Due date: **By the end of your lab session**

CSE-271: Object-Oriented Programming
Exercise #1
Max Points: 20

Name:	

For your own convenient reference – You should first save/rename this document using the naming convention `MUid_Exercise1.docx` (example: `raodm_Exercise1.docx`) prior to proceeding with this exercise.

Objectives: The objectives of this exercise are to:
1. Configure and start using Eclipse for Java programming
2. Learn to run Eclipse's style checker
3. Develop a simple Java program in Eclipse using skills from CSE-174
4. Recap Javadoc documentation to the program
5. Practice working with the debugger
6. Practice the procedure for style checking and submitting programs

Fill in answers to all of the questions. For some of the questions you can simply copy-paste appropriate text from Eclipse output into this document. You may discuss the questions or seek help from your neighbor, TA, and/or your instructor.

Part #1: One time setup of Eclipse (IDE)
Estimated time: 20 minutes

In this course we will be using Eclipse (IDE) for Java programming. You will need to setup and configure Eclipse. This is a one-time setup (as long as you are using the same computer) that involves the following steps:
1. If you are using your own personal computer then you will need to install Java and Eclipse via the following URLs:
   a. Java Development Kit (JDK): https://www.oracle.com/java/technologies/downloads/#java11
   b. Eclipse (IDE): https://www.eclipse.org/downloads/
2. Next configure Eclipse's code formatter and check style plug-in to be consistent with the CSE department's programming style guidelines as shown in the 1st video on this Canvas page: Configuring Eclipse for Java programming.

Due date: **By the end of your lab session**

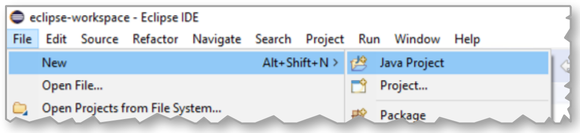
Due date: **By the end of your lab session**

Part #2: Creating a Java project in Eclipse

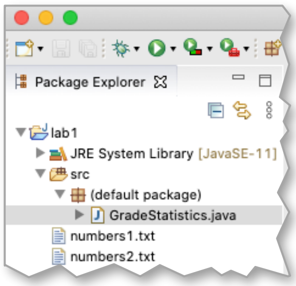
Estimated time: < 10 minutes

Next, create a new Java project in Eclipse and download the supplied starter code to it via the following procedure:

1. Launch Eclipse (IDE) using the default workspace.
2. Next, from Eclipse's main menu use the option File → New → Java project to create a new Java project titled `lab1`. Don't create any modules.


eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
New Alt+Shift+N >
Open File...
Open Projects from File System...
Java Project
Project...
Package

3. From Canvas, download the supplied starter code for this lab directly to the directory of your newly created Java project while paying attention to the following:
   a. The source (`.java`) files should be placed in the `src` folder
   b. The data files should be placed at the top-level folder
   c. In Eclipse, refresh the project. Your files should appear as shown in the adjacent screenshot



Part #3: Implement and test methods

Estimated time: 20 minutes

In this part of the exercise, you are required to implement the following three methods in the supplied starter program. **Test each method as you implement them by running your program in Eclipse**.

1. **`printGrades`**: This method should print the array of grades. Each grade is printed to 1 decimal place and is separated by a comma and blank space (*i.e.*, `", "`). For example given the array { 1.23, 2.47, 5.3321} this method prints them as `1.2, 2.5, 5.3`. Tips for this method:
   a. Given *n* grades, print the first *n* – 1 grades in a loop and then print the last grade
   b. It will be convenient to print via `System.out.printf("%.1f, ", grades[i]);`
2. **`getMean`**: Compute the average of all the grades supplied to this method. This method adds all the grades to find `sum`. It then divides the `sum` by the number of entries to compute the average.
3. **`getMinMax`**: This method is partially implemented for you. You need to complete the implementation of this method. This method computes and returns the minimum and maximum values in the list of grades. Note that the grades are unsorted (*i.e.*, the values in the array are in no specific order).

Due date: **By the end of your lab session**

Expected outputs:
Here are expected outputs from running the program, once all the methods have been correctly implemented. User inputs are shown in **bold**.

Test #1:

```
Enter name of data file to process: numbers1.txt
The grades are: 75.3, 66.8, 95.3, 100.0, 50.2, 78.9
Mean = 77.8
Min = 50.2 Max = 100.0
```

Test #2:

```
Enter name of data file to process: numbers2.txt
The grades are: 84.0, 97.0, 63.0, 98.0, 66.0, 92.0, 81.0, 90.0,
75.0, 53.0, 54.0, 50.0, 93.0, 70.0, 95.0, 89.0, 91.0, 87.0,
82.0, 68.0
Mean = 78.9
Min = 50.0 Max = 98.0
```

Part #4: Add Javadoc

Estimated time: 15 minutes

Background: Properly documenting each method in a program is an important skill that is valued at all technical jobs. Good documentation takes practice. Luckily Eclipse provides place holders to help with documentation.

Exercise: Add Javadoc (similar to documentation for other methods) to the comments to the first 2 methods (i.e., `populateGrades` and `printStatistics`) in the supplied starter code.

Part #5: Trace operations of the program via the debugger

Estimated time: 15 minutes

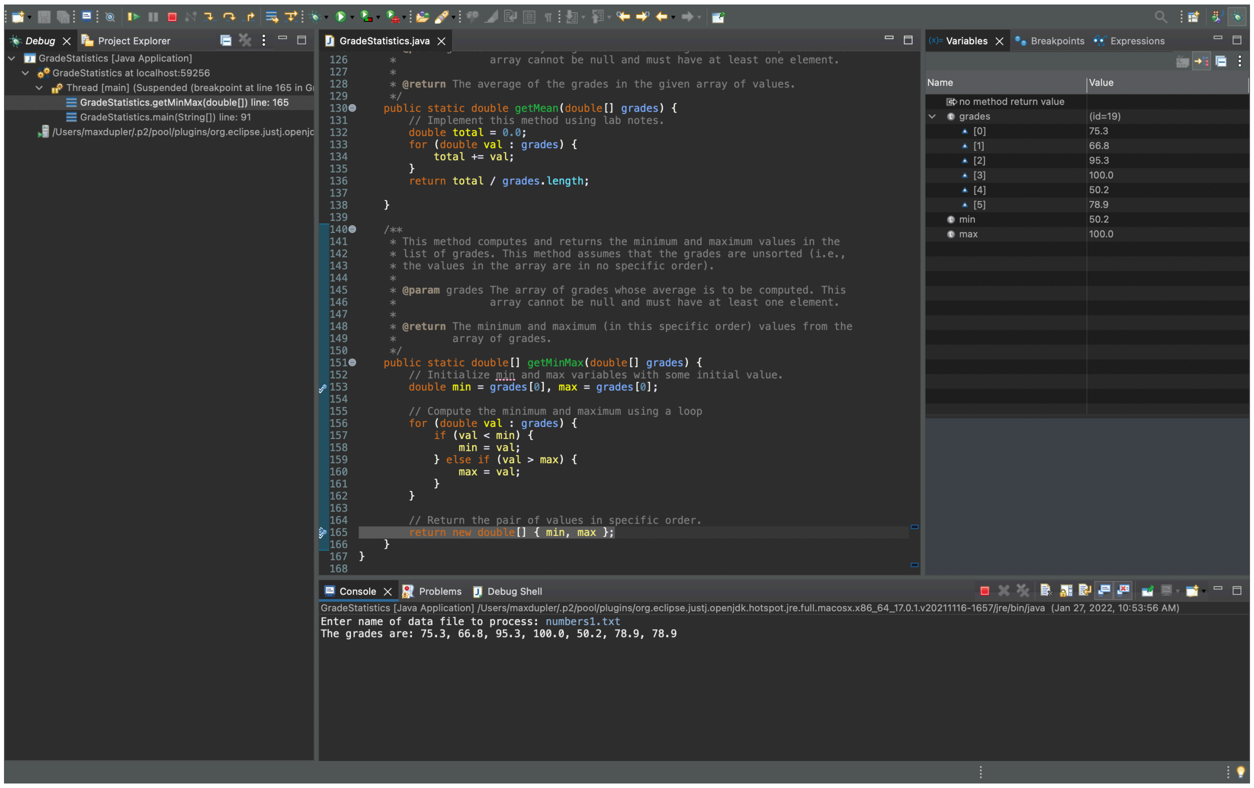
Background: The debugger is a very important tool that you should learn how to use. It will come in handy in future laboratory exercises and homework. Consequently, you should become very comfortable with using the debugger.

Exercise: Complete this part of the exercise using the following procedure:

1. If you have not used the debugger in Eclips, first review its usage via the videos posted on canvas.
2. In order to use the debugger, you first need to set a breakpoint. Set a breakpoint at the first line of the `getMinMax` method – *i.e.*, "`double min = grades[0], max = grades[0];`"

3. Run the program via the debugger and use `numbers1.txt` file as the input.
4. The debugger will stop at the breakpoint you have set. When prompted switch over to the debug perspective.
5. Step through the operations of the `getMinMax` method until you get to the last line of the method (just before returning values).
6. Make a screenshot of your Eclipse window and place it in the space below:

Place screenshot of debugger here:



```
GradeStatistics [Java Application]
  GradeStatistics at localhost:59256
    Thread [main] (Suspended (breakpoint at line 165 in Gr
      GradeStatistics.getMinMax(double[]) line: 165
      GradeStatistics.main(String[]) line: 91
  /Users/maxdupler/.p2/pool/plugins/org.eclipse.justj.openjc
```

```
126      *                 array cannot be null and must have at least one element.
127      *
128      * @return The average of the grades in the given array of values.
129      */
130     public static double getMean(double[] grades) {
131         // Implement this method using lab notes.
132         double total = 0.0;
133         for (double val : grades) {
134             total += val;
135         }
136         return total / grades.length;
137 
138     }
139 
140     /**
141      * This method computes and returns the minimum and maximum values in the
142      * list of grades. This method assumes that the grades are unsorted (i.e.,
143      * the values in the array are in no specific order).
144      *
145      * @param grades The array of grades whose average is to be computed. This
146      *                 array cannot be null and must have at least one element.
147      *
148      * @return The minimum and maximum (in this specific order) values from the
149      *          array of grades.
150      */
151     public static double[] getMinMax(double[] grades) {
152         // Initialize min and max variables with some initial value.
153         double min = grades[0], max = grades[0];
154 
155         // Compute the minimum and maximum using a loop
156         for (double val : grades) {
157             if (val < min) {
158                 min = val;
159             } else if (val > max) {
160                 max = val;
161             }
162         }
163 
164         // Return the pair of values in specific order.
165         return new double[] { min, max };
166     }
167 }
168 
```

Name	Value
no method return value	
grades	(id=19)
[0]	75.3
[1]	66.8
[2]	95.3
[3]	100.0
[4]	50.2
[5]	78.9
min	50.2
max	100.0

```
GradeStatistics [Java Application] /Users/maxdupler/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_17.0.1.v20211116-1657/jre/bin/java  (Jan 27, 2022, 10:53:56 AM)
Enter name of data file to process: numbers1.txt
The grades are: 75.3, 66.8, 95.3, 100.0, 50.2, 78.9, 78.9
```

Part #6: Checking style in Eclipse

Estimated time: < 10 minutes

Background: All programs will be required to adhere to the CSE department's style guidelines. Eclipse includes a source code formatter that will format the source code to be consistent with the department guidelines. In addition, Eclipse's check style plug-in will help check style. This will be the same style checks that are run when you submit your programs via Canvas.

Exercise: Run and fix style issues in the following manner:

1. In Eclipse, right click on the program and in the pop up meu select `Checkstyle` → `Check code with Checkstyle` option.
2. This will run the style checker and any style violations will be displayed as errors.
3. If there are style errors, fix them and repeat from step #1 until all style errors are fixed.

Part #7: Submit to Canvas via CODE plug-in

Estimated time: < 5 minutes

Background: Most (if not all) programming homework/project are required to be submitted via the CSE department's CODE plug-in. The plug-in has been designed to facilitate learning and teaching in programming centric courses such as this one. The CODE plug-in is setup to ensure: (1) your program compiles, (2) passes style checks, and (3) correctly operates for test cases. This eliminates issues with accidental incorrect submissions, incorrect solutions, etc. while promoting higher quality learning and outcomes for all.

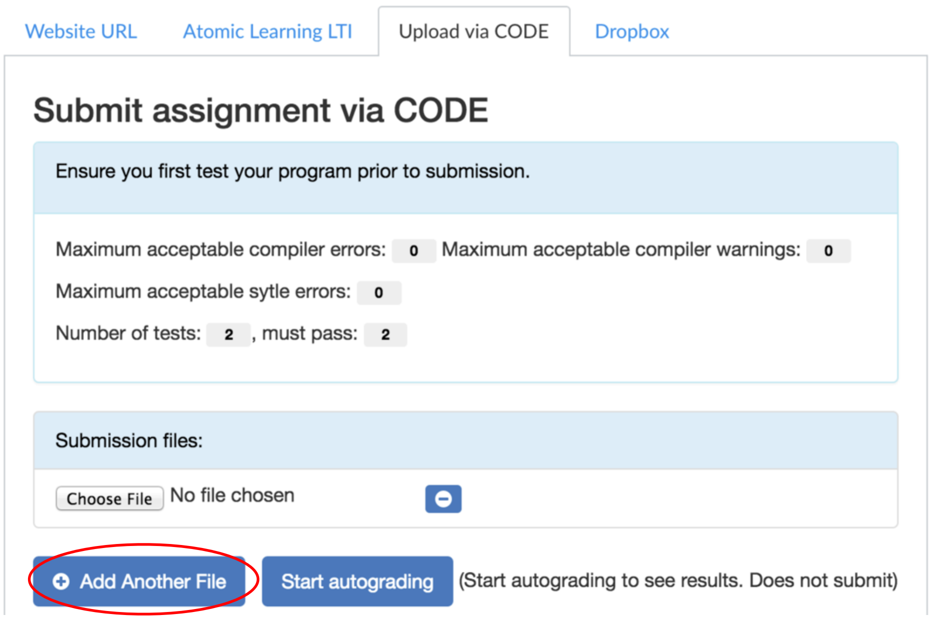
Exercise: In this part of the exercise, you will be submitting the necessary files via the Canvas CODE plug-in.

1. If this is your first time using the CODE plug-in then review the submission process via the following brief video demonstration -- https://youtu.be/P2bWUt5KqbU.
2. Save this MS-Word document as a PDF file – **Do not upload Word documents. Only submit PDF file**.

3. Practice submitting solutions via the CODE plug-in by submitting the following files:
   a. The `GradeStatistic.java` starter code
   b. Your lab notes for this exercise (*i.e.*, this document) saved as a PDF file.

4. Upload the files using the "`Upload via CODE`" tab shown in the screenshot below:

Due date: **By the end of your lab session**



Submit assignment via CODE

Ensure you first test your program prior to submission.

Maximum acceptable compiler errors: 0 Maximum acceptable compiler warnings: 0

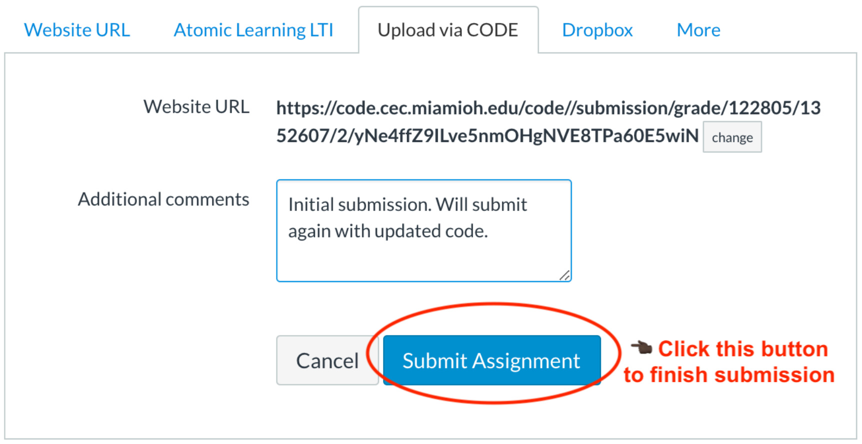
Maximum acceptable sytle errors: 0

Number of tests: 2 , must pass: 2

Submission files:

(Start autograding to see results. Does not submit)

Ensure you actually **submit** the URL generated by CODE plug-in in the final step as shown in the Video demonstration and in the screenshot below:



Website URL https://code.cec.miamioh.edu/code//submission/grade/122805/1352607/2/yNe4ffZ9ILve5nmOHgNVE8TPa60E5wiN

Additional comments: Initial submission. Will submit again with updated code.