GALWAY-MAYO INSTITUTE OF TECHNOLOGY

Department of Computing & Mathematics

H.Dip in Software Development

Advanced Object Oriented Software Development
Main Assignment (50%)

A Java Indexing and Dictionary API

Introduction

You are required to design and develop an indexing API in Java that allows a word index to be created from an e-book (or maybe even a URL). Logically, the index should contain a list of words that relate to the page numbers they appear on and the dictionary defined meaning of the word:

Word	Details
Orchard	**Definitions:** "Orchard","n.","A garden." "Orchard","n.","An inclosure containing fruit trees; also, the fruit trees, collectively; -- used especially of apples, peaches, pears, cherries, plums, or the like, less frequently of nutbearing trees and of sugar maple trees." **Pages:** {1, 17, 19, 87, 65}
Curiosity	**Definitions:** "Curiosity","n.","The state or quality or being curious; nicety; accuracy; exactness; elaboration." "Curiosity","n.","Disposition to inquire, investigate, or seek after knowledge; a desire to gratify the mind with new information or objects of interest; inquisitiveness." "Curiosity","n.","That which is curious, or fitted to excite or reward attention." **Pages:** {4, 23, 37, 41, 88}

```
Map<String, WordDetail> index = new HashMap<String, WordDetail>(); //Average O(1)
Map<String, WordDetail> index = new TreeMap<String, WordDetail>(); //Guaranteed O(log(n))
```

Your application should parse a specified e-book and build an index by examining every word in each sentence. A number of e-books, in TXT format, are available on Moodle - you can assume that a page of text corresponds to ***40 lines of text or blank spaces***. Your API should also parse the 15Mb file "**dictionary.csv**", which contains a set of over 176,000 words and their definitions.

Your application should provide a useful abstraction (in the form of an interface) of the type of functionality you would expect from an index, e.g. a successful search for a word should return a list of page numbers. Extra marks will be given for designs that exhibit more behavior. Please note the following:

- The user of the API should not know how you are building the index, i.e. you should ***encapsulate your data structure***.
- The API should contain a method that ***outputs the index***, with indentations, as text.
- All API operations on the index should be ***case insensitive***.
- The ***definitions*** of words read from the dictionary should only be added to words that exist in the text or URL being parsed.
- A list of ***ignore words*** have been provided on Moodle. Your API should exclude these words from the index.
- ***Document your code using the JavaDoc standard***. JavaDocs are an industry standard for inline documentation of Java classes. A quick JavaDoc tutorial, from the University of Birmingham, is available on Moodle. Eclipse provides full support for the JavaDoc standard, including intelli-sense drop down suggestions.

Note that the whole point of this assignment is for you to demonstrate an understanding of the principles of object-oriented design by using abstraction, encapsulation, composition, inheritance and polymorphism WELL throughout the application. Hence, you are also required to provide a UML diagram of your design (a photograph of a sketch will suffice). Please pay particular attention to how your application must be packaged and submitted.

Deployment and Submission

- ***The project must be submitted by midnight on Sunday 10th April 2016.*** The project must be submitted as a Zip archive ***(not a rar or WinRar file)*** using the Moodle upload utility. You can find the area to upload the project under the " A Java Indexing and Dictionary API - (50%) Assignment Upload" heading in the "Notices and Assignments" section of Moodle.
- The name of the Zip archive should be <*id*>.zip where <*id*> is your GMIT student number.
- The Zip archive should have the following structure (do NOT submit the assignment as an Eclipse project):

Marks	**Category**
src	A directory that contains the packaged source code for your application.
README.txt	A text file outlining the main features of your application, with a set of instructions for running the programme.
design.png	A UML diagram of your API design. Your UML diagram should only show the relationships between the key classes in your design. Do not show methods or variables in your class diagram.
docs	A directory containing the JavaDocs for your application.
index.jar	A Java Application Archive containing the classes in you application. Use the following syntax to create a JAR from a command prompt: ***jar –cf index.jar ie/gmit/*.class***

Marking Scheme

Marks for the project will be applied using the following criteria:

Marks	**Category**
(50%)	Robustness
(10%)	Abstraction
(10%)	Encapsulation
(20%)	Packaging & Distribution (including docs and UML)
(10%)	Documented (and relevant) extras.