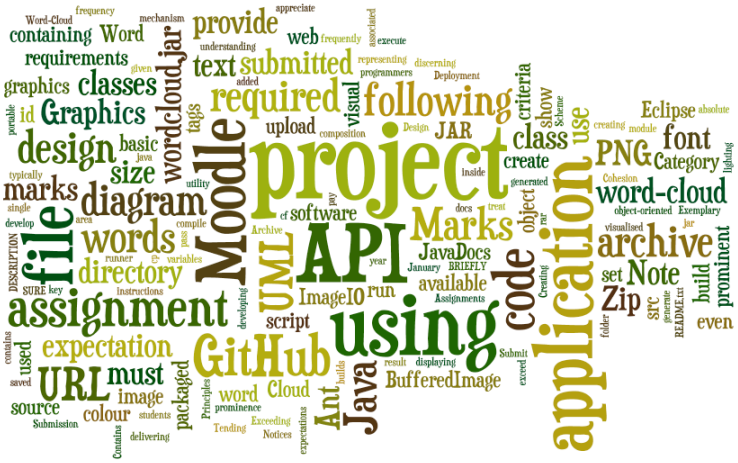
GALWAY-MAYO INSTITUTE OF TECHNOLOGY

Department of Computing & Mathematics

B.Sc. Software Development – Advanced Object-Oriented Design Principles & Patterns (2015)

ASSIGNMENT DESCRIPTION & SCHEDULE

A Java Word Cloud API



Note: This assignment will constitute 50% of the total marks for this module.

Overview

Word-clouds are a mechanism for creating a visual representation of text and are used to display a visual summary of the most prominent words used on a web page, a news forum or a social media web site. A word-cloud is comprised of a set of tags, with each tag representing a single word. The prominence of a word is typically estimated from its occurrence frequency in a text and is visualised using a large font size or different font colour.

You are required to develop a Java Word-Cloud API that can parse a file or a URL to generate a PNG (portable network graphics) file with a word-cloud displaying the most prominent words in decreasing font size, style and colour. Note that your API should ignore frequently occurring words and HTML tags, in the case of a word-cloud generated from a URL. A list of common words is provided in the file ***stopwords.txt***.

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 and to JavaDoc your code. Please pay particular attention to how your application must be packaged and submitted. Marks will be deducted if you deviate (even slightly…) from the requirements. Finally, as 4th year software students, you should appreciate that, if your code does not compile you cannot pass the assignment…

Deployment and Submission

- ***The project must be submitted by midnight on Sunday 10th January 2016*** using both Moodle and GitHub.

GitHub:

Submit the HTTPS clone URL, e.g. https://github.com/myaccount/my-repo.git of the public repository for your project. All your source code should be available at the GitHub URL. *You should try to use GitHub while developing your software and not just push changes at the end. It will make you more employable next May…*

Moodle

- 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 "Word Cloud 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
wordcloud.jar	A Java Archive containing your API and a runner class with a main() method. The application should be run as follows: **java –cp ./wordie.jar ie.gmit.sw.Runner** You can create the JAR file using Ant or with the following command from inside the "bin" folder of the Eclipse project: **jar –cf wordcloud.jar ***
src	A directory that contains the packaged source code for your application.
README.txt	Contains a description of the application, extra functionality added and the steps required to run the application. This file should BRIEFLY provide the instructions required to execute the project
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.
build.xml	An Ant build script that compiles the code in the ***src*** directory and builds a JAR archive called ***wordcloud.jar.*** Use the Ant build script available on Moodle. MAKE SURE THAT THE ANT SCRIPT USES THE CURRENT DIRECTORY (./). **You will lose marks if you use absolute paths. A sample Ant script is available on Moodle.**

Marking Scheme

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

Marks	Category
(40%)	Robustness
(10%)	Cohesion
(10%)	Coupling
(10%)	JavaDocs and UML Diagram
(10%)	Packaging & Distribution (GitHub and Moodle)
(10%)	Unit Tests
(10%)	Documented (and relevant) extras.

You should treat this assignment as a project specification. Any deviation from the requirements will result in a loss of marks. Each of the categories above will be scored using the following criteria:

- 0–30% Not delivering on basic expectation
- 40-50% Meeting basic expectation
- 60–70% Tending to exceed expectation
- 80-90% Exceeding expectations
- 90-100% Exemplary

Creating a PNG Image from Text

The Java 2D API provides a rich set of classes for manipulating images. The capabilities of the *BufferedImage*, *Graphics* and *ImageIO* classes are amply sufficient for this project. We can create a *BufferedImage* of a given size and use its associated *Graphics* object to draw text onto an image. The image can then be converted to a PNG and saved using the *ImageIO* class. For the more intrepid and discerning programmers amongst you, the *Graphics* object can be cast to a *Graphics2D* type, provide an even richer graphics environment that includes lighting, shadowing, ghosting and other effects.

```
import java.awt.*;
import java.awt.image.*;
import javax.imageio.*;
import java.io.*;

public class ReallySimpleWordCloud{
  public static void main(String args[]) throws Exception{
    Font font = new Font(Font.SANS_SERIF, Font.BOLD, 62);
    BufferedImage image = new BufferedImage(600, 300, BufferedImage.TYPE_4BYTE_ABGR);
    Graphics graphics = image.getGraphics();
    graphics.setColor(Color.red);
    graphics.setFont(font);
    graphics.drawString("Object-Oriented", 0, 100);

    font = new Font(Font.SANS_SERIF, Font.ITALIC, 42);
    graphics.setFont(font);
    graphics.setColor(Color.yellow);
    graphics.drawString("Software Development", 10, 150);

    font = new Font(Font.MONOSPACED, Font.PLAIN, 22);
    graphics.setFont(font);
    graphics.setColor(Color.blue);
    graphics.drawString("2012 Assignment", 40, 180);

    graphics.dispose();
    ImageIO.write(image, "png", new File("image.png"));
  }
}
```