TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS — 2011

END OF YEAR

SWEN425

DESIGN PATTERNS

Time Allowed: 3 Hours

Instructions:

- This examination will be marked out of **180** marks.
- Read each question carefully before attempting it.
- Answer all six questions. Each question has the same value, and should take approximately 30 minutes to answer.
- You may answer the questions in any order. Make sure you clearly identify the question you are answering.
- Many of the questions require you to discuss an issue, or to express and justify an opinion. For such questions, the assessment will take into account the *evidence* you present and any *insight* you demonstrate.
- Some of the questions ask for examples from object-oriented languages or of design patterns. Your answers need only refer to object-oriented languages or patterns discussed in the course, but you may refer to other programming languages and patterns if you wish.
- You may write code samples and/or draw diagrams to illustrate your answers to any of the questions.
- This exam is open book. Non-electronic reference books and handwritten notes are permitted.

Question 1. Creational Patterns [30 marks]

(a) [5 marks] The Java `Collection` interface includes a Factory Method called `iterator` that returns an Iterator. Explain why the `iterator` method is a Factory Method, and explain why a Factory Method is necessary in this design.

(b) [5 marks] Explain how to change `Collection` to use the Prototype pattern to create iterators. Do you think this is a better design than using a Factory Method?

(c) [5 marks] The Java `List` class implements `Collection`, and includes the following four methods:

```
              // Returns an Iterator over this list
     Iterator iterator()

              // Returns a ListIterator over this list
ListIterator listIterator()

              // Returns a ListIterator over this list starting from "from"
ListIterator listIterator(int from)

              // Returns a view of this list between "from" and "to"
        List subList(int from, int to)
```

Does this make `List` an Abstract Factory? Explain why or why not.

(d) [5 marks] The Singleton pattern is often "considered harmful". Explain the problems with Singletons. Are there ways of mitigating those problems, or should we abandon the Singleton pattern altogether?

(e) [5 marks] Java lets you write array literals - as in:

```
int[] arr = new int[] {1,2,3};
```

but not literals for Collections — in C♯ we could write something like this:

```
List<Integer> arr = new List<Integer>{1,2,3};
```

but we can't do that in Java. Explain how you could use the Builder pattern to solve this problem. Do you think a Builder is a good solution here?

(f) [5 marks] Should the Flyweight pattern be considered a creational pattern? Why or why not?

Question 2. Structural Patterns [30 marks]

(a) [15 marks] The class diagram below shows the design of a Font subsystem, made up of the following classes:

Typeface: a typeface such as "Helvetica" or "Times"

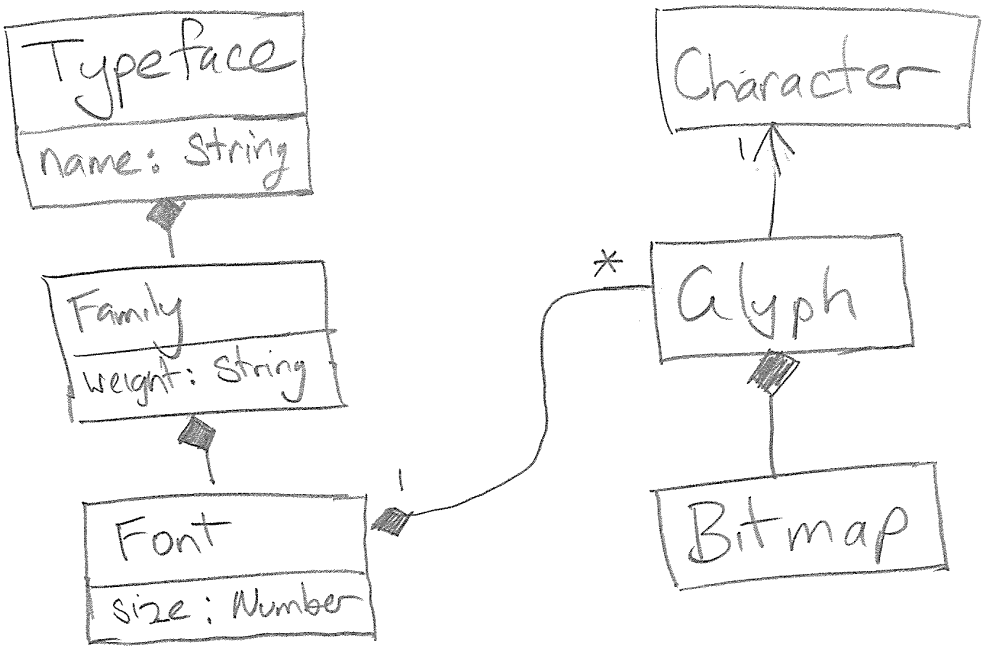
Family: a font family such as "Helvetica Bold" or "Helvetica Italic"

Font: a particular font such as "Helvetica Bold 12pt"

Glyph: a character shape from a font, such as the "A" from "Helvetica Bold 12pt"

Character: a 32-bit Unicode Character, like character number 65 ("A") or character number 9,731 ("Snowman")

Bitmap: a 32-bit deep bitmap of a Glyph rendered at screen resolution.



Describe how you could use the Flyweight and Proxy patterns to manage the memory use of the font subsystem.

(b) [15 marks] *"A Bridge is just an Adaptor plus a Strategy?"* Thomas J. "Tad" Peckish

Sketch a UML diagram of the Bridge pattern. Using your diagram, make clear how a Bridge could (potentially) be a combination of the Adaptor and Strategy patterns. List all the Participants of the Bridge pattern, showing how each Bridge Participant is a Participant in one or both of the Adaptor and Bridge patterns. What are the advantages and disadvantages of thinking about the Bridge pattern as a combination of these two other patterns?

Question 3. Behavioural Patterns [30 marks]

(a) [10 marks] Sketch the code for an Internal Iterator and an External Iterator over a simple collection. Identify the distinguishing benefits and liabilities of each type of Iterator, and give an example of the appropriate use of each. When — in general — should you use an External Iterator, and when should you use an Internal Iterator?

(b) [10 marks] Both the State and Memento patterns make part of the internal state of a Context or Originator object into a separate new object (the State or Memento object). Discuss the general advantages and disadvantages of this kind of refactoring. What are the other commonalities and differences between these two patterns?

(c) [10 marks] The Observer and Mediator patterns are often used together.

(i) [5 marks] Explain how you can use a Mediator to help implement an Observer.

- Outline how the Participants of the two patterns are related
- Identify a problem that using Observer creates
- Explain how incorporating a Mediator into the Observer solves the problem.

(ii) [5 marks] Explain how you can use an Observer to help implement a Mediator.

- Outline how the Participants of the two patterns are related
- Identify a problem that using Mediator creates
- Explain how incorporating an Observer into the Mediator solves the problem.

Question 4. Composition vs Inheritance [30 marks]

(a) [10 marks] Compare the Chain of Responsibility and Template Method patterns. You should give an example of the use of each pattern, describe how each pattern lets you extend the code of an object-oriented framework, and consider whether (or when) Chain of Responsibility can be replaced by Template Method, and vice versa.

(b) [10 marks] The Command and Strategy patterns let you parameterise clients with different requests or different algorithms. Compare and contrast how these patterns use composition and inheritance. Discuss how composition and inheritance give rise to the benefits and liabilities of the patterns.

(c) [10 marks] The Facade pattern is described using composition. Can you implement a Facade using inheritance instead? If you can, explain the strengths and weaknesses of this design, compared to the traditional facade. If you cannot, explain why not.

Question 5. Patterns in System Design [30 marks]

(a) [10 marks] Describe how you could use the Composite pattern to implement a simple programming language used for programming moisture vaporators. A short example vaporator program is shown below. You should sketch class diagrams, interaction diagrams, and sample code for key parts of your design.

```
PROGRAM VAPORATOR;
  AUTHOR C3P0.
  TRANSLATOR R2D2.

BEGIN
  SET TARGET_FLOW = 10
  SET FULL_TANK_LEVEL = 100
  WHILE (VAPORATION_RATE < TARGET_FLOW)
    BOOST_VAPORATION
  WHILE (TANK_LEVEL <= FULL_TANK_LEVEL)
    WAIT 1 MINUTE
```

(b) [10 marks] Describe how you could extend your design using the Interpreter and Visitor patterns to run those programs. Again, show class diagrams, interaction diagrams, and sample code for key parts of your design.

(c) [10 marks] Describe how you could further extend your design using the Observer pattern to support debugging running programs. Again, show class diagrams, interaction diagrams, and sample code for key parts of your design.

Question 6. Patterns vs Programming [30 marks]

(a) [20 marks]

The *Design Patterns* book was published in 1995. The code in the book does not use C++ templates (or Java or C♯'s generics) and makes little use of lambda expressions (C♯, Scala and Python's lambdas, Smalltalk or Ruby blocks, otherwise known as anonymous functions).

Choose at least two patterns to illustrate how langauge features interact with designs. Explain, giving examples, how the widespread adoption of generics and lambdas affects the patterns' implementations, and their Consequences.

(b) [10 marks]

> *Them **Design Patterns** ain't really about Design.*
> *We should'a called them **Programmin' Patterns.***
>
> Attributed to Thomas J. "Tad" Peckish.

Discuss.

****