

Lab 7 – Introduction to Objects

Aim

This week is focused on objects and being comfortable with using them. This will require you to be comfortable with using methods, and with using arrays.

Tips:

Objects are a very important Java concept, so it is important you understand what you are doing. If unsure, ask.

Stock Objects

Lecture 6 introduced the concept of using objects. Rather than have static classes and multiple variables and arrays, we can create objects to group data together.

- Create an Empty NetBeans project called Lab7, with a main class LabStock. Do not make any changes to this main class yet.
- Create a new Java class. This is going to be our data class, so call it “StockPrice”. You should now have 2 empty Java classes: your main class, and a new StockPrice class.
- Note that StockPrice should have no Main class, why not?
- The Lecture notes discuss creating a person object, and this week, you should create a StockPrice object. Do not give them initial values, just **declare** their name and type. E.g.

<u>Name</u>	<u>Example Data</u>	<u>Description</u>
• Ticker	BABA	The NYSE stock code
• Date	2019-8-21	The date for the quote
• Open	92.11	The opening price at start of day
• High	93.5	The highest price during the day
• Low	91.45	The lowest price during the day
• Close	92.01	The closing price at end of day
• Volume	35071161	The total number of stocks traded during the day

- What data type will each instance variable be? (TIP: we can use a String for the date for now; we will change it to something more useful later).
- You will also need to create a constructor. Again, consult your lecture notes for guidance. A constructor requires parameters, which are used to initialise your variables.
- Add a System.out.println in the constructor, so you will know when a new StockPrice has been created.

Encapsulation

As discussed in your lecture, variables and methods should be private unless they need to be accessed by other classes or objects.

- Your instance variables (the variables in the StockPrice class) should be set to private. However, doing this means that we will not be able to access our data! How can we solve this?
- We can add “getters” for each instance variable
- Create getter methods like this for each instance variable. This will let you access the variables:

```
public String getTicker(){  
    return ticker;  
}
```

Creating a StockPrice Object

You now have a StockPrice class, which means that you are ready to use it to create tickers!

- Open your main (controller) class. You can now create an object, and instantiate it. Consult your lecture notes and ask a TA if you are not sure how to do this.
- You need to create a variable of **type** StockPrice, and you need to create a **new** StockPrice, call the constructor in StockPrice, and **pass in** suitable arguments.

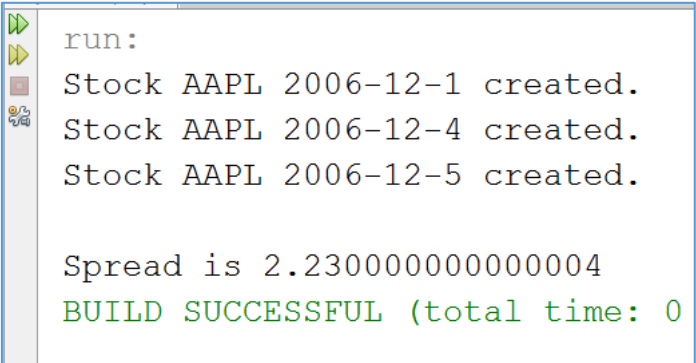
```
StockPrice sp1 = new StockPrice("AAPL", "1999-9-9", 100, 90, 110, 95, 765432100);
```

- This is how we create an **object**.
- Try to create 3 different stock prices with their own unique attributes, call them *sp1*, *sp2*, *sp3* (some examples have been provided at the end of this sheet)

Using your Methods

You should have created a getTicker() method in your StockPrice class, use it to return the ticker name of sp1 and display it in your main method. Here, we do not change the getter. We will call the system.out in the main method:

- Use your getter methods to display information about each attribute of your stock price
- Create a new method that will **return** the spread. The spread is the difference between the highest price and the lowest price, i.e. high – low. This method should be created in your StockPrice class
- Call this method and display the results of sp1

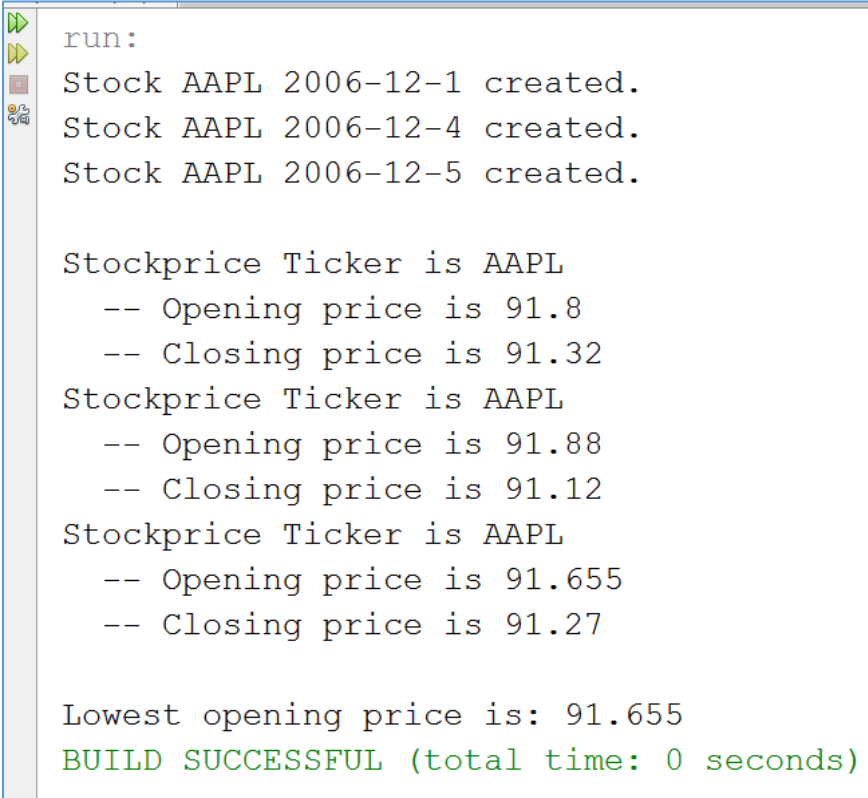


```
run:  
Stock AAPL 2006-12-1 created.  
Stock AAPL 2006-12-4 created.  
Stock AAPL 2006-12-5 created.  
  
Spread is 2.2300000000000004  
BUILD SUCCESSFUL (total time: 0
```

StockPrice Arrays

You worked with arrays last week, but an array can be of any type. For example you can create an array of StockPrice objects!

- Create an array of 3 StockPrice objects, and then use a For loop to view the ticker, the opening price, and the closing price of each StockPrice. Ask if you don't understand!
- In week 6, you learned how to search arrays for useful information. We want to find the ticker with the lowest opening price. Create a method that will receive your StockPrice array, loop through it, and display the lowest opening price.



```
run:
Stock AAPL 2006-12-1 created.
Stock AAPL 2006-12-4 created.
Stock AAPL 2006-12-5 created.

Stockprice Ticker is AAPL
-- Opening price is 91.8
-- Closing price is 91.32
Stockprice Ticker is AAPL
-- Opening price is 91.88
-- Closing price is 91.12
Stockprice Ticker is AAPL
-- Opening price is 91.655
-- Closing price is 91.27

Lowest opening price is: 91.655
BUILD SUCCESSFUL (total time: 0 seconds)
```

StockPrice toString() Method

Currently, you have used the getters to output individual information about a stock price, but you may want to output all the information at once

- Create a toString() method in your StockPrice class
- Modify your previous array to use your toString() method and display all information

Example StockPrice Data

Ticker	Date	Open	High	Low	Close	Volume
AAPL	2006-12-1	91.8	92.33	90.1	91.32	28395700
AAPL	2006-12-4	91.88	92.05	90.5	91.12	25340600
AAPL	2006-12-5	91.655	92.33	90.87	91.27	23672800

Lab Exercise : Student Class

Create a new Java class “Student”. Note that there is no Main method in the “Student” class.

- **Define** private members and **declare** their name and type as follows:

<u>Name</u>	<u>Type</u>	<u>Example Data</u>	<u>Description</u>
name	String	Teng	Student name
id	String	1101	Student ID number
chinese	double	92	student's Chinese score
math	double	93	student's Math score
english	double	91	student's English score
studentAmount	static int	0	total number of students

- Create a constructor to initialize all the private member variables. **Please be noted that to record how many Student objects have been created, each time when the constructor is invoked, we should add 1 to the value of the static variable studentAmount.**
- **Note that the studentAmount variable refers to the total number of students created, and should be static, because if we create 10 students objects, the value of studentAmount in EACH OBJECT should be 10. If you are unsure, please watch the lecture videos again, which provide examples.**
- Create a method “getTotalScore” to return the total score of a student.
- Create a method “getAverageScore” to return the average of a student.
- Override the “toString()” method and print student' name, id, average score and total score.

StudentTester Class

There may be cases where two identical students have been registered for the university, due to an administration error. Using your lecture notes, create a method in Student that will compare it (this Student) with a student that is passed in (that Student).

- Two students are the same if their name, id, and test scores are identical, and the method should return a boolean value
- You should test it by calling it in your main method:

```
Boolean same = student1.sameStudent(student2);
```

StudentTester Class

- Create a method “Student[] createStuArray()” to initialize and return an array of Student objects. Ask the user to input the number of students. Use a for loop to create and store Student objects into an array.
- Create a method “Student getBestStudent(Student [] stuArray)” to return a Student object which has the highest total score within the student array.

- Create a main method to invoke these two methods above. Print the number of students and the student information which has the highest total score.

Hints

- The main structure of StudentTester may look like:

```
14 public class StudentTester {
15     public static Student[] createStuArray() {...22 lines }
37     public static Student getBestStudent(Student[] stuArray)
38     {...10 lines }
48     public static void main(String[] args){
49         Student[] stuArray;
50         stuArray=createStuArray();
51         Student s=getBestStudent(stuArray);
52         System.out.println("There are "+ Student.studentAmount+" students in the array!");
53         System.out.println("The best student is: ");
54         System.out.println(s);
55     }
56 }
```

- The output window may look like:

```
run:
Input the number of students: 2
Input student name: Teng
Input student id: 1234
Input student Chinese score: 90
Input student Math score: 95
Input student English score: 91
Input student name: Erick
Input student id: 92
Input student Chinese score: 95
Input student Math score: 97
Input student English score: 100
There are 2 students in the array!
The best student is:
Name: Erick id: 92 average score: 97.33333333333333 total score: 292.0
BUILD SUCCESSFUL (total time: 28 seconds)
```