Java

Listing 2.

/WEB-INF/src/com/devx/tradingapp/web/PortfolioBeanItem.java
/WEB-INF/src/com/devx/tradingapp/business/Portfolio.java

```
/WEB-INF/src/com/devx/tradingapp/web/PortfolioBeanItem.java
package com.devx.tradingapp.web;

import net.neurotech.quotes.Quote;

public class PortfolioItemBean {
    private String symbol;

    private int shares;

    private Quote quote;

    private double currentValue;

    private double gainLoss;

    public int getShares() {
        return shares;
    }

    public void setShares(int quantity) {
        this.shares = quantity;
    }

    public String getSymbol() {
        return symbol;
    }

    public void setSymbol(String symbol) {
        this.symbol = symbol;
    }

    public double getCurrentValue() {
        return currentValue;
    }

    public void setCurrentValue(double currentValue) {
        this.currentValue = currentValue;
    }

    public Quote getQuote() {
        return quote;
    }

    public void setQuote(Quote quote) {
        this.quote = quote;
    }

    public double getGainLoss() {
        return gainLoss;
    }

    public void setGainLoss(double valueChange) {
        this.gainLoss = valueChange;
    }
}



/WEB-INF/src/com/devx/tradingapp/business/Portfolio.java

package com.devx.tradingapp.business;

import java.util.Iterator;
import java.util.Map;
```

```
public class Portfolio {

    private float cash;

    //maps symbol string to shares
    private Map sharesPerSymbol;

    public Portfolio(float cash, Map sharesPerSymbol) {
        this.cash = cash;
        this.sharesPerSymbol = sharesPerSymbol;
    }

    public float getCash() {
        return cash;
    }

    public boolean contains(String symbol) {
        return sharesPerSymbol.containsKey(symbol);
    }

    public int getNumberOfShares(String symbol) {
        Object shares = sharesPerSymbol.get(symbol);

        if (shares instanceof String) {
            return Integer.parseInt((String) shares);
        } else if (shares instanceof Integer) {

            return ((Integer) shares).intValue();
        } else {
            throw new RuntimeException("Application error");
        }
    }

    public Iterator getSymbolIterator() {
        return sharesPerSymbol.keySet().iterator();
    }

    public void buyStock(String symbol, int sharesBought, float purchasePrice) {
        cash -= sharesBought * purchasePrice;
        if (sharesPerSymbol.containsKey(symbol)) {
            int currentShares = getNumberOfShares(symbol);
            sharesPerSymbol.put(symbol, new Integer(currentShares
                    + sharesBought));
        } else {
            sharesPerSymbol.put(symbol, new Integer(sharesBought));
        }
    }

    public void sellStock(String symbol, int sharesSold, float sellPrice) {
        cash += sharesSold * sellPrice;
        int currentShares = getNumberOfShares(symbol);
        int sharesLeft = currentShares - sharesSold;
        if (sharesLeft == 0) {
            sharesPerSymbol.remove(symbol);
        } else {
            sharesPerSymbol.put(symbol, new Integer(sharesLeft));
        }

    }

    public boolean canBuy(int shares, float purchasePrice) {
        if ((shares * purchasePrice) <= cash) {
            return true;
        } else {
            return false;
        }
    }
}
```