Introduction To Programming
Tutorial 2
(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 instructor. 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 2…

5.1: As shown in the week 2 materials under Canvas→Modules, Java has 8 primitive data types.

1. Create a new Eclipse *project* then a new Java class named TypeDemo inside that project.

2. Now, inside the *main* method of your TypeDemo class, create 8 variables; one for each primitive data type. The names of the variables should be descriptive of the type of information. E.g. We store the age of something in a variable called 'age' as shown below:

```
int age;
```

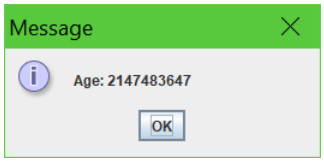
I.e. do not name the variable *a*, *b*, etc. Use your imagination and try and give the variable a "purpose". When you do it, your *int* variable must be named differently what's shown in the example here. (Gayan will discuss what makes a bad variable name during the live lecture)

3. Next, for variables of integer (whole number) data types (hint: there are four, not just int), assign the largest value that you are allowed to assign to each (refer to values given under Oracle Java Tutorials Primitive Data Types page for exact values). E.g: For int, the largest value is given as 2^{31}-*1*, we can then use a calculator to find out that this value is 2147483647 and hard-code it to the variable as follows:

```
age=2147483647;
```
or
```
int age=2147483647;
```

For variables of other data types that you create, assign compatible values for each. (Gayan will discuss how you can find out the maximum values for numerical data types without calculating them)

4. Finally, compose a single String variable to contain a summary of the field/variable (one per line) names and their values and then display it (you may also use System.out… instead of JOptionPane...). E.g.



Of course, the above example only has the *int* field/variable and its value.
Yours must have all 8 and just use your imagination for the variable names!

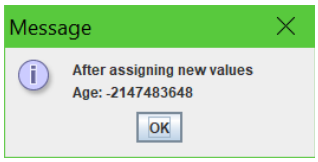
5.2: Continue to modify the TypeDemo class (from step 5) above as per the following instructions (there is no need to show/submit the version from 5.1 separately): After displaying the String (from exercise 1), increment the existing values of the number variables by 1. You can do this by saying either:

```
age+=1;
or
age++;
```

For the non-numerical variables (e.g. boolean, etc.), assign completely new new values.

Hint: When re-assigning values, we do not mention the data type (*int*, in this case) as we have already declared the variable with a data type before.

Finally, create a similar message to the one that you created at the end of exercise 1, create a second message that shows the new values, after the new values were re-assigned. E.g.



Again, yours must show all 8 fields/variable names and their new values (and you can use System.out… instead). If you do this right, the integer values will now show negative values (Optional: How does a positive number suddenly become a negative one? Listen to Gayan's explanation during the live lecture lesson)

In summary, after completion of exercise 2, your program should produce two outputs, showing "before" and "after" summaries of the variables and their contents.

Optional Exercise: You have seen the += and ++ operators (as in age+=1 or age++). Have you seen the rest of the Java operators? Which of these are important in everyday programming? What is operator precedence? There is no need to write answers to these but please follow Gayan's explanation during the compulsory lecture lesson for this tutorial (weekly live lesson) to find out what they are, which ones are important, relevant to this unit, etc.

6. Your Assignment 1 also requires you to store values like character names, relevant numbers, etc. in variables. For this week's dummy submission of Assignment 1, at least copy the code from TypeDemo (exercises 1 and 2 above) in to your Assignment 1's java file, get it to work and submit your Assignment 1's java file via Canvas→Assignments→Assignment 1 (you **can improve and re-submit any number of times before the deadline**). Tip: At this stage it is fine if you do not have user inputs; programs are easier to write when they are developed iteratively. You can safely ignore this step if you have made your final submission; dummy submissions are for those who are yet to get started.

Submission Checklist:
1. Format your code (e.g. Eclipse→Source→Format).
2. Go to Canvas→Assignments→**Independent Investigative Effort 2** and select 'submit assignment'.
3. 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 **`TypeDemo.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.
4. 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