**CS308 Folio-Tracker**

**Chloe Forsyth xeb12185@uni.strath.ac.uk**

**Tom Maxwell gvb12182@uni.strath.ac.uk**

**Ryan Gibson yfb13226@uni.strath.ac.uk**

**Amanda Renny gpb12164@uni.strath.ac.uk**

**Design rational:**

We decided that we would use a development method known as the Rational Unified process as this centers largely around iterative programming and revolves around four key stages; inception, elaboration, construction and transition. The inception stage does not really apply here as we were given a project brief as the task and as the program will not be ran live by anyone; nor does the transition phase apply as the program will not be ran live and there is no “real” customer. However the methodology as a whole worked well with our own style. The elaboration phase was useful as it allowed us to break the program down into constituent parts and work out what we needed to do for each of them. The construction phase was again key to us as this is where the actual programming took place. After something had been programmed we could go back and see what needed changed and how it fitted in with the rest of the program as a whole.

**API Design inc specs:**

Tracker.java: Handles the creation, deletion and viewing of portfolios. Manages the refresh rate of updates to the model.

Contained methods:

Public Tracker()

Effects:

public void addPortfolio()

Modifies:

public void addStock(Portfolio p)

Modifies:

public void addShares(Stock s)

Modifies:

Portfolio.java: Interface of a Portfolio object. Handles all Portfolio-specific activities and data, including the addition and deletion of stocks and getting stock related details.

Contained methods:

public synchronized Stock newStock(String ticker);

Modifies: this

Effects: Creates a new Stock object

public synchronized boolean deleteStock(Object o);

Modifies: this

Effects: Deletes given Stock from the Portfolio. Returns True if successful, otherwise returns False.

public synchronized List<Stock> getStocks();

Effects: Returns this.stocks

public synchronized double getTotalValue();

Effects: Returns this.totalValue

public synchronized double getNetGain();

Effects: Returns this.netGain

public String getName();

Effects: returns this.name

Public void update(observable o, object arg);

Effects: updates this

Modifies: this

int noStocks()

Effects: returns the number of stocks within this

Synchronized void update()

private void calculateNetGain()

Effects: this.netGain = netGain of all stocks

Modifies: this

private void calculateTotalValue()

Effects: this.totalValue = totalValue of all stocks

Modifies: this

Stock.java: Interface of a Stock object. Handles all information required of a publically traded stock. Can add and remove shares, and get stock-specific details.

Contained methods:

public boolean addShares(int noShares, double initialValue);

Modifies: this

Effects: Adds shares to this.shares, Returns True if successful, otherwise returns False.

public boolean removeShares(int noShares);

Modifies: this

Effects: Removes shares from this.shares, Returns True if successful, otherwise returns False.

public String getTicker();

Effects: Returns this.ticker

public String getName();

Effects: Returns this.name

public double getTotalSpent();

Effects: Returns this.totalSpent

public String getExchange();

Effects: Returns this.exchange

public double getCurrentValue();

Effects: Returns this.currentValue

public double getHoldingValue();

Effects: Returns this.holdingValue

public int getNumberOfShares();

Effects: Returns this.noShares

public double getNetGain();

Effects: Returns this.netGain

public double getClosingPrice();

Effects: Returns this.closingPrice

public double getOpeningPrice();

Effects: Returns this.openingPrice

public double getDailyChange();

Effects: Returns this.dailyChange

public double getDailyMax();

Effects: Returns this.dailyMax

public double getDailyMin();

Effects: Returns this.dailyMin

public double getVolume();

Effects: Returns this.volume

Model.java: Interface of a model object.

public Portfolio createPortfolio(String name);

Effects: Creates a new portfolio in the model and returns it

Modifies: this

public boolean deletePortfolio(Object o);

Effects: Deletes the given portfolio

Modifies: this

public List<Portfolio> getPortfolios();

Effects: returns List<Portfolio> portfolios

**Class Diagram:**

**Assertions:**

None, because Ricky left us in the lurch with 36 hours to go.

**JUnit Tests:**

None, because Ricky left us in the lurch with 36 hours to go.