Introduction To Programming
Tutorial 3
(See Canvas→Assignments for due dates and marks)

Note: Do not include your name, student ID or any personally identifiable info in your submission as the submission may be used for peer reviews; your submission will not be lost as Canvas keeps track of these internally.

Please follow all of the steps below in the given sequence:

1. Read all unread announcements and unread replies to announcements under Canvas→Announcements.

2.1 Do any missed tutorials before going further.

2.2 Watch any unwatched recordings of the compulsory **Weekly Live Lecture** and any important videos in the Extra Videos Playlist.

2.3 If you need help in addition to what has been shown in the compulsory weekly live lecture, you are also expected to speak to your **group tutor via discussion forums** and attend/watch their live sessions. Please note that group tutors cannot debug your assessment code on your behalf as debugging is a part of every programming assessment.

2.4 **If you still have have any unresolved questions** or **if you need further feedback**, post the relevant parts of your submitted work in a new post under Canvas→Discussions→Tutorial discussions and ask from your group tutor. E.g. you can ask "*In the live lesson Gayan did ___ with ___. I didn't do ___ so should I be doing this as well?*", etc. Please note that the university requires teaching to be conducted in an equitable manner so your tutors will require you to post questions in the discussion forums.

3. Check any available feedback of your previous submissions and if you have any unresolved questions or if you need further feedback, post the relevant parts of your submitted work in a new post under Canvas→Discussions→Tutorial discussions and ask from your tutor. E.g. you can ask "*Gayan showed _____ but I did mine like _____, so which is the better approach and why?*", etc. Please note that the university requires teaching to be conducted in an equitable manner so please only use email for matters such as special consideration.

4. Follow the materials under Canvas→Modules→Week 3…

5.1 Step 4 (above) showed that we can use if-statements to perform checks and we can get the user to enter values for our programs to process. Create a new Java project and get either the 'full example of `Scanner`' or 'full example of `JOptionPane`' from the 'taking inputs' notes to work.

5.2 Continue to the modify the program from 5.1 as follows (must follow instructions exactly):

1. Rename the class so that it says "`public class CPT120GradeMaker`" instead.
2. Ask the user to enter 3 separate mark inputs either by using JoptionPane or Scanner: (1) An "IIE total", (2) an "assignment total" and (3) an "exam total". What data type would you use for storing the values and explain what other options would have been there (along with pros and cons) in a code comment.
3. Calculate the non-exam total by adding the IIE total with the assignment total. The non-exam total should be capped to 50, if it goes over. Explain in a comment how you achieve this. E.g. if the IIE total is 25 and the assignment total is 27, the non-exam total must be 50.
4. Calculate the final course mark (an integer) by adding the non-exam total with the exam total then round up any decimal places to the next nearest int. E.g. if the non-exam total is 50 and the exam total is 33.1, the final course mark must be rounded up from 83.1 to 84. Hint: You need to independently investigate in the Math class documentation for a suitable method that would round values up to the nearest int and additionally any related examples online. (Your tutor will not be able to debug or give you the answer to this or related tasks before submission so please ask all questions in a general manner.)
5. Using the criteria given in the RMIT Higher Education Grades page for numerical mark ranges, determine and display either the grade code or grade alongside the calculated total. E.g. "84 HD".

Note: Input validation is covered in the while-loops lesson. There is no need to validate the user inputs. You may assume that the user will enter positive values and in the requested data type (e.g. will not enter text when a int is asked for, etc.).

Submission Checklist:
1. Ensure that your code does not have any red dots (Java errors) as code with such errors cannot be tested/marked and will receive 0 for that submission. If your code has red-dots, refer back to similar code and fix the error or remove the code that is causing the problem. You must not leave any commenteded out code in your submissions. Yellow dots are warnings and these are different.
2. Ensure that you have added comments to your .java file explaining what you have done and any potential alternative approaches.
3. Format your code (e.g. Eclipse→Source→Format).
4. Go to Canvas→Assignments→**Independent Investigative Effort 3** and select 'submit assignment'.
5. Select to attach files from your computer, navigate to your Eclipse workspace folder→Project folder→src folder and select the (one) final version of your `CPT120GradeMaker.java` file. Please do not submit more than 1 file as it delays the marking process. You can also context select on the .java file name from package/project explorer inside Eclipse and find its exact location. Only the last submission is the official submission.
5. Verify your submitted files as shown during the week 1 chat session.

Having trouble with usernames, passwords, access, etc.? Please call the RMIT IT Service and Support Centre for quick help on 03-9925 8888 and remember to ask for a reference number and pass it on to your instructor.

Need extensions or special consideration? Please follow details and process below:
https://www.rmit.edu.au/students/student-essentials/assessment-and-exams/assessment/special-consideration