Strategy.jl

- struct Portfolio
- rules
  - (i.e. if stock A < stock B for 8 time units: buy A)
- AnonymousFunction:: Process_info (Portfolio, data_point)

hector the investor

Portfolio.jl

- Vars
  - Struct::Portfolio
    - liquid capital
      - float
    - holdings
      - ticker:share
  - Struct:: PortfolioState
    - currentPortfolio
    - startDate
    - currentDate
- NOTE:
  - Figure out how to handle dividends
    - auto-reinvest
    - turn to capital

**buy(date, stock, amount_dollar)**

- trade some amount of liquid capital for the corresponding amount of stock

**sell(date, stock, amount_shares)**

- trade some amount of stock for the corresponding amount of liquid capital

**riskReward()**

- sharpe ratio?
  - the rate of return above the "risk-free rate"/volatility
  - (r_p - r_f)/o_p
    - r_p = expected portfolio return
    - r_f = risk free rate
    - o_p = portfolio std dev
  -

**returnVolatility(timeframe, PortfolioState)**

- "Volatility"?
  - the standard deviation of a stock's past returns

**EvaluateValue()**

- return overall value of portfolio (i.e. ticker*share + liquid capital)

StockTicker.jl

- Holds a struct with exchange and symbol
- on initialization, checks that the ticker exists in the database at the given time
- start date is when teh company became public, end_date (when applicable) is when the company went off the market

**stockTicker(exchange, symbol, start_date, end_date)**

- returns the data associated with the ticker (denoted by an exchange:stock pair) at the given time unit
- start

Simulator.jl

- Historical_data
- Array::Strategies[]
  - e.g. User generated, Baseline_strategy (S&P), etc.
- Timeframe

**Plot(Strategies[], start_time, end time)**

- Draws

MarketDB.jl

- Holds historical data in a time series
- Use JuliaDB if 1.0 or convert to 0.6??
- TimeSeries otherwise

**query(date/time, exchange, stock)**

- returns the data associated with the ticker (denoted by an exchange:stock pair) at the given time unit

**validTicker(exchange, stock)**

- returns true if information exists in the database for the given exchange:stock pair

Price at time

PortfolioDB.jl

- Holds the states of the portfolio over the run of the simulation

**query(date/time)**

- returns the Portfolio information associated with the given date/time

**write(date/time, PortfolioState, riskReward, Volatility, Value)**

- writes an immutable data point to the database for use with visualization/data analysis