CS 635 Advanced Object-Oriented Design and Programming
Fall Semester, 2022
Assignment 4

San Diego State University -- This page last updated on 12/2/22

Flyweight

Due Dec 13

In this assignment we will investigate the flyweight pattern. We will look at a flyweight for characters. The example in the text shows an example of an object-oriented word processor that uses objects to represent characters. One reason they represent characters as objects is that they use the Composite pattern to represent the contents of a document. Some of the contents, images for example, must be objects. So in a language like Java all the elements of the document need to be objects. In a document we have letters from an alphabet and font information. In theory each character could have different font (font name, point size and style). In practice the font does not change very often. So we will use the Flyweight to save space. You will create a Character class that will store only the unicode code point of the character. You need a Flyweight factory that given a unicode code point (a char in Java) returns the Flyweight character object for the character. You need to have a single point of access to the same Flyweight factory from anywhere in your program. Since documents tend to use the same fonts repeatedly we will also use a Flyweight factory for fonts. In this factory the input will be a triple: the font name (Times, Courier, etc), point size (12, 13, etc) and style (bold, italic, underline, etc). This factory also needs a single point of access.

For the Character Flyweight to work we need to a way to story the extrinsic state of the character objects, that is the font information. For this we will use a RunArray. A RunArray keeps track of runs in a sequence. For example if we have a document that starts with 250 characters in font A, then has 10 characters in font B and finally 320 characters in font A. The RunArray needs to store the runs: 250, 10 and 320. It also has to store the font that is associated with each run. Given any index (0 to 579 or 1 to 580) the run array will return the font used by the character in that location of the document. So give index 12 the RunArray will return Font A, given the index 255 will return Font B. When adding runs to the RunArray one needs to indicate the index the run starts at, the length of the run and the value at the run array. So for the current example we might have:

```
RunArray test = new RunArray();
test.addRun(0, 250, fontA);
test.addRun(250, 10, fontB);
test.appendRun(320, fontA);
```

If the run is appended the run array can determine the start index of the run.

The goal of the Flyweight is to save space. So the question is how much space does this save for sample documents. To answer this question on have to be able to compute the space of objects. For example on a 64bit machine an empty object in Java takes up 16 bytes, a font object takes up 72 bytes. A tool to measure the size of Java objects can be found at: http://java.dzone.com/articles/java-getting-size-object. Later I will give you some data from sample documents. You will use the data to compute how much space using flyweights saves.

Grading

Item	Percent of Grade
Working Code	10%
Unit Tests	10%
Proper implementation of Patterns	60%
Quality of Code	10%
Computing Space Savings	10%

Sample Text

CS 635 Advanced Object-Oriented Design & Programming
Fall Semester, 2018
Doc 17 Mediator, Flyweight, Facade, Demeter, Active Object
Nov 19, 2019

Sample Text Information

The sample text contains a total of 356 characters. There are 54 different characters in the text. There are two font runs. The first run contains 144 characters. The second run contains 212 characters.

The following chart indicates the frequency of the characters in the text. For example there is one character (4) that occurred four times, there are two characters (n, t) that occurred 18 times each in the text.

Frequency of Occurrence	Number of Characters with Given Frequency
41	1
34	1
22	1
17	1
16	1
15	1
14	1
10	3
9	3
8	2
7	3
6	4
5	4
4	5
3	4
2	6
1	13

What to do with the Sample Text

The point of the Flyweight pattern is to save space. The question is how much space we will save using the pattern. We have two ways to store the above text. The first way is to store 356 character objects with all their state (intrinsic and extrinsic). So we could store the 356 characters in an array. The other way is to use the flyweights. But the flyweight pattern uses additional data structures which require space. We have the flyweight factories and the run array, and we still need an array of size 356 spots to point to flyweight objects. So the goal is to compute the space required to store the document the normal way (array of 356 non-flyweight objects) versus using the flyweights.