```
MainFrame{
    if(newStock.pressed){
     newStockEntry.Open(){
          if(submit.pressed){
               if(not Database.search(stockName)){
                    calculateFuture(Textfields.getText)
                    Database.storeData(Textfields.getText)
                    StockFuture.Open{
                         if(home.pressed){
                              MainFrame.Open
                         }else if(comapre.pressed){
                              compareStockSelection.Open(code later)
                         }else if(exit.pressed){
                              quit()
                         }
                    }
               }else{
                    existingStockErrorFrame.open(){
                         if(overideStock.pressed){
                              calculateFuture(Textfields.getText)
                              Database.storeData(values)
                              StockFuture.Open (code above)
                         }else if(editName.pressed){
                              calculateFuture(Textfields.getText)
                              Database.storeData(newName+values)
                              StockFuture.Open (code above)
                         }else if(home.pressed){
                              MainFrame.Open
                         }else if(exit.pressed){
                              quit()
                         }
                    }
               }
          }
     }
     }else if(compareStock.pressed){
          compareStockSelection.Open{
               if(go.pressed){
                    Database.getInfo(stockOne stockTwo)
                    compare(stockOne stockTwo)
                    stockComparisonDisplay.Open{
                         if(back.pressed){
                              compareStockSelection.Open
                         }
                         }else if(home.pressed){
```

```
                              MainFrame.Open
                        }else if(exit.pressed){
                              quit()
                        }

                  }
            }else if(home.pressed){
                  MainFrame.Open
            }else if(exit.pressed){
                  quit()
            }
      }
}else if(editStock.pressed){
      Database.edit(stock values)
}else if(help.pressed){
      helpFrame.Open()
}else if(exit.pressed){
      quit()
}
}
```

Each cell is a column in their respective table

DATABASE: StockInformation

TABLE: StockValues

Company (Primary Key) (VarChar)	Sector (VarChar)	Price (Float)	1 day Change (Float)	1 year high (Float)	1 year low (Float)	Enterprise value (Float)	Market Cap (Float)	Altman z score (Float)	Piotroski f score (Float)	Modified c score (Float)	***CONTINUED BELLOW***

Earning Yield (Float)	Operating Margin (Float)	Free Cash Flow (Float)	Long Term Debt (Float)	Net Worth (Float)	Dividend Yield (Float)	Enterprise Value EBIDTA (Float)	Price Sales (Float)	CashFlow (Float)

TABLE: PredictedReturn

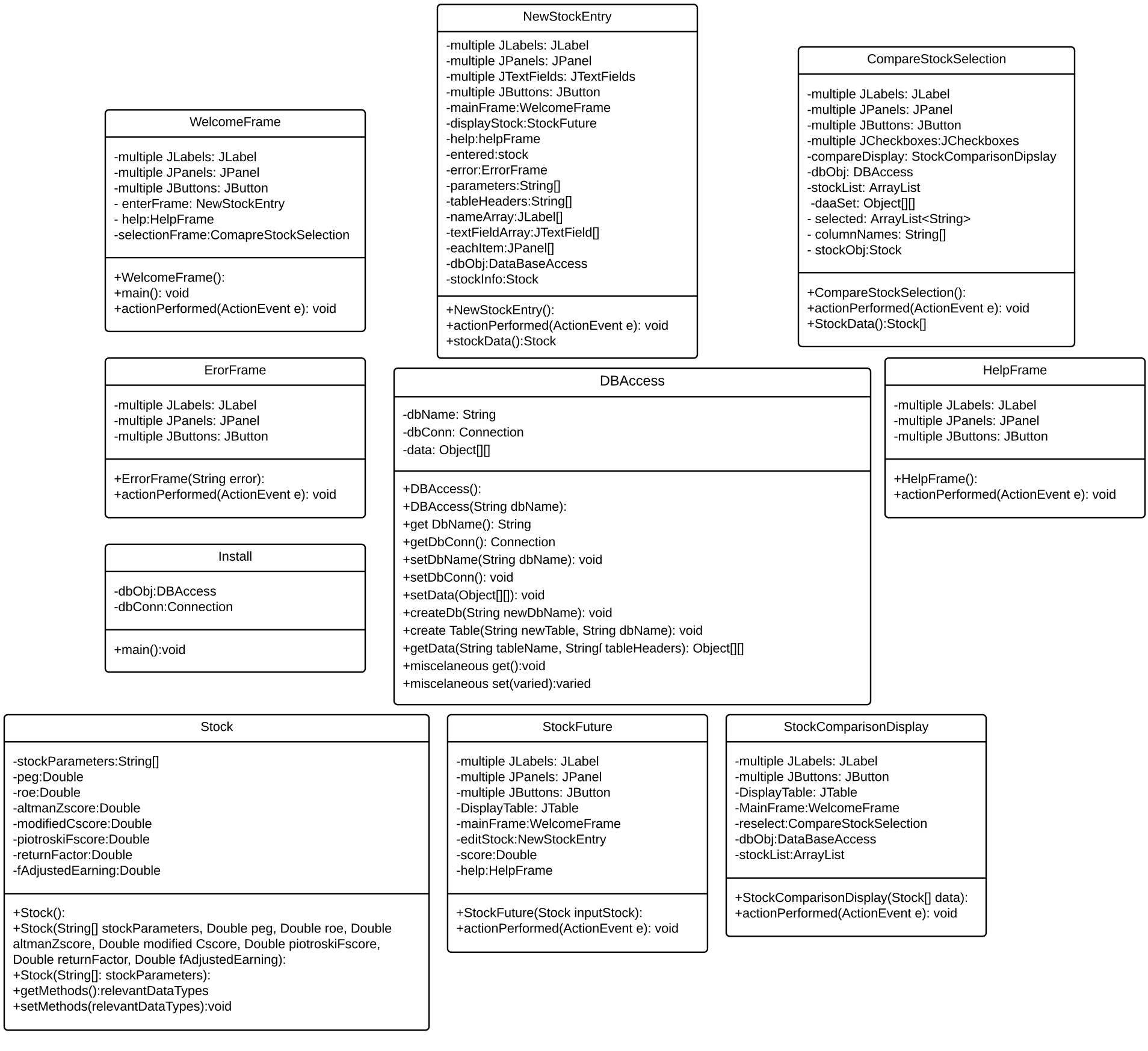
Company (Primary Key) (VarChar)	1 day return (Float)	1 week return (Float)	1 month return (Float)	1 year return (Float)	3 year return (Float)	5 year return (Float)	10 year return (Float)

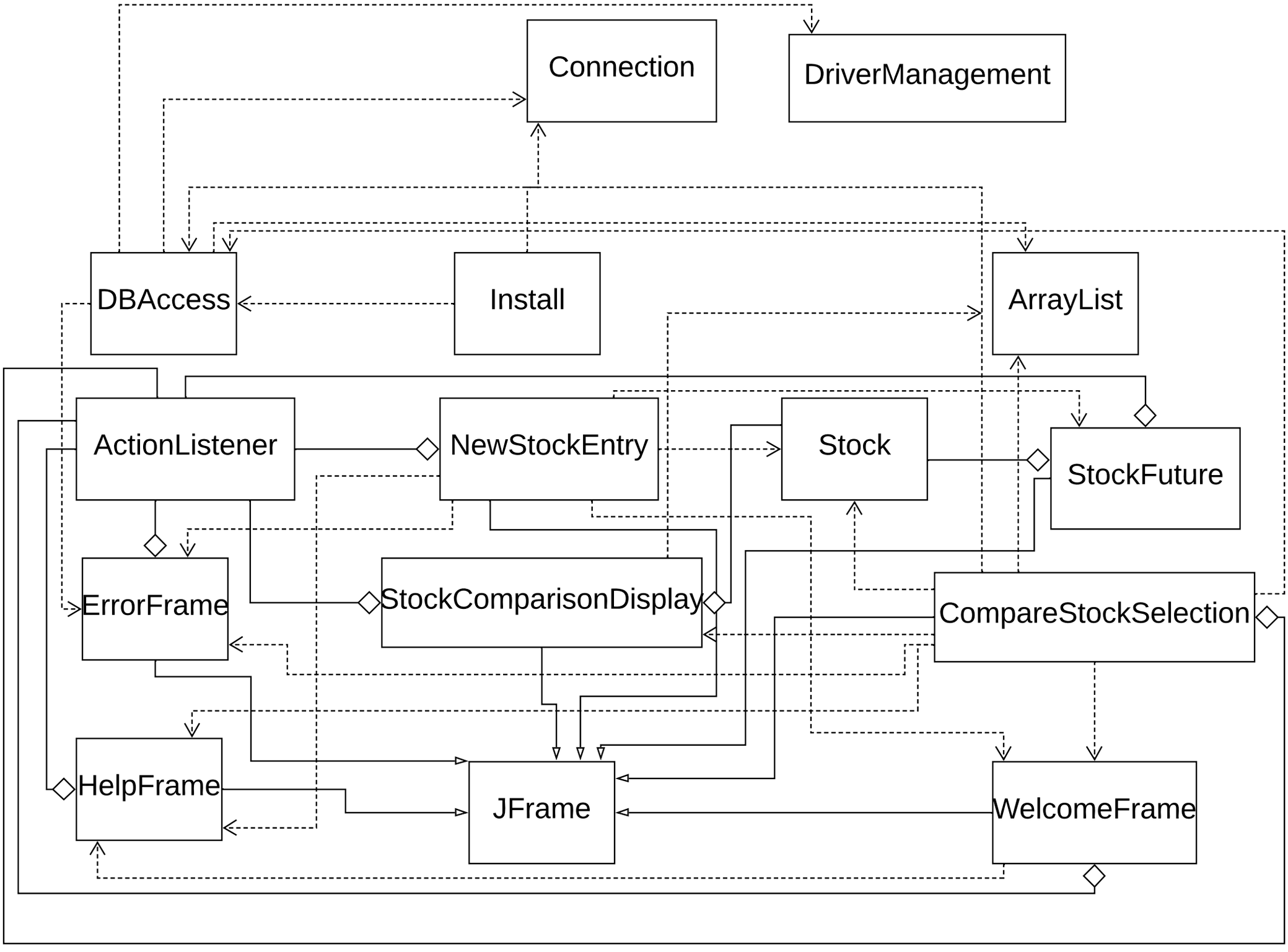
TABLE: PredictedGrowth

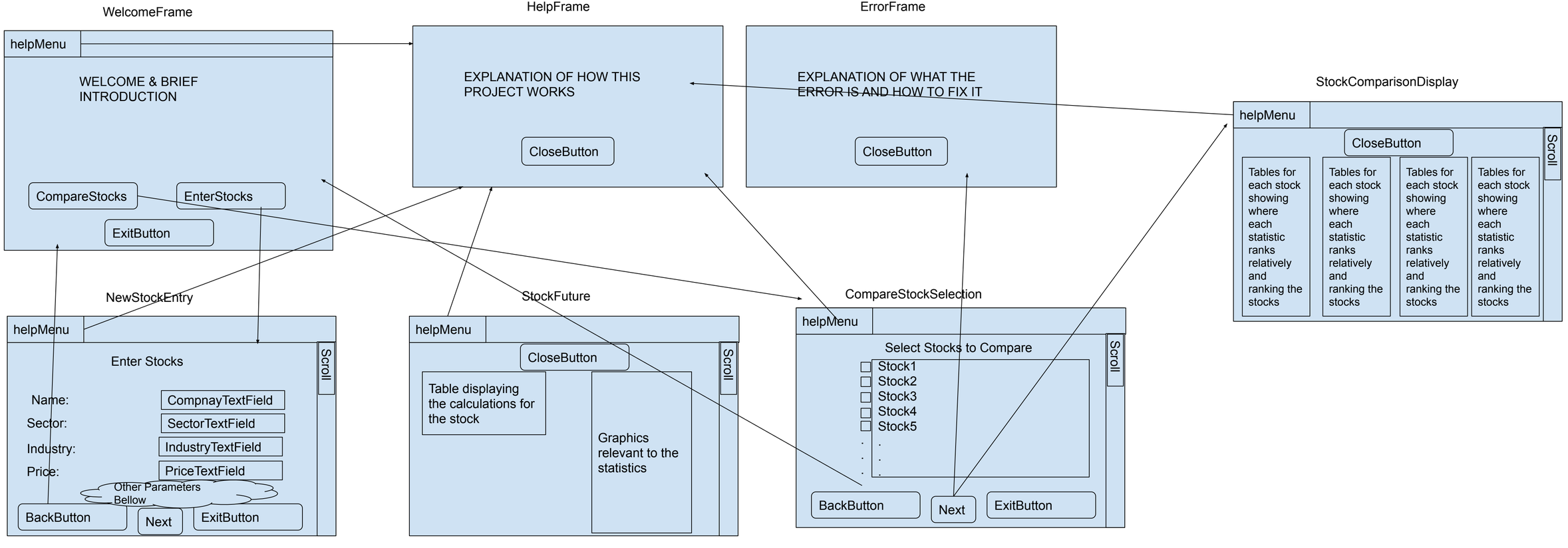
Company (Primary Key) (VarChar)	1 year revenue growth (Float)	1 year eps growth (Float)	1 year book value growth (Float)	3 year revenue growth (Float)	3 year eps growth (Float)	3 year book growth (Float)	5 year revenue growth (Float)	5 year eps growth (Float)	5 year book growth (Float)

WelcomeFrame	This is the first frame and the home frame. It will serve as a hub of activities for the user, where they can go to any place.
NewStockEntry	This will be where the user can enter information for a stock.
CompareStockSelection	This will be a frame that displays all the stocks in a database, the user can select some and then information is retrieved.
ErrorFrame	A frame used to display if an error has happened increases robustness.
DBAccess	This class will contain several methods to interact with the database.
Install	This is a separate program but will be responsible for installing the database and table.
Stock	A class that will be used to create objects of stock, it will have information about the stock as well as methods to calculate the relevant information.
StockFuture	This will come after the user has entered information in “NewStockEntry”, it will be the place that predicts the future performance of the stock as well as its future viability.
StockComparisonDisplay	This will come after the user has selected stocks from “CompareStockSelection”, it will compare the information across the stocks and determine which stock is the best one to invest in.
HelpFrame	This is a frame that provides general troubleshooting information accessible at any given point in the program.

UML Classes

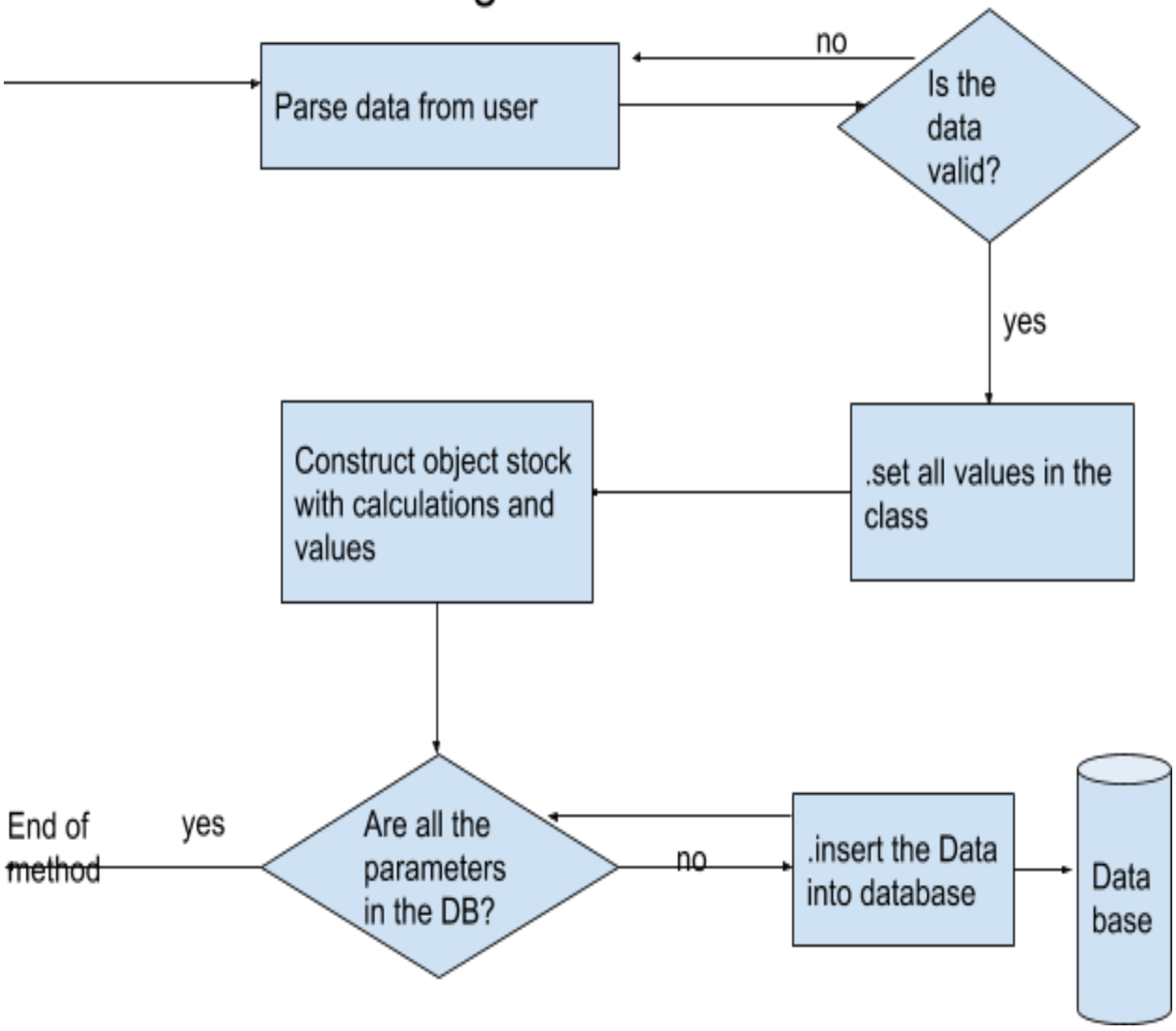

NewStockEntry
-multiple JLabels: JLabel
-multiple JPanels: JPanel
-multiple JTextFields: JTextFields
-multiple JButtons: JButton
-mainFrame:WelcomeFrame
-displayStock:StockFuture
-help:helpFrame
-entered:stock
-error:ErrorFrame
-parameters:String[]
-tableHeaders:String[]
-nameArray:JLabel[]
-textFieldArray:JTextField[]
-eachItem:JPanel[]
-dbObj:DataBaseAccess
-stockInfo:Stock
+NewStockEntry():
+actionPerformed(ActionEvent e): void
+stockData():Stock
CompareStockSelection
-multiple JLabels: JLabel
-multiple JPanels: JPanel
-multiple JButtons: JButton
-multiple JCheckboxes:JCheckboxes
-compareDisplay: StockComparisonDipslay
-dbObj: DBAccess
-stockList: ArrayList
-daaSet: Object[][]
- selected: ArrayList<String>
- columnNames: String[]
- stockObj:Stock
+CompareStockSelection():
+actionPerformed(ActionEvent e): void
+StockData():Stock[]
WelcomeFrame
-multiple JLabels: JLabel
-multiple JPanels: JPanel
-multiple JButtons: JButton
- enterFrame: NewStockEntry
- help:HelpFrame
-selectionFrame:ComapreStockSelection
+WelcomeFrame():
+main(): void
+actionPerformed(ActionEvent e): void
ErorFrame
-multiple JLabels: JLabel
-multiple JPanels: JPanel
-multiple JButtons: JButton
+ErrorFrame(String error):
+actionPerformed(ActionEvent e): void
DBAccess
-dbName: String
-dbConn: Connection
-data: Object[][]
+DBAccess():
+DBAccess(String dbName):
+get DbName(): String
+getDbConn(): Connection
+setDbName(String dbName): void
+setDbConn(): void
+setData(Object[][]): void
+createDb(String newDbName): void
+create Table(String newTable, String dbName): void
+getData(String tableName, String[tableHeaders): Object[][]
+miscelaneous get():void
+miscelaneous set(varied):varied
HelpFrame
-multiple JLabels: JLabel
-multiple JPanels: JPanel
-multiple JButtons: JButton
+HelpFrame():
+actionPerformed(ActionEvent e): void
Install
-dbObj:DBAccess
-dbConn:Connection
+main():void
Stock
-stockParameters:String[]
-peg:Double
-roe:Double
-altmanZscore:Double
-modifiedCscore:Double
-piotroskiFscore:Double
-returnFactor:Double
-fAdjustedEarning:Double
+Stock():
+Stock(String[] stockParameters, Double peg, Double roe, Double altmanZscore, Double modified Cscore, Double piotroskiFscore, Double returnFactor, Double fAdjustedEarning):
+Stock(String[]: stockParameters):
+getMethods():relevantDataTypes
+setMethods(relevantDataTypes):void
StockFuture
-multiple JLabels: JLabel
-multiple JPanels: JPanel
-multiple JButtons: JButton
-DisplayTable: JTable
-mainFrame:WelcomeFrame
-editStock:NewStockEntry
-score:Double
-help:HelpFrame
+StockFuture(Stock inputStock):
+actionPerformed(ActionEvent e): void
StockComparisonDisplay
-multiple JLabels: JLabel
-multiple JButtons: JButton
-DisplayTable: JTable
-MainFrame:WelcomeFrame
-reselect:CompareStockSelection
-dbObj:DataBaseAccess
-stockList:ArrayList
+StockComparisonDisplay(Stock[] data):
+actionPerformed(ActionEvent e): void


Connection
DriverManagement
DBAccess
Install
ArrayList
ActionListener
NewStockEntry
Stock
StockFuture
ErrorFrame
StockComparisonDisplay
CompareStockSelection
HelpFrame
JFrame
WelcomeFrame

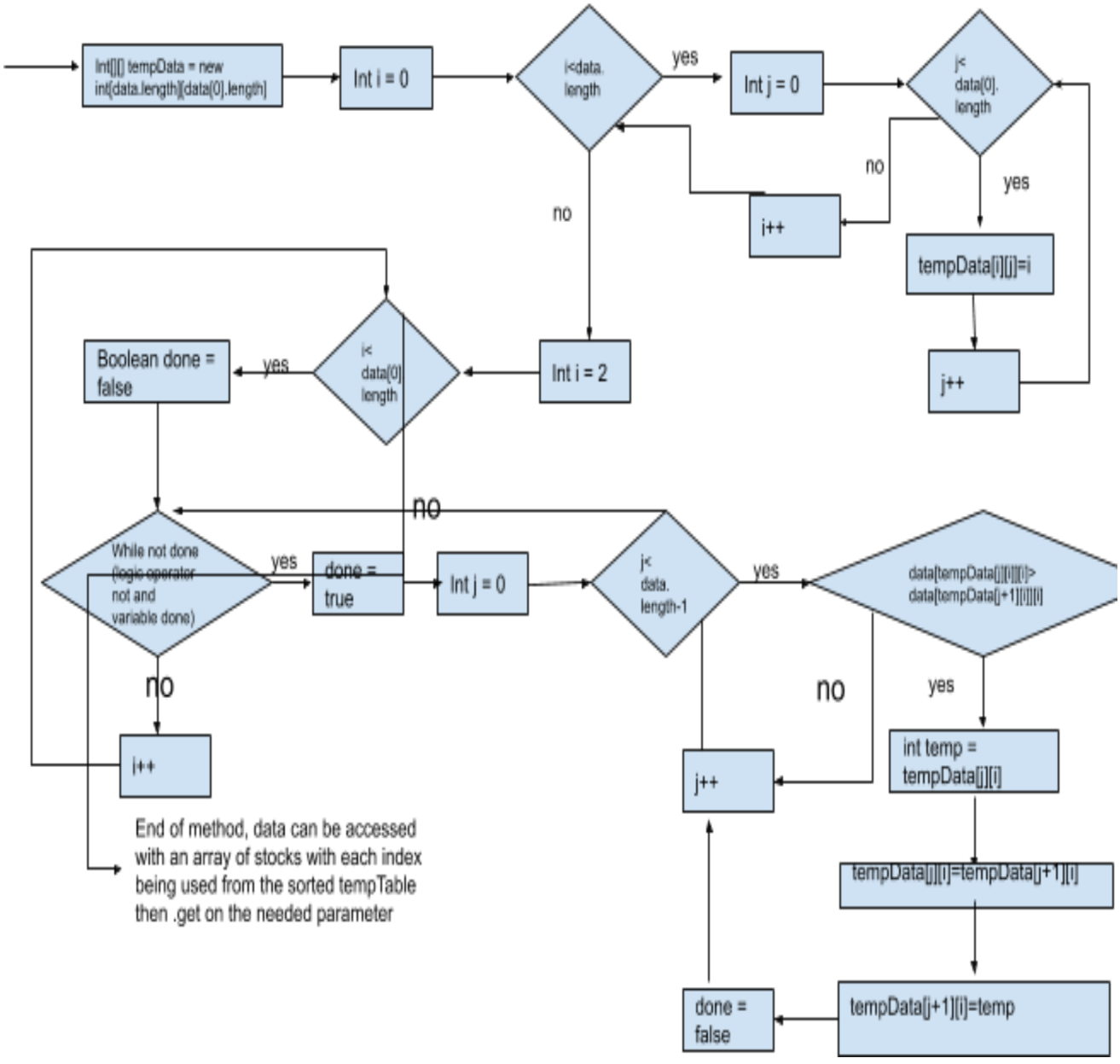

WelcomeFrame
helpMenu
WELCOME & BRIEF INTRODUCTION
CompareStocks
EnterStocks
ExitButton
HelpFrame
EXPLANATION OF HOW THIS PROJECT WORKS
CloseButton
ErrorFrame
EXPLANATION OF WHAT THE ERROR IS AND HOW TO FIX IT
CloseButton
StockComparisonDisplay
helpMenu
CloseButton
Scroll
Tables for each stock showing where each statistic ranks relatively and ranking the stocks
Tables for each stock showing where each statistic ranks relatively and ranking the stocks
Tables for each stock showing where each statistic ranks relatively and ranking the stocks
Tables for each stock showing where each statistic ranks relatively and ranking the stocks
NewStockEntry
helpMenu
Enter Stocks
Scroll
Name:
CompnayTextField
Sector:
SectorTextField
Industry:
IndustryTextField
Price:
PriceTextField
Other Parameters Bellow
BackButton
Next
ExitButton
StockFuture
helpMenu
CloseButton
Scroll
Table displaying the calculations for the stock
Graphics relevant to the statistics
CompareStockSelection
helpMenu
Select Stocks to Compare
Scroll
Stock1
Stock2
Stock3
Stock4
Stock5
BackButton
Next
ExitButton

Array	Used to store information of the stocks
Stock	Object that holds the parameters of all the stocks and has the relevant behaviors
ArrayList	Holds an indefinite amount of info that is recalled from the database or selected
Database	Stores the information about the stocks
ResultSet	Stores the database info and is used to transcribe to an ArrayList
Array[][]	Used to contain all the information of the database for the java portion to access

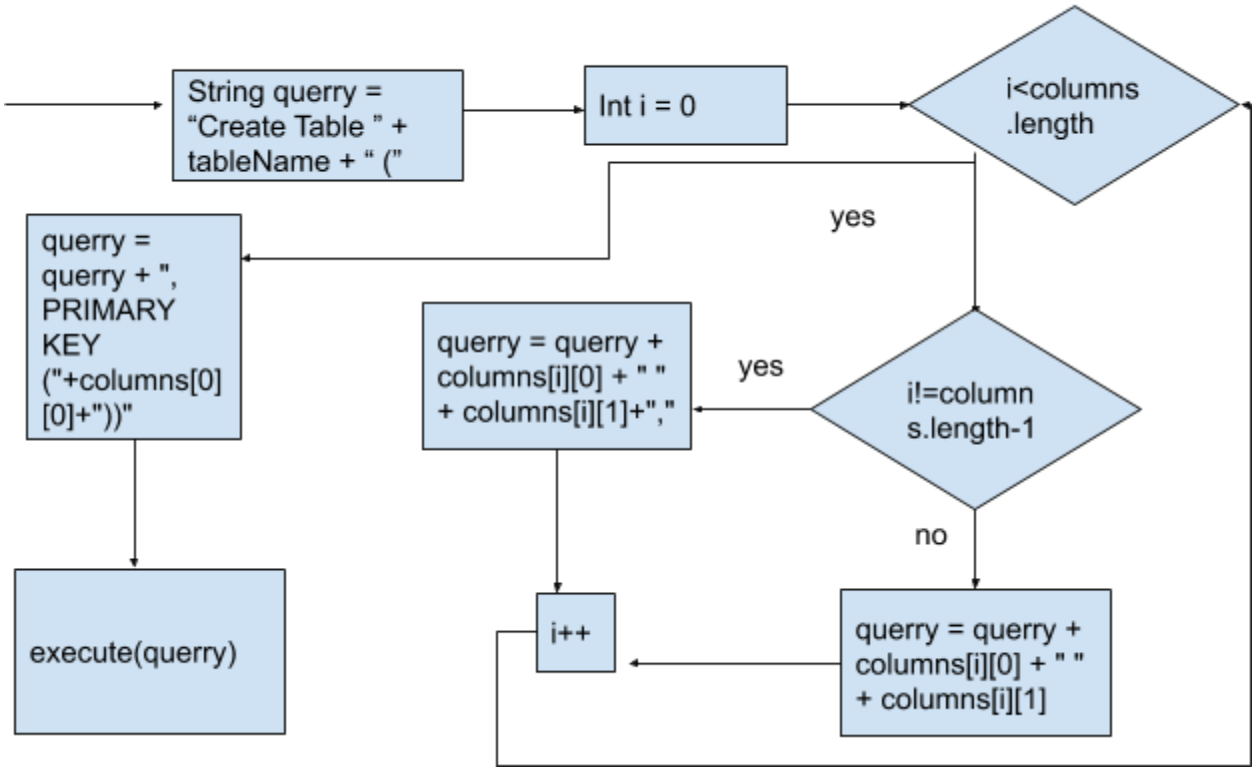
Storing Stock Information



Comparing Stock Information



Creation of a New Table



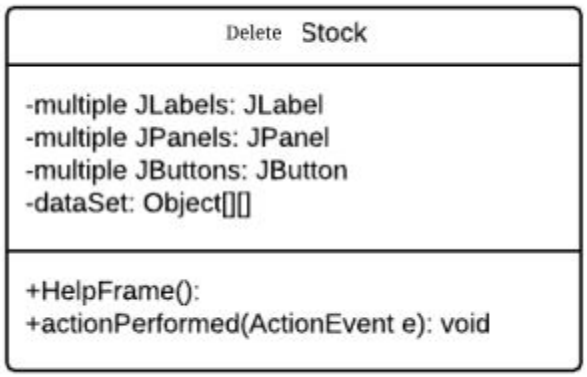
Success Criteria	Testing Plan
To be able to evaluate the performance of a stock based on input parameters	The client will test the program against previous calculations they have run
Store the results in a database	The data can be viewed in the database to see if it was properly stored
Recall results from the database	If the data accurately matches the information that was previously entered then the data was properly recalled.
Compare different stocks to determine the best one	The client will determine the rankings themself and evaluate how correct the program is
Evaluate the future performance of stocks	This can be checked against past information or the client can also verify this
Feature to overwrite previous data	Seeing if the user is able to edit the data if something is different
Rate different aspects of a stock	The client will check this against their evaluation of the stocks
It should be robust and able to handle any input	Arbitrary values can be entered without crashing the application

Amendments

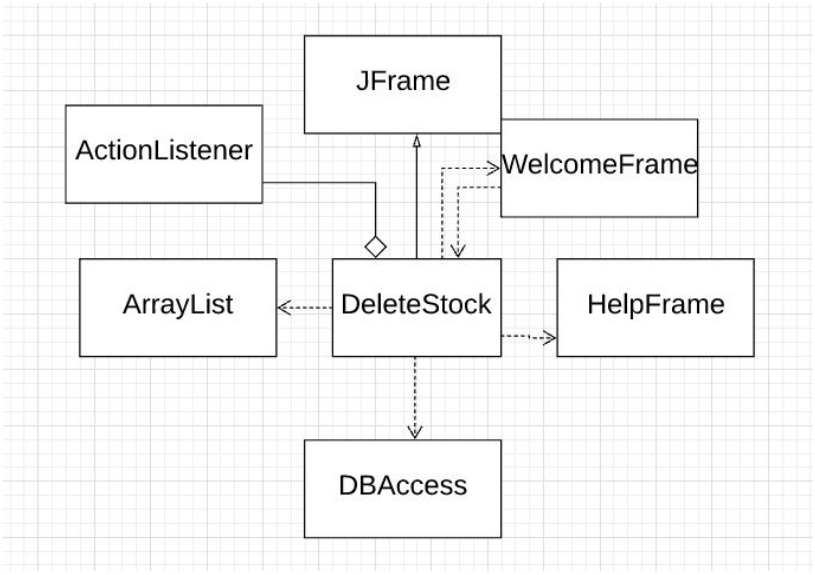
Criterion B:

- **NewStockEntry renamed to EnterStock**
- **Some minor variables (not in the UML) were added to classes IE counters**
- **Added Class Delete Stock**
  -

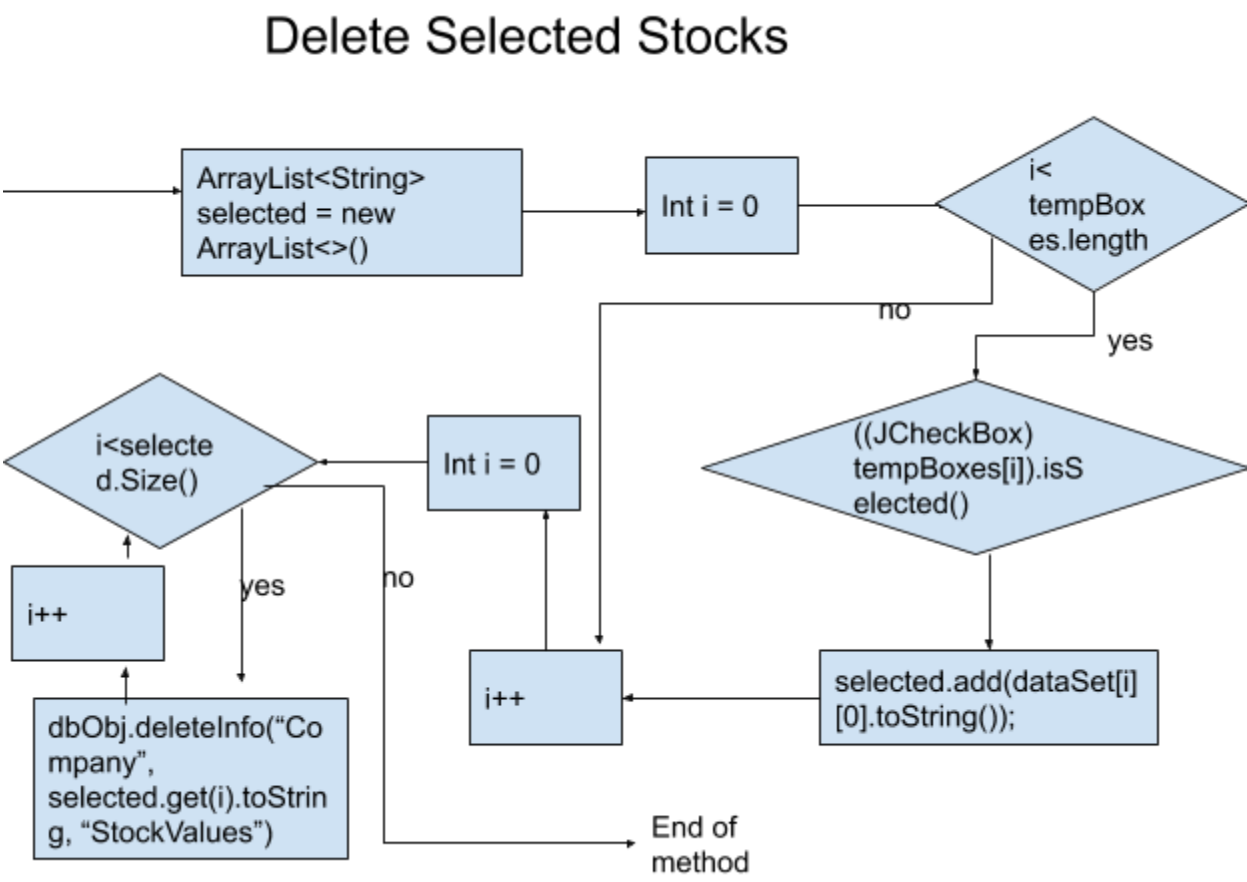
DeleteStock	This is a frame that the user will navigate to be able to delete stocks from the database.



-



-



-

- **Changed Database Columns**
  - Added columns to StockValues
    -

EPS (Float)	NetIncome (Float)	ShareholderEquity (Float)

  - Removed columns from StockValues
    -

1 day change (Float)	1 year income (Float)	1 year low (Float)