Computing Science and Mathematics

CSCU9T4 Practical (22nd January) Spring 2018

The aim of the first few practical sessions is to implement a Training record program. We start with a reduced version, to do just records of single runs. This program is to support a sports trainer who wants to keep a record of the times and distances her athletes run over the season. You are given a simple program with the ability to let you add and look up records over the years. Dates should be entered using DD MM YYYY format. Times in hh:mm:ss format.

The existing program makes use of Java library classes such as `ArrayList` and processes the contents of a list using a `ListIterator`. Your extensions must also use iterators and parameterised types. We assume that you will be using the Java API to gain more information about iterators and their use. Your extensions should be coded as elegantly and efficiently as possible.

You are reminded that while it is OK to discuss general points with your colleagues, copying is a serious offence. If there is any suggestion of sharing of code between class members then you will be asked to submit your code and it will be checked for plagiarism.

Checkpoint

Demonstrate to a tutor your modified TrainingRecord program. It should include the (functioning) FindAllByDate button. If you don't finish this work today, finish it off in your own time and get it checked at the next class.

THE ABSOLUTE DEADLINE FOR CHECKING THIS WORK IS THE LAB ON MONDAY 5TH FEBRUARY 2018.

1. Copy **My Computer**: **Groups on Wide**: **CSCU9T4: Practicals: Practical 1 (Training Record)** to your home folder. Open your version of TrainingRecord in **BlueJ** and run the program. There are eight TextFields, two Buttons, and a TextArea. The Add button is used to add a new entry, represented as Name, Month, Day, Year, Hours, Minutes, Seconds and Distance to the list. The LookUpByDate button is used to find a single entry on a given day/month/year combination.

   Run the program and add new entries. Note there are some entries hard-coded in the file to help you test the program. What is the underlying object model here? Sketch it using UML to be sure you understand.

P.T.O.

2. Add a new button FindAllByDate to the GUI. Add statements to the actionPerformed method to handle this button. Use a place holder, e.g. when it is pressed the string 'Not implemented yet' is displayed. Test this works.

3. Implement the FindAllByDate button. Now when you enter a Day and Month and Year and click FindAllByDate, the program will display the names and times of all runs for that day (not just the first one). The output should be shown in the output area with one run entry per line.

   Use lookupEntry as a model for your new method to look up all entries on a particular day. Start by asking which entry lookupEntry returns. How can you make it return all entries in a single string? (The advantage of this is that all the work of constructing the output string is in the TrainingRecord class (rather than the TrainingRecordGUI class).

4. (optional) Experiment with the program to see what kinds of input it will accept. It is not very robust, is it? It fails, with an exception, if the value entered for Day, Month or Year is not an integer. And it accepts the empty string as a Name. It also allows you to add the same entries over and over. Improve the program so that it handles bad input in a sensible way, for example, by displaying a suitable error message in the TextArea. You may assume that name, day, month, year is a unique key for running records (i.e. your athletes do at most one run per day).

You will be adding to this code next week.