Security.h----------

```
#ifndef SECURITY_H
#define SECURITY_H

#include <string>

class Security
{
public:
  // create a security with the given symbol
  Security(std::string the_symbol);

  // company symbol
  std::string get_symbol();

  // update the current share value
  void set_share_value(double current_share_value);

  // current share value
  double get_share_value() const;

  // update the number of holdings
  void set_holdings(int number_of_holdings);

  // current number of holdings
  int get_holdings() const;

  // total current value of all holdings
  double market_worth() const;

private:
  std::string symbol;       // security identifier
  double share_value = 0;   // each share's current market value
  int holdings = 0;         // number of total shares of security held
};

#endif
```

Security.cpp----------

```
//File security.cpp
//Security class implementation
//Carter Mooring
//CPSC 223

#include "security.h"
#include <iostream>
using namespace std;

//update the symbol
Security::Security(std::string the_symbol)
{
  symbol = the_symbol;
}

//get the symbol
std::string Security::get_symbol()
{
  return symbol;
}

//update the share value
void Security::set_share_value(double current_share_value){
  share_value = current_share_value;
}

//get the share value
void Security::get_share_value() const
{
  return share_value;
}

//update the number of holdings
void Security::set_holdings(int number_of_holdings){
  holdings = number_of_holdings;
}

//get the number of holdings
int Security::get_holdings() const
{
  return holdings;
}
```

```
//return the market worth
double Security::market_worth() const
{
  return return holdings * share_value;
}
```

Stocks.h----------

```
#ifndef STOCK_SHARE_H
#define STOCK_SHARE_H

#include <string >
#include "security.h"

class Stock : public Security {
public:
  // create a stock with the given company symbol
  Stock(std::string the_symbol);

  // set the purchase price of the holdings
  void set_purchase_price(double the_purchase_price);

  // purchase price
  double get_purchase_price() const;

  // compute the net worth of the stock holdings
  double sell_value() const;

private:
  double purchase_price = 0;    // price per holding
};

#endif
```

Stocks.cpp----------

```
//File stock.cpp
//Stock class implementation
//Carter Mooring
//CPSC 223

#include "stock.h"
#include <iostream>
using namespace std;

//the symbol for the stock
Stock::Stock(std::string the_symbol)
{
}

//update the purchase price
void Stock::set_purchase_price(double the_purchase_price)
{
  purchase_price = the_purchase_price;
}

//get the purchase price
double Stock::get_purchase_price() const
{
  return purchase_price;
}

//compute the sell value
double Stock::sell_value() const
{
}
```

Stock.option.h----------

```
#ifndef STOCK_OPTION_H
#define STOCK_OPTION_H

#include <string >
#include "stock.h"

class StockOption : public Stock
{
public:
  // crate a stock option with the given symbol
  StockOption(std::string the_symbol);

  // set the strike price per share
  void set_strike_price(double the_strike_price);

  // strike price per share
  double get_strike_price() const;

  // the net worth of the option
  double sell_value() const;

private:
  double strike_price = 0;    // price to sell per holding
};
```

Stock.option.cpp----------

```
//File stock.option.cpp
//Stock.option class implementation
//Carter Mooring
//CPSC 223

#include "stock.option.h"
#include <iostream>
using namespace std;

//the stock option symbol
StockOption::StockOption(std::string the_symbol)
{
}
```

```
//update the strike price value
void StockOption::set_strike_price(double the_strike_price)
{
  strike_price = the_strike_price;
}

//get the strike price value
double StockOption::get_strike_price() const
{
  return strike_price;
}

//the networth of the sell value
double StockOption::sell_value() const
{
}
```

Hw1.cpp----------

```
#include <iostream>
#include <assert.h>
#include "security.h"
#include "stock.h"
#include "stock_option.h"

using namespace std;

// returns true if stock value is positive
bool should_sell(Stock& the_stock);


int main(int argc, char** argv)
{
Security s1("GOOG");
s1.set_share_value(1245);
s1.set_holdings(120);
assert(s1.get_share_value() == 1245);
assert(s1.get_holdings() == 120);
assert(s1.market_worth() == 1245*120);
```

```
Stock s2("APPL");
s2.set_share_value(204);
s2.set_holdings(76);
s2.set_purchase_price(175);
assert(s2.get_share_value()
assert(s2.get_holdings() ==
assert(s2.get_purchase_price() == 175);
assert(s2.market_worth() == 204*76);
assert(s2.sell_value() == (204-175)*76);

StockOption s3("AMZN");
s3.set_share_value(1823);
s3.set_holdings(500);
s3.set_purchase_price(5.25);
s3.set_strike_price(1828);
assert(s3.get_share_value() == 1823);
assert(s3.get_holdings() == 500);
assert(s3.get_purchase_price() == 5.25);
assert(s3.get_strike_price() == 1828);
assert(s3.market_worth() == 1823*500);
assert(s3.sell_value() == (1828-5.25)*500 - 1823*500);

assert(should_sell(s2) == true);
assert(should_sell(s3) == false);
}

bool should_sell(Stock& the_stock)
{
  return the_stock.sell_value() > 0;
}
```

One brief paragraph describing your testing strategy and possible improvements:

Unfortunately I was unable to get my code to run from the command line on my machine so I am unsure if it works but I hope it does! Usually though, I try to test my code every couple lines so that I don't end up with an untraceable error after a lot of effort and change to the code. I suppose this method could be improved if I test after I finish a certain task rather than a couple lines of code so I don't spend as much time compiling.

One brief paragraph on implementation issues/challenges and how you addressed them:

I guess this brings me back to my first paragraph and the main issue I had was not being able to compile and run code from my command line. I tried various methods online and also used the line given in the homework to try and compile but it wasn’t working. What I tried figuring out was how to navigate to my files from the command line but I think they may be private so I was unable to find the right path to them. I will also address this by asking you for help after class.