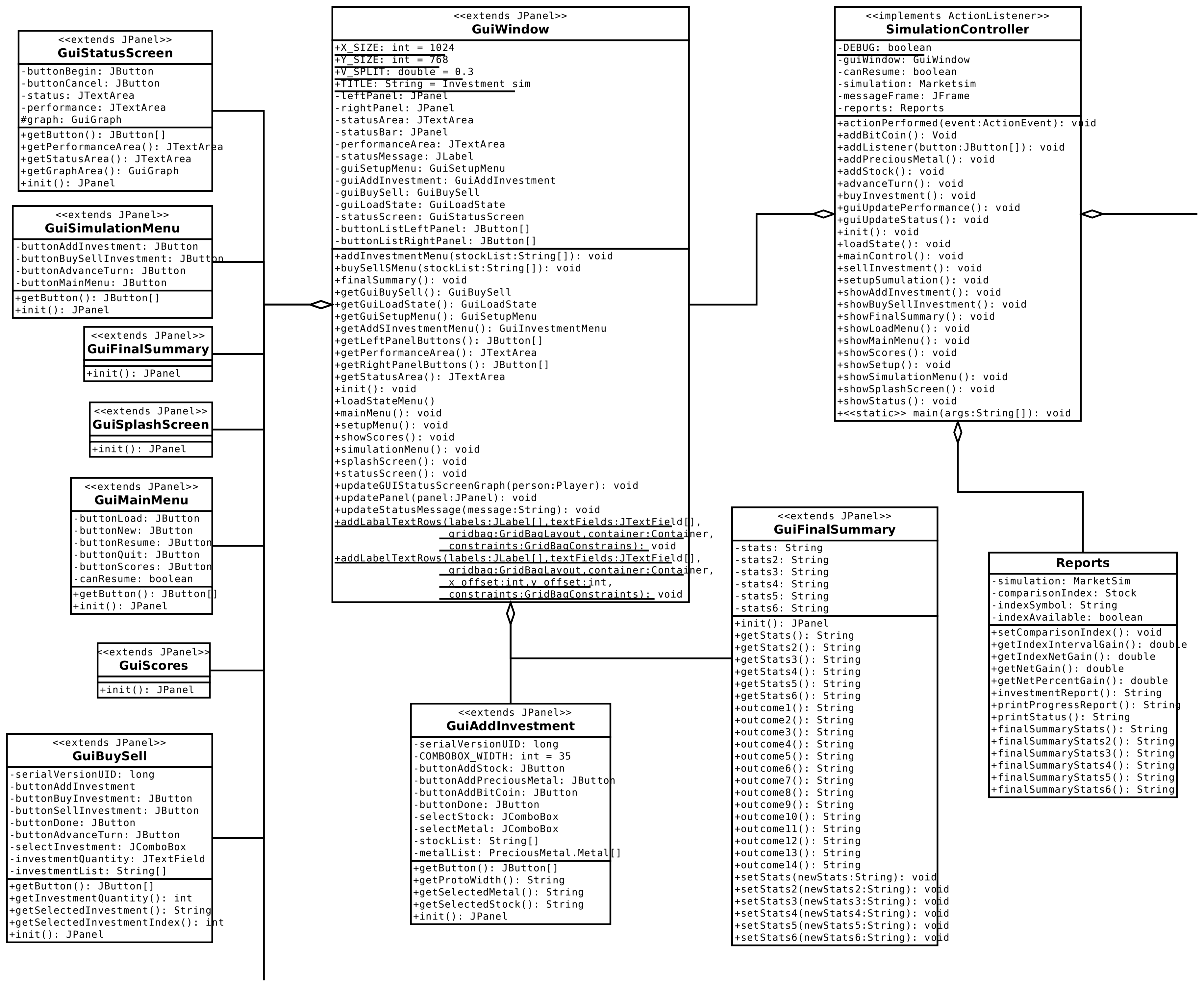
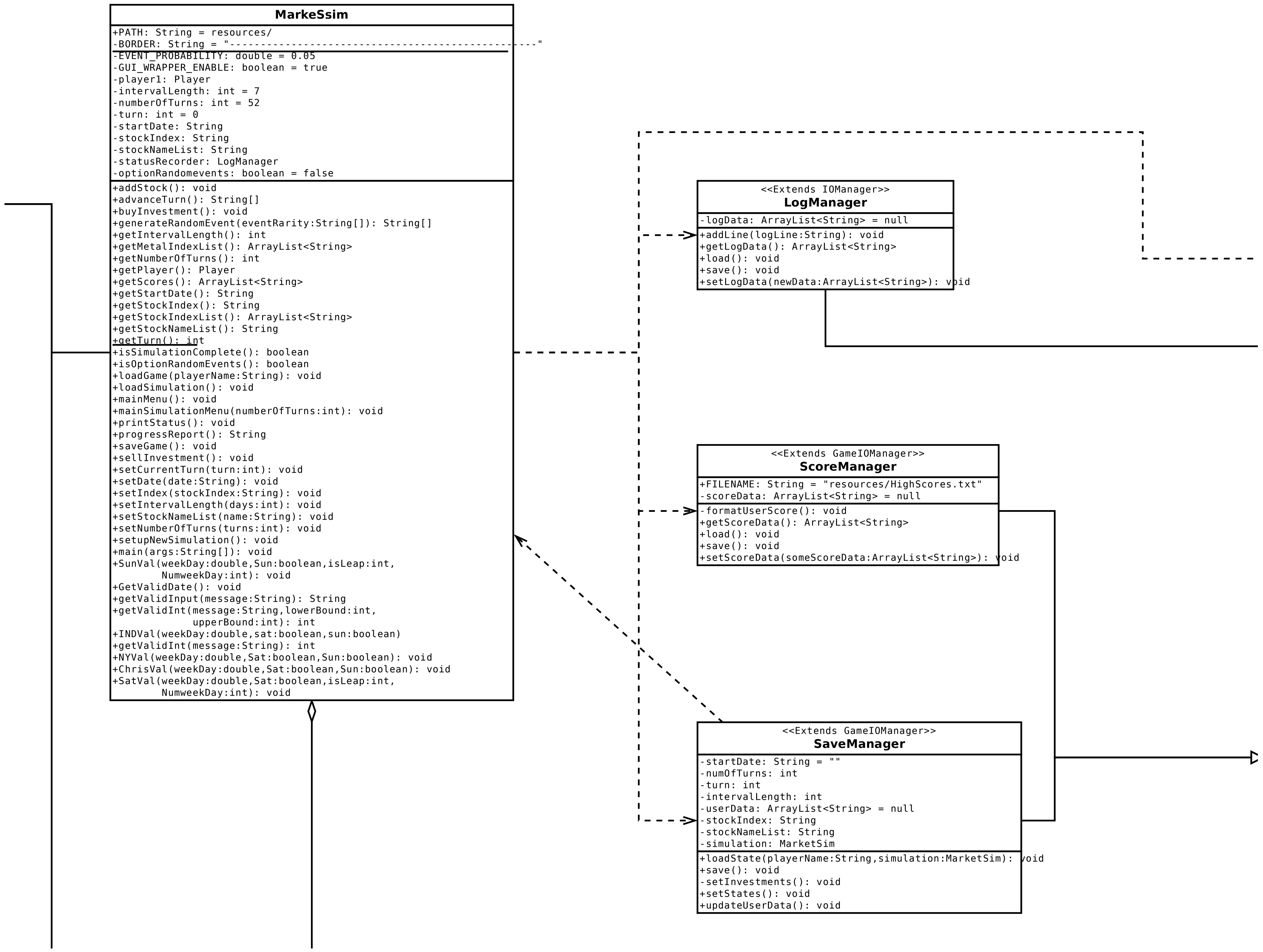
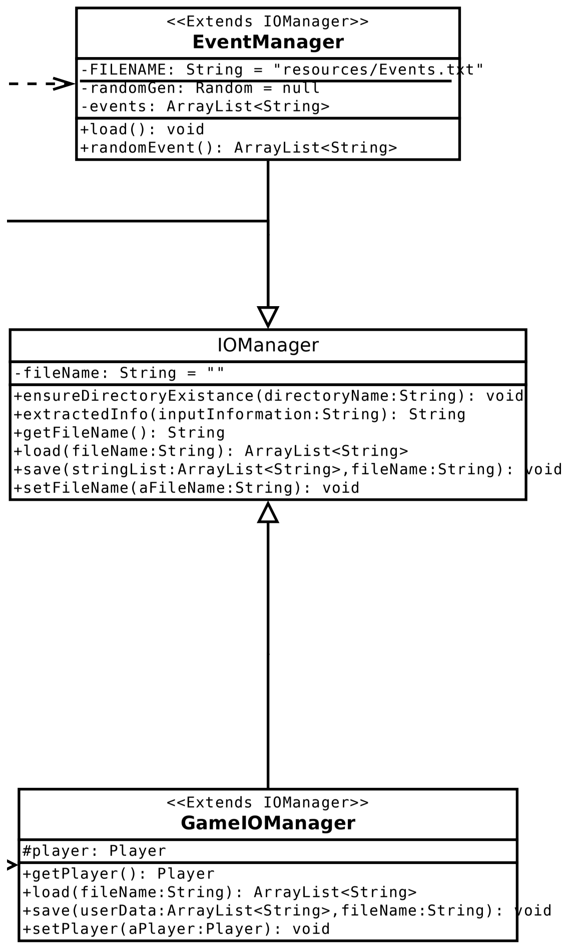
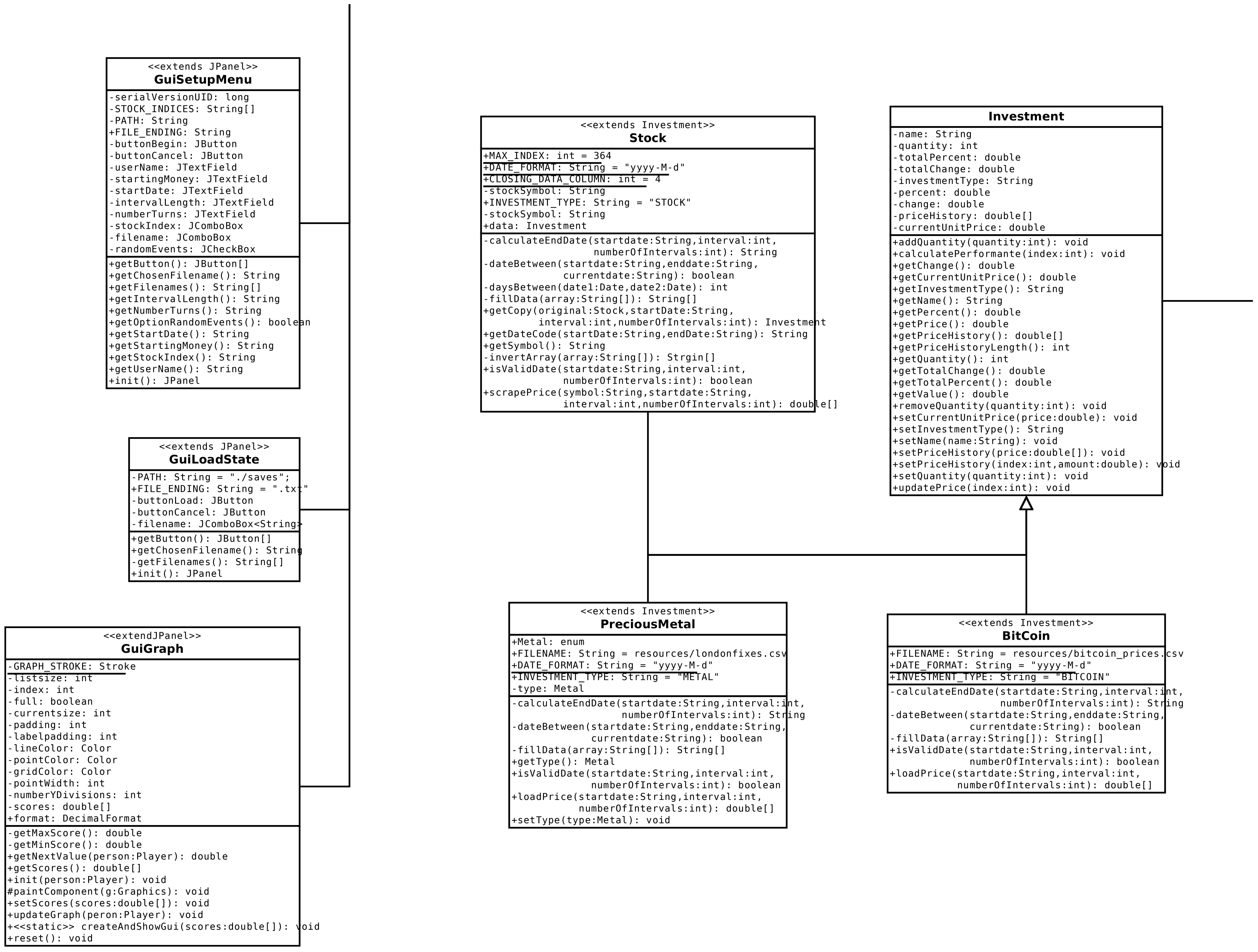
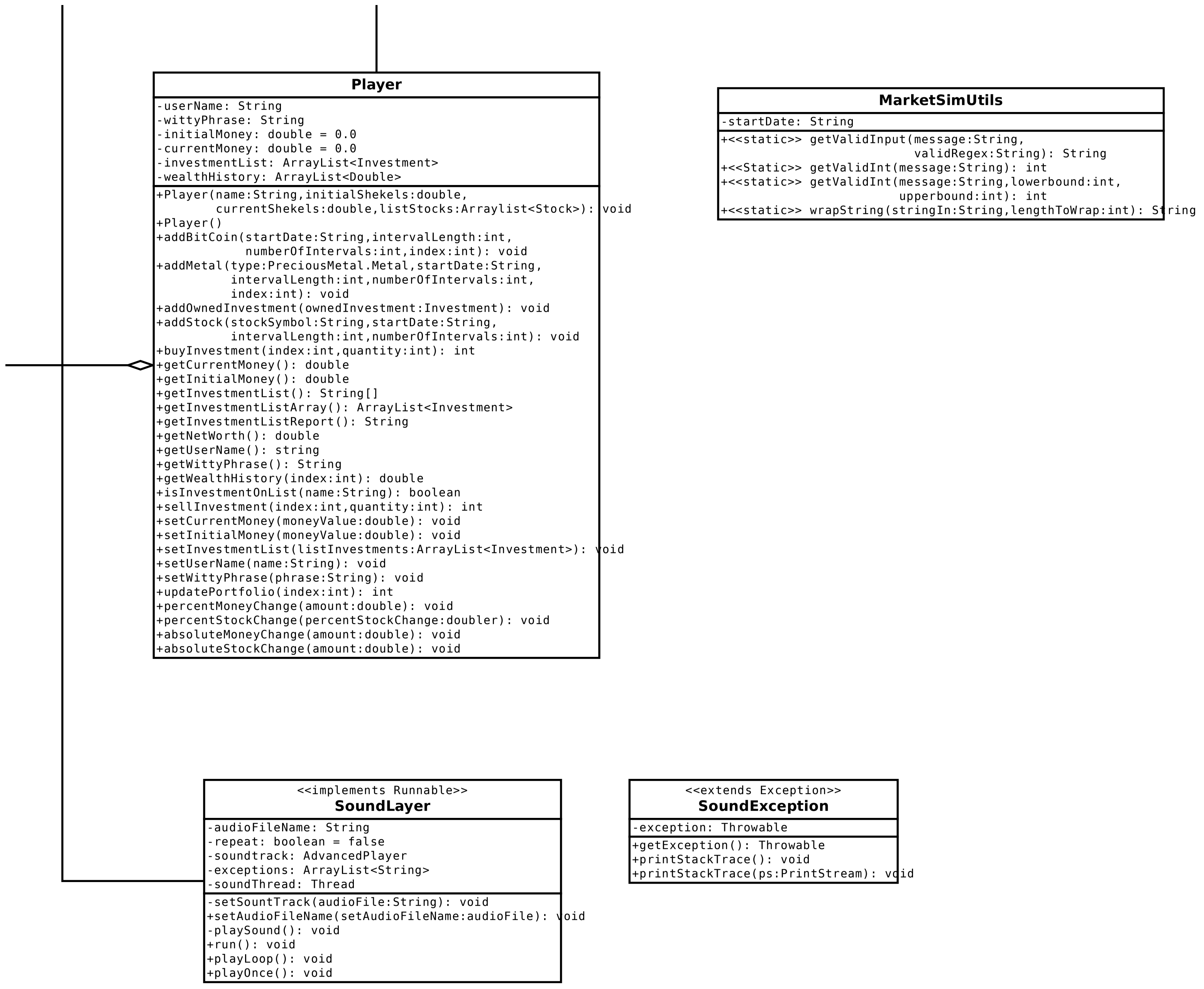

<<extends JPanel>>
GuiStatusScreen
-buttonBegin: JButton
-buttonCancel: JButton
-status: JTextArea
-performance: JTextArea
#graph: GuiGraph
+getButton(): JButton[]
+getPerformanceArea(): JTextArea
+getStatusArea(): JTextArea
+getGraphArea(): GuiGraph
+init(): JPanel
<<extends JPanel>>
GuiSimulationMenu
-buttonAddInvestment: JButton
-buttonBuySellInvestment: JButton
-buttonAdvanceTurn: JButton
-buttonMainMenu: JButton
+getButton(): JButton[]
+init(): JPanel
<<extends JPanel>>
GuiFinalSummary
+init(): JPanel
<<extends JPanel>>
GuiSplashScreen
+init(): JPanel
<<extends JPanel>>
GuiMainMenu
-buttonLoad: JButton
-buttonNew: JButton
-buttonResume: JButton
-buttonQuit: JButton
-buttonScores: JButton
-canResume: boolean
+getButton(): JButton[]
+init(): JPanel
<<extends JPanel>>
GuiScores
+init(): JPanel
<<extends JPanel>>
GuiBuySell
-serialVersionUID: long
-buttonAddInvestment
-buttonBuyInvestment: JButton
-buttonSellInvestment: JButton
-buttonDone: JButton
-buttonAdvanceTurn: JButton
-selectInvestment: JComboBox
-investmentQuantity: JTextField
-investmentList: String[]
+getButton(): JButton[]
+getInvestmentQuantity(): int
+getSelectedInvestment(): String
+getSelectedInvestmentIndex(): int
+init(): JPanel
<<extends JPanel>>
GuiWindow
+X_SIZE: int = 1024
+Y_SIZE: int = 768
+V_SPLIT: double = 0.3
+TITLE: String = Investment sim
-leftPanel: JPanel
-rightPanel: JPanel
-statusArea: JTextArea
-statusBar: JPanel
-performanceArea: JTextArea
-statusMessage: JLabel
-guiSetupMenu: GuiSetupMenu
-guiAddInvestment: GuiAddInvestment
-guiBuySell: GuiBuySell
-guiLoadState: GuiLoadState
-statusScreen: GuiStatusScreen
-buttonListLeftPanel: JButton[]
-buttonListRightPanel: JButton[]
+addInvestmentMenu(stockList:String[]): void
+buySellSMenu(stockList:String[]): void
+finalSummary(): void
+getGuiBuySell(): GuiBuySell
+getGuiLoadState(): GuiLoadState
+getGuiSetupMenu(): GuiSetupMenu
+getAddSInvestmentMenu(): GuiInvestmentMenu
+getLeftPanelButtons(): JButton[]
+getPerformanceArea(): JTextArea
+getRightPanelButtons(): JButton[]
+getStatusArea(): JTextArea
+init(): void
+loadStateMenu()
+mainMenu(): void
+setupMenu(): void
+showScores(): void
+simulationMenu(): void
+splashScreen(): void
+statusScreen(): void
+updateGUIStatusScreenGraph(person:Player): void
+updatePanel(panel:JPanel): void
+updateStatusMessage(message:String): void
+addLabalTextRows(labels:JLabel[],textFields:JTextField[], gridbag:GridBagLayout,container:Container, constraints:GridBagConstrains): void
+addLabelTextRows(labels:JLabel[],textFields:JTextField[], gridbag:GridBagLayout,container:Container, x_offset:int,y_offset:int, constraints:GridBagConstraints): void
<<extends JPanel>>
GuiAddInvestment
-serialVersionUID: long
-COMBOBOX_WIDTH: int = 35
-buttonAddStock: JButton
-buttonAddPreciousMetal: JButton
-buttonAddBitCoin: JButton
-buttonDone: JButton
-selectStock: JComboBox
-selectMetal: JComboBox
-stockList: String[]
-metalList: PreciousMetal.Metal[]
+getButton(): JButton[]
+getProtoWidth(): String
+getSelectedMetal(): String
+getSelectedStock(): String
+init(): JPanel
<<implements ActionListener>>
SimulationController
-DEBUG: boolean
-guiWindow: GuiWindow
-canResume: boolean
-simulation: Marketsim
-messageFrame: JFrame
-reports: Reports
+actionPerformed(event:ActionEvent): void
+addBitCoin(): Void
+addListener(button:JButton[]): void
+addPreciousMetal(): void
+addStock(): void
+advanceTurn(): void
+buyInvestment(): void
+guiUpdatePerformance(): void
+guiUpdateStatus(): void
+init(): void
+loadState(): void
+mainControl(): void
+sellInvestment(): void
+setupSumulation(): void
+showAddInvestment(): void
+showBuySellInvestment(): void
+showFinalSummary(): void
+showLoadMenu(): void
+showMainMenu(): void
+showScores(): void
+showSetup(): void
+showSimulationMenu(): void
+showSplashScreen(): void
+showStatus(): void
+<<static>> main(args:String[]): void
<<extends JPanel>>
GuiFinalSummary
-stats: String
-stats2: String
-stats3: String
-stats4: String
-stats5: String
-stats6: String
+init(): JPanel
+getStats(): String
+getStats2(): String
+getStats3(): String
+getStats4(): String
+getStats5(): String
+getStats6(): String
+outcome1(): String
+outcome2(): String
+outcome3(): String
+outcome4(): String
+outcome5(): String
+outcome6(): String
+outcome7(): String
+outcome8(): String
+outcome9(): String
+outcome10(): String
+outcome11(): String
+outcome12(): String
+outcome13(): String
+outcome14(): String
+setStats(newStats:String): void
+setStats2(newStats2:String): void
+setStats3(newStats3:String): void
+setStats4(newStats4:String): void
+setStats5(newStats5:String): void
+setStats6(newStats6:String): void
Reports
-simulation: MarketSim
-comparisonIndex: Stock
-indexSymbol: String
-indexAvailable: boolean
+setComparisonIndex(): void
+getIndexIntervalGain(): double
+getIndexNetGain(): double
+getNetGain(): double
+getNetPercentGain(): double
+investmentReport(): String
+printProgressReport(): String
+printStatus(): String
+finalSummaryStats(): String
+finalSummaryStats2(): String
+finalSummaryStats3(): String
+finalSummaryStats4(): String
+finalSummaryStats5(): String
+finalSummaryStats6(): String


MarkeSsim
+PATH: String = resources/
-BORDER: String = "---"
-EVENT_PROBABILITY: double = 0.05
-GUI_WRAPPER_ENABLE: boolean = true
-player1: Player
-intervalLength: int = 7
-numberOfTurns: int = 52
-turn: int = 0
-startDate: String
-stockIndex: String
-stockNameList: String
-statusRecorder: LogManager
-optionRandomevents: boolean = false
+addStock(): void
+advanceTurn(): String[]
+buyInvestment(): void
+generateRandomEvent(eventRarity:String[]): String[]
+getIntervalLength(): int
+getMetalIndexList(): ArrayList<String>
+getNumberOfTurns(): int
+getPlayer(): Player
+getScores(): ArrayList<String>
+getStartDate(): String
+getStockIndex(): String
+getStockIndexList(): ArrayList<String>
+getStockNameList(): String
+getTurn(): int
+isSimulationComplete(): boolean
+isOptionRandomEvents(): boolean
+loadGame(playerName:String): void
+loadSimulation(): void
+mainMenu(): void
+mainSimulationMenu(numberOfTurns:int): void
+printStatus(): void
+progressReport(): String
+saveGame(): void
+sellInvestment(): void
+setCurrentTurn(turn:int): void
+setDate(date:String): void
+setIndex(stockIndex:String): void
+setIntervalLength(days:int): void
+setStockNameList(name:String): void
+setNumberOfTurns(turns:int): void
+setupNewSimulation(): void
+main(args:String[]): void
+SunVal(weekDay:double,Sun:boolean,isLeap:int,
        NumweekDay:int): void
+GetValidDate(): void
+getValidInput(message:String): String
+getValidInt(message:String,lowerBound:int,
             upperBound:int): int
+INDVal(weekDay:double,sat:boolean,sun:boolean)
+getValidInt(message:String): int
+NYVal(weekDay:double,Sat:boolean,Sun:boolean): void
+ChrisVal(weekDay:double,Sat:boolean,Sun:boolean): void
+SatVal(weekDay:double,Sat:boolean,isLeap:int,
        NumweekDay:int): void
<<Extends IOManager>>
LogManager
-logData: ArrayList<String> = null
+addLine(logLine:String): void
+getLogData(): ArrayList<String>
+load(): void
+save(): void
+setLogData(newData:ArrayList<String>): void
<<Extends GameIOManager>>
ScoreManager
+FILENAME: String = "resources/HighScores.txt"
-scoreData: ArrayList<String> = null
-formatUserScore(): void
+getScoreData(): ArrayList<String>
+load(): void
+save(): void
+setScoreData(someScoreData:ArrayList<String>): void
<<Extends GameIOManager>>
SaveManager
-startDate: String = ""
-numOfTurns: int
-turn: int
-intervalLength: int
-userData: ArrayList<String> = null
-stockIndex: String
-stockNameList: String
-simulation: MarketSim
+loadState(playerName:String,simulation:MarketSim): void
+save(): void
-setInvestments(): void
+setStates(): void
+updateUserData(): void


<<Extends IOManager>>
EventManager
-FILENAME: String = "resources/Events.txt"
-randomGen: Random = null
-events: ArrayList<String>
+load(): void
+randomEvent(): ArrayList<String>
IOManager
-fileName: String = ""
+ensureDirectoryExistance(directoryName:String): void
+extractedInfo(inputInformation:String): String
+getFileName(): String
+load(fileName:String): ArrayList<String>
+save(stringList:ArrayList<String>,fileName:String): void
+setFileName(aFileName:String): void
<<Extends IOManager>>
GameIOManager
#player: Player
+getPlayer(): Player
+load(fileName:String): ArrayList<String>
+save(userData:ArrayList<String>,fileName:String): void
+setPlayer(aPlayer:Player): void


<<extends JPanel>>
GuiSetupMenu
-serialVersionUID: long
-STOCK_INDICES: String[]
-PATH: String
+FILE_ENDING: String
-buttonBegin: JButton
-buttonCancel: JButton
-userName: JTextField
-startingMoney: JTextField
-startDate: JTextField
-intervalLength: JTextField
-numberTurns: JTextField
-stockIndex: JComboBox
-filename: JComboBox
-randomEvents: JCheckBox
+getButton(): JButton[]
+getChosenFilename(): String
+getFilenames(): String[]
+getIntervalLength(): String
+getNumberTurns(): String
+getOptionRandomEvents(): boolean
+getStartDate(): String
+getStartingMoney(): String
+getStockIndex(): String
+getUserName(): String
+init(): JPanel
<<extends JPanel>>
GuiLoadState
-PATH: String = "./saves";
+FILE_ENDING: String = ".txt"
-buttonLoad: JButton
-buttonCancel: JButton
-filename: JComboBox<String>
+getButton(): JButton[]
+getChosenFilename(): String
-getFilenames(): String[]
+init(): JPanel
<<extendJPanel>>
GuiGraph
-GRAPH_STROKE: Stroke
-listsize: int
-index: int
-full: boolean
-currentsize: int
-padding: int
-labelpadding: int
-lineColor: Color
-pointColor: Color
-gridColor: Color
-pointWidth: int
-numberYDivisions: int
-scores: double[]
+format: DecimalFormat
-getMaxScore(): double
-getMinScore(): double
+getNextValue(person:Player): double
+getScores(): double[]
+init(person:Player): void
#paintComponent(g:Graphics): void
+setScores(scores:double[]): void
+updateGraph(peron:Player): void
+<<static>> createAndShowGui(scores:double[]): void
+reset(): void
<<extends Investment>>
Stock
+MAX_INDEX: int = 364
+DATE_FORMAT: String = "yyyy-M-d"
+CLOSING_DATA_COLUMN: int = 4
-stockSymbol: String
+INVESTMENT_TYPE: String = "STOCK"
-stockSymbol: String
+data: Investment
-calculateEndDate(startdate:String,interval:int,
numberOfIntervals:int): String
-dateBetween(startdate:String,enddate:String,
currentdate:String): boolean
-daysBetween(date1:Date,date2:Date): int
-fillData(array:String[]): String[]
+getCopy(original:Stock,startDate:String,
interval:int,numberOfIntervals:int): Investment
+getDateCode(startDate:String,endDate:String): String
+getSymbol(): String
-invertArray(array:String[]): Strgin[]
+isValidDate(startdate:String,interval:int,
numberOfIntervals:int): boolean
+scrapePrice(symbol:String,startdate:String,
interval:int,numberOfIntervals:int): double[]
Investment
-name: String
-quantity: int
-totalPercent: double
-totalChange: double
-investmentType: String
-percent: double
-change: double
-priceHistory: double[]
-currentUnitPrice: double
+addQuantity(quantity:int): void
+calculatePerformante(index:int): void
+getChange(): double
+getCurrentUnitPrice(): double
+getInvestmentType(): String
+getName(): String
+getPercent(): double
+getPrice(): double
+getPriceHistory(): double[]
+getPriceHistoryLength(): int
+getQuantity(): int
+getTotalChange(): double
+getTotalPercent(): double
+getValue(): double
+removeQuantity(quantity:int): void
+setCurrentUnitPrice(price:double): void
+setInvestmentType(): String
+setName(name:String): void
+setPriceHistory(price:double[]): void
+setPriceHistory(index:int,amount:double): void
+setQuantity(quantity:int): void
+updatePrice(index:int): void
<<extends Investment>>
PreciousMetal
+Metal: enum
+FILENAME: String = resources/londonfixes.csv
+DATE_FORMAT: String = "yyyy-M-d"
+INVESTMENT_TYPE: String = "METAL"
-type: Metal
-calculateEndDate(startdate:String,interval:int,
numberOfIntervals:int): String
-dateBetween(startdate:String,enddate:String,
currentdate:String): boolean
-fillData(array:String[]): String[]
+getType(): Metal
+isValidDate(startdate:String,interval:int,
numberOfIntervals:int): boolean
+loadPrice(startdate:String,interval:int,
numberOfIntervals:int): double[]
+setType(type:Metal): void
<<extends Investment>>
BitCoin
+FILENAME: String = resources/bitcoin_prices.csv
+DATE_FORMAT: String = "yyyy-M-d"
+INVESTMENT_TYPE: String = "BITCOIN"
-calculateEndDate(startdate:String,interval:int,
numberOfIntervals:int): String
-dateBetween(startdate:String,enddate:String,
currentdate:String): boolean
-fillData(array:String[]): String[]
+isValidDate(startdate:String,interval:int,
numberOfIntervals:int): boolean
+loadPrice(startdate:String,interval:int,
numberOfIntervals:int): double[]



```
Player
-userName: String
-wittyPhrase: String
-initialMoney: double = 0.0
-currentMoney: double = 0.0
-investmentList: ArrayList<Investment>
-wealthHistory: ArrayList<Double>
+Player(name:String,initialShekels:double,
        currentShekels:double,listStocks:Arraylist<Stock>): void
+Player()
+addBitCoin(startDate:String,intervalLength:int,
            numberOfIntervals:int,index:int): void
+addMetal(type:PreciousMetal.Metal,startDate:String,
          intervalLength:int,numberOfIntervals:int,
          index:int): void
+addOwnedInvestment(ownedInvestment:Investment): void
+addStock(stockSymbol:String,startDate:String,
          intervalLength:int,numberOfIntervals:int): void
+buyInvestment(index:int,quantity:int): int
+getCurrentMoney(): double
+getInitialMoney(): double
+getInvestmentList(): String[]
+getInvestmentListArray(): ArrayList<Investment>
+getInvestmentListReport(): String
+getNetWorth(): double
+getUserName(): string
+getWittyPhrase(): String
+getWealthHistory(index:int): double
+isInvestmentOnList(name:String): boolean
+sellInvestment(index:int,quantity:int): int
+setCurrentMoney(moneyValue:double): void
+setInitialMoney(moneyValue:double): void
+setInvestmentList(listInvestments:ArrayList<Investment>): void
+setUserName(name:String): void
+setWittyPhrase(phrase:String): void
+updatePortfolio(index:int): int
+percentMoneyChange(amount:double): void
+percentStockChange(percentStockChange:doubler): void
+absoluteMoneyChange(amount:double): void
+absoluteStockChange(amount:double): void
```

```
MarketSimUtils
-startDate: String
+<<static>> getValidInput(message:String,
                          validRegex:String): String
+<<Static>> getValidInt(message:String): int
+<<static>> getValidInt(message:String,lowerbound:int,
                        upperbound:int): int
+<<static>> wrapString(stringIn:String,lengthToWrap:int): String
```

```
<<implements Runnable>>
SoundLayer
-audioFileName: String
-repeat: boolean = false
-soundtrack: AdvancedPlayer
-exceptions: ArrayList<String>
-soundThread: Thread
-setSountTrack(audioFile:String): void
+setAudioFileName(setAudioFileName:audioFile): void
-playSound(): void
+run(): void
+playLoop(): void
+playOnce(): void
```

```
<<extends Exception>>
SoundException
-exception: Throwable
+getException(): Throwable
+printStackTrace(): void
+printStackTrace(ps:PrintStream): void
```