REPORT :

Source code available at: https://github.com/ernestjumbe/ministockmarket

The project code consists of three classes; the Good class, the Game class, and the game test class. Each class resides in it's own file in the same directory; the ministockmarket directory. Below are description of each class and the methods within each class:

The Good class

The purpose of this class is to construct a Good object. This represents a type of good to be traded in the mini stock market. The class allows its construction to take the name of the good the minimum and maximum trading prices. The class also holds the state of amount of goods at a given time and the price in a given state of the instance of the class.
Creating a Good object is done as follows `new Good("Sugar", 13.00, 20.00)` where the first argument is the String name of the good to be traded and the second and third arguments are the double minimum and double maximum prices respectively. The decision to initialize the class with this information is that this information is constant from the creation to the destruction of the object.

The class contains a single method `roundPrice();` this method takes no arguments and returns a double `currentprice;` which is assigned as the current price of the Good object created in a given state. Calling this method on an object allows its current price to be randomly generated and set. The random current price is generated using the `Math.random()`

The Game class

This class handles the logic of the game. To construct an instance of the Game class the initial amount of money a player must be provided as follows: new Game(10000.00); The game object also holds variables the provide information of the game in various states at different stages of the game. The include the current round of the game in int currentround; , the totalstock in goods the game game object is holding at a given time the amount is calculated in double totalstock; the cost of storage of the goods in double storageprice; . on initialization the class creates an array to hold a number of Good objects. This allows the creation of the first two Good objects in the game as per requirements and addition of more goods as the game progresses and the player “unlocks” new levels allowing the trading of more goods.

The core logic for game play is contained in the run(); method that does not return anything. This is the method called to run the game. On entering this method the goods in the requirements of the game are created. The variables are declared. A String s; which is used for string input by the played and int amount; used for integer inputs by the player these variables are used in cases the user is prompted to enter a string or an integer and used by the StdIn class. A variable boolean canplay; with the value of true is also created. This variable is the test that determines whether the main game loop can continue to run.

Within main game loop a test is made to check weather new levels can be opened. And information that allows the player to make decisions on whether to buy or sell certain goods is provided. Information on the current round of play is also provided to allow the player to keep track of how many rounds of the game they have played. Before a player may buy or sell goods a conditional statement checks if the player as enough funds to continue. If the condition is not met the player is provided with information that the game has ended and the canplay variable is set to false thus exiting the main game loop and ending the game. if the condition is met the player is prompted to buy or sell goods in the goodArray which are not null.

The checkStock(); method takes two arguments the amount to sell and the amount of stock in a Good object. If the amount the user wishes to sell exceeds the stock the user has available in the object the user cannot sell the amount and it prompted to try again.

If a player chooses to buy a good, they are prompted to enter the amount of the good they wish to buy. The current price of the good is accessed and the amount they wish tto buy and the current price of the good are passed to the buyStock(); method. this method takes two arguments and return nothing. Within this method the current price of the good and the amount of the good to be bought are multiplied. The resulting multiplicand is checked by the checkFunds(); method which takes two arguments; the resulting multicand and the and the funds the player has available. If the the amount the player wishes exceeds the amount the player has available the player is not permitted to purchase the goods. If the player has sufficient funds the cost of the goods is subtracted from the players available funds and the the goods bought are added to the instance of the good class the game is handling at the time.

If the player wishes to sell a good the current price of the good and the amount of the good is passed to the sellStock(); method. This method multiplies the amount to be sold and the current price of the good. The multiplicand is added to the players total funds and the amount of goods sold is subtracted from the stock value of the Good object instance.

when the options to buy and sell goods are complete the round count is increased by 1 and the cost of storing the total amount of goods in the stock value of each instance of the Good class passed to the chargeStock(); method. This method multiplies the the total in cost with the value ot the storage price variable. The result of this calculation is subtracted from the funds available to the plater and a new round beings.

The GameTest class

This class contains the main method to run the game. Within the class a game object is instantiated with the initial amount of funds available to the player. The run method of this instance of the Game class is called to begin a game.

Bugs

If a player is unable to buy stock the program continues and does not offer the player the opportunity to attempt to buy a different amount of stock of a certain good.

There is no means to terminate a game expect through terminating the program using for example a keyboard interrupt Ctrl + c

```
1   public class GameTest{
2     public static void main(String[] args){
3       Game mygame = new Game(10000.00);
4       mygame.run();
5     }
6   }
```

```
1   public class Good {
2     String name;
3     double currentprice;
4     double minprice;
5     double maxprice;
6     double stock = 0;
7
8     Good(String n, double min, double max){
9       name = n;
10      minprice = min;
11      maxprice = max;
12    }
13
14    // Set the price for a specific round
15    public double roundPrice(){
16      double range = Math.abs(maxprice - minprice);
17      currentprice = (Math.random() * range) + (minprice <= maxprice ? minprice :
    maxprice);
18      return currentprice;
19    }
20
21
22  }
```

```
public class Game {

  double currentfunds;

  Game(double funds){
    currentfunds = funds;
  }
  int currentround = 1;
  double totalstock;
  double storageprice = 3.00;

  Good[] goodArray = new Good[3];

  public void run(){

    goodArray[0] = new Good("Sugar", 13.00, 20.00);
    goodArray[1] = new Good("Steel", 21.00, 54.00);

    String s;
    int amount;
    boolean canplay = true;
    boolean cansell = false;
    boolean testsell = true;

    while(canplay){
      double roundendstock = 0;
      addGood(currentfunds);
      System.out.println("Round number: " + currentround);
      for(int i = 0; i < goodArray.length; i++) {
        if (goodArray[i] != null) {
          goodArray[i].roundPrice();
        }
      }
      System.out.println("You have $ " + currentfunds + " available.");
      System.out.println("Goods in stock:");

      for(int i = 0; i < goodArray.length; i++) {
        if (goodArray[i] != null) {
          System.out.println("You have: " + goodArray[i].stock + " of " + goodArray
[i].name);
        }
      }
      for(int i = 0; i < goodArray.length; i++) {
        if (goodArray[i] != null) {
          System.out.println(goodArray[i].name + " price in this round: " +
goodArray[i].roundPrice());
        }
      }


      if (currentfunds > 0 ){
        for(int i = 0; i < goodArray.length; i++){
          if (goodArray[i] != null) {
            System.out.println("Buy " + goodArray[i].name + " ? (Enter Y or n):");
            s = StdIn.readString();
            if (s.equals("Y")){
              System.out.print("Enter amount to buy in tons. (example 100): ");
              amount = StdIn.readInt();
              buyStock(amount, goodArray[i].currentprice);
              goodArray[i].stock += amount;
              System.out.println("Funds: $ " + currentfunds);
            }
          }
        }
        for(int i = 0; i < goodArray.length; i++){
        if (goodArray[i] != null){
          System.out.println("Sell " + goodArray[i].name + " ? (Enter Y or n):");
          s = StdIn.readString();
          if (s.equals("Y")){
            System.out.print("Enter amount to sell in tons. (example 100): ");
            //amount = StdIn.readInt();
            //cansell = checkStock(amount, goodArray[i].stock);
            while (testsell){
              amount = StdIn.readInt();
              cansell = checkStock(amount, goodArray[i].stock);
              if (cansell) {
                sellStock(amount, goodArray[i].stock);
                goodArray[i].stock -= amount;
                testsell = false;
              } else {
                System.out.println("You do not have " + amount + " tons of " +
goodArray[i].name + " to sell");
                System.out.print("Enter amount to sell in tons. (example 100): ");
              }
            }
            System.out.println("Funds: $ " + currentfunds);
          }
        }
      }
        for(int i = 0; i < goodArray.length; i++){
        if (goodArray[i] != null) {
          roundendstock += goodArray[i].stock;
        }
      }
```

```
92
93        System.out.println("Round end stock :" + roundendstock);
94        totalstock = roundendstock;
95        chargeStorage(totalstock);
96        currentround++;
97        System.out.println("Total stock :" + totalstock);
98        System.out.println();
99        } else {
100         System.out.println("You balance is " + currentfunds + " you do not have
    enough money to continue playing.");
101         canplay = false;
102       }
103     }
104   }
105
106   // Buy stock
107   public void buyStock(double units, double priceperunit){
108     // Calculate the total price for the of stock to be purchased.
109     double totalprice = units * priceperunit;
110     //check if the funds are avaiable to purchase this amount of goods of a certain
    type.
111     if (checkFunds(totalprice, currentfunds)) {
112       // Subract the total stock purchased from the funds available.
113       currentfunds -= totalprice;
114     } else {
115       System.out.println("You do not have enough money to but this amount of
    goods.");
116     }
117   }
118
119   public void sellStock(double units, double priceperunit){
120     // Calculate the total price for the of stock sold.
121     double totalprice = units * priceperunit;
122     // Add the total of the stock sold to the funds available.
123     currentfunds += totalprice;
124   }
125
126   // Charge storage
127   private void chargeStorage(double stock){
128     // Calculate the cost of storage for current round.
129     double storagecost = stock * storageprice;
130     // Subtract the storage cost from the funs available.
131     currentfunds -= storagecost;
132   }
133
134   private boolean checkStock(double tosell, double available) {
135     if (tosell <= available) {
136       return true;
137     } else {
138       return false;
139     }
140   }
141
142   // Check funds
143   private boolean checkFunds(double tospend, double available){
144     if (tospend < available){
145       return true;
146     } else {
147       return false;
148     }
149   }
150
151
152
153   private void printFunds(){
154     System.out.println("You have $" + currentfunds + " available.");
155   }
156
157   private void addGood(double funds){
158     if (funds >= 15000 && goodArray[2] == null){
159       goodArray[2] = new Good("Gold", 50.00, 120.00);
160       System.out.println("******** Congratulations you can now trade Gold
    ********");
161     } else {
162       System.out.println("Nothing to add!");
163     }
164   }
165 }
```