ALGONQUIN COLLEGE

OBJECT ORIENTED PROGRAMMING (JAVA) (CST8284)

Course Overview & Introduction

LAB 2

2-D Arrays

&

Use of Debuggers

LAB 2: Preparations

The essence of this lab, is to learn to work with 2-dimensional arrays, and the use of a debugger.

Preparations:

- Review the information on **Debugging** provided to you in Weeks #1 and #2 **Hybrid** tasks on Brightspace course page.

- Review the concept of **two dimensional arrays** using the link provided to you and/or other resources such as:

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

PART 1

REVIEW AND UPDATE YOUR JAVA CODE

ALGONQUIN COLLEGE

The Outcomes to Demonstrate

- Show your professor your **updated version** of the code
- Run and show the **Javadoc** for the comments in this code
- Demonstrate to your professor (**using** the code you **updated**) why and how to correctly use the Eclipse debugger focusing on:
  - Breakpoints
  - Single stepping (*step into* and *step over*)
  - Inspecting variables

Part 1: Description

- In this part, you will work with a 2D array. You will **update** the Java code provided to you so that it performs additional tasks. **The code shows:**

- The **number** of **traders** who conduct business on importation of food spices:
  - Across **9 cities** in Ontario
  - Over a period of one year - **12 months**.

- The **total** number of the traders for each month of the year for each city.

- Format your **output** using `printf`

Just a moment…

- Before you continue, have you reviewed the hybrid materials on how to use a **debugger**?
- Have you reviewed the hybrid material on **2D-arrays**?
- If not, **take a break now** and review the hybrid materials **before** you **continue**!
- You may **not** be able to successfully complete this lab without these reviews.

Part 1 - You are Required to:

- **Create** a new project in eclipse – **Lab 2**
- **Load** and review the code file into your Eclipse editor (**Spice.java**).
- DO **NOT** load the sample output file on eclipse **SpiceOutput**. It's just an example output.
- **Open** and **review** the **sample output** file named SpiceOutput on notepad to understand the required pattern of the **output**.

Part 1 - You are Required to: (2)

- **Update** the portions in the code file **Spice.java** as **specified.** Use a **nested** **for loop** to compute and print the sum of the traders in the array.

  - **Compute** the **sum** of the food spice traders for each **month** specified.

  - **Use** `printf` to format and **print** the each column **sum**.

  - Insert Javadoc style comments into your code

  The program sample output is shown in **SpiceOutput** file for your guidance.

PART 2

DEMONSTRATING THE USE OF A DEBUGGER

ALGONQUIN COLLEGE

You are Required to:

- Demonstrate executing your code in different **perspectives** (e.g. debug and Java mode)
- **Identify** different view panels in debug mode
- **Select** a **breakpoint** in the code you have updated to demonstrate your work
- **Explore** and **analyze** the **variables** in your code

Important…

Some important concepts you need to know in order to make effective use of a debugger:

- Breakpoints
- Single Stepping
- Inspecting Variables

- Review the **Hybrid resources** and course materials to learn how the Eclipse debugger works to complete this part.

Starting the Debug Mode

- At the top bar of your Eclipse screen:
  - Click on **Window**, and then scroll down to **Perspective**. Scroll to **Open Perspective** and then select **Debug**
  - Or you can: Select **Run** and then select **Debug**
- If it is your first time running **Debug**, agree to the debug mode pop up window seen on your screen in the debug perspective.

Starting the Debug Mode (2)

- Notice that your screen changes. Explore the additional view panels included. What are they?

- Notice the **Debug** and **Java** icons at the far top right corner of your screen. You can use them to go back and forth on Java or Debug **modes.**

- Example of the new view panels that appear are the **Variables**, **Breakpoint** and **Expressions** (at the top right corner of your debug mode screen). What do they mean?

Inserting a Breakpoint in your code

- Ensure that your eclipse editor shows line numbers for your code
- Insert a **breakpoint** in your **updated** code:
  - Double click on the blue line margin corresponding to the desired code line number
  - **OR you can**: do a **right click** on the margin of the desired line number and then select **Toggle Breakpoint.**
- Explore how the **toggle breakpoint** **and disable breakpoint** work

Explore the following Step Commands in debug mode

- **Step into** - You can use this button to step into a method you wish to debug one step after another
- **Step over** – You can use this button to skip a method you do not wish to debug when invoked
- **Step Return -** You can use this button at the end of your debugging and having the debugger back to an initial point of start.

Explore the following Step Commands in debug mode (2)

- **Terminate** – You can use this button to stop the running of a program if there are no further analysis required, or if errors are encountered. The terminated program could e in the debug or normal mode.

- **Resume** – You can use this button to restart execution of the program again from any suspended state. This continues till another breakpoint.

Demonstrating Your Work

- **To obtain your mark, show your Professor**:
  - The code you updated
  - Run your program and show that the output of your code is correct.
  - Generate Javadoc to document the comments in your code and show the Javadoc output

Demonstrating Your Work (2)

- **Show your Professor how to**:
  - Select a **breakpoint** in your updated code
  - Run your code in debug mode (as a Java application):
    - Go to **Run** and select **Debug As** then select **Java Application**
    - Click **Yes** in the pop up window to agree to the debug perspective (if it pops up)
    - The breakpoint will be **highlighted** in your code.

Exploring the Step into and Step over

- Click on the **step into** icon at the top bar of your screen.
  - What do you observe?

- Click on the **step over** icon at the top bar of your screen.
  - What do you observe?

Demonstrating your Work (3)

Explore the variables

- The Variables, Breakpoint and Expression view panels are at the right-hand top corner of your screen.
- **Focus** on the Variables panel and carefully inspect the two columns (Name and Value).
- Observe the names of the **variables,** the **values** stored and **how**.

Demonstrating your Work (4)

- What **types** (of values) do you see?
- Click on **>** beside **traders** in the Name column
- Click on **>** beside **[1]** in the Name column
- Any changes in the **Value** Column?
- What does this change mean?

- Discuss any observations with your professor to obtain your marks.

Rubrics

Total marks for Lab 2 is 4%. You must submit and demo this lab to get any marks.

- Accurate completion of code and output – **1%**
- Javadoc style comments and generation - **1%**
- Correct analysis and use of the debugger - **2%**

Important: *Note that your analysis and correct use of the debugger in this lab depends on an accurately completed code. If you do not get the code right, you stand a risk of loosing the entire lab marks.*

Test your Knowledge further…

- When do you need to use a 2D array?
- Why should you consider switching to the **debug mode** from the **run** mode?
- Why should you ***step into*** a method?
- When should you ***step over*** a method?
- Why and where do you examine the code **variables** using the debugger? What should you expect to see?

References – Course textbooks

- Big Java Early Objects, 7/E. Author: Horstmann, C. Wiley. ISBN: eText: 978-1-119-49909-1 or loose-leaf paper: 978-1-119-74020-9.

- More Helpful Textbooks:

Java How to Program, Early Objects Plus MyProgrammingLab with Pearson eText -- Access Card Package, 11/E. Author: Deitel ISBN: 978013480027 1

Finalizing…

- Remember to **demonstrate your work** to your lab professor to receive your marks
- **Marks** are all or **nothing**. Show your work fully
- Remember to **review your hybrid tasks** as specified for each week
- Remember to **keep ahead** by checking if there are any more assessments due this week