Lab 5

To pass this exercise you must:

- Complete the exercise below
- After completing the task, submit your zipped java source files to Canvas for assessment
- Discuss your work with your tutor for feedback
- Try to get ok from your tutor before you submit by the end of the tutorial

Exercise

Task:
1. Define a class called `Employee` that will be used represent an employee entity. The class should contain instance variables for name (String), employee ID (Integer) and salary (Double). Include a constructor that takes parameters to initialise the instance variables. Include a getter and a setter method for each variable. Include a `toString()` method that returns a string representation of the employee which should consist all the instance variables.

2. Create a tester class `EmployeeTester` with a main method.
   1) Ask for user inputs to create at least 3 employee objects with different values.
   2) Use an ArrayList object to add each employee object in
   3) Call the toString() method to display the objects
   4) For each object, change the value of a variable that you choose from the three instance variables with the setter method and display the value by calling the getter method.
   5) Call the toString() method to display the objects by using a loop for the **ArrayList** object.

Submission
Zip your java files and submit the zipped file to the Canvas for assessment.

Marking scheme

- The employee class (**5 marks**) & the tester class (**5 marks**). Both class should also meet the following requirements:
  1. The employee class is defined as required & the tester class works as required (2 marks).
  2. A proper class header comment should follow java doc style (0.5 marks).
  3. A proper method header comment for each method, which has the following information (1 mark).
     a) javadoc comment beginning with /** and end with */
     b) purpose of the method.
     c) @pararm name of the parameter and description (if the method takes a parameter)
     d) @return name of the retune variable and description (if the method returns a value)
  4. Readability: name conventions (variable name, constant name, class name), meaningful names, indentation, comments for each variable (0.5 marks).
  5. Keep the user well informed with proper messages (1 mark).