**Module**: Level 8 – Programming – SD1 & GD1

**CA**: Final Exam 2021 - Online

**Value**: 30% of Module Mark

**Date**: 5th May 2021 @ 11am

**Duration**: 2 hours 15mins (including upload)

**Instructions:** Students cannot source assistance from any person or website other than Moodle and their own files.

Part A – 70 Marks – 80 mins. Upload complete application (4 files) to Moodle assignment.

Part B – 30 Marks – 45 mins. Upload methods only to Moodle quiz.

* **Part A** of the Exam is open book as regards course notes, examples and exercises
* Download the FOUR files from Moodle:   
   ***Activity.java, FitnessTracker.java, MainAppV1.java, MainAppV2.java***
* Use the code in these classes as the starting point for answering the questions.
* Students must place code in appropriate location (e.g. Place the testing code in the correct MainApp file after the appropriate comment)
* Final upload version must be runnable from *main()* in ***MainAppV2.java*** & ***MainAppV3.java .*** Place ***c***omments around any non-runnable code.
* The exact structure of the code is left upto the student but to obtain maximum marks, **students must write formatted maintainable** code which promotes code reuse and makes good and appropriate use of programming constructs e.g. loops, methods, arguments etc.
* ***TESTING*** is important but you do not need to create either an array or ArrayList in the ***MainApp*** files ***–*** just use some sample Activity objects and test as appropriate.
* ***UPLOAD***:
  + Create a ZIP file using your own name as the name of the file.
  + Copy in the FOUR/FIVE ***.java*** files.
  + Upload it to the appropriate Moodle link.

#### Question 1 (10 marks)

#### Question 2 (8 marks)

#### Question 3 (20 marks)

#### Question 4 (22 marks)

#### Question 5 (10 marks)

A fitness social media app is storing its users activities’ information. Users are able to share their walks, runs, swims, etc. by adding the information saved on their Fitness Tracker, such as FitBit, Apple Watch etc.

#### Question 1 (10 marks)

Modify ***Activity.java*** as follows:

1. Insert an additional field ***type*** to hold activity type details e.g. run, walk, swim, cycle etc. Insert additional getter() and setter(). Update the constructor.
2. Add two additional constructors (including the default no-argument constructor).
3. Insert an appropriate toString() method for the Activity class.
4. Insert test code for the Activity class in ***MainAppV1.java*** where indicated. This test code should confirm that a) b) c) are working correctly.

#### Question 2 (8 marks)

Modify ***Activity.java*** as follows, testing each modification from ***MainAppV1.java***

1. Write a method called ***increaseDuration()*** which will increase the duration of an activity by a specified number of minutes. TEST***.***
2. Code a compareTo() method for **Activity** based on duration. TEST by comparing two Activity objects in ***MainAppV1***.

#### Question 3 (20 marks)

Modify ***FitnessTracker.java*** as follows (with testing in ***MainAppV2*** where indicated):

1. Insert an additional field (an ArrayList) to hold details of all the activities belonging to a **FittnessTracker** - name your field: **activities**.
2. Update current **FitnessTracker** as appropriate (constructors, getter() setter(), toString()).
3. Write a method called **addActivity()** to add an activity to a tracker.
4. Test your modifications (a) (b) (c) at start of MainAppV2 (see comments in code) by creating a **FitnessTracker** object and inserting 5 **Activity** objects.
5. Write a method **displayAllActivities()** to display all activities’ details neatly in a table. TEST in MainAppV2.

#### Question 4 (22 marks)

Modify your ***FitnessTracker*** class as follows (with appropriate testing in ***MainAppV2***):

1. Write a method to determine and output the activities’ date and time (only) of activities with a durationt longer than a particular number of minutes. Call your method ***displayLongerActivities().***
2. Write a method to count how many activities allow access to map details. Call your method ***countMaps().***
3. Write a method to calculate the average duration of a Fitness Tracker’s activities. Call your method ***calculateAverageDurations().***
4. Write a method to determine if any activities are of a particular type. Modify the method making it as generally useful as possible.
5. Display the activities’ details sorted by duration.

#### Question 5 (10 marks)

#### Only attempt this question when Q1-Q4 are working correctly.

#### In a new file called MainAppV3:

1. Improve the user interface of your application by coding a single level menu of options.
2. Include JavaDoc comments for each of your methods in ***FitnessTracker.***
3. Insert some appropriate additional functionality for TOP Marks!

**Declare & create an ArrayList**:

ArrayList<Type> name = new ArrayList<>();

|  |
| --- |
| **Method Summary – ArrayList class** |

|  |  |
| --- | --- |
| add(value) | appends value at end of list |
| add(index, value) | inserts given value at the given index, shifting current and subsequent values (if any) to the right |
| clear() | removes all elements of the list |
| indexOf(value) | returns first index where given value is found in list (-1 if not found) |
| get(index) | returns the value at given index |
| remove(index) | removes/returns value at given index, shifting subsequent values to the left |
| set(index, value) | replaces value at given index with given value |
| size() | returns the number of elements in list |
| toString() | returns a string representation of the list |
| addAll(list)  addAll(index, list) | adds all elements from the given list to this list  (at the end of the list, or inserts them at the given index) |
| contains(value) | returns true if given value is found somewhere in this list |
| containsAll(list) | returns true if this list contains every element from given list |
| equals(list) | returns true if given other list contains the same elements |
| lastIndexOf(value) | returns last index value if found in list (-1 if not found) |
| remove(value) | finds and removes the given value from this list |
| removeAll(list) | removes any elements found in the given list from this list |
| retainAll(list) | removes any elements not found in given list from this list |
| subList(from, to) | returns the sub-portion of the list between indexes from (inclusive) and to (exclusive) |
| toArray() | returns the elements in this list as an array |

|  |  |
| --- | --- |
| **Method Summary – String class** | |
| char | [**charAt**](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html#charAt(int))(int index)            Returns the character at the specified index. |
| int | [**compareTo**](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html#compareTo(java.lang.String))([String](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html) anotherString)         Compares two strings lexicographically (alphabetically). |
| int | [**compareToIgnoreCase**](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html#compareToIgnoreCase(java.lang.String))([String](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html) str)            Compares two strings lexicographically, ignoring case. |
| [String](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html) | [**concat**](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html#concat(java.lang.String))([String](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html) str)         Concatenates the specified string to the end of this string. |
| boolean | [**equals**](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\Local%20Settings\Temporary%20Internet%20Files\OLK6C\j2sdk1.4.0docsapijavalangString.html#equals(java.lang.Object))([Object](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\Local%20Settings\Temporary%20Internet%20Files\OLK6C\j2sdk1.4.0docsapijavalangObject.html) anObject)           Compares this string to the specified object. |
| boolean | [**equalsIgnoreCase**](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html#equalsIgnoreCase(java.lang.String))([String](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html) anotherString)         Compares this String to another String, ignoring case. |
| int | [**indexOf**](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html#indexOf(int))(int ch)            Returns the index within this string of the first occurrence of the specified character. |
| int | [**indexOf**](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html#indexOf(int, int))(int ch, int fromIndex)        Returns the index within this string of the first occurrence of the character, starting search at the specified index. |
| int | [**indexOf**](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html#indexOf(java.lang.String))([String](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html) str)            Returns the index within this string of the first occurrence of the substring. |
| int | [**indexOf**](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html#indexOf(java.lang.String, int))([String](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html) str, int fromIndex)          Returns the index within this string of the first occurrence of the specified substring, starting at the specified index. |
| int | [**length**](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html#length())()            Returns the length of this string. |
| [String](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html) | [**substring**](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html#substring(int))(int beginIndex)         Returns a new string that is a substring of this string. |
| [String](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html) | [**substring**](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html#substring(int, int))(int beginIndex, int endIndex)       Returns a new string that is a substring of this string. |
| [String](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html) | [**toLowerCase**](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html#toLowerCase())() Converts all of the characters in this String to lower case |
| [String](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html) | [**toUpperCase**](file:///G:\Documents\DKIT\Courses\WINNT\Profiles\annl\0203Alison\Question%20Sheets\jdk1.3docsapijavalangString.html#toUpperCase())()            Converts all of the characters in this String to upper case |