CSE 174 – Fall 2021

lab2: 20 points – Due Saturday, Sep 4, 2021 by 11:59pm

NOTE: For indentation, if you still haven't set your tab to 4 spaces in jGrasp please go ahead and do it now. You can find the instructions for that on canvas. Or just use the space key on the keyboard instead to add 4 spaces.

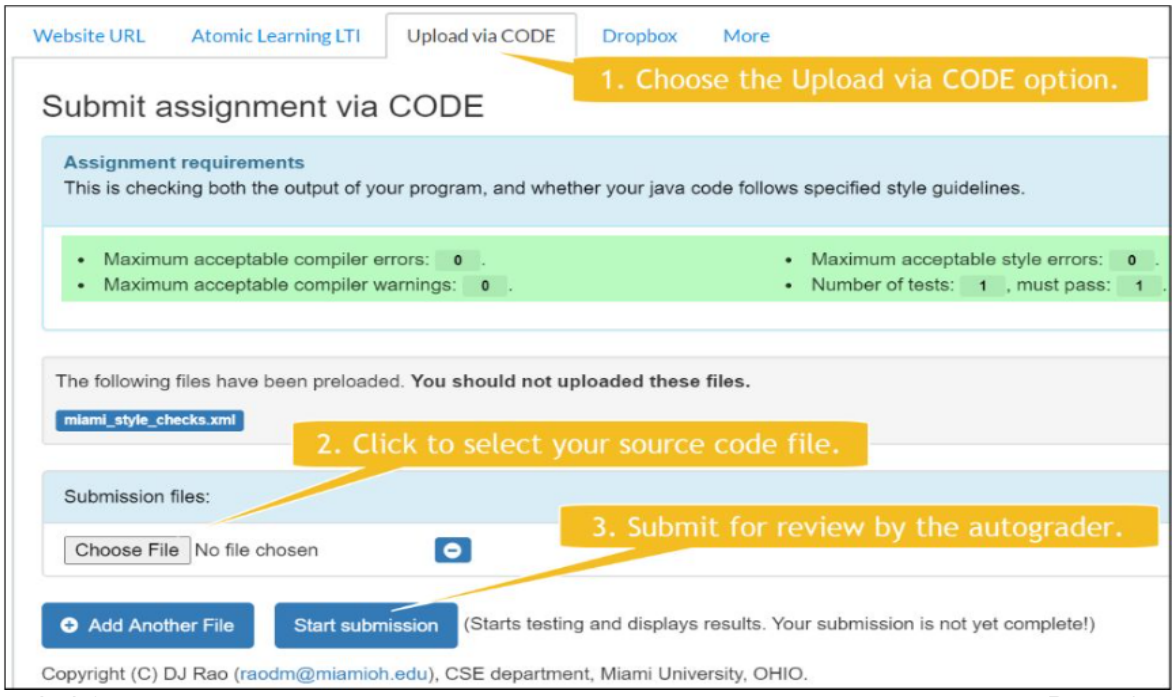
Part 1: Submitting assignments using CODE autograder plugin on Canvas

Many assignments in CSE 174 will be set up to use the "CODE" plugin on Canvas (https://code.cec.miamioh.edu/code/). The CODE plugin allows you to upload your source code file for an assignment, and then automatically check:

- to see if executing your compiled code is producing the expected result on some sample runs.
- to see if your source code follows standard style guidelines.

In this part of the lab, you will submit a file to Canvas, review the autograder output, adjust your code, and resubmit until it passes all tests.

1. Download AreaChecker.java.
2. In jGRASP, open this file, compile it, and run it. Verify that it produces output. The output is not correct, but don't fix it yet. We want to see how the Canvas autograder shows mistakes.
3. Log in on Canvas, and find the "Lab 2.1 - CODE Autograder" assignment. Click the "Submit Assignment" button.
4. Select the "Upload Via CODE" option.

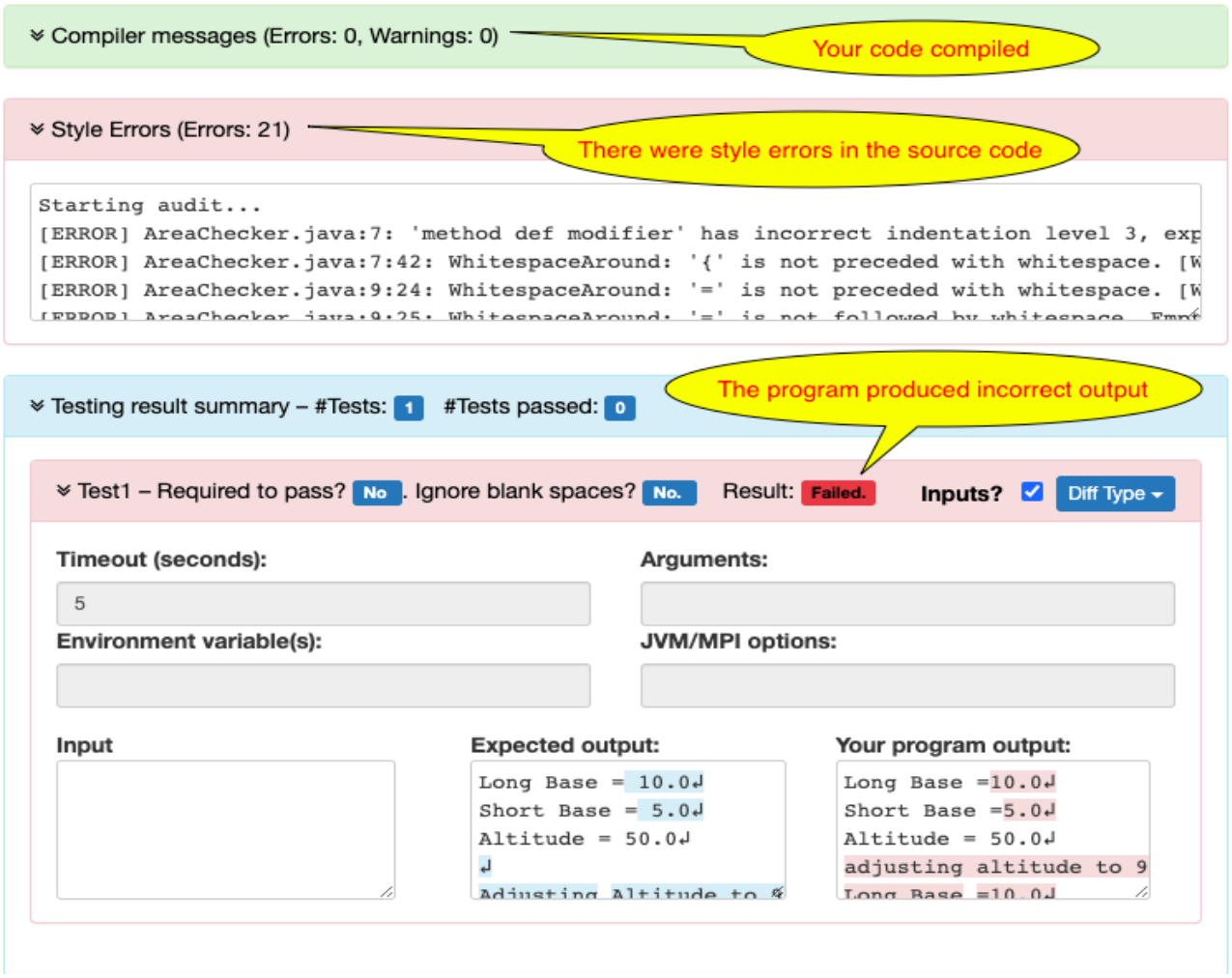


5. Review the section highlighted above in green, which tells you:

    - Code with compiler errors and warnings will not be accepted.
    - Your source code will be checked to see if it meets style guidelines. (zero style errors will be allowed)
    - There is one autograder test, and you must pass that test in order for your code to be accepted.

6. Use the "Choose File" button to browse to the AreaChecker.java file.
7. After you have selected the file, click the "Start submission" button to begin the autograder process.
8. It will take about 10-20 seconds to see your results. You should see results like the following, which indicate:

    - The source code was successfully compiled.
    - The source code has several style errors.
    - Running the compiled code did not produce the expected results.



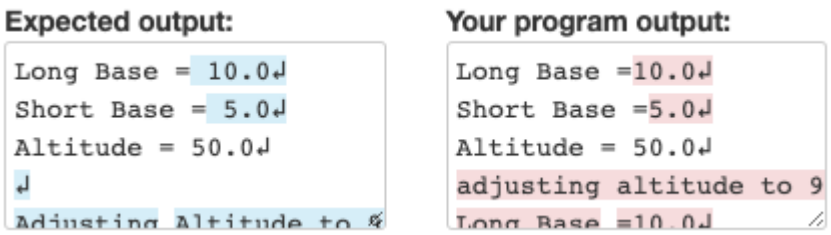
9. Let's focus first on fixing the style errors. Read through the autograder's style errors and use them to help you fix your code in jGRASP. Note that because there were a lot of style errors, not all errors are shown. See if you can find all of them on your own.

    - All errors indicate the source code line number and column number of the error. For example, line 7 has 3 spaces for indentation (which should be 4), or on line 7, column 42, before { should be a space.
    - Some of the errors indicate that the indentation is incorrect. Each level of indentation should be 4 spaces from the previous level. On Canvas find “Configuring jGRASP for Style Guidelines” to adjust “Tab Size” to 4, then use tab to fix the indentation.

10. You need to read errors carefully in order to understand what the issue is. If it says something that you can’t understand, please ask your instructor or TA.

11. Once you think you have fixed all the style errors, submit your code again to make sure there is no style error anymore. You can do it by clicking on the Start **a new submission** button.
12. Next step is to focus on fixing the errors in your output. The autograder shows side by side results of what your code produced, and what is expected, with the differences highlighted. This is known as the "Diff Output" (because it highlights differences). On the left, the "Expected" shows what your program should have produced. On the right it shows what your program actually did produce.

    - You can grab each window from the bottom-left corner and expand it to see more details.
    - The ↲ character shows that there is a linefeed (or Enter, which means you need to go to the next line) at the end of the line.

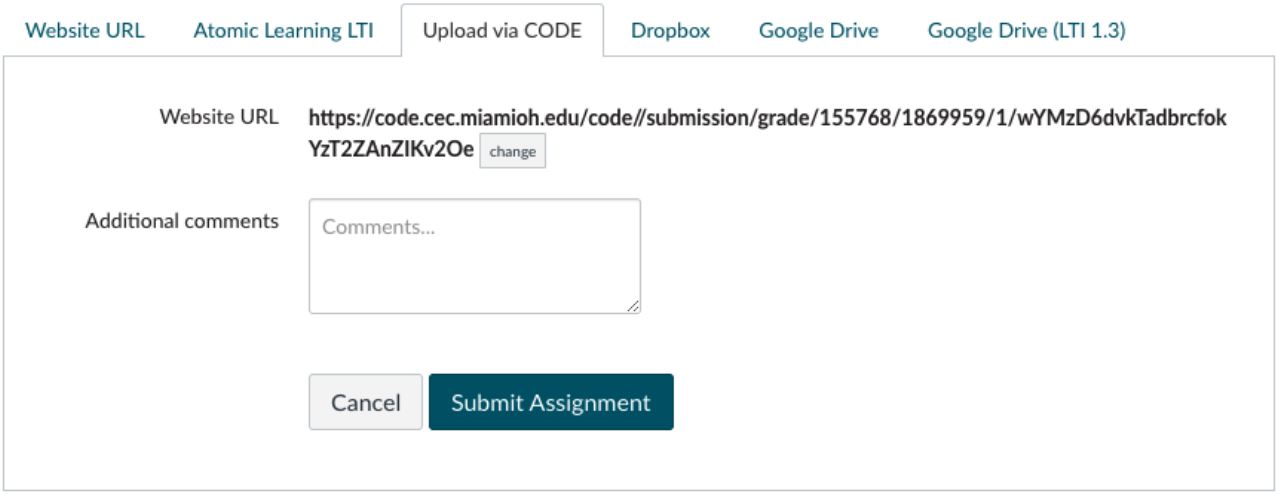


13. A few things need to be fixed in your code in order to produce the correct output:
    - Some of the output is missing spaces.
    - Empty lines need to be added in some places.
    - Some of the letters need to be uppercase

<continue to the next page>

14. Another way of fixing your style errors before submitting your code is to use the style checker in jGRASP. On canvas find “Configuring jGRASP for Style Guidelines” and learn how you can do it.
15. When you think you have fixed all style errors and all problems with your program's output, compile and run it in jGRASP to test it out. See if you can find any additional errors.
16. When you think you have everything completely correct, click the **Start a new submission** button and re-upload and resubmit to the autograder. Keep repeating this until the autograder indicates that your submission passes all requirements. When this happens, you will see the following:

17. When you see the "Congratulations", you should then click the green Accept button to use this submission as your final submission. **Note that this does not necessarily mean that your submission meets all of your instructor's requirements. It just means that it passed certain tests of the autograder.**
18. After clicking on the Accept button, it takes you to another window where it gives you a website url link.



19. You can click on Submit Assignment to **officially submit your code**.

Scoring Rubric:

- **Code meets output and style requirements (6 points):** You will receive full credit for this if the autograder accepts your submission. Otherwise your score will be zero.

<continue to the next page>

Part 2: Calculating Future Profits

Background

The Initech projects its profits for next 3 years based on the following table:

Predicted Incomes	Next Year	Second Year	Third Year
2,000,000	5.1%	6.0%	8.0%
2,500,000	7.2%	8.0%	10.5%
3,000,000	9.3%	10.1%	13.0%
4,000,000	11.2%	13.2%	16.8%

Problem

Your job is to create a new java class named **ProjectProfits** which:

- Calculates the average profit for each year.
- Calculate the total of the average profits for the next 3 years.
- Generates the following output:

```
----jGRASP exec: java ProjectProfits
The average profit for the next year: 252250.0
The average profit for the second year: 287750.0
The average profit for the third year: 371125.0

Total profit in next 3 years: 911125

----jGRASP: operation complete.
```

Example of calculating the average profit for the next year:

1. Calculate the 5.1% of 2000000 = 102000
2. Calculate the 7.2% of 2500000
3. Calculate the 9.2% of 3000000
4. Calculate the 11.2% of 4000000
5. Calculate the average of all the results you got from step 1-4.

<continue to the next page>

Tips & Rules

1. Always use **meaningful** variable names. Variable names such as "a", "b", "c" sound good at first till you have to go back and read your code.
2. Always pick the best variable **types** for the variables. If you need the decimal point, use a data type that has one.
3. You **can not** use a **double type value or variable**.
4. It seems like that using **constant variables** for holding predicted incomes from the table is a good option since their value always stays the same. So, use constant variables for those.
5. Save all the values inside variables first, then use those variables in your calculations instead of using the values directly. For example: Instead of having a line like this:
   float total = 32452 * 3232 / 100.0;
   You should save those values inside variables first and then use those variables in your calculations:
   int profit1 = 32452;
   int profit2 = 3232;
   float percentage1 = 100.0f;
   Float total = profit1 * profit2 / percentage1;
6. Format your **output** to match the above as closely as possible. This includes spacing, capitalization, punctuation and empty lines.
7. Remember to follow style guidelines as you write your code.

8. **Style Advice:**
   - Declare and initialize your constant variables at the beginning of your code.
   - Do **not declare and Initialize** other variables until you are **ready** to **use** them for the first time.
   - If you are using jGRASP style checkers, you will get style errors because of the naming of your constant variables. It's okay, you can ignore the errors related to the constant variables. You won't get those errors when you submit your code through Code Plugin.

Submission

When you believe that you have correctly solved the problem, and that your code follows the style guidelines, submit it on canvas for "Lab 2.2 - Calculating Future Profits". If the autograder rejects your solution, fix it and resubmit it again and again until there is no error. You will lose points if you submit your code with style errors.

<continue to the next page>

Scoring Rubric:

- **Program produces correct output and follows style guidelines (5 points):** You will receive full credit for this if the autograder accepts your submission and your submission has no style errors. Otherwise your score will be zero.
- **Program uses variables wisely (3 points):** Your code uses meaningful variable names and correct data types.
- **Program uses variables in calculations (2 points):** Your code saves the values inside variables first, then uses those variables in calculations.
- **Program uses constant variables (2 Points):** Your code uses constant variables.
- **Program has comments at the top and inside your code (2 Points):** Your code should have comments at the top with your name, also a few comments inside your code to explain necessary parts of your code.