarios after two months? We assume that William was able to earn money needed to buy lorry (30000£). When the market price of lorry rises to 40000£ William will buy lorry for the Exercise price 30000£ and he will have equipment worth 40000£, so 10000£ more than he paid. When price of lorry didn’t change William can buy lorry then he will loose only 1000£ of call premium that he paid for Julia. In the last scenario when price drop to 20000£ William have alternative to reject buying the lorry from Julia.

At this point, there is one essential fact to be aware, in call options buyer in our case William has a right to buy the lorry but not an obligation. In all those scenarios he can refuse to purchase a truck. Julia as the seller has a duty to sell lorry at the exercise price if buyer wants to – in that case we say that buyer exercises the option.

One more fact to mention is Julia’s possibility of changing call premium value. If there is more than one potential customer for instance apart from William also investor Greg and Christian are interested in buying Julia’s option, she can rise call premium price to 2500£.

At this stage, we should consider a more practical example that is based on a real option.

On 16 June 2017 the current stock price of Apple Inc. is $142.27. We are basing on data from <http://www.nasdaq.com>.

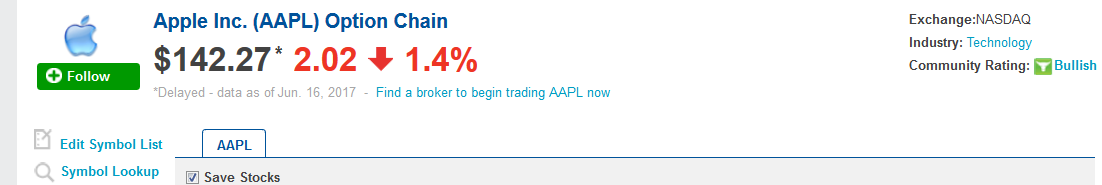


Figure 4 Apple current stock price[[2]](#footnote-2)

Investor assumes that Apple Inc. option price is going to increase to $155. In that case, he buys $150 call option - the exercise price . Call premium for that option is $1.

We are considering a scenario in which investor had good intuition and option rose to $155 at the time to expiration . In that situation investor can buy that option for $150 even though the current stock price for that option is $5 higher. Investor earned $5 - $1 of Call premium value, so he has $4 of profit. In a second scenario investor was wrong and option price decrease to level of $130 investor loses one dollar but he is not obligated to buy that option. That situation is illustrated on Figure 5.



Figure 5 Apple Inc. option example illustration[[3]](#footnote-3)

We are going to present an easy explanation of Put options.

Hence put options are not considered in the Black-Scholes formula which is an intrinsic part of our theis we will only describe a short example to clarify put options.

Referring to the intuitive example from [20] we are going to demonstrate Put options relating to the insurance company.

William bought brand new robot vacuums for 900£, and he is worried about its potential minor damage or for the unexpected situation when the robot will be broken. To be sure about vacuum condition William is asking Julia for insurance – Put option for the full price of vacuum 900£, so that is Exercise price (striking price). She agrees for selling him an insurance for 50£ (Call premium) for two years which is the time to expiration .

Similar to Call option William’s vacuum during that year could be fully functional without any destruction. In this scenario, William pays Julia 50£ of call premium for Put option (insurance).

Next possibility is a situation where vacuum was damaged, but it is possible to fix it. The cost of repair is 150£. Julia covers all repair expenses. Hence it was a part of the contract.

The last situation is when robot vacuum is entirely damaged. In that case, Julia also has to cover all expenses needed to replace or buy a new one. We assume that the cost of new vacuum is the same as its initial price, i.e. 900£.

We visualised all situations on Figure 6.

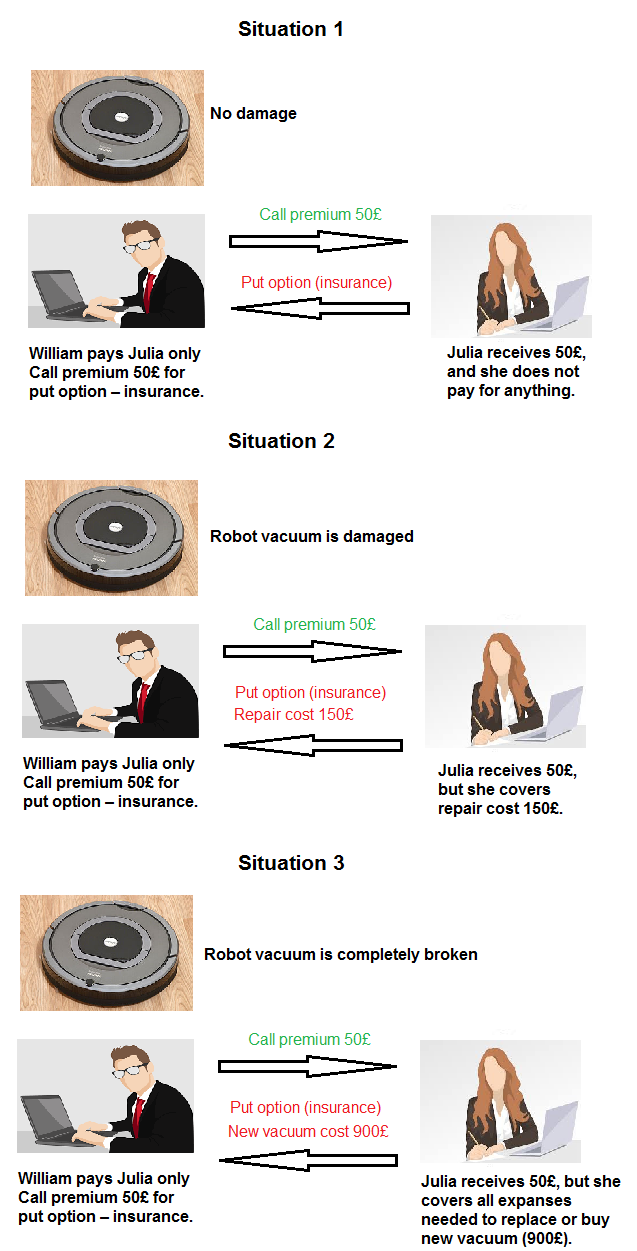


Figure 6 Put option possible situations[[4]](#footnote-4)

### Understanding Black-Scholes formula

To understand that model an introduction to the Black-Scholes formula from Khan Academy [11] was highly helpful. We are going to explain the Black-Scholes formula as detailed and easy to understand as possible.

Now we are going to recall the Black-Scholes formula for a European call option. There is a significant difference between a European call option and an American one. This is a possibility to exercise a European call option at only one time – on the exercise date, by contrast, we can exercise American call option at any point.

The European call option in Black-Scholes formula is expressed in the following formulation:

|  |  |
| --- | --- |
|  | (3‑1) |

|  |  |
| --- | --- |
|  | (3‑2) |

|  |  |
| --- | --- |
|  | (3‑3) |

As those symbols are defined in nomenclature.

We should now consider factors with significant impact on stock option pricing. Firstly the stock option price is obviously is paramount important. Another factor according is the exercise price according to investopedia.com[[5]](#footnote-5) exercise price is:

*“The exercise price is the price at which an underlying security can be purchased (call option) or sold (put option).”*

Investors also care about the risk-free interest rate . This is the theoretical rate of return without any loss in a selected period of time.

We need to think for how long we need to exercise selected option. That siltation is described with a parameter called time until option exercise or just the time to expiration .

The last parameter the Standard deviation is seemingly the most unclear when look at Black-Scholes formula for the first time. We need to know how volatile the stock is. Volatility is described as a standard deviation of log returns for selected security [11].

To make that parameter more clear, we are going to show an example chart in Figure 7.

Figure 7 Example chart of stock price of two companies - Company A and Company B

As we can see the price of Company A is changing more dynamically than the one of Company B., The price differences of Company A in the period under consideration are significantly higher compared to Company B., The prices of Company A are more dispersed. Hence we say that Company B is less volatile.

The volatility of Company A is much bigger than that of Company B, so this means that the standard deviation of Company A is higher > . We can also say that Company A could generate us higher profit (income) or loss. Higher volatility makes options more interesting when it comes to generate higher potential profit. Later we will discuss about volatility impact in the formula.

Now we will explain the function. is nothing more than the cumulative distribution function, which in our case is the standard normal distribution [21]. Next parameter is whichmeans discounted back exercise price .

So overall we have Current stock price determined by a some probability , hence and later we subtracting it from discounted exercise price multiplied by probability with different argument .

We can split the equation (3‑1) into two parts:

- is what we get

- is what we pay

The call option value is the Stock price subtracted from discounted back exercise price.

Next, we want to focus on parameter . In the ratio the current stock price to striking price (exercise price) we can see that higher values of or lower values of will result in a higher ratio value. When that ratio has a high value the value of is higher so after applying that parameter to equation (3‑2) we wil get higher probalities and .

The result in that situation will be the higher chance of exercising that price.

Before we discuss the next factor that has significant influence for the European call options , we will derive equation (3‑3). Deriving that equation will give us more inside on its behaviour.

We substitute for equation described in (3‑2):

|  |  |
| --- | --- |
|  | (3‑4) |

Subtracting with a common denominator:

|  |  |
| --- | --- |
|  | (3‑5) |
|  | (3‑6) |

Putting outside the bracket:

|  |  |
| --- | --- |
|  | (3‑7) |

Finally, we have following equation for :

|  |  |
| --- | --- |
|  | (3‑8) |

We observed the Standard deviation has strong impact for the Black-Scholes equation in both formulations presented in (3‑2) and (3‑8). The impact of in in numerator is quadratic. So a higher value of will result overall in a higher value of as result, what is equivalent to a higher of probability and finally get higher value of European call options .

On the other hand, in we subtract in numerator: . This means that in the second part of the European call option equation we will a get lower value of that is what we are paying, as long as the value of is gets higher.

So overall having greater value in the Standard deviation will result in the higher values of European call options and we can also think about it as the higher the volatility is, the higher values of European call options we get.

We are going to discuss Black-Scholes formula in practical usage. As it is described in [15] to use that formula we need to have the stock option price which is easy to get, the exercise price is part of a contract, there are many sourcs for the risk-free interest rate , also information like government debt allows to estimate that value. Next value Is the time to expiration which is also given we know when option expires and current date. The last parameter the Standard deviation is not as obvious as previous parameters to estimate. The only way to do that is refer to the history. Only historical prices of the Standard deviation could give us some necessary data to estimate that. Usually we look at historical value of the Standard deviation over some period of time where its value hasn’t changed drastically and we, can use that value in the Black-Scholes formula. Despite that we need to remember that it would be the estimated value.

Potential investors should be are aware that options trading is a continuous process, so they should look at the European call option price and check actual trade value. It could be for instance 10€. When we are looking for this actual price of 10€ that means market believes this is a proper price for that option. We can say that this value is evaluated by actual market transactions.

At this point we observe that most data are given, we know stock option price , the exercise price , the risk-free interest rate , the time to expiration and the price of the European call option . Knowing all this values allows us to use Black-Scholes formula in order to find of the Standard deviation , we transform Black-Scholes formula in a way where is unknown variable. We need to look at actual options pricing, when we know that we can evaluate what volatility the market expects to be. Assuming that the current European call option price is 10€ and we know all other variables apart from the Standard deviation d we can easily evaluate what volatility market expects to be.

When the market is getting more unpredictable and unstable like on Figure 8, it means that the current European call option price is higher, we will pay more, what is equivalent that the Standard deviation or volatility is higher. That kind of volatility is called Implied volatility.

Definition of Implied volatility from [16] is:

“Implied volatility is the estimated volatility of a security's price. (..) Implied volatility is a way of estimating the future fluctuations of a security's worth based on certain predictive factors.”

Figure 8 Company with a high volatility

We cannot assume that the Standard deviation will be given straightforwardly. We are using all those values to calculate the price for the European call options **.**

## The Technical Analysis

We are basing in this thesis only on the prices from the past and analysing those prices; we are trying to forecast future movement applying mathematical and statistical methods in our software. To achieve those aims method like the Technical analysis should be very sufficient.

The Technical analysis[[6]](#footnote-6) is the method that helps the investor to make the decision whenever he wants to buy, sell or hold stocks. With the Technical analysis, the historical prices are considered in opposition to the Fundamental analysis in which we consider factors like company financial statement, management efficiency or macroeconomic.

### Data collection

To collect the stocks data we use data from Quandl[[7]](#footnote-7) website using quandl4j [[8]](#footnote-8) API to collect the stocks data (prices for the selected date range).

Our application allows searching through Quandl WIKI database which offers stock prices dividends and splits for 3000 U.S. Companies.

In our software to see the selected stock prices, you can simply put the stock symbol into Search engine press Enter or click Search. The application will automatically show the selected company share prices in the chosen time range. For data Displaying we use JavaFX LineChart.

Below presented example output for the Microsoft Corporation (stock symbol MSFT).

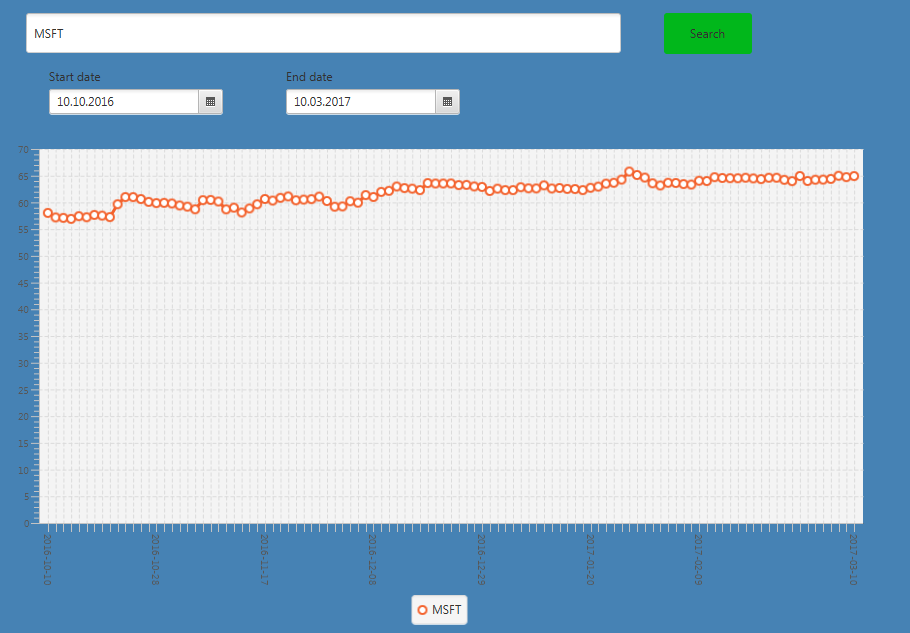


Figure 9 Microsoft share prices from 10.10.2016 to 10.03.2017 with a daily interval of data collection applied. The orange line presents actual stock prices.

Moreover, Microsoft share prices in our application with applied Monthly data collection interval looks as on below example:

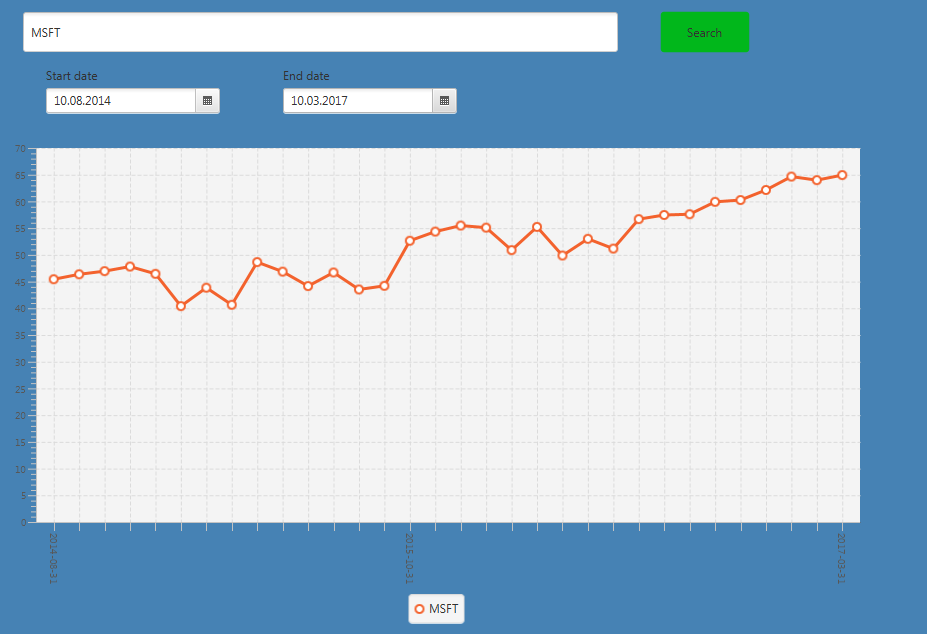


Figure 10 Microsoft stock prices from 10.08.2014 to 10.03.2017 with a monthly interval of data collection applied. The orange line presents actual stock prices.

On above examples presented in Figure 9 and Figure 10, we can see that even on Figure 9 selected period is much shorter, than on the next figure we have much more prices points on the first chart. Daily interval in our results will apply only for Short-time periods (maximum six months) for the longer periods that kind of data collection would be redundant and cost additional computational resources. For analysis period over half year interval is sufficient enough to perform trend and make prices times series analysis.

### Regression line

In the thesis application, we use the method called the Regression line. It is beneficial when it comes to evaluating the trend, how the market behaves, do we have uptrend[[9]](#footnote-9) or downtrend[[10]](#footnote-10)?

The regression line is just a line showing trend of the selected data. The standard formula for the Regression line is as follow[[11]](#footnote-11):

|  |  |
| --- | --- |
|  | (3‑9) |

Where,

– Dependent variable

– Slope of the line

–Constant term of y-intercept. Point where the y-axis is intersected with the line

– Independent variable

To produce a Regression line we need to calculate and.

For the evaluation the following formula is applied[[12]](#footnote-12):

|  |  |
| --- | --- |
|  | (3‑10) |

Where,

Correlation between and

Standard deviation of

Standard deviation of

can be calculated using the following formula12:

|  |  |
| --- | --- |
|  | (3‑11) |

Where,

– Slope of the line

– The mean of

– The mean of

We will present the couple example to visualise how the Regression line looks applied for the real-life stocks prices.

First case Apple stocks prices are shown with applied regression Line.

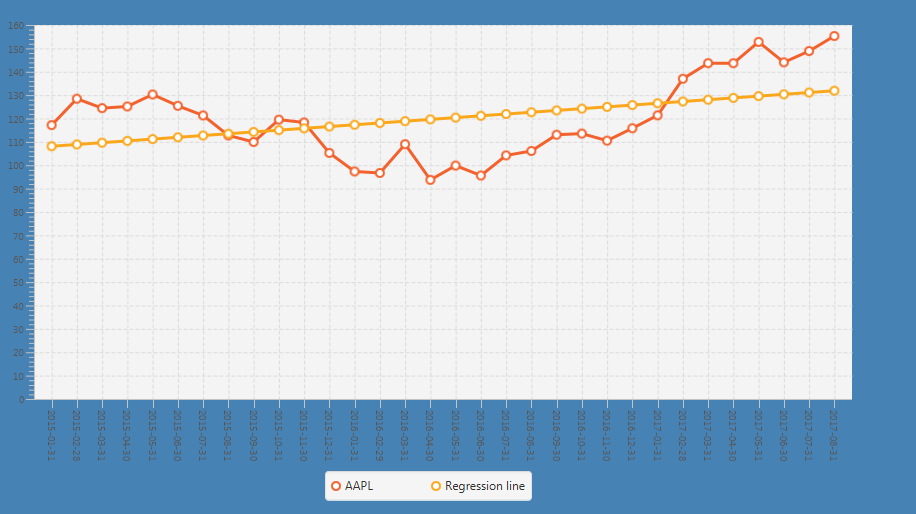


Figure 11 Apple stock prices from 11.01.2015 to 10.08.2017 with the monthly interval. The orange line presents actual stock prices, and the yellow line shows the applied regression line.

We can see that between 11.01.2015 and 10.08.2017 there is uptrend on the Apple shares. In this case, value is positive, and it takes the value of 0.7662050953079177.

We are calculating Regression line using Apache Common math library[[13]](#footnote-13) and the SimpleRegression class.

### Decision making

In real decision making even if we have various indicators they still cannot make the decision instead of us. We have to make it on our own based on our intuition, knowledge, experience or pure speculation.

The application that we created decides for us. It is basing on the Regression line method and some trend assumption.

1. We are dividing the whole downloaded from Quandl selected company prices data series on five chunks (parts).
2. We are calculating Regression line for all collected prices – General Regression line
3. For the last fifth chunk we are implicitly calculating local Regression line and see what the trend in the last period is.
4. Having those two trend lines (indicators), we are choosing one of 4 possible decisions:
   1. Local Regression line and General Regression are > 0. There is an uptrend.

The application output opinion for that stock: BUY

* 1. Local Regression line > 0 and General Regression are < 0. There is downtrend, but the local Regression line gave a positive signal. The application output opinion for that stock: BUY
  2. Local Regression line and General Regression are < 0. There is downtrend for both indicators; there are no signals for buying stocks. The application output opinion for that stock: SELL
  3. Local Regression line < 0 and General Regression are > 0. The last chunk of considered period has descending trend, but based on those data it seems to that Long-term trend is growing.

The application output opinion for that stock: HOLD

The example presented on Figure 12 will explain the whole opition graphically making process.

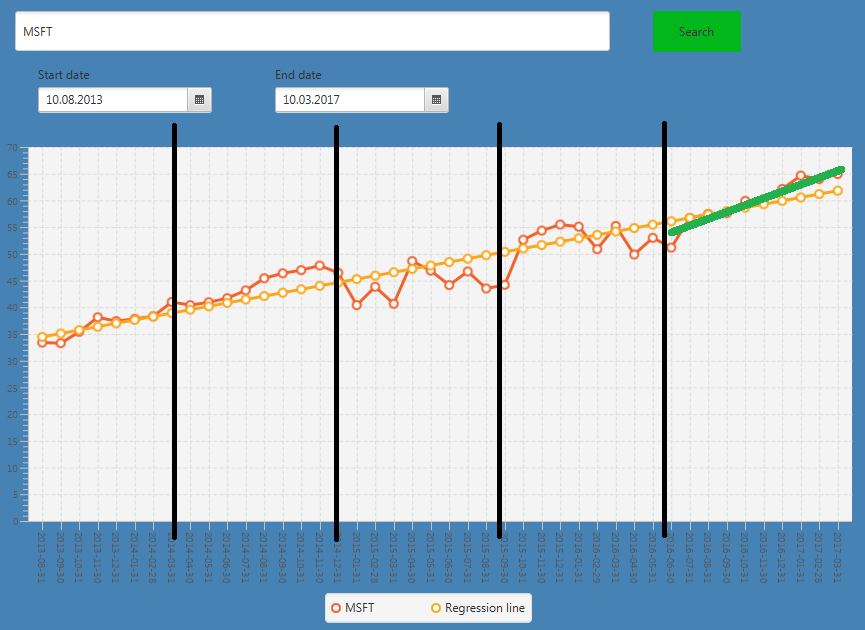


Figure 12 Example of decision-making process. The whole prices times series of Microsoft are divided into five chunks (parts). The yellow line presents the General Regression line, and the green line is an example of the local Regression line for the last fifth data prices part.

### Strategy evaluation approach – periods to consider

The crucial part of our work was to evaluate our strategy. We analysed many companies prices, and we consider three investment periods:

* Short-term – up to six months
* Mid-term – up to two years
* Long-term – over two years

The assumption is what would be our revenue presently (considered date 10.08.2017) if we would follow the software opinion and buy, sell or keep our stocks. We are considering three analysis period, analogical to the investments periods. We have three analysis periods Short (maximum six months), Mid (maximum two years), and Long-term (over two years).

If we are considering analysis period from 10.10.2016 to 10.03.2016 (5 months), that means we are performing our evaluation on Short-term analysis period. We will cover the tests for all three analysis period Short, Mid, and the Long-term.

The main assumption is that the investor is buying each stock for the same money amount. For instance, if we have 10000£ to spend and the program output gave us the opinion of buying ten stock companies that mean we are buying the stocks of each company for exactly 1000£. That assumption simplifies the calculations and for a bigger amount of money (like 100000£) will not have significant influence on the given results.

The next assumption is stocks purchasing date and the day of profit/loss evaluation, which for each considered period is always **10.08.2017**. For that day we are validating our results. Days of operation making on stocks (Buy, Hold, Sell) are

* Short-term is 10.03.2017
* Mid-term is 10.08.2016
* Long-term is 10.08.2016

Moreover, the Analysis periods are respectively

* Short-term from 10.10.2016 to 10.03.2017 (5 months)
* Mid-term 10.08.2014 to 10.08.2016 (2 years)
* Long-term 10.08.2011 to 10.08.2016 (5 years)

# Results

In this section, we will discuss in detail about our software implementation. I

## Used technology

Our solution of the particular business app is developed in Java programming language. We consider creating the application with a graphical interface in order to make it easy to use for the user. The Application should be responsive, intuitive and provide proper results. By decent results, we mean results that should be mathematically correct and output should not be incorrect because the mistake that was done in a software itself.

### Hardware used the application

We are using the Personal computer (PC) with the following hardware:

Table 1 PC hardware utilised in the business application project

|  |  |
| --- | --- |
| **Type of hardware** | **Description** |
| Processor | AMD Phenom II X4 955 QuadCore 3.2GHz |
| RAM | 4096 MB DDR3 |
| GPU | Nvida GeForce GT 730 1GB |
| Hard drive | SSD 256 GB |

According to our predictions, this PC should be powerful enough to perform all calculations fast and efficiently.

### Software used for the application

In our project, we will use apart from Java language we will use JavaFX technology.

Results in our thesis will be obtained by using the following software:

Table 2 Software utilised in the business application solution

|  |  |
| --- | --- |
| **Type of software** | **Description** |
| Operating system | Windows 7 64 bit |
| Integrated Development Environment (IDE) | IntelliJ IDEA Ultimate 2017.1 |
| Editor | Microsoft Word 2016 |
| Charts maker | Microsoft Excel 2016 |

### External libraries

We are going to use various external libraries in our application.

A first external library that we are going to us is the JavaFX. This technology is defined as a set of Java libraries[[14]](#footnote-14). The JavaFX enables developers for creating the graphical interface for an application which is being designed. This set of libraries was formed to provide to clients graphical applications that are consistent across different platforms.

To be more acquainted with that technology we will show the Architecture diagram of JavaFX.



Table 3 Architecture diagram of JavaFX[[15]](#footnote-15)

The Next external software that helps in designing an application is Scene Builder, which will be used during projects design process. Scene Builder is the graphical layout tool which allows the developer to set his software components graphically. Oracle stopped to develop this tool[[16]](#footnote-16), but fortunately, it is still under development thanks to Gluon team[[17]](#footnote-17). Gluon Scene Builder version will be used for this thesis purposes.

## Design pattern to use

In this subsection, we are going to discuss our application development approach.

We decided to create a graphical application which allows the user to input parameters and gives back as an output one possible recommendation buy, hold, sell. Taking into account the fact that our solution is based on JavaFX it is almost certain that the application will be developed using the Model/View/Controller Design Pattern [22]. Even a new project in IntelliJ IDEA IDE gives a template with automatically generated Controller class.

## Project mock-ups

We created the application’s mock-ups to have a design prototype that would be implemented later in the final JavaFX project.

Software used for the mock-ups purpose is Balsamiq Mockups 3.5.13[[18]](#footnote-18).

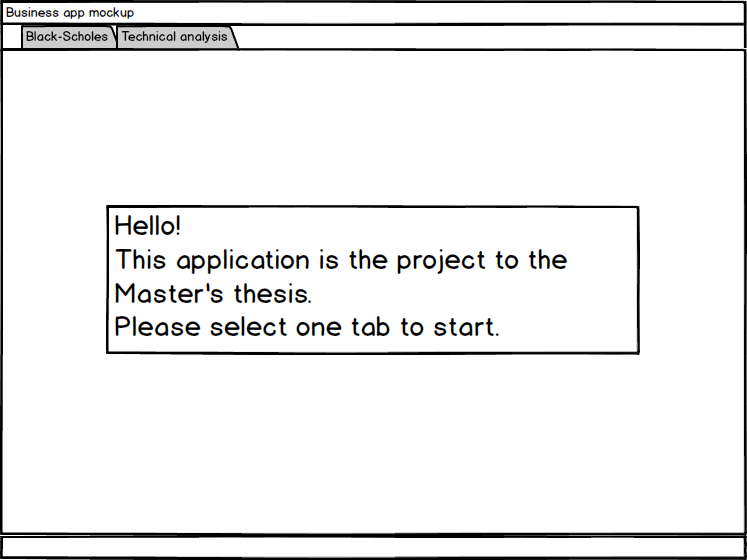


Figure 13 The main application menu

The application will be divided into two sections Results for the Black-Scholes equation and the Technical analysis. After selection of one bookmark, we will be directed into specific application mode.

The first is Black-Scholes which is presented below.

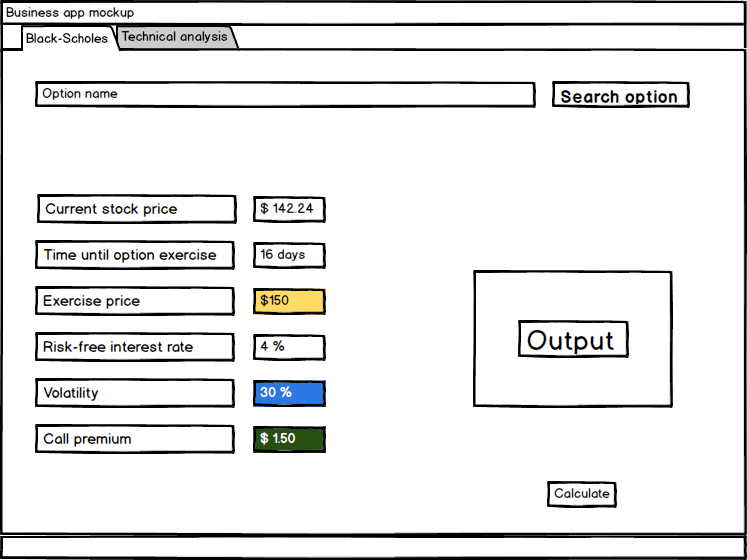


Figure 14 The application mock-up in the Black-Scholes model

In this mode, the user can select an option by its name from a database. Once it is selected, all fields will be automatically filled. However, the purpose of that simulation is to give the output to the user if it is recommended to buy particular option, hold it or sell. The user is also allowed to modify all parameters he wants to.

The most significant values for an investor to know is Exercise price, volatility and call premium, all these values are highlighted in Figure 14. If the user will leave one parameter empty an application consider it as a parameter to find and it will calculate it according to the Black-Scholes formula.

The next tab is the Technical analysis feature.

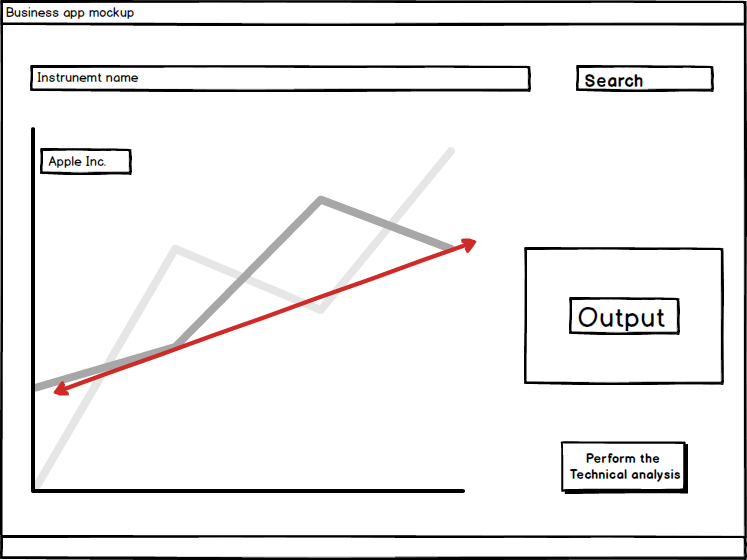


Figure 15 The application mock-up in the Technical analysis mode

Similar as in the Black-Scholes model we are getting Financial instrument name from a database, and we get as an output its price chart. In the Figure 15 example, we can see a red line which stands for the Technical analysis analytic line. After clicking the “Perform the Technical Analysis button” we are going to see analysis lines (like the red line from the Figure 15, and also output as a text saying if selected option is worth to buying, selling or holding.

## Validation of results

### Comparing results with the other external source

Results are consistent with other websites. We cannot refer to any book, but we can compare our results with the other websites.

In the first order we will show some comparison with the another solution done by

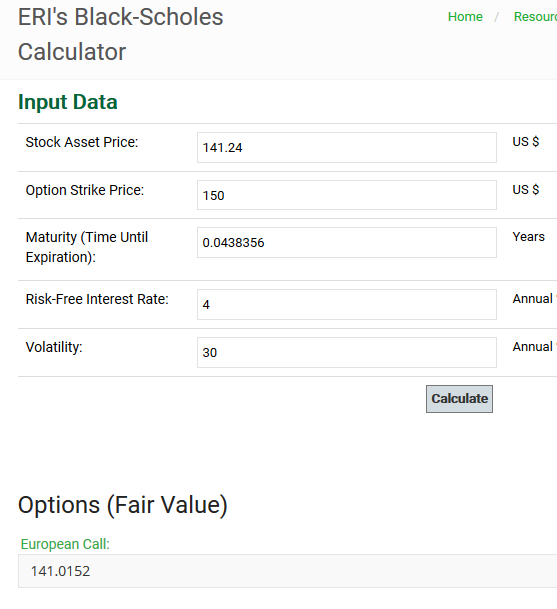
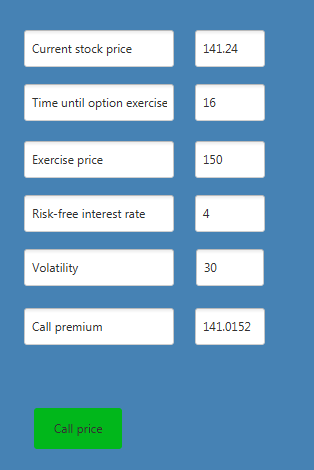
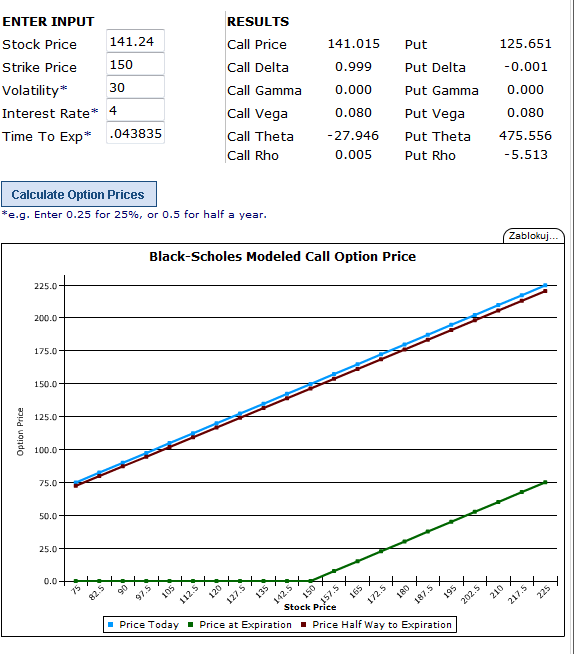


Figure Comparing results from external website with the thesis application



Our solution gives the same results as on applications mentioned above. Nevertheless, we cannot rely on some other works which have no reviews. To strengthen our assumption of the correctness of the results we will perform the test, especially Unit tests.

**PART B**

### Unit tests

Unit tests are carried out in our work using the JUnit[[19]](#footnote-19) testing framework. To set the testing framework in the IntelliJ IDEA Ultimate 2017.1 IDE, we decided to use Apache Maven comprehension tool[[20]](#footnote-20). Thanks to Maven configuring additional tools is very easy, and it is only needed apart from set the project to use maven just add selected dependencies to the Maven pom.xml file.

Also, the other tools are retrieved with maven in our project like The Apache Common Mathematics Library[[21]](#footnote-21) or Quandl for Java API (quandl4j)[[22]](#footnote-22).

Unit tests were performed using JUnit functions like assertArrayEquals(expectedArray[], actualArray[], delta) which takes as the third parameter delta.

Delta is the acceptable difference between expected and actual values. The Round-off error causes the difference. For instance expected value equals 12.76 could be expressed in the output as 12.666666666666666.

In thetests, we took fair delta value of 0.01.

#### Black-Scholes Unit tests - Probability density function (PDF) test

Black-Scholes formula uses Normal distribution in its calculations. In the implementation, we used[[23]](#footnote-23) we are taking the double value to the function that calculates Normal distribution result based on probability density function[[24]](#footnote-24).

|  |  |
| --- | --- |
|  | (4‑1) |

In standard Normal distribution =1 and = 0, so N(,) is in our case N(0,1).

That gives us

|  |  |
| --- | --- |
|  | (4‑2) |

So ultimately probability density for N(0,1) is following

|  |  |
| --- | --- |
|  | (4‑3) |

The formula from the equation (4‑3) is applied in our solution. We need to test if that method is implemented correctly and given valid results.

In order to do that we will calculate formula (4‑3) in the Microsoft Excel 2016 and to strengthen our calculations accuracy as well in MATLAB R2015a for a selected values and make the test which will validate if applied by us solution returns the same results as the ones calculated by us.

Code snippet for PDF calculation in that we performed in MATLAB

fun = @(x) (1/sqrt(2\*pi) ).\* ( exp((-0.5\*(x).^2)) );

probalityDensity = zeros(size(testVector));

%Calculates Gaussian Probability density function

for i=1:length(testVector)

probalityDensity(i) = fun(testVector{i})

end

Table 4 Unit test of Probability density function. Comparing values calculated by our own using MATLAB R2015a and Excel 2016 with our software output

|  |  |
| --- | --- |
| **Probability density function Unit test** | |
| **Values type** | **Values** |
| Values to test | 10.2,1.12,2.32, 2.1,0.23,0.31,0.12,0.93,0.45-17.22,-11.09,-0.23,-0.43 |
| Expected Probability Density Values calculated in Excel 2016 | 1.02073E-23,0.213069147,0.0270481, 0.043983596,0.388528585,0.380226355,  0.396080212,0.258880547,0.360526962,  1.624E-65,7.84122E-28,0.388528585, 0.3637136 |
| Expected Probability Density Values calculated in MATLAB R2015a | 1.02073055943061e-23,0.213069146775718, 0.02704809954688180.0439835959804272, 0.388528585315836,0.380226354751325, 0.396080211793656,0.258880546731149, 0.360526962461648,1.62400043794223e-65, 7.84121714679792e-28,0.388528585315836, 0.363713600373713 |
| **Test Result** | **Passed** |

#### Black-Scholes Unit tests - Cumulative distribution function (CDF) test

In the Black-Scholes model to evaluate (Call premium) it is needed to know the Cumulative distribution function of ande as in (1‑1).

Cumulative distribution function [23] is defined as following

|  |  |
| --- | --- |
|  | (4‑4) |

Where,

***-*** Probability density function (PDF)

To calculate CFD in the solution that we used from[[25]](#footnote-25) author uses Taylor approximation.

For selected value , the Probability density functionwill be needed for our calculations. The pseudocode of algorithm used in our project to calculate CFD is the following

|  |
| --- |
| **Input:**  *value*    **Output:**  *Cumulative distribution function (*CDF*) for value*  **Algorithm:**  **If**  *> 8*  **Return** 1  **Endif**  **If**  *< -8*  **Return** 0  **Endif**      **for** **to**  **do**        **Endfor**  CDF =  **Return** CDF |

In the end, we calculate PDF which is expressed as in above pseudocode.

To evaluate this solution we checked the Standard Normal Distribution Table [24]. The vector of values to test was created, and we compared the results given by our software with the values for the Standard Normal Distribution Table.

Table 5 Unit test of the Cumulative distribution function. Comparing values calculated by our software with Cumulative distribution function values from the Standard Normal Distribution Table.

|  |  |
| --- | --- |
| **Cumulative distribution function Unit test** | |
| **Values type** | **Values** |
| Values to test | 0.01, 0.08, 0.44, 0.12, 0.24, 0.57, 0.87, 0.97 |
| Expected Cumulative distribution function for the big input values calculated in Excel 2016 | 0.5, 0.5319, 0.67, 0.5478, 0.5948, 0.7157, 0.8078, 0.8340 |
| **Test Result** | **Passed** |

The next test for the CFD is the one that checks how the function handles for the big input values. We assume that for the CFD input values larger than 8 and lower than -8 output should the equals 1 and 0 respectively.

Table 6 Unit test of the Cumulative distribution function. Comparing values calculated by our software with the Microsoft Excel 2016 NORM.S.DIST() function with the true Cumulative flag.

|  |  |
| --- | --- |
| **Cumulative distribution function Unit test** | |
| **Values type** | **Values** |
| Big values to test | 12312.43,3432534.56,3425432.54, 77987.78,-542234.43,-53252.47, -654563.34, -865447.65 |
| Expected Cumulative distribution function for the big input values calculated in Excel 2016 | 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0 |
| **Test Result** | **Passed** |

#### Black-Scholes Unit tests - Call premium calculation

Having Probability density function (PDF) and Cumulative distribution function (CDF) functionalities tested we can perform Call premium calculations test. In the 4.4.1 subsection, we compared results given by our solution with the externals with

To do those test, we selected the sets of values such as Current stock price, Time until option exercise, Option striking price (Exercise price), Risk-free interest rate and Standard deviation (Volatility).

For each vector of above results, we took values at index i, and we calculated i results. We have i some values in each vector so that we will get i results.

Code snippet of the test

**for** (**int** i = 0; i < s.**length**; ++i) {  
 *assertEquals*(Tools.*CalculateBlackScholesModel*(S[i], t[i], K[i], r[i], s[i]), expectedResults[i], 0.01);  
}

Table 7 Unit test of Call premium. Comparing values calculated by our software with the Microsoft Excel 2016 output computed using Excel functions such as SQRT(), LN() or NORM.S.DIST(), function with the true Cumulative flag.

|  |  |
| --- | --- |
| **Call premium calculation Unit test** | |
| **Values type** | **Values** |
| Current stock prices | 141.24, 23.4, 43.2, 324.3, 21.5, 34.2, 12.4 |
| Time until option exercise | 16, 12, 23, 11, 43, 12, 4 |
| Option striking price (Exercise price) | 150.0, 160.0, 155.5 ,170.0, 180.7, 177.0, 171.0 |
| Risk-free interest rate | 4.0, 4.1, 1.5, 5.5, 4.8, 6.3, 7.1, |
| Standard deviation (Volatility). | 30, 40, 45, 35, 37, 42, 47 |
| Expected Call premium calculation values calculated in Excel 2016 | 141.0151808, 23.38399643, 43.19999874, 323.7891892, 21.49999999, 34.19031615, 11.84739255 |
| **Test Result** | **Passed** |

#### Technical Analysis Unit tests – Min-max value indicator

This functionality allows seeing for a particular trend. It connects the lowest price with the highest price of the chosen company for the selected time range. This is the simplest indicator that we used in the Technical Analysis.

The image presents our application with Min-max value indicator applied for the selected stock.

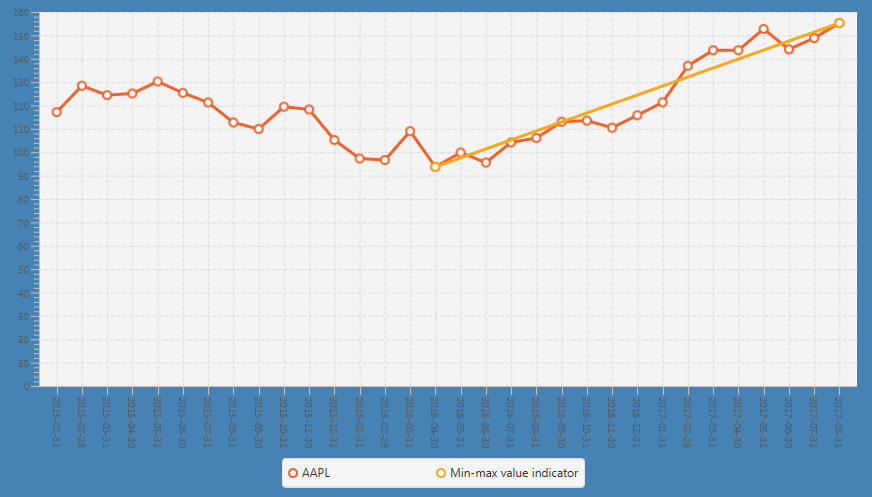


Figure 17 Our application output showing Min-max value indicator applied for the Apple stocks for the period from 11.01.2015 to 10.08.2017 with a monthly interval of prices.

To find the minimum and maximum value of the stock two functions were implemented. In order to evaluate its results accuracy, the tests were performed.

For a function that finds the minimum value in the selected vector of stocks prices for the chosen period, we created the vectors with known and distinctive minimal price.

Example array for finding the minimum value in the vector used for the tests:

Double[] testArray2 = {32131.0, 4.0, 0.2, 3545.0, 5.0, 36.0, 3.0, 5.0, 7545.0, 3.0, 3.0, 7.0, 2.0, 47.0, 4.0, 2.0, 675.0, 3267.0, 3.0, 6.0};

In above example the minimum value is 0.2 and it has index of 2 in Java (the third position).

We applied 3 test arrays to the tests including above presented.

Table 8 Unit test of the minimum value in the selected vector function. Comparing index number of minimum value evaluated by our software with the minimum index assessed by a human (author).

|  |  |
| --- | --- |
| **The minimum value in the selected vector Unit test** | |
| **Test vector** | **Expected result – index of minimum element in the test vector** |
| Test array 1 | 6 |
| Test array 2 | 2 |
| Test array 3 | 7 |
| **Test Result** | **Passed** |

The similar test was performed for the function that finds The maximum value in the selected prices vector.

Example array for finding the maximum value in the vector used for the tests:

Double[] testArray = {10.0, 23.0, 43.0, 5.0, 432.0, 5.0, 36.0, 2.0, 3.0, 2.0, 2.0};

As in the finding the minimum value function in finding the maximum vector value function we used 3 test arrays including above one.

Table 9 Unit test of the minimum value in the selected vector function. Comparing index number of minimum value evaluated by our software with the minimum index assessed by a human (author).

|  |  |
| --- | --- |
| **The maximum value in the selected vector Unit test** | |
| **Test vector** | **Expected result – index of maximum element in the test vector** |
| Test array 1 | 4 |
| Test array 2 | 0 |
| Test array 3 | 7 |
| **Test Result** | **Passed** |

#### Technical Analysis Unit tests – Derivatives calculation tests

Our solution allows the user to see derivatives of the selected stocks for the selected period.

Below image shows our application output with calculated derivatives.

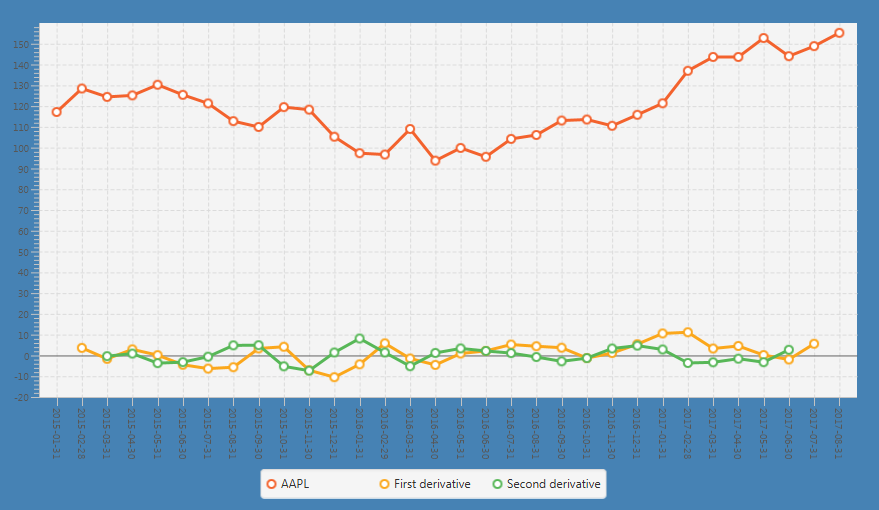


Figure 18 Our application output showing calculated derivatives applied for the Apple stocks for the period from 11.01.2015 to 10.08.2017 with a monthly interval of prices.

We are estimating derivatives with the following method

|  |  |
| --- | --- |
|  | (4‑5) |

Where,

- is date when stock had particular price

– is calculated derivative

– is the stock price for selected date

To test that approach we inputted the set of values to Excel, and we calculated results using the formula (4‑5).

Table 10 Unit test of the first derivative function. Comparing values calculated by our software with the results from Microsoft Excel 2016 with applied (4‑5) equation.

|  |  |
| --- | --- |
| **Calculating the first derivative function Unit test** | |
| **Values type** | **Values** |
| Values to test | 4.26, 5.6, 6.54, 5.365, 5.88, 6.08, 7.99, 7.52,5.87, 8.08, 9.12, 10.41 |
| Expected the first derivative values calculated in Excel 2016 | 1.14, -0.1175, -0.33, 0.3575, 1.055, 0.72, -1.06, 0.28, 1.625, 1.165 |
| **Test Result** | **Passed** |

To calculate the second derivative we can just derivate derivative.

|  |  |
| --- | --- |
|  | (4‑6) |

To test the second derivative function, we derivate the first derivative in Excel the following formula was used equivalent to (4‑6).

|  |  |
| --- | --- |
| Table 11 Unit test of the second derivative function. Comparing values calculated by our software with the results from Microsoft Excel 2016 with applied (4‑7) equation. | (4‑7) |

|  |  |
| --- | --- |
| **Calculating the second derivative function Unit test** | |
| **Values type** | **Values** |
| Values to test | 4.26, 5.6, 6.54, 5.365, 5.88, 6.08, 7.99, 7.52,5.87, 8.08, 9.12, 10.41 |
| Expected the second derivative values calculated in Excel 2016 | -0.735, 0.2375, 0.6925, 0.18125,-1.0575, -0.22, 1.3425, 0.4425, |
| **Test Result** | **Passed** |

#### Technical Analysis Unit tests – Moving average tests

The Moving average belongs to the Quantitive Forecasting Methods [25]. This is particular usage to smoothing method.

Smoothing methods are

The moving average is the technique used when the selected values series has no significant trend. It provides a particular average of the values series in order to make that data series less dispersed, and the new smoothed data provides investor more clear trend of the selected data series (stock prices).

Basing on the Apple share prices example, we will show how the moving average series looks on the Line chart.

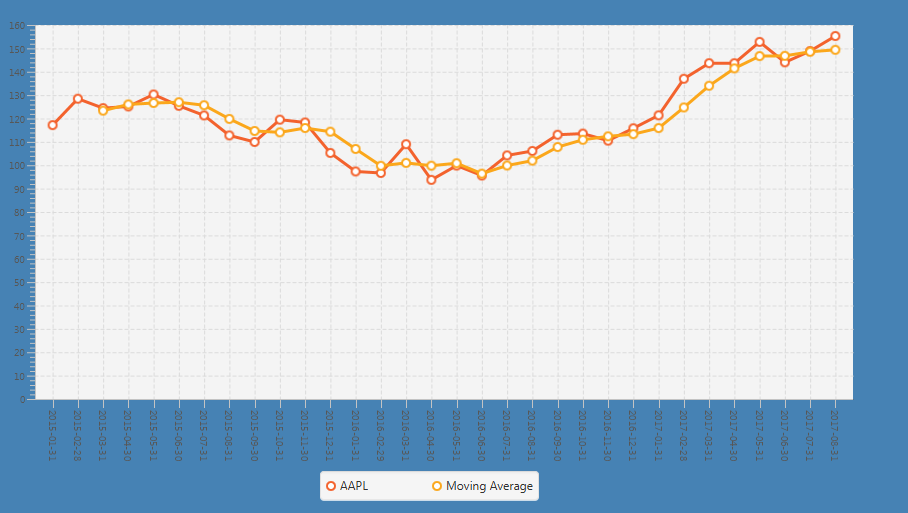


Figure 19 Our application output showing calculated moving average applied to the Apple stocks for the period from 11.01.2015 to 10.08.2017 with a monthly interval of prices. The Orange line is actual Apple share prices; yellow line is smoothed prices series (moving average).

Moving average calculation is given by the following equation [25]:

|  |  |
| --- | --- |
|  | (4‑8) |

In our computations, we selected three the most recent value period, so in our case, the Moving average equation takes the form

|  |  |
| --- | --- |
|  | (4‑9) |

Basing on the example from[[26]](#footnote-26) and also validating it by our Excel data calculation we have the following results

Table 12 Values calculated using three-period Moving average method (4‑9).

|  |  |
| --- | --- |
| **Moving average** |  |
| Test values | Results |
| 10 | N/A |
| 12 | N/A |
| 16 | 12.66667 |
| 13 | 13.66667 |
| 17 | 15.33333 |
| 19 | 16.33333 |
| 15 | 17 |
| 20 | 18 |
| 22 | 19 |
| 19 | 20.33333 |
| 21 | 20.66667 |
| 19 | 19.66667 |

Above table shows that for two first values we do not have any result. For used in our software three periods Moving average we need three the most recent values, so the first value that we can compute an example from the Table 12 is 12.66667, because

|  |  |
| --- | --- |
|  | (4‑10) |

Moreover, the resulting second analogically equals

|  |  |
| --- | --- |
| The rest values were calculated analogically as well.  For that functionality, the proper test was performed. | (4‑11) |

|  |  |
| --- | --- |
| **Calculating the Moving average function Unit test** | |
| **Values type** | **Values** |
| Values to test | 10.0, 12.0, 16.0, 13.0, 17.0, 19.0, 15.0, 20.0, 22.0, 19.0, 21.0, 19.0 |
| Expected the Moving average values calculated in Excel 2016 | 12.66666667, 13.66666667, 15.33333333, 16.33333333, 17, 18, 19, 20.33333333, 20.66666667, 19.66666667 |
| Expected the Moving average values from the resource presentation26 | 12.67, 13.67, 15.33, 16.33, 17.00, 18.00, 19.00, 20.33, 20.67, 19.67 |
| **Test Result** | **Passed** |

## Software opinion results

The one of the most important parts of the thesis is to examine if the application output gives us information that is properly evaluating movement that should be profitable in the future.

As it was specified in subsection 3.2.4 we have three Analysis periods to consider. Short term is the most speculative[[27]](#footnote-27) type of evaluation. In such period, many movements can occur not always related to real company financial status and its results. In our tests, we took into account 51 U.S. Companies.

Results cover the Mid-term output of our application. In that time interval and for a Long-term analysis we are not considering Apple (AAPL) and Peabody Energy Corporation Interactive (BTU), since Apple had a stock split in 09.07.2014[[28]](#footnote-28) and BTU set to exit bankruptcy[[29]](#footnote-29). The split was also done on Ball Corporation. (BLL)[[30]](#footnote-30), as well we are not taking that company into account. For Mid-term w chose the same Companies as for the Short-term research.

Crucial indicator for our research is opinion accuracy in %. It is calculated by the following equation:

|  |  |
| --- | --- |
|  | (4‑12) |

We distinct software output opinions over 50% accuracy as successful, which is equivalent to probability greater than .

Table 13 The application outputs considering the set of 51 (49, since 2 of them are not considered - NC) U.S. Companies. The table shows Potential revenue, Opinion of about potential market movement (BUY, HOLD, SELL), and Opinion accuracy calculated by the thesis application. Considered Short-term period on which the analysis was performed from 10.10.2016 to 10.03.2017 and prices data are loaded in the daily interval.

|  |  |  |  |
| --- | --- | --- | --- |
| **Company  symbol** | **Revenue in %** | **Output  opinion** | **Opinion accuracy** |
| BLL | -45.6876 | HOLD | NC |
| BTU | 1272.464 | HOLD | NC |
| AAPL | 11.59 | BUY | right |
| ABBV | 6.92 | BUY | right |
| ADBE | 19.68 | BUY | right |
| ADSK | 21.90 | BUY | right |
| AET | 14.77 | BUY | right |
| AFL | 12.22 | BUY | right |
| AIZ | 3.36 | BUY | right |
| AKAM | -27.89 | BUY | wrong |
| ALL | 14.76 | BUY | right |
| ALLE | 5.98 | BUY | right |
| ALXN | 3.92 | BUY | right |
| AMAT | 10.26 | BUY | right |
| AME | 17.01 | BUY | right |
| AMGN | -6.80 | BUY | wrong |
| AMP | 8.85 | BUY | right |
| AMT | 20.47 | BUY | right |
| AMZN | 12.25 | BUY | right |
| A | 12.31 | BUY | right |
| AON | 17.81 | BUY | right |
| APH | 10.23 | BUY | right |
| AVB | 6.35 | BUY | right |
| AVY | 14.20 | BUY | right |
| AXP | 6.53 | BUY | right |
| BA | 30.32 | BUY | right |
| BAC | -4.70 | BUY | wrong |
| BAX | 14.80 | BUY | right |
| BBT | -2.16 | BUY | wrong |
| BCR | 27.77 | BUY | right |
| BDX | 6.97 | BUY | right |
| BEN | 0.12 | BUY | right |
| BIIB | -3.79 | BUY | wrong |
| BK | 7.85 | BUY | right |
| BLK | 9.91 | BUY | right |
| BMS | -15.69 | BUY | wrong |
| BMY | -2.97 | BUY | wrong |
| BRK\_B | 0.79 | BUY | right |
| BWA | 7.44 | BUY | right |
| C | 9.32 | BUY | right |
| MSFT | 9.98 | BUY | right |
| ABC | -10.68 | HOLD | wrong |
| AIG | -0.13 | HOLD | wrong |
| AIV | 5.32 | HOLD | right |
| APA | -15.17 | HOLD | wrong |
| APC | -29.39 | HOLD | wrong |
| APD | 4.50 | HOLD | right |
| AZO | -26.71 | HOLD | wrong |
| BBY | 33.44 | HOLD | right |
| BF\_B | 3.53 | HOLD | right |
| BSX | 6.57 | HOLD | right |
| BXP | -7.02 | HOLD | wrong |
| IBM | -20.24 | HOLD | wrong |
|  | Total profit in % |  | Accuracy  in % |
|  | **5.03** |  | **74.51** |
|  | Total profit in % BUY only |  |  |
|  | **8.02** |  |  |
|  | Potential  profit/loss in % |  |  |
|  | -4.66 |  |  |

The results are surprising; seemingly the most speculative Short-term period brought almost 75% decision accuracy. Those results are even over our expectations. The total profit of investments would be 5.03% in 5 months. This result also includes all lose (wrong decisions).

The another aspect is that we did not have any SELL signals for that period, so in an exception, for above-presented example, we decided to introduce “BUY only” indicator. When we ignore HOLD outputs, we can observe that our profit would rise to over 8%. If we decide to buy only stocks with the HOLD opinion, we will lose 4.64%.

One reason of that is the current economic state; recently stocks hit records high[[31]](#footnote-31). Nevertheless, our strategy, for now, seems to be effective.

Respectively we are considering the Mid-term results. In this case, we are collecting price data with a monthly interval; each price is given at the end of each month in selected period. The period of the Mid-term period which we chose is two years, so the maximum duration for the Mid-term case.

Below the results given by the simulation are presented:

Table 14 The application outputs considering the set of 53 U.S. Companies (50, since 3 of them are not considered - NC). The table shows Potential revenue, Opinion about potential market movement (BUY, HOLD, SELL), and opinion accuracy calculated by the thesis application. Considered period on which the analysis was performed from 10.08.2014 to 10.08.2016 and prices data are loaded in the monthly interval.

|  |  |  |  |
| --- | --- | --- | --- |
| **Company  symbol** | **Revenue in %** | **Output  opinion** | **Opinion accuracy** |
| AAPL | 43.76852 | BUY | NC |
| BLL | -50.0561 | BUY | NC |
| BTU | 1272.464 | SELL | NC |
| ABBV | 6.036429 | BUY | right |
| ABC | -10.2617 | BUY | wrong |
| ADBE | 44.02703 | BUY | right |
| ADSK | 74.09619 | BUY | right |
| AET | 27.50544 | BUY | right |
| AFL | 10.4551 | BUY | right |
| AIG | 6.779661 | BUY | right |
| AIV | 0.748899 | BUY | right |
| ALL | 35.63814 | BUY | right |
| ALLE | 10.60967 | BUY | right |
| AMAT | 59.14426 | BUY | right |
| AMGN | -1.2381 | BUY | wrong |
| AMT | 17.71335 | BUY | right |
| AMZN | 24.50817 | BUY | right |
| A | 23.27931 | BUY | right |
| AON | 26.49176 | BUY | right |
| APC | -16.9675 | BUY | wrong |
| APD | -2.69073 | BUY | wrong |
| APH | 29.37947 | BUY | right |
| AVB | 2.889953 | BUY | right |
| AVY | 16.82552 | BUY | right |
| AZO | -34.5921 | BUY | wrong |
| BA | 76.0508 | BUY | right |
| BAC | 62.86293 | BUY | right |
| BAX | 22.86008 | BUY | right |
| BBT | 25.72044 | BUY | right |
| BBY | 77.23095 | BUY | right |
| BCR | 43.35834 | BUY | right |
| BDX | 13.28799 | BUY | right |
| BF\_B | -51.1987 | BUY | wrong |
| BIIB | -8.97465 | BUY | wrong |
| BLK | 15.3504 | BUY | right |
| BMS | -17.8261 | BUY | wrong |
| BRK\_B | 20.22633 | BUY | right |
| BSX | 9.966499 | BUY | right |
| BXP | -15.4265 | BUY | wrong |
| IBM | -12.4877 | BUY | wrong |
| MSFT | 23.07825 | BUY | right |
| AIZ | 21.47912 | SELL | wrong |
| AKAM | -10.9765 | SELL | right |
| ALXN | -0.27889 | SELL | right |
| AME | 32.67075 | SELL | wrong |
| AMP | 50.21939 | SELL | wrong |
| APA | -12.6486 | SELL | right |
| AXP | 30.61477 | SELL | wrong |
| BEN | 19.88189 | SELL | wrong |
| BK | 30.39659 | SELL | wrong |
| BMY | -6.58633 | SELL | right |
| BWA | 33.1949 | SELL | wrong |
| C | 47.89879 | SELL | wrong |
|  | Summary  profit in % |  | Accuracy  in % |
|  | **16.70** |  | **64.00** |
|  | Potential  profit/loss in % |  |  |
|  | **19.66** |  |  |

This time results are also very satisfactory, we achieved 16.70% of profit in one year, as we are considering that we will possess stocks on 10.08.2017. We have to mention also the fact that if we invested opposite to program output, we would gain even more because 19,66%. Nevertheless, that type of investment according to stocks trend line has no logical basis. Data from the past at least considering two years period for Ameriprise Financial, Inc. (AMP) did not give any signal to buy. The most for us significant indicator is decision accuracy which is equal 64%.

The last considered period is long-term analysis. For it, we took five years extended analysis period. Below our results for backwards analysis using an extended period of 5 years.

Table 15 The application outputs considering the set of 53 U.S. Companies (50, since 3 of them are not considered - NC). The table shows Potential revenue, Opinion about potential market movement (BUY, HOLD, SELL), and opinion accuracy calculated by the thesis application. Considered period on which the analysis was performed from 10.08.2011 to 10.08.2016 and prices data are loaded in the monthly interval.

|  |  |  |  |
| --- | --- | --- | --- |
| **Company  symbol** | **Revenue in %** | **Output  opinion** | **Opinion accuracy** |
| ABBV | 6.036429 | BUY | right |
| ADBE | 44.02703 | BUY | right |
| ADSK | 74.09619 | BUY | right |
| AET | 27.50544 | BUY | right |
| AFL | 10.4551 | BUY | right |
| AIV | 0.748899 | BUY | right |
| AIZ | 21.47912 | BUY | right |
| ALL | 35.63814 | BUY | right |
| ALLE | 10.60967 | BUY | right |
| AMAT | 59.14426 | BUY | right |
| AMGN | -1.2381 | BUY | wrong |
| AMT | 17.71335 | BUY | right |
| AMZN | 24.50817 | BUY | right |
| A | 23.27931 | BUY | right |
| AON | 26.49176 | BUY | right |
| APA | -12.6486 | BUY | wrong |
| APD | -2.69073 | BUY | wrong |
| APH | 29.37947 | BUY | right |
| AVB | 2.889953 | BUY | right |
| AVY | 16.82552 | BUY | right |
| AZO | -34.5921 | BUY | wrong |
| BAX | 22.86008 | BUY | right |
| BCR | 43.35834 | BUY | right |
| BDX | 13.28799 | BUY | right |
| BLK | 15.3504 | BUY | right |
| BMS | -17.8261 | BUY | wrong |
| BMY | -6.58633 | BUY | wrong |
| BRK\_B | 20.22633 | BUY | right |
| BSX | 9.966499 | BUY | right |
| BXP | -15.4265 | BUY | wrong |
| IBM | -12.4877 | BUY | wrong |
| MSFT | 23.07825 | BUY | right |
| ABC | -10.2617 | HOLD | wrong |
| AIG | 6.779661 | HOLD | right |
| AKAM | -10.9765 | HOLD | wrong |
| ALXN | -0.27889 | HOLD | wrong |
| AME | 32.67075 | HOLD | right |
| AMP | 50.21939 | HOLD | right |
| APC | -16.9675 | HOLD | wrong |
| AXP | 30.61477 | HOLD | right |
| BA | 76.0508 | HOLD | right |
| BAC | 62.86293 | HOLD | right |
| BBT | 25.72044 | HOLD | right |
| BBY | 77.23095 | HOLD | right |
| BEN | 19.88189 | HOLD | right |
| BF\_B | -51.1987 | HOLD | wrong |
| BIIB | -8.97465 | HOLD | wrong |
| BK | 30.39659 | HOLD | right |
| BWA | 33.1949 | HOLD | right |
| C | 47.89879 | HOLD | right |
|  | Summary  profit in % |  | Accuracy  in % |
|  | **17.29** |  | **72.00** |
|  | Total profit BUY only in % |  | HOLD  only profit  in % |
|  | **14.86** |  | **21.94** |

What was unexpected that there are no SELL outputs in above results. These are similar to mid-term one and would achieve this time over 17% profit in a year. If we ignored HOLD opinion outputs, we would achieve worse result than following the application output.

Interesting is HOLD only result in which we assume to buy only stocks with the HOLD simulation opinion, but in real life it would be the really unusual type of investment.

# Conclusions and further work

During our research for the thesis, we obtained auspicious results. For each period type, we achieved a significant profit, the minimum one was 14.86%. Of course, the World economy is solid and in United Stated stocks hit the records regularly, but opinion given by our solution seems to be promising. Nevertheless, when we look at another statistics, i.e. opinion Accuracy the conclusions are getting more clear. In the worst case researched scenario the minimum accuracy was at the level of 60% for the Long-term analysis. The Mid-term and Short-term analysis had astonishing levels of 64% and 74.51% accuracy respectively.

Also, potential profit is noticeable. For the Short, Mid and Long -term investment we achieved 5.03%, 16.70% and 17.29% respectively. Comparing that with the fact that in August 2017 interest rate in United Kingdom bank’s deposit is only 2.0%[[32]](#footnote-32) even over 17% profit is significant. The highest interest rate in Europe is in Ukraine 16.50%, but that investment presently could be very risky.

We must mention that not always the opinion given by our application was the most beneficial. In the Mid-term period, we could achieve higher profit if instead of selling we would buy selected stocks. In the Long-term scenario, we could gain more if we would buy only stocks with the HOLD opinion. We believe those examples show that investor has to consider often many others financial indicators apart from the Technical Analysis if he wants to forecast more accurate. Nevertheless, the simulation is based on only past data we conclude that basing only on that data it is not only possible but very likely to make a profit.

### Further work

In the future we would like to focus on making new other decision-making methods and performing even more detailed research. In this paper, we were based on U.S. stock. Apart from that market, we have much more European and not only stock markets like London Stock Exchange, Warsaw Stock Exchange, Frankfurt Stock Exchange or Emerging markets[[33]](#footnote-33) stock exchanges.

Our next aim is to analyse options using Black-Scholes model and make analogical simulation outputs (BUY, HOLD, SELL) but this time the most probably we would base our analysis on Implied volatility calculated using Black-Scholes model or retrieved from external resources.

The next improvement taken into account by us would be the application interface enhancement in order to bring the user the highest quality simulation. The issues like minor bugs fix and performance improvement are also considered.

### Final conclusions

The thesis research shows that forecasting based on only past stock prices is not only possible but very likeable. The average accuracy based on our tests is equal 70.5% and average profit based on Mid and the Long-term period is 17%. That accuracy level which can also be considered as probability, which is significantly higher than making simple random choice with probability . The average profit is also greater than any even the highest European deposit offer. We believe that combination of Statistics, Mathematics and Software development is future of finance and our results strengthen that statement.

REFERENCES

[1] Taylor, M. P., & Allen, H. (1992). The use of technical analysis in the foreign exchange market. Journal of International Money and Finance, 11(3), 304-314. doi:10.1016/0261-5606(92)90048-3

[2] Lo, A. W., Mamaysky, H., & Wang, J. (2000). Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. Journal of Finance, 55(4), 1705-1765. Retrieved from www.scopus.com

[3] Neely, C., Weller, P., & Dittmar, R. (1997). Is technical analysis in the foreign exchange market profitable? A genetic programming approach. Journal of Financial and Quantitative Analysis, 32(4), 405-426. Retrieved from www.scopus.com

[4] Bessembinder, H., & Chan, K. (1998). Market efficiency and the returns to technical analysis. Financial Management, 27(2), 5-17. Retrieved from www.scopus.com

[5] Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. Journal of Political Economy, 81(3), 637-657. Retrieved April 2017 from www.scopus.com

[6] Merton, R. C. (1973). THEORY OF RATIONAL OPTION PRICING. Bell J Econ Manage Sci, 4(1), 141-183. Retrieved from www.scopus.com

[7] Merton, R. C. (1976). Option pricing when underlying stock returns are discontinuous. Journal of Financial Economics, 3(1-2), 125-144. doi:10.1016/0304-405X(76)90022-2

[8] Cox, J. C., Ross, S. A., & Rubinstein, M. (1979). Option pricing: A simplified approach. Journal of Financial Economics, 7(3), 229-263. doi:10.1016/0304-405X(79)90015-1

[9] Rusu, V., & Rusu C. (2003). Forecasting methods and stock market

Analysis.). Retrieved April 2017, from http://creative-mathematics.ubm.ro/issues/creative\_2003\_12\_103\_110.pdf

[10] Kuepper J., Technical Analysis: The Basic Assumptions. Retrieved April 2017, from: http://www.investopedia.com/university/technical/techanalysis1.asp

[11] Khan S., Introduction to the Black-Scholes formula. Retrieved April 2017, from:

https://www.khanacademy.org/economics-finance-domain/core-finance/derivative-securities/black-scholes/v/introduction-to-the-black-scholes-formula

[12] Folger J., Options Pricing: Black-Scholes Model. Retrieved April 2017, from: http://www.investopedia.com/university/options-pricing/black-scholes-model.asp

[13] Black-Scholes Model. Retrieved April 2017, from: http://www.investopedia.com/terms/b/blackscholes.asp

[14] Investopedia Staff. Introduction To Fundamental Analysis. Retrieved May 9, 2017, from: <http://www.investopedia.com/university/fundamentalanalysis/>

[15] Khan S., Implied volatility. Retrieved June 2017, from:

<https://www.khanacademy.org/economics-finance-domain/core-finance/derivative-securities/black-scholes/v/implied-volatility>

[16] Implied Volatility - IV - Investopedia. (n.d.). Retrieved June 2017 from <http://www.investopedia.com/terms/i/iv.asp>

[17] Underlying Asset - Investopedia. (n.d.). Retrieved from <http://www.investopedia.com/terms/u/underlying-asset.asp>

[18] Options Basics: What Are Options? - Investopedia. (n.d.). Retrieved from <http://www.investopedia.com/university/options/option.asp>

[19] Bracker K., Introduction to Options -- Part One. Retrieved June 2017 from <https://www.youtube.com/watch?v=92kmHlctjdk&list=PLj0kNr0hjxJKw-IwVtPBv8mUMd9Iv_AsY>

[20] Profits Run, Call Options & Put Options Explained Simply In 8 Minutes (How To Trade Options For Beginners). Retrieved June 2017 from

<https://www.youtube.com/watch?v=EfmTWu2yn5Q>

[21] MANN, P. R. (2010). The Standard Normal Distribution. In INTRODUCTORY STATISTICS (7thth ed., pp. 259-266). Retrieved June 19, 2017, from <http://ftp.cats.com.jo/Stat/Statistics/Introductory%20Statistics%20(7th%20Ed).pdf>

[22] Leff, A., & Rayfield, J. T. (2001). Web-application development using the Model/View/Controller design pattern. Paper presented at the *Proceedings - 5th IEEE International Enterprise Distributed Object Computing Conference, , 2001-January*(January) 118-127. doi:10.1109/EDOC.2001.950428

[23] Zwillinger, D., Kokoska, S. (2010). CRC Standard Probability and Statistics Tables and Formulae. CRC Press. p. 49.

[24] Cewe, A., & Nachorowska, H., Pancer I. (2012). Tablice matematyczne (Wydanie drugie, p. 130).

[25] Narodzonek-Karpowska M., & Prof. Dr. Toporowski W. Smoothing methods Retrieved August 16, 2017, from <http://www.statoek.wiso.uni-goettingen.de/veranstaltungen/graduateseminar/SmoothingMethods_Narodzonek-Karpowska.pdf>

APPENDICES

Source Code

**Main class**

**package** Thesis;  
  
**import** javafx.application.Application;  
**import** javafx.fxml.FXMLLoader;  
**import** javafx.scene.Parent;  
**import** javafx.scene.Scene;  
**import** javafx.stage.Stage;  
  
**public class** Main **extends** Application {  
  
 **public static void** main(String[] args) {  
 *launch*(args);  
 }  
  
 @Override  
 **public void** start(Stage primaryStage) **throws** Exception {  
 Parent root = FXMLLoader.*load*(getClass().getResource(**"design.fxml"**));  
 root.getStylesheets().add(**"/sets.css"**);  
 primaryStage.setTitle(**"Thesis application"**);  
 primaryStage.setScene(**new** Scene(root, 1000, 800));  
  
 primaryStage.show();  
 }  
}

**package** Thesis.View;  
  
**import** Thesis.Model.QuotesReader;  
**import** Thesis.Model.TechnicalAnalysisTools;  
**import** Thesis.Model.Tools;  
**import** javafx.scene.chart.XYChart;  
**import** org.threeten.bp.LocalDate;  
  
**import** java.util.ArrayList;  
**import** java.util.InputMismatchException;  
  
*/\*\*  
 \* Created by Wiktor Bednarek on 2017-08-09.  
 \*/***public class** TechnicalAnalysisView {  
  
 **private** String **technicalAnalysisStockQuote**;  
  
 **private** LocalDate **startDate**;  
  
 **private** LocalDate **endDate**;  
  
 **private** QuotesReader **stockDataReader**;  
  
  
 **private boolean areStockDataLoaded**;  
  
 **public** TechnicalAnalysisView(String technicalAnalysisStockQuote, LocalDate startDate, LocalDate endDate) {  
 **this**.**technicalAnalysisStockQuote** = technicalAnalysisStockQuote;  
 **this**.**startDate** = startDate;  
 **this**.**endDate** = endDate;  
 **stockDataReader** = **new** QuotesReader();  
 **areStockDataLoaded** = **false**;  
 }  
  
 **public** TechnicalAnalysisView() {  
 **stockDataReader** = **new** QuotesReader();  
 }  
  
 **public** XYChart.Series prepareTheTechnicalAnalysisChart() {  
  
 **stockDataReader**.showSelectedQuote(**technicalAnalysisStockQuote**, **startDate**, **endDate**);  
 *//stockDataReader.getPriceForSelectedDate(technicalAnalysisStockQuote, endDate);* XYChart.Series series = **new** XYChart.Series();  
 series.setName(**technicalAnalysisStockQuote**);  
  
 **try** {  
 **if** (**stockDataReader**.getPrices().size() == **stockDataReader**.getDates().size()) {  
 **for** (**int** i = 0; i < **stockDataReader**.getPrices().size(); ++i) {  
 series.getData().add(**new** XYChart.Data(**stockDataReader**.getDates().get(i).toString(), **stockDataReader**.getPrices().get(i)));  
 }  
 }  
  
  
 *//TechnicalAnalysisLineChart.getData().addAll(series);* } **catch** (InputMismatchException exception) {  
 System.***out***.print(exception.getMessage());  
 }  
  
 **areStockDataLoaded** = **true**;  
 **return** series;  
 }  
  
  
 **public** XYChart.Series addTheTechnicalAnalysisDataToTheChart() {  
 XYChart.Series TechnicalAnalysisSeries = **new** XYChart.Series();  
 TechnicalAnalysisSeries.setName(**"Min-max value indicator"**);  
  
  
 **stockDataReader**.findTheDataToCreateMinMaxValueIndicator();  
 System.***out***.println(**"Reader data: "** + **stockDataReader**.getTechnicalAnalysisPriceData().get(1));  
  
  
 TechnicalAnalysisSeries.getData().add(**new** XYChart.Data(**stockDataReader**.getTechnicalAnalysisDatesIndicesData().get(0).toString(), **stockDataReader**.getTechnicalAnalysisPriceData().get(0)));  
 TechnicalAnalysisSeries.getData().add(**new** XYChart.Data(**stockDataReader**.getTechnicalAnalysisDatesIndicesData().get(1).toString(), **stockDataReader**.getTechnicalAnalysisPriceData().get(1)));  
  
*/\*  
 for (int i = 0; i < size; ++i) {  
 TechnicalAnalysisSeries.getData().add(new XYChart.Data(stockDataReader.getTechnicalAnalysisDatesData().get(i).toString(), stockDataReader.getTechnicalAnalysisPriceData().get(i)));  
 }  
 \*/* **return** TechnicalAnalysisSeries;  
 }  
  
  
 **public** XYChart.Series firstDerivative() {  
 **int** derivativeNumber = 1;  
  
 XYChart.Series firstDerivativeSeries = **new** XYChart.Series();  
 firstDerivativeSeries.setName(**"First derivative"**);  
 ArrayList<Double> deviatedVector = Tools.*calculateDerivative*(**stockDataReader**.getPrices());  
  
 **for** (**int** i = 0; i < deviatedVector.size(); ++i) {  
 firstDerivativeSeries.getData().add(**new** XYChart.Data(**stockDataReader**.getDates().get(i + derivativeNumber).toString(), deviatedVector.get(i)));  
 System.***out***.println(**"Derivated value "** + deviatedVector.get(i));  
 }  
 System.***out***.println(**"derivative vector size: "** + deviatedVector.size());  
  
 **return** firstDerivativeSeries;  
 }  
  
  
 **public** XYChart.Series secondDerivative() {  
 **int** derivativeNumber = 2;  
  
 XYChart.Series secondDerivativeSeries = **new** XYChart.Series();  
  
 secondDerivativeSeries.setName(**"Second derivative"**);  
 ArrayList<Double> deviatedVector = Tools.*calculateSecondDerivative*(**stockDataReader**.getPrices());  
  
 **for** (**int** i = 0; i < deviatedVector.size(); ++i) {  
 secondDerivativeSeries.getData().add(**new** XYChart.Data(**stockDataReader**.getDates().get(i + derivativeNumber).toString(), deviatedVector.get(i)));  
 System.***out***.println(**"2 Derivated value "** + deviatedVector.get(i));  
 }  
 System.***out***.println(**"2 derivative vector size: "** + deviatedVector.size());  
  
 **return** secondDerivativeSeries;  
 }  
  
  
 **public** XYChart.Series movingAverageSeries(**int** periodNumber) {  
  
  
 XYChart.Series movingAverageSeries = **new** XYChart.Series();  
  
 movingAverageSeries.setName(**"Moving Average"**);  
  
 ArrayList<Double> movingAverageVector = TechnicalAnalysisTools.*movingAverage*(**stockDataReader**.getPrices(), periodNumber);  
  
 **for** (**int** i = 0; i < movingAverageVector.size(); ++i) {  
 movingAverageSeries.getData().add(**new** XYChart.Data(**stockDataReader**.getDates().get(i + (periodNumber - 1)).toString(), movingAverageVector.get(i)));  
  
 }  
  
  
 **return** movingAverageSeries;  
 }  
  
  
 **public** XYChart.Series regressionLineSeries() {  
  
  
 XYChart.Series resressionLineSeries = **new** XYChart.Series();  
  
 resressionLineSeries.setName(**"Regression line"**);  
 ArrayList<Double> regressionLineVector = TechnicalAnalysisTools.*simpleRegression*(**stockDataReader**.getPrices());  
  
 **for** (**int** i = 0; i < regressionLineVector.size(); ++i) {  
 resressionLineSeries.getData().add(**new** XYChart.Data(**stockDataReader**.getDates().get(i).toString(), regressionLineVector.get(i)));  
 }  
  
  
 **return** resressionLineSeries;  
 }  
  
  
  
 **public** String getTechnicalAnalysisStockQuote() {  
 **return technicalAnalysisStockQuote**;  
 }  
  
 **public** QuotesReader getStockDataReader() {  
 **return stockDataReader**;  
 }  
  
 **public** ArrayList<Double> getStockPrices() {  
 **return stockDataReader**.getPrices();  
 }  
  
}

**package** Thesis.View;  
  
**import** javafx.scene.control.ProgressBar;  
  
*/\*\*  
 \* Created by Wiktor Bednarek on 2017-08-17.  
 \*/***public class** SimulationViewTechnicalAnalysis **implements** Runnable {  
  
  
 **private int stockCounter**;  
 **private int companiesNamesSize**;  
  
 **public** SimulationViewTechnicalAnalysis(**int** stockCounter, **int** companiesNamesSize) {  
 **this**.**stockCounter** = stockCounter;  
 **this**.**companiesNamesSize** = companiesNamesSize;  
 }  
  
 **public** ProgressBar progressBar() {  
 ProgressBar simulationProgressBar = **new** ProgressBar((**double**) **stockCounter** / **companiesNamesSize**);  
  
  
 System.***out***.println((**double**) **stockCounter** / **companiesNamesSize**);  
 **return** simulationProgressBar;  
 }  
  
 @Override  
 **public void** run() {  
  
 }  
}

**package** Thesis.Model;  
  
**import** java.util.ArrayList;  
  
*/\*\*  
 \* Created by Wiktor Bednarek on 2017-08-09.  
 \*/***public class** Tools {  
  
 **public static int** indexOfMaxElement(ArrayList<Double> list) {  
 **int** maxIndex = 0;  
 **double** maxValue = list.get(0);  
 **for** (**int** i = 1; i < list.size(); ++i) {  
 **if** (maxValue < list.get(i)) {  
 maxValue = list.get(i);  
 maxIndex = i;  
 }  
 }  
  
 **return** maxIndex;  
 }  
  
 **public static int** indexOfMinElement(ArrayList<Double> list) {  
 **int** minIndex = 0;  
 **double** minValue = list.get(0);  
 **for** (**int** i = 1; i < list.size(); ++i) {  
 **if** (minValue > list.get(i)) {  
 minValue = list.get(i);  
 minIndex = i;  
 }  
 }  
  
 **return** minIndex;  
 }  
  
  
 */\*\*  
 \* Gaussian calculation.  
 \* Probability density function for Normal distribution  
 \* Based on http://introcs.cs.princeton.edu/java/22library/Gaussian.java.html. Retrieved June 4 2017.  
 \*/* **public static double** Gaussian(**double** val) {  
  
 **return** Math.*exp*(-val \* val / 2) / Math.*sqrt*(2 \* Math.***PI***);  
 }  
  
 */\*\*  
 \* Normal distribution calculation Taylor approximation  
 \* Based on http://introcs.cs.princeton.edu/java/22library/Gaussian.java.html. Retrieved June 4.  
 \*/* **public static** Double cumulativeDistributionFunction(**double** val) {  
  
 **if** (val < -8.0) **return** 0.0;  
 **if** (val > 8.0) **return** 1.0;  
 **double** sum = 0.0;  
 **double** tmp = val;  
 **try** {  
  
 **for** (**int** i = 3; sum + tmp != sum; i += 2) {  
 sum += tmp;  
 tmp = tmp \* val \* val / i;  
 }  
  
  
 } **catch** (Exception e) {  
 System.***err***.println(e);  
 }  
  
  
 **return** 0.5 + sum \* *Gaussian*(val);  
 }  
  
 */\*\*  
 \* Calculates Call premium  
 \*/* **public static double** CalculateBlackScholesModel(**double** S, **int** timeInDays, **double** K, **double** r, **int** s) {  
  
 *//Time Until Option Exercise in years* **double** t = timeInDays / 365.0;  
 **double** d1;  
 **double** d2;  
 **double** CallPremium;  
  
 d1 = (Math.*log*(S / K) + (r + (Math.*pow*(s, 2) / 2)) \* t) / (s \* Math.*sqrt*(t));  
 d2 = d1 - s \* Math.*sqrt*(t);  
 CallPremium = (S \* *cumulativeDistributionFunction*(d1)) - (*cumulativeDistributionFunction*(d2) \* K \* Math.*exp*(-r \* t));  
 **return** CallPremium;  
  
 }  
  
  
 */\*\*  
 \* See https://stackoverflow.com/questions/373186/mathematical-function-differentiation-with-c/373265#373265. Retrieved June 4 2017.  
 \*  
 \** ***@return*** *\*/* **public static** ArrayList<Double> calculateDerivative(ArrayList<Double> toDerivate) {  
 **int** step = 1;  
 ArrayList<Double> derivedVector = **new** ArrayList<Double>(toDerivate.size());  
 **if** (toDerivate.isEmpty() == **false**) {  
 **for** (**int** i = 1; i < toDerivate.size() - 1; ++i) {  
 derivedVector.add((toDerivate.get(i + 1) - toDerivate.get(i - 1)) / 2 \* step);  
 }  
 } **else** {  
 System.***out***.println(**"Vector to derivate is empty"**);  
 }  
  
 **return** derivedVector;  
 }  
  
  
 **public static** ArrayList<Double> calculateSecondDerivative(ArrayList<Double> toDerivate) {  
 **return** *calculateDerivative*(*calculateDerivative*(toDerivate));  
 }  
  
  
  
  
}

**package** Thesis.Model;  
  
**import** org.apache.commons.math3.stat.regression.SimpleRegression;  
  
**import** java.util.ArrayList;  
  
*/\*\*  
 \* Created by Wiktor Bednarek on 2017-08-12.  
 \*/***public class** TechnicalAnalysisTools {  
  
  
 **private static int** *movingAveragePeriodNumber*;  
  
  
 **private static** Double regressionSlope;  
  
 **public** TechnicalAnalysisTools() {  
  
 regressionSlope = 0.0;  
  
 }  
  
 **public static** ArrayList<Double> movingAverage(ArrayList<Double> dataVector, **int** periodNumber) {  
 **int** counter = 0;  
 *movingAveragePeriodNumber* = periodNumber;  
 ArrayList<Double> movingAverageVector = **new** ArrayList<Double>();  
 **double** tmpSum = 0;  
  
 **while** (counter < dataVector.size() - (periodNumber - 1)) {  
 **for** (**int** i = 0; i < periodNumber; ++i) {  
 tmpSum += dataVector.get(i + counter);  
 }  
  
 tmpSum /= periodNumber;  
 movingAverageVector.add(tmpSum);  
 tmpSum = 0;  
 counter++;  
 }  
  
 **return** movingAverageVector;  
 }  
  
 **public static** ArrayList<Double> simpleRegression(ArrayList<Double> toRegressionY) {  
  
 ArrayList<Double> regressionValuesVector = **new** ArrayList<Double>();  
 SimpleRegression regression = *calculateSimpleRegressionCoefficients*(toRegressionY);  
  
  
 **double** intercept = regression.getIntercept();  
 regressionSlope = regression.getSlope();  
  
 **for** (**int** i = 0; i < toRegressionY.size(); ++i) {  
 regressionValuesVector.add(intercept + regressionSlope \* (i + 1));  
 }  
  
 **return** regressionValuesVector;  
 }  
  
 **public static** SimpleRegression calculateSimpleRegressionCoefficients(ArrayList<Double> toRegressionSlope) {  
 SimpleRegression regression = **new** SimpleRegression();  
  
 **for** (**int** i = 0; i < toRegressionSlope.size(); ++i) {  
 regression.addData(i + 1, toRegressionSlope.get(i));  
 }  
  
 **return** regression;  
  
 }  
  
 **public static** Double simpleRegressionSlope(ArrayList<Double> toRegressionSlope) {  
 SimpleRegression regression = **new** SimpleRegression();  
 ArrayList<Double> regressionValuesVector = **new** ArrayList<Double>();  
  
  
 **for** (**int** i = 0; i < toRegressionSlope.size(); ++i) {  
 regression.addData(i + 1, toRegressionSlope.get(i));  
 }  
  
  
 **return** regression.getSlope();  
 }  
  
 **public static** Double getRegressionSlope(ArrayList<Double> toRegressionY) {  
  
  
 **if** (regressionSlope != **null**) {  
 **return** regressionSlope;  
 } **else** {  
 SimpleRegression regression = *calculateSimpleRegressionCoefficients*(toRegressionY);  
 regressionSlope = regression.getSlope();  
 }  
 **return** regressionSlope;  
 }  
  
 **public int** getMovingAveragePeriodNumber() {  
 **return** *movingAveragePeriodNumber*;  
 }  
  
}

**package** Thesis.Model;  
  
**import** com.jimmoores.quandl.\*;  
**import** org.threeten.bp.LocalDate;  
  
**import** java.util.ArrayList;  
**import** java.util.Collections;  
  
*/\*\*  
 \* Created by Wiktor Bednarek on 2017-08-03.  
 \*/***public class** QuotesReader {  
  
  
 **private** ArrayList<Double> **prices**;  
  
 **private** ArrayList<LocalDate> **dates**;  
  
 **private boolean isStockLoaded**;  
  
 **private** QuandlSession **session**;  
  
  
 **private** ArrayList<Double> **TechnicalAnalysisPriceData**;  
  
  
 **private** ArrayList<LocalDate> **TechnicalAnalysisDatesIndicesData**;  
  
  
 **public** QuotesReader() {  
 **isStockLoaded** = **false**;  
 *//Initializes Quandl session with selected API key* **session** = QuandlSession.*create*(**"FpXT6Px8-fcFF7jVaEA7"**);  
 }  
  
  
 **public void** showSelectedQuote(String quoteSymbol, LocalDate startDate, LocalDate endDate) {  
  
  
 **prices** = **new** ArrayList<Double>();  
 **dates** = **new** ArrayList<LocalDate>();  
  
 LocalDate START\_DATE = startDate;  
 LocalDate END\_DATE = endDate;  
  
 System.***out***.println(**"CURRENT DATE: "** + END\_DATE);  
 **final int** CLOSE\_COLUMN = 4;  
  
  
 TabularResult tabularResultMulti = **session**.getDataSets(  
 MultiDataSetRequest.Builder  
 .*of*(  
 QuandlCodeRequest.*singleColumn*(**"WIKI/"** + quoteSymbol, CLOSE\_COLUMN),  
 QuandlCodeRequest.*allColumns*(**"DOE/RWTC"**)  
 )  
 .withStartDate(START\_DATE)  
 .withEndDate(END\_DATE)  
 .withFrequency(Frequency.MONTHLY)  
 .build());  
  
  
**for** (**final** Row row : tabularResultMulti) {  
 LocalDate date = row.getLocalDate(**"Date"**);  
 Double value = row.getDouble(**"WIKI/"** + quoteSymbol + **" - Close"**);  
  
 **prices**.add(value);  
 **dates**.add(date);  
  
  
}  
  
 Collections.*reverse*(**prices**);  
 Collections.*reverse*(**dates**);  
 **isStockLoaded** = **true**;  
}  
  
 **public** Double getPriceForSelectedDate(String quoteSymbol, LocalDate selectedDate) {  
 **final int** CLOSE\_COLUMN = 4;  
  
 TabularResult tabularResultMulti = **session**.getDataSets(  
 MultiDataSetRequest.Builder  
 .*of*(  
 QuandlCodeRequest.*singleColumn*(**"WIKI/"** + quoteSymbol, CLOSE\_COLUMN),  
 QuandlCodeRequest.*allColumns*(**"DOE/RWTC"**)  
 )  
 .withStartDate(selectedDate)  
 .withEndDate(selectedDate)  
 .withFrequency(Frequency.***DAILY***)  
 .build());  
  
 System.***out***.println(**"tabularResultMulti.size(): "** + tabularResultMulti.size());  
 LocalDate date = tabularResultMulti.get(0).getLocalDate(**"Date"**);  
 Double value = tabularResultMulti.get(0).getDouble(**"WIKI/"** + quoteSymbol + **" - Close"**);  
 System.***out***.println(**"ONE Value on date "** + date + **" was "** + value);  
  
 **return** value;  
  
 }  
  
 **public void** findTheDataToCreateMinMaxValueIndicator() {  
  
 **if** (**isStockLoaded** == **true**) {  
  
  
 **double** maxStockValue = Collections.*max*(**prices**);  
 **double** minStockValue = Collections.*min*(**prices**);  
  
 **TechnicalAnalysisPriceData** = **new** ArrayList<Double>();  
 **TechnicalAnalysisPriceData**.add(minStockValue);  
 **TechnicalAnalysisPriceData**.add(maxStockValue);  
  
  
 **TechnicalAnalysisDatesIndicesData** = **new** ArrayList<LocalDate>();  
 **TechnicalAnalysisDatesIndicesData**.add(**dates**.get(Tools.*indexOfMinElement*(**prices**)));  
 **TechnicalAnalysisDatesIndicesData**.add(**dates**.get(Tools.*indexOfMaxElement*(**prices**)));  
  
 System.***out***.println();  
 System.***out***.println(**"Index of min element: "** + Tools.*indexOfMinElement*(**prices**));  
 System.***out***.println(**"Index of max element: "** + Tools.*indexOfMaxElement*(**prices**));  
  
 } **else** {  
 System.***out***.println(**"Stock is not loaded"**);  
  
 }  
  
  
 }  
  
  
 **public** ArrayList<Double> getPrices() {  
 **return prices**;  
 }  
  
 **public** ArrayList<LocalDate> getDates() {  
 **return dates**;  
 }  
  
 **public** ArrayList<Double> getTechnicalAnalysisPriceData() {  
 **return TechnicalAnalysisPriceData**;  
 }  
  
 **public** ArrayList<LocalDate> getTechnicalAnalysisDatesIndicesData() {  
 **return TechnicalAnalysisDatesIndicesData**;  
 }  
  
  
  
  
}

**package** Thesis.Model;  
  
**public class** DataBase {  
  
  
 **private final static** String[] ***dataBase*** = {  
  
 */\*  
 "AAPL",  
 "BLL",  
 "BTU",  
 \*/* **"ABBV"**,  
  
 **"ABC"**,  
 **"ADBE"**,  
 **"ADSK"**,  
  
  
 **"AET"**,  
 **"AFL"**,  
 **"AIG"**,  
 **"AIV"**,  
 **"AIZ"**,  
 **"AKAM"**,  
 **"ALL"**,  
 **"ALLE"**,  
  
 **"ALXN"**,  
 **"AMAT"**,  
 **"AME"**,  
 **"AMGN"**,  
 **"AMP"**,  
 **"AMT"**,  
 **"AMZN"**,  
 **"A"**,  
  
 **"AON"**,  
 **"APA"**,  
 **"APC"**,  
 **"APD"**,  
 **"APH"**,  
 **"AVB"**,  
  
 **"AVY"**,  
 **"AXP"**,  
 **"AZO"**,  
 **"BA"**,  
 **"BAC"**,  
 **"BAX"**,  
  
 **"BBT"**,  
 **"BBY"**,  
 **"BCR"**,  
 **"BDX"**,  
 **"BEN"**,  
 **"BF\_B"**,  
 **"BIIB"**,  
 **"BK"**,  
 **"BLK"**,  
  
 **"BMS"**,  
  
 **"BMY"**,  
  
 **"BRK\_B"**,  
 **"BSX"**,  
  
 **"BWA"**,  
 **"BXP"**,  
  
 **"IBM"**,  
  
  
 **"MSFT"**,  
 **"C"**,  
  
  
 };  
  
  
 **public** DataBase() {  
  
 }  
  
  
 **public static** String[] getDataBase() {  
 **return *dataBase***;  
 }  
  
}

**package** Thesis.Controller;  
  
**import** Thesis.Model.QuotesReader;  
**import** Thesis.Model.TechnicalAnalysisTools;  
**import** Thesis.View.TechnicalAnalysisView;  
**import** javafx.fxml.FXML;  
**import** javafx.fxml.Initializable;  
**import** javafx.scene.control.Button;  
**import** javafx.scene.control.ProgressBar;  
**import** javafx.scene.control.TextArea;  
**import** org.threeten.bp.LocalDate;  
  
**import** java.net.URL;  
**import** java.util.ArrayList;  
**import** java.util.ResourceBundle;  
  
*/\*\*  
 \* Created by Wiktor Bednarek on 2017-08-17.  
 \*/***public class** SimulationControllerTechnicalAnalysis **implements** Initializable {  
  
  
 @FXML  
 **protected** ProgressBar **simulationProgressBar**;  
  
 @FXML  
 **protected** Button **startSimulationButton**;  
  
 @FXML  
 **protected** TextArea **resultsTextArea**;  
  
 **private** LocalDate **startDateBasedOnBPLibrary**;  
 **private** LocalDate **endDateBasedOnBPLibrary**;  
 **private** Double **regressionLineSlopeValue**;  
 **private** ArrayList<Double> **stocksPrices**;  
 **private boolean isDecisionWasCorrect**;  
 **private** LocalDate **presentDate**;  
  
  
 **private** ArrayList<Double> **companiesRevenues**;  
 **private** ArrayList<String> **companiesNames**;  
 **private** ArrayList<String> **decisionMade**;  
 **private** ArrayList<String> **decisionCorrectness**;  
 **private int stockCounter**;  
 **private double wrongResultCounter**;  
 **private double rightResultCounter**;  
  
  
 **public** SimulationControllerTechnicalAnalysis() {  
  
  
 }  
  
  
 @Override  
 **public void** initialize(URL url, ResourceBundle rb) {  
  
 }  
  
  
 **public void** initializeSimulationTA(ArrayList<String> stockSymbols, LocalDate startDateBasedOnBPLibrary, LocalDate endDateBasedOnBPLibrary) {  
 **this**.**companiesNames** = stockSymbols;  
 **this**.**startDateBasedOnBPLibrary** = startDateBasedOnBPLibrary;  
 **this**.**endDateBasedOnBPLibrary** = endDateBasedOnBPLibrary;  
  
 **this**.**regressionLineSlopeValue** = 0.0;  
 **this**.**stocksPrices** = **new** ArrayList<Double>();  
 **this**.**presentDate** = LocalDate.*of*(2017, 8, 10);  
  
 **this**.**companiesRevenues** = **new** ArrayList<Double>();  
 **this**.**decisionCorrectness** = **new** ArrayList<String>();  
 **this**.**decisionMade** = **new** ArrayList<>();  
 **this**.**stockCounter** = 0;  
 **this**.**rightResultCounter** = 0;  
 **this**.**wrongResultCounter** = 0;  
  
  
  
 }  
  
  
 **public void** progressBar() {  
  
 **simulationProgressBar**.setProgress((**double**) **stockCounter** / **companiesNames**.size());  
  
 System.***out***.println((**double**) **stockCounter** / **companiesNames**.size());  
  
  
  
 }  
  
  
 **public** ArrayList<Double> dividePricesAndGetLastPricesChunk(**int** numberOfChunks) {  
  
  
  
 **int** lastChunkSize = **stocksPrices**.size() / numberOfChunks;  
 ArrayList<Double> lastChunkValues = **new** ArrayList<Double>();  
 **for** (**int** i = **stocksPrices**.size() - lastChunkSize; i < **stocksPrices**.size(); ++i) {  
 lastChunkValues.add(**stocksPrices**.get(i));  
 }  
  
 **return** lastChunkValues;  
  
 }  
  
 **public void** checkInvestmentReturn(String output) {  
  
 TechnicalAnalysisView taView = **new** TechnicalAnalysisView();  
 Double todayPrice = taView.getStockDataReader().getPriceForSelectedDate(**companiesNames**.get(**stockCounter**), **presentDate**);  
 Double theLastConsideredPrice = **stocksPrices**.get(**stocksPrices**.size() - 1);  
  
 **companiesRevenues**.add(**stockCounter**, ((todayPrice / theLastConsideredPrice) \* 100) - 100);  
  
  
 **if** (output.equals(**"BUY"**)) {  
 **if** (**companiesRevenues**.get(**stockCounter**) > 0) {  
 **isDecisionWasCorrect** = **true**;  
 decisionEvaluation(**isDecisionWasCorrect**);  
  
 } **else** {  
 **isDecisionWasCorrect** = **false**;  
 decisionEvaluation(**isDecisionWasCorrect**);  
  
 }  
  
 } **else if** (output.equals(**"SELL"**)) {  
 **if** (**companiesRevenues**.get(**stockCounter**) > 0) {  
 **isDecisionWasCorrect** = **false**;  
 decisionEvaluation(**isDecisionWasCorrect**);  
 } **else** {  
 **isDecisionWasCorrect** = **true**;  
 decisionEvaluation(**isDecisionWasCorrect**);  
 }  
  
 } **else** {  
 **if** (**companiesRevenues**.get(**stockCounter**) > 0) {  
 **isDecisionWasCorrect** = **true**;  
 decisionEvaluation(**isDecisionWasCorrect**);  
  
 } **else** {  
 **isDecisionWasCorrect** = **false**;  
 decisionEvaluation(**isDecisionWasCorrect**);  
  
 }  
 }  
  
  
 }  
  
  
 **public void** decisionEvaluation(**boolean** decisionResult) {  
  
  
 **if** (decisionResult == **true**) {  
 **rightResultCounter**++;  
 **decisionCorrectness**.add(**stockCounter**, **"right"**);  
 } **else** {  
 **wrongResultCounter**++;  
 **decisionCorrectness**.add(**stockCounter**, **"wrong"**);  
 }  
  
  
 }  
  
 **public void** decisionMaker() {  
 **int** numberOfChunks = 5;  
  
  
 Double chunkSlopeValue = TechnicalAnalysisTools.*calculateSimpleRegressionCoefficients*(dividePricesAndGetLastPricesChunk(numberOfChunks)).getSlope();  
 *//System.out.println("CHUNK SLOPE: " + chunkSlopeValue + "Regression line value: " + regressionLineSlopeValue);* String result = **""**;  
 **if** (chunkSlopeValue > 0.0 && **regressionLineSlopeValue** > 0.0) {  
 result = **"BUY"**;  
 **decisionMade**.add(**stockCounter**, result);  
 checkInvestmentReturn(result);  
  
 } **else if** (chunkSlopeValue > 0.0 && **regressionLineSlopeValue** < 0.0) {  
 result = **"BUY"**;  
 **decisionMade**.add(**stockCounter**, result);  
 checkInvestmentReturn(result);  
  
 } **else if** (chunkSlopeValue < 0.0 && **regressionLineSlopeValue** < 0.0) {  
 result = **"SELL"**;  
 **decisionMade**.add(**stockCounter**, result);  
 checkInvestmentReturn(result);  
  
 } **else** {  
 result = **"HOLD"**;  
 **decisionMade**.add(**stockCounter**, result);  
 checkInvestmentReturn(result);  
  
 }  
  
  
 }  
  
  
 **public void** performSimulation() {  
  
  
 **long** start = System.*currentTimeMillis*();  
  
 QuotesReader stockDataObject = **new** QuotesReader();  
  
 System.***out***.println(**"Companies names: "** + **companiesNames**);  
  
  
 **for** (String company : **companiesNames**) {  
  
  
 System.***out***.println(**"COMPANY name: "** + company);  
 stockDataObject.showSelectedQuote(company, **startDateBasedOnBPLibrary**, **endDateBasedOnBPLibrary**);  
 **this**.**stocksPrices** = stockDataObject.getPrices();  
 System.***out***.println(**"STOCKS PRICES: "** + **stocksPrices**);  
  
 **regressionLineSlopeValue** = TechnicalAnalysisTools.*getRegressionSlope*(**stocksPrices**);  
 decisionMaker();  
 **stockCounter**++;  
  
}  
 **double** end = (System.*currentTimeMillis*() - start) / 1000.0;  
  
 String resultOutput = **"Companies names: "** + **companiesNames** + **"\n"** +  
 **"Companies Revenues: "** + **companiesRevenues** + **"\n"** +  
 **"Decision made: "** + **decisionMade** + **"\n"** +  
 **"Decision accuracy: "** + **decisionCorrectness** + **"\n"** +  
 **"Percentage Accuracy: "** + (**rightResultCounter** / (**wrongResultCounter** + **rightResultCounter**)) \* 100.0 + **"\n"** +  
 **"Time elapsed: "** + end + **" seconds"**;  
 System.***out***.println(resultOutput  
 );  
 **this**.**resultsTextArea**.setText(resultOutput);  
  
 }  
  
  
}

**package** Thesis.Controller;  
  
**import** Thesis.Model.DataBase;  
**import** Thesis.Model.TechnicalAnalysisTools;  
**import** Thesis.Model.Tools;  
**import** Thesis.View.TechnicalAnalysisView;  
**import** javafx.fxml.FXML;  
**import** javafx.fxml.FXMLLoader;  
**import** javafx.fxml.Initializable;  
**import** javafx.scene.Parent;  
**import** javafx.scene.Scene;  
**import** javafx.scene.chart.CategoryAxis;  
**import** javafx.scene.chart.LineChart;  
**import** javafx.scene.chart.NumberAxis;  
**import** javafx.scene.chart.XYChart;  
**import** javafx.scene.control.Button;  
**import** javafx.scene.control.DatePicker;  
**import** javafx.scene.control.TextArea;  
**import** javafx.scene.control.TextField;  
**import** javafx.scene.input.KeyCode;  
**import** javafx.scene.input.KeyEvent;  
**import** javafx.stage.Stage;  
**import** org.threeten.bp.LocalDate;  
  
**import** java.io.IOException;  
**import** java.net.URL;  
**import** java.time.format.DateTimeFormatter;  
**import** java.util.ArrayList;  
**import** java.util.Arrays;  
**import** java.util.Locale;  
**import** java.util.ResourceBundle;  
  
  
**public class** Controller **implements** Initializable {  
  
 */\*\*  
 \* Object {****@link*** *javafx.scene.control.Button} searches financial instrument typed into BlackScholesSearchTextField.  
 \* It binds Graphical JavaFX Button with that object.  
 \*/* @FXML  
 **protected** Button **BlackScholesSearchButton**;  
  
 */\*\*  
 \* Object {****@link*** *javafx.scene.control.Button} calculates one parameter which left in an empty field and shows result in that field in the application's Black-Scholes tab.  
 \* It binds Graphical JavaFX Button with that object.  
 \*/* @FXML  
 **protected** Button **calculateBlackScholes**;  
  
 */\*\*  
 \* Object {****@link*** *javafx.scene.control.Button} searches financial instrument typed into BlackScholesSearchTextField.  
 \* It binds Graphical JavaFX Button with that object.  
 \*/* @FXML  
 **protected** Button **TechnicalAnalysisSearchButton**;  
  
 */\*\*  
 \* Object {****@link*** *javafx.scene.control.Button} performs the Technical analysis for selected financial instrument. When button is clicked the lines on the chart in the application's Technical Analysis tab are showed.  
 \* It binds Graphical JavaFX Button with that object.  
 \*/* @FXML  
 **protected** Button **performTechnicalAnalysisButton**;  
  
 */\*\*  
 \* Object {****@link*** *javafx.scene.control.TextField} input text field for a selected financial instrument to find in the application's Black-Scholes tab.  
 \* It binds Graphical JavFX TextFlied with that object.  
 \*/* @FXML  
 **protected** TextField **BlackScholesSearchTextField**;  
  
 */\*\*  
 \* Object {****@link*** *javafx.scene.control.TextField} input text field for a Current stock price value.  
 \* Value can be typed from the user's keyboard into that field. If all the rest Black–Scholes–Merton model parameter are given in the application ane user requested calculation  
 \* and this field is empty, it shows calculated value of that parameter.  
 \* It binds Graphical JavFX TextFlied with that object.  
 \*/* @FXML  
 **protected** TextField **CurrentStockPrice**;  
  
 */\*\*  
 \* Object {****@link*** *javafx.scene.control.TextField} input text field for a Time until option exercise value.  
 \* Value should be an integer number showing the number of days until option exercise.  
 \* It binds Graphical JavFX TextFlied with that object.  
 \*/* @FXML  
 **protected** TextField **TimeUntilOptionExercise**;  
  
 */\*\*  
 \* Object {****@link*** *javafx.scene.control.TextField} input text field for a Exercise price value.  
 \* It binds Graphical JavFX TextFlied with that object.  
 \*/* @FXML  
 **protected** TextField **ExercisePrice**;  
  
 */\*\*  
 \* Object {****@link*** *javafx.scene.control.TextField} input text field for a Risk-free interest rate value.  
 \* It binds Graphical JavFX TextFlied with that object.  
 \*/* @FXML  
 **protected** TextField **RiskFreeInterestRate**;  
  
 */\*\*  
 \* Object {****@link*** *javafx.scene.control.TextField} input text field for a Volatility percentage value.  
 \* It binds Graphical JavFX TextFlied with that object.  
 \*/* @FXML  
 **protected** TextField **Volatility**;  
  
 */\*\*  
 \* Object {****@link*** *javafx.scene.control.TextField} input text field for a Call premium value.  
 \* It binds Graphical JavFX TextFlied with that object.  
 \*/* @FXML  
 **protected** TextField **CallPremium**;  
  
 */\*\*  
 \* Object {****@link*** *javafx.scene.control.TextField} output commentary for current option. Opinion is based on the Black–Scholes–Merton model.  
 \* It binds Graphical JavFX TextArea with that object.  
 \*/* @FXML  
 **protected** TextArea **BlackScholesOutputTextArea**;  
  
 */\*\*  
 \* Object {****@link*** *javafx.scene.control.TextField} input text field for a selected financial instrument to find in the application's Technical Analysis tab.  
 \* It binds Graphical JavFX TextFlied with that object.  
 \*/* @FXML  
 **protected** TextField **TechnicalAnalysisSearchTextField**;  
  
 */\*\*  
 \* Object {****@link*** *javafx.scene.control.TextField} output commentary for current financial instrument. Opinion is based on the Technical Analysis.  
 \* It binds Graphical JavFX TextArea with that object.  
 \*/* @FXML  
 **protected** TextArea **TechnicalAnalysisOutputTextArea**;  
  
  
 */\*\*  
 \* Object {****@link*** *javafx.scene.chart.CategoryAxis} stores data presented on the X axis in the Technical Analysis tab.  
 \* This axis represents selected stock quote price.  
 \* It binds Graphical JavFX TextArea with that object.  
 \*/* @FXML  
 **protected** CategoryAxis **TechnicalAnalysisXAxis**;  
  
  
 */\*\*  
 \* Object {****@link*** *javafx.scene.chart.NumberAxis} stores data presented on the Y axis in the Technical Analysis tab.  
 \* This axis represents date.  
 \* It binds Graphical JavFX TextArea with that object.  
 \*/* @FXML  
 **protected** NumberAxis **TechnicalAnalysisYAxis**;  
  
  
 */\*\*  
 \* Object {****@link*** *javafx.scene.chart.LineChart} graphical representation of the prices of selected stock quote in the Technical Analysis tab in the specific dates.  
 \* It binds Graphical JavFX TextArea with that object.  
 \*/* @FXML  
 **protected** LineChart<?, ?> **TechnicalAnalysisLineChart**;  
  
  
 @FXML  
 **protected** DatePicker **startDateTechnicalAnalysis**;  
  
 @FXML  
 **protected** DatePicker **endDateTechnicalAnalysis**;  
  
 @FXML  
 **protected** Button **derivativeButton**;  
  
 @FXML  
 **protected** Button **showAnalysisReportButton**;  
  
 @FXML  
 **protected** Button **movingAverageButton**;  
  
 @FXML  
 **protected** Button **regressionLineButton**;  
  
  
 **protected** String **symbolOfStockInTechnicalAnalysis**;  
 **private** TechnicalAnalysisView **taView**;  
 **private boolean isTechnicalAnalysisStockLoaded**;  
 **private boolean isTechnicalAnalysisTrendLineDisplayed**;  
 **private boolean areTechnicalAnalysisDerivativesShowed**;  
 **private boolean isMovingAverageDisplayed**;  
 **private boolean areDerivativesVisible**;  
 **private boolean isRegressionLineDisplayed**;  
 **private** DateTimeFormatter **formatDate**;  
 *//Decision making relevant values* **private** XYChart.Series **firstDerivative**;  
 **private** XYChart.Series **secondDerivative**;  
 **private** Double **regressionLineSlope**;  
 **private** ArrayList<Double> **stockPrices**;  
  
 **public** Controller() {  
 **isTechnicalAnalysisStockLoaded** = **false**;  
 **isTechnicalAnalysisTrendLineDisplayed** = **false**;  
 **areTechnicalAnalysisDerivativesShowed** = **false**;  
 **isMovingAverageDisplayed** = **false**;  
 **areDerivativesVisible** = **false**;  
 **isRegressionLineDisplayed** = **false**;  
 **endDateTechnicalAnalysis** = **new** DatePicker();  
 **startDateTechnicalAnalysis** = **new** DatePicker();  
  
 *//endDateTechnicalAnalysis.setValue(LocalDate.parse(formatDate.format(LocalDate.now())));* }  
  
 */\*\*  
 \* Overloads initialize in order to set initial data in the application GUI.  
 \*/* @Override  
 **public void** initialize(URL url, ResourceBundle rb) {  
 **startDateTechnicalAnalysis**.setValue(java.time.LocalDate.*of*(2011, 8, 10));  
 *//endDateTechnicalAnalysis.setValue(java.time.LocalDate.now().minusDays(1));* **endDateTechnicalAnalysis**.setValue(java.time.LocalDate.*of*(2016, 8, 10));  
  
  
 disableTechnicalAnalysisButtons(**true**);  
 }  
  
 **public void** disableTechnicalAnalysisButtons(**boolean** isDisabled) {  
 **showAnalysisReportButton**.setDisable(isDisabled);  
 **performTechnicalAnalysisButton**.setDisable(isDisabled);  
 **movingAverageButton**.setDisable(isDisabled);  
 **regressionLineButton**.setDisable(isDisabled);  
 **derivativeButton**.setDisable(isDisabled);  
  
 }  
  
 @FXML  
 **public void** onEnterPressedTechnicalAnalysis(KeyEvent keyEvent) {  
 **if** (keyEvent.getCode() == KeyCode.***ENTER***) {  
 showQuotes();  
 }  
 }  
  
 */\*\*  
 \* Calculates and displays Call premium  
 \*/* **public void** CalculateBlackScholesModel() {  
  
 **double** S = Double.*valueOf*(**CurrentStockPrice**.getText());  
 **int** timeInDays = Integer.*valueOf*(**TimeUntilOptionExercise**.getText());  
 **double** K = Double.*valueOf*(**ExercisePrice**.getText());  
 **double** r = Double.*valueOf*(**RiskFreeInterestRate**.getText());  
 **int** s = Integer.*valueOf*(**Volatility**.getText());  
  
 **CallPremium**.setText(String.*format*(Locale.***ENGLISH***, **"%.4f"**, Double.*valueOf*(Tools.*CalculateBlackScholesModel*(S, timeInDays, K, r, s))));  
  
  
 }  
  
 **public void** showQuotes() {  
 disableTechnicalAnalysisButtons(**true**);  
 **TechnicalAnalysisLineChart**.getData().clear();  
 **symbolOfStockInTechnicalAnalysis** = **TechnicalAnalysisSearchTextField**.getText();  
  
 *//Create org.threeten.bp.LocalDate objects from java.time.LocalDate. Quandl4j requires org.threeten.bp.LocalDate not java.time.LocalDate.* LocalDate startDateBasedOnBPLibrary = LocalDate.*of*(**startDateTechnicalAnalysis**.getValue().getYear(), **startDateTechnicalAnalysis**.getValue().getMonthValue(), **startDateTechnicalAnalysis**.getValue().getDayOfMonth());  
 LocalDate endDateBasedOnBPLibrary = LocalDate.*of*(**endDateTechnicalAnalysis**.getValue().getYear(), **endDateTechnicalAnalysis**.getValue().getMonthValue(), **endDateTechnicalAnalysis**.getValue().getDayOfMonth());  
  
 **taView** = **new** TechnicalAnalysisView(**symbolOfStockInTechnicalAnalysis**, startDateBasedOnBPLibrary, endDateBasedOnBPLibrary);  
 **TechnicalAnalysisLineChart**.getData().addAll(**taView**.prepareTheTechnicalAnalysisChart());  
  
 *//Set all interface boolean* **isTechnicalAnalysisStockLoaded** = **true**;  
 setTechnicalAnalysisDisplayStatusValues(**false**);  
 calculateAllTechnicalAnalysisRelevantValues();  
 disableTechnicalAnalysisButtons(**false**);  
  
  
 }  
  
 **public void** setTechnicalAnalysisDisplayStatusValues(**boolean** displayStatus) {  
 **isTechnicalAnalysisTrendLineDisplayed** = displayStatus;  
 **areTechnicalAnalysisDerivativesShowed** = displayStatus;  
 **isMovingAverageDisplayed** = displayStatus;  
 **isRegressionLineDisplayed** = displayStatus;  
 }  
  
 **public void** calculateAllTechnicalAnalysisRelevantValues() {  
 **firstDerivative** = **taView**.firstDerivative();  
 **secondDerivative** = **taView**.secondDerivative();  
 **stockPrices** = **taView**.getStockPrices();  
 **regressionLineSlope** = TechnicalAnalysisTools.*simpleRegressionSlope*(**stockPrices**);  
 }  
  
 **public void** performTechnicalAnalysis() {  
  
  
 **if** (**isTechnicalAnalysisStockLoaded** == **true** && **isTechnicalAnalysisTrendLineDisplayed** == **false**) {  
 **TechnicalAnalysisLineChart**.getData().addAll(**taView**.addTheTechnicalAnalysisDataToTheChart());  
 *// TechnicalAnalysisLineChart.getData().add(2, taView.addTheTechnicalAnalysisDataToTheChart());* **isTechnicalAnalysisTrendLineDisplayed** = **true**;  
  
 } **else** {  
 System.***out***.println(**"First load selected stock data"**);  
 }  
  
  
 }  
  
  
  
 **public void** showDerivative() {  
  
  
 **if** (**isTechnicalAnalysisStockLoaded** == **true** && **areTechnicalAnalysisDerivativesShowed** == **false**) {  
  
 **TechnicalAnalysisLineChart**.getData().addAll(**firstDerivative**);  
 **TechnicalAnalysisLineChart**.getData().addAll(**secondDerivative**);  
 **areTechnicalAnalysisDerivativesShowed** = **true**;  
} **else** {  
 System.***out***.println(**"First load selected stock data"**);  
 }  
 }  
  
  
 **public void** showMovingAverage() {  
 **if** (**isTechnicalAnalysisStockLoaded** == **true** && **isMovingAverageDisplayed** == **false**) {  
  
 **TechnicalAnalysisLineChart**.getData().add(**taView**.movingAverageSeries(3));  
 **isMovingAverageDisplayed** = **true**;  
  
 } **else** {  
 System.***out***.println(**"Moving average can't be processes. \nFirst load selected stock data"**);  
 }  
 }  
  
 **public void** showRegressionLine() {  
 **if** (**isTechnicalAnalysisStockLoaded** == **true** && **isRegressionLineDisplayed** == **false**) {  
  
 **TechnicalAnalysisLineChart**.getData().add(**taView**.regressionLineSeries());  
 **isRegressionLineDisplayed** = **true**;  
  
 } **else** {  
 System.***out***.println(**"Moving average can't be processed. \nFirst load selected stock data"**);  
 }  
 }  
  
  
 @FXML  
 **public void** showAnalysisReport() {  
  
  
 FXMLLoader loader = **new** FXMLLoader();  
 loader.setLocation(getClass().getResource(**"/analysis.fxml"**));  
  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
  
 AnalysisReportController report = loader.getController();  
 report.initializeAnalysisReportController(**symbolOfStockInTechnicalAnalysis**, **startDateTechnicalAnalysis**, **endDateTechnicalAnalysis**, **regressionLineSlope**, getStockPrices());  
  
 report.performAnalysisReport();  
  
  
 Parent reportScene = loader.getRoot();  
 Stage analysisStage = **new** Stage();  
 analysisStage.setTitle(**"Analysis Report"**);  
 analysisStage.setScene(**new** Scene(reportScene, 800, 600));  
 analysisStage.showAndWait();  
}  
  
  
 @FXML  
 **public void** makeAnalysisSimulation() {  
 LocalDate simulationStartDate = LocalDate.*of*(**startDateTechnicalAnalysis**.getValue().getYear(), **startDateTechnicalAnalysis**.getValue().getMonthValue(), **startDateTechnicalAnalysis**.getValue().getDayOfMonth());  
 LocalDate simulationEndDate = LocalDate.*of*(**endDateTechnicalAnalysis**.getValue().getYear(), **endDateTechnicalAnalysis**.getValue().getMonthValue(), **endDateTechnicalAnalysis**.getValue().getDayOfMonth());  
 String[] companies = {**"AAPL"**, **"MSFT"**, **"IBM"**, **"FB"**};  
 *//String[] companies = {"AAPL", "MSFT"};* DataBase dataBase = **new** DataBase();  
 ArrayList<String> companiesArray = **new** ArrayList<String>(Arrays.*asList*(DataBase.*getDataBase*()));  
 System.***out***.println(**"COMPANIES: "** + companiesArray);  
  
  
 FXMLLoader loader = **new** FXMLLoader();  
 loader.setLocation(getClass().getResource(**"/simulationTechnicalAnalysis.fxml"**));  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 SimulationControllerTechnicalAnalysis simulation = loader.getController();  
 simulation.initializeSimulationTA(companiesArray, simulationStartDate, simulationEndDate);  
 *//simulation.performSimulation();* Parent reportScene = loader.getRoot();  
 Stage analysisStage = **new** Stage();  
 analysisStage.setTitle(**"Analysis Report"**);  
 analysisStage.setScene(**new** Scene(reportScene, 800, 600));  
 analysisStage.showAndWait();  
  
  
 }  
  
 **public** String getSymbolOfStockInTechnicalAnalysis() {  
 **return** symbolOfStockInTechnicalAnalysis;  
 }  
  
 **public** ArrayList<Double> getStockPrices() {  
 **return** stockPrices;  
 }  
  
  
  
  
}

**package** Thesis.Controller;  
  
**import** Thesis.Model.TechnicalAnalysisTools;  
**import** Thesis.View.TechnicalAnalysisView;  
**import** javafx.fxml.FXML;  
**import** javafx.fxml.Initializable;  
**import** javafx.print.PrinterJob;  
**import** javafx.scene.Group;  
**import** javafx.scene.Node;  
**import** javafx.scene.Parent;  
**import** javafx.scene.control.Button;  
**import** javafx.scene.control.DatePicker;  
**import** javafx.scene.control.TextArea;  
**import** javafx.scene.text.Text;  
**import** javafx.scene.text.TextFlow;  
**import** javafx.stage.Stage;  
**import** org.threeten.bp.LocalDate;  
  
**import** java.net.URL;  
**import** java.util.ArrayList;  
**import** java.util.ResourceBundle;  
  
*/\*\*  
 \* Created by Wiktor Bednarek on 2017-08-13.  
 \*/***public class** AnalysisReportController **implements** Initializable {  
  
 @FXML  
 **protected** Button **printTechnicalAnalysisReport**;  
  
 @FXML  
 **protected** TextArea **analysisContent**;  
  
 @FXML  
 **protected** Text **stockDescription**;  
  
 @FXML  
 **protected** Text **conclusion**;  
  
 @FXML  
 **protected** TextFlow **report**;  
  
 @FXML  
 **protected** Group **finalReportContent**;  
  
  
 **private** Parent **root**;  
 **private** Stage **analysisStage**;  
  
 **private** DatePicker **startDateTechnicalAnalysis**;  
 **private** DatePicker **endDateTechnicalAnalysis**;  
 **private** Double **regressionLineSlopeValue**;  
 **private** String **stockSymbol**;  
 **private** ArrayList<Double> **stocksPrices**;  
 **private boolean isDecisionWasCorrect**;  
 **private** LocalDate **presentDate**;  
  
 **public** AnalysisReportController() {  
 **conclusion** = **new** Text(**""**);  
 **stockDescription** = **new** Text(**""**);  
 **finalReportContent** = **new** Group();  
 }  
  
 @Override  
 **public void** initialize(URL url, ResourceBundle rb) {  
  
  
 }  
  
  
 **public void** initializeAnalysisReportController(String stockSymbol1, DatePicker startDateTechnicalAnalysis1, DatePicker endDateTechnicalAnalysis1, Double regressionLineSlope1, ArrayList<Double> stocksPrices) {  
 **this**.**stockSymbol** = stockSymbol1;  
 **this**.**startDateTechnicalAnalysis** = startDateTechnicalAnalysis1;  
 **this**.**endDateTechnicalAnalysis** = endDateTechnicalAnalysis1;  
 **this**.**regressionLineSlopeValue** = regressionLineSlope1;  
 **this**.**stocksPrices** = stocksPrices;  
 *//presentDate = LocalDate.now().minusDays(1);* **this**.**presentDate** = LocalDate.*of*(2017, 8, 10);  
  
 }  
  
 **public** String regressionSlopeSign() {  
 String slopeValue;  
 **if** (**regressionLineSlopeValue** > 0.1) {  
 slopeValue = **"positive"**;  
 } **else if** (**regressionLineSlopeValue** < -0.1) {  
 slopeValue = **"negative"**;  
 } **else** {  
 slopeValue = **"zero"**;  
 }  
  
 **return** slopeValue;  
  
  
 }  
  
  
 **public** ArrayList<Double> dividePricesAndGetLastPricesChunk(**int** numberOfChunks) {  
 **int** lastChunkSize = **stocksPrices**.size() / numberOfChunks;  
 ArrayList<Double> lastChunkValues = **new** ArrayList<Double>();  
 **for** (**int** i = **stocksPrices**.size() - lastChunkSize; i < **stocksPrices**.size(); ++i) {  
 lastChunkValues.add(**stocksPrices**.get(i));  
 }  
  
 **return** lastChunkValues;  
  
 }  
  
 **public** String checkInvestmentReturn(String output) {  
 String investmentReturn = **""**;  
 TechnicalAnalysisView taView = **new** TechnicalAnalysisView();  
  
 Double theLastConsideredPrice = **stocksPrices**.get(**stocksPrices**.size() - 1);  
  
 Double todayPrice = taView.getStockDataReader().getPriceForSelectedDate(**stockSymbol**, **presentDate**);  
  
 Double percentageRevenue = ((todayPrice / theLastConsideredPrice) \* 100) - 100;  
  
 **if** (output.equals(**"BUY"**)) {  
 **if** (percentageRevenue > 0) {  
 **isDecisionWasCorrect** = **true**;  
 investmentReturn = decisionMessage(output, percentageRevenue, **isDecisionWasCorrect**);  
  
 } **else** {  
 **isDecisionWasCorrect** = **false**;  
 investmentReturn = decisionMessage(output, percentageRevenue, **isDecisionWasCorrect**);  
  
 }  
  
 } **else if** (output.equals(**"SELL"**)) {  
 **if** (percentageRevenue > 0) {  
 **isDecisionWasCorrect** = **false**;  
 investmentReturn = decisionMessage(output, percentageRevenue, **isDecisionWasCorrect**);  
 } **else** {  
 **isDecisionWasCorrect** = **true**;  
 investmentReturn = decisionMessage(output, percentageRevenue, **isDecisionWasCorrect**);  
 }  
  
 } **else** {  
 **if** (percentageRevenue > 0) {  
 **isDecisionWasCorrect** = **true**;  
 investmentReturn = decisionMessage(output, percentageRevenue, **isDecisionWasCorrect**);  
  
 } **else** {  
 **isDecisionWasCorrect** = **false**;  
 investmentReturn = decisionMessage(output, percentageRevenue, **isDecisionWasCorrect**);  
  
 }  
 }  
  
  
 **return** investmentReturn;  
 }  
  
  
 **public** String decisionMessage(String output, Double percentageRevenue, **boolean** decisionResult) {  
 String decision;  
 **if** (decisionResult == **true**) {  
 decision = **"right"**;  
 } **else** {  
 decision = **"wrong"**;  
 }  
  
 **return "If you would follow the opinion of "** + output + **" in the last day of the selected period i.e on "** + **endDateTechnicalAnalysis**.getValue() + **".\n"** +  
 **"Presently on "** + **presentDate** + **" your revenue would be "** + String.*format*(**"%.2f"**, percentageRevenue) + **"%.\n"** +  
 **"That means the program output decision was "** + decision;  
 }  
  
 **public void** decisionMaker() {  
 **int** numberOfChunks = 5;  
 Double chunkSlopeValue = TechnicalAnalysisTools.*calculateSimpleRegressionCoefficients*(dividePricesAndGetLastPricesChunk(numberOfChunks)).getSlope();  
 System.***out***.println(**"CHUNK SLOPE: "** + chunkSlopeValue + **"Regression line value: "** + **regressionLineSlopeValue**);  
 String result = **""**;  
 **if** (chunkSlopeValue > 0.0 && **regressionLineSlopeValue** > 0.0) {  
 result = **"BUY"**;  
 **conclusion**.setText(**"General regression line is "** + regressionSlopeSign() +  
 **" and the last 1/"** + numberOfChunks + **" chunk of considered period has growing trend\n"** +  
 **"on the market is uptrend\n"** +  
 **"consider to "** + result + **" the stocks\n\n"** + checkInvestmentReturn(result)  
 );  
 **conclusion**.setStyle(**"-fx-font-color: green; -fx-font-size: 16; -fx-fill: #2d803c"**);  
 } **else if** (chunkSlopeValue > 0.0 && **regressionLineSlopeValue** < 0.0) {  
 result = **"BUY"**;  
 System.***out***.println(**"Regression line value: "** + **regressionLineSlopeValue**);  
 **conclusion**.setText(**"General regression line is "** + regressionSlopeSign() +  
 **" and the last 1/"** + numberOfChunks + **" chunk of considered period has growing trend\n"** +  
 **"this could be good buy signal\n"** +  
 **"consider to "** + result + **" the stocks\n\n"** + checkInvestmentReturn(result)  
 );  
 **conclusion**.setStyle(**"-fx-font-color: green; -fx-font-size: 16; -fx-fill: #2d803c"**);  
 } **else if** (chunkSlopeValue < 0.0 && **regressionLineSlopeValue** < 0.0) {  
 result = **"SELL"**;  
 **conclusion**.setText(**"General regression line is "** + regressionSlopeSign() +  
 **" and the last 1/"** + numberOfChunks + **" chunk of considered period has descending trend\n"** +  
 **"there is no signals for buy stocks\n"** +  
 **"consider "** + result + **" the stocks\n\n"** + checkInvestmentReturn(result)  
 );  
 **conclusion**.setStyle(**"-fx-font-color: red; -fx-font-size: 16; -fx-fill: red"**);  
 } **else** {  
 result = **"HOLD"**;  
 **conclusion**.setText(**"General regression line is "** + regressionSlopeSign() +  
 **" and the last 1/"** + numberOfChunks + **" chunk of considered period has descending trend\n"** +  
 **"basing on those data it seems to that long-term trend is growing anyway\n"** +  
 **"consider "** + result + **" the stocks\n\n"** + checkInvestmentReturn(result)  
 );  
 **conclusion**.setStyle(**"-fx-font-color: gray; -fx-font-size: 16; -fx-fill: gray"**);  
 }  
  
  
 }  
  
  
 **public void** performAnalysisReport() {  
  
 decisionMaker();  
 **stockDescription**.setStyle(**"-fx-font-size: 16"**);  
 **stockDescription**.setText(**"Hello you have chosen "** + **stockSymbol** + **" company to analyse.\n"** +  
 *//"You have selected period from " + startDateTechnicalAnalysis " to " + endDateTechnicalAnalysis.toString()* **"Regression slope value is "** + regressionSlopeSign() +  
 **"\n\nFINAL CONCLUSION\n"**);  
  
 }  
  
  
 */\*\*  
 \* Prints analysis report  
 \* See https://ijavayou.wordpress.com/2016/02/08/javafx-easy-way-to-save-scenesnodes-as-pdf/; Retrieved August 13, 2017.  
 \*/* **public void** printReport() {  
 Node tryIt = **finalReportContent**;  
  
 PrinterJob toPrint = PrinterJob.*createPrinterJob*();  
 **if** (toPrint != **null**) {  
 toPrint.showPrintDialog(**analysisStage**);  
 toPrint.printPage(tryIt);  
 toPrint.endJob();  
  
 }  
  
 }  
  
}

*<?***xml version="1.0" encoding="UTF-8"***?>  
  
<?***import javafx.scene.control.Button***?>  
<?***import javafx.scene.control.Label***?>  
<?***import javafx.scene.control.ProgressBar***?>  
<?***import javafx.scene.control.TextArea***?>  
<?***import javafx.scene.layout.AnchorPane***?>  
<?***import javafx.scene.text.Font***?>*<**AnchorPane prefHeight="599.0" prefWidth="782.0" xmlns="http://javafx.com/javafx/8.0.112" xmlns:fx="http://javafx.com/fxml/1" fx:controller="Thesis.Controller.SimulationControllerTechnicalAnalysis"**>  
 <**children**>  
 <**ProgressBar fx:id="simulationProgressBar" layoutX="266.0" layoutY="470.0" prefHeight="18.0" prefWidth="250.0" progress="0.0"** />  
 <**Button fx:id="startSimulationButton" layoutX="579.0" layoutY="507.0" mnemonicParsing="false" onAction="#performSimulation" prefHeight="40.0" prefWidth="139.0" style="-fx-background-color: #01b71b" text="Perform simulation"** />  
 <**Label layoutX="149.0" layoutY="14.0" prefHeight="65.0" prefWidth="484.0" text="Hello, please press Perform simulation button and wait for the results"**>  
 <**font**>  
 <**Font size="14.0"** />  
 </**font**>  
 </**Label**>  
 <**TextArea fx:id="resultsTextArea" layoutX="97.0" layoutY="93.0" prefHeight="356.0" prefWidth="588.0"** />  
   
 </**children**>  
</**AnchorPane**>

*<?***xml version="1.0" encoding="UTF-8"***?>  
  
<?***import javafx.scene.control.Button***?>  
<?***import javafx.scene.Group***?>  
<?***import javafx.scene.layout.AnchorPane***?>  
<?***import javafx.scene.text.Text***?>*<**AnchorPane xmlns:fx="http://javafx.com/fxml/1" prefHeight="515.0" prefWidth="721.0"  
 xmlns="http://javafx.com/javafx/8.0.112" fx:controller="Thesis.Controller.AnalysisReportController"**>  
 <**children**>  
  
 <**Button fx:id="movingAverageButton" layoutX="558.0" layoutY="469.0" mnemonicParsing="false"  
 onAction="#printReport" prefHeight="40.0" prefWidth="149.0" style="-fx-background-color: #01b71b"  
 text="Print"**/>  
  
 <**Group fx:id="finalReportContent"**>  
  
 <**Text fx:id="stockDescription" layoutX="53.0" layoutY="86.0" text="stock description"**/>  
  
 <**Text fx:id="conclusion" fill="#6ac667" layoutX="53.0" layoutY="168.0" text="conclusion"**/>  
 </**Group**>  
 *<!--  
 Try also  
 <Group />  
 <Text fx:id="conclusion" fill="#6ac667" layoutX="146.0" layoutY="349.0" text="JUST TEXT">  
 <font>  
 <Font size="14.0" />  
 </font></Text>  
 -->* </**children**>  
</**AnchorPane**>

*<?***xml version="1.0" encoding="UTF-8"***?>  
  
<?***import javafx.scene.chart.CategoryAxis***?>  
<?***import javafx.scene.chart.LineChart***?>  
<?***import javafx.scene.chart.NumberAxis***?>  
<?***import javafx.scene.control.Button***?>  
<?***import javafx.scene.control.DatePicker***?>  
<?***import javafx.scene.control.Label***?>  
<?***import javafx.scene.control.Tab***?>  
<?***import javafx.scene.control.TabPane***?>  
<?***import javafx.scene.control.TextField***?>  
<?***import javafx.scene.layout.AnchorPane***?>*<**AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="804.0" prefWidth="958.0" style="-fx-background-color: #4682b4" xmlns="http://javafx.com/javafx/8.0.112" xmlns:fx="http://javafx.com/fxml/1" fx:controller="Thesis.Controller.Controller"**>  
 <**children**>  
 <**TabPane prefHeight="800.0" prefWidth="1000.0" tabClosingPolicy="UNAVAILABLE"**>  
 <**tabs**>  
 <**Tab text="Technical Analysis"**>  
 <**content**>  
 <**AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="785.0" prefWidth="1020.0"**>  
 <**children**>  
  
 <**LineChart fx:id="TechnicalAnalysisLineChart" layoutX="14.0" layoutY="144.0" prefHeight="495.0" prefWidth="875.0"**>  
  
 <**xAxis**>  
 <**CategoryAxis fx:id="TechnicalAnalysisXAxis" side="BOTTOM"** />  
 </**xAxis**>  
 <**yAxis**>  
 <**NumberAxis fx:id="TechnicalAnalysisYAxis" side="LEFT"** />  
 </**yAxis**>  
 </**LineChart**>  
 <**TextField fx:id="TechnicalAnalysisSearchTextField" layoutX="38.0" layoutY="23.0" onKeyPressed="#onEnterPressedTechnicalAnalysis" prefHeight="40.0" prefWidth="595.0" text="MSFT"** />  
 <**Button fx:id="TechnicalAnalysisSearchButton" layoutX="676.0" layoutY="23.0" mnemonicParsing="false" onAction="#showQuotes" prefHeight="40.0" prefWidth="88.0" style="-fx-background-color: #01b71b" text="Search"** />  
  
 *<!--  
 Different text position in the button text="&#10;Technical Analysis"  
 <Button fx:id="performTechnicalAnalysis" layoutX="340.0" layoutY="488.0" mnemonicParsing="false" prefHeight="40.0" prefWidth="149.0" style="-fx-background-color: #01b71b" text="&#10;Technical Analysis" /> -->* <**Button fx:id="performTechnicalAnalysisButton" layoutX="304.0" layoutY="658.0" mnemonicParsing="false" onAction="#performTechnicalAnalysis" prefHeight="40.0" prefWidth="149.0" style="-fx-background-color: #01b71b" text="Min-max value"** />  
  
 <**Button fx:id="showAnalysisReportButton" layoutX="684.0" layoutY="658.0" mnemonicParsing="false" onAction="#showAnalysisReport" prefHeight="40.0" prefWidth="139.0" style="-fx-background-color: #01b71b" text="Show Analysis Report"** />  
 <**DatePicker fx:id="startDateTechnicalAnalysis" layoutX="61.0" layoutY="99.0"** />  
 <**DatePicker fx:id="endDateTechnicalAnalysis" layoutX="298.0" layoutY="99.0"** />  
 <**Label layoutX="61.0" layoutY="78.0" text="Start date"** />  
 <**Label layoutX="298.0" layoutY="78.0" text="End date"** />  
 <**Button fx:id="derivativeButton" layoutX="104.0" layoutY="658.0" mnemonicParsing="false" onAction="#showDerivative" prefHeight="40.0" prefWidth="100" style="-fx-background-color: #01b71b" text="Derivatives"** />  
 <**Button fx:id="movingAverageButton" layoutX="304.0" layoutY="717.0" mnemonicParsing="false" onAction="#showMovingAverage" prefHeight="40.0" prefWidth="149.0" style="-fx-background-color: #01b71b" text="Moving average"** />  
 <**Button fx:id="regressionLineButton" layoutX="104.0" layoutY="717.0" mnemonicParsing="false" onAction="#showRegressionLine" prefHeight="40.0" prefWidth="100" style="-fx-background-color: #01b71b" text="Regression line"** />  
 <**Button fx:id="makeAnalysisSimulationButton" layoutX="684.0" layoutY="717.0" mnemonicParsing="false" onAction="#makeAnalysisSimulation" prefHeight="40.0" prefWidth="139.0" style="-fx-background-color: #01b71b" text="Make simulation"** />  
  
 </**children**>  
 </**AnchorPane**>  
 </**content**>  
 </**Tab**>  
 <**Tab text="Black-Scholes"**>  
 <**content**>  
 <**AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="800.0" prefWidth="1000.0"**>  
 <**children**>  
 <**Button fx:id="calculateBlackScholes" layoutX="52.0" layoutY="497.0" mnemonicParsing="false" onAction="#CalculateBlackScholesModel" prefHeight="40.0" prefWidth="88.0" style="-fx-background-color: #01b71b" text="Call price"** />  
 <**TextField layoutX="42.0" layoutY="173.0" prefHeight="37.0" prefWidth="150.0" text="Time until option exercise"** />  
 <**TextField layoutX="42.0" layoutY="338.0" prefHeight="37.0" prefWidth="150.0" text="Volatility"** />  
 <**TextField layoutX="42.0" layoutY="284.0" prefHeight="37.0" prefWidth="150.0" text="Risk-free interest rate"** />  
 <**TextField layoutX="42.0" layoutY="230.0" prefHeight="37.0" prefWidth="150.0" text="Exercise price"** />  
 <**TextField layoutX="42.0" layoutY="119.0" prefHeight="37.0" prefWidth="150.0" text="Current stock price"** />  
 <**TextField layoutX="42.0" layoutY="397.0" prefHeight="37.0" prefWidth="150.0" text="Call premium"** />  
 <**TextField fx:id="CurrentStockPrice" layoutX="213.0" layoutY="119.0" prefHeight="37.0" prefWidth="70.0" text="141.24"** />  
 <**TextField fx:id="TimeUntilOptionExercise" layoutX="213.0" layoutY="173.0" prefHeight="37.0" prefWidth="70.0" text="16"** />  
 <**TextField fx:id="ExercisePrice" layoutX="213.0" layoutY="230.0" prefHeight="37.0" prefWidth="70.0" text="150"** />  
 <**TextField fx:id="RiskFreeInterestRate" layoutX="213.0" layoutY="284.0" prefHeight="37.0" prefWidth="70.0" text="4"** />  
 <**TextField fx:id="Volatility" layoutX="214.0" layoutY="338.0" prefHeight="37.0" prefWidth="68.0" text="30"** />  
 <**TextField fx:id="CallPremium" layoutX="213.0" layoutY="397.0" prefHeight="37.0" prefWidth="70.0"** />  
 <**TextField fx:id="BlackScholesSearchTextField" layoutX="42.0" layoutY="23.0" prefHeight="40.0" prefWidth="595.0"** />  
 <**Button fx:id="BlackScholesSearchButton" layoutX="659.0" layoutY="23.0" mnemonicParsing="false" onAction="#showQuotes" prefHeight="40.0" prefWidth="88.0" style="-fx-background-color: #01b71b" text="Search"** />  
  
 <**DatePicker fx:id="date" layoutX="42.0" layoutY="79.0"** />  
 </**children**>  
 </**AnchorPane**>  
 </**content**>  
 </**Tab**>  
 </**tabs**>  
 </**TabPane**>  
 </**children**>  
</**AnchorPane**>

1. Own image based on images from: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQ1MGaEKMk8D\_JZdqTnwRuslKvIN4HjcxoO3jWjaw30J90YIHnzlA;<https://encryptedtbn0.gstatic.com/images?q=tbn:ANd9GcQqmE3beeLfEPIu6y1PahYrB152RlY5qbaiRugmooDsJD02JgsF>;https://cdn.pixabay.com/photo/2016/04/25/07/15/man-1351317\_960\_720.png. All pictures retrieved 19 June 2017. [↑](#footnote-ref-1)
2. Own image based on screenshot retrieved 19 June 2017 from

   http://www.nasdaq.com/symbol/aapl/option-chain [↑](#footnote-ref-2)
3. Own image based on screenshot retrieved 19 June 2017 from http://eoddata.com/stockquote/NASDAQ/AAPL.htm [↑](#footnote-ref-3)
4. Own image based on images mentioned in footnotes1 and image retrieved 19 June from

   https://upload.wikimedia.org/wikipedia/commons/thumb/c/c6/IRobot\_Roomba\_780.jpg/320px-IRobot\_Roomba\_780.jpg [↑](#footnote-ref-4)
5. Retrieved April 2017, from: http://www.investopedia.com/terms/e/exerciseprice.asp [↑](#footnote-ref-5)
6. <http://www.investopedia.com/university/technical/>. Retrieved August 16, 2017. [↑](#footnote-ref-6)
7. <https://www.quandl.com/>. Retrieved August 16, 2017. [↑](#footnote-ref-7)
8. <http://quandl4j.org/>. Retrieved August 2017. [↑](#footnote-ref-8)
9. <http://www.investopedia.com/terms/u/uptrend.asp>. Retrieved August 16, 2017. [↑](#footnote-ref-9)
10. http://www.investopedia.com/terms/d/downtrend.asp. Retrieved August 16, 2017. [↑](#footnote-ref-10)
11. [21] p. 567 [↑](#footnote-ref-11)
12. http://onlinestatbook.com/2/regression/intro.html. Retrieved August 16, 2017. [↑](#footnote-ref-12)
13. <http://commons.apache.org/proper/commons-math/>. Retrieved August 2017. [↑](#footnote-ref-13)
14. Retrieved July 2017, from: http://docs.oracle.com/javafx/2/architecture/jfxpub-architecture.htm [↑](#footnote-ref-14)
15. Retrieved July 2017, http://docs.oracle.com/javafx/2/architecture/jfxpub-architecture.htm [↑](#footnote-ref-15)
16. Retrieved July 2017, https://dzone.com/articles/bye-bye-javafx-scene-builder [↑](#footnote-ref-16)
17. Retrieved July 2017, http://gluonhq.com/products/scene-builder/ [↑](#footnote-ref-17)
18. Retrieved July 2017, https://balsamiq.com/download/ [↑](#footnote-ref-18)
19. <http://junit.org/>. Retrieved August 2017. Version of Junit framework used in my evaluation tests is 4.8.2 [↑](#footnote-ref-19)
20. <https://maven.apache.org/>. Retrieved August 2017. [↑](#footnote-ref-20)
21. <http://commons.apache.org/proper/commons-math/>. Retrieved August 2017. [↑](#footnote-ref-21)
22. <http://quandl4j.org/>. Retrieved August 2017. [↑](#footnote-ref-22)
23. Implementation based on http://introcs.cs.princeton.edu/java/22library/Gaussian.java.html. Retrieved June 4. [↑](#footnote-ref-23)
24. MANN, P. R. (2010). The Standard Normal Distribution. In INTRODUCTORY STATISTICS (7thth ed., p. 258 ). [↑](#footnote-ref-24)
25. http://introcs.cs.princeton.edu/java/22library/Gaussian.java.html. Retrieved June 4, 2017. [↑](#footnote-ref-25)
26. <http://www.statoek.wiso.uni-goettingen.de/veranstaltungen/graduateseminar/SmoothingMethods_Narodzonek-Karpowska.pdf> page18. Retrieved August 16, 2017. [↑](#footnote-ref-26)
27. <http://www.investopedia.com/terms/s/speculator.asp>. Retrieved August 17, 2017 [↑](#footnote-ref-27)
28. <https://www.stocksplithistory.com/apple/>. Retrieved August 17, 2017 [↑](#footnote-ref-28)
29. <http://www.foxbusiness.com/markets/2017/03/17/coal-miner-peabody-set-to-exit-bankruptcy.html>. Retrieved August 17, 2017. [↑](#footnote-ref-29)
30. <https://www.splithistory.com/bll/>, Retrieved August 17, 2017. [↑](#footnote-ref-30)
31. <http://www.businessinsider.com/stocks-hit-record-highs-2017-6?IR=T>. Retrieved August 17, 2017. [↑](#footnote-ref-31)
32. <https://europe.deposits.org/>. Retreived August 18, 2017. [↑](#footnote-ref-32)
33. <https://investors.pl/jak-inwestowac/abc-inwestora/emerging-markets-TcgNDv>. Retrieved August 18, 2017. [↑](#footnote-ref-33)