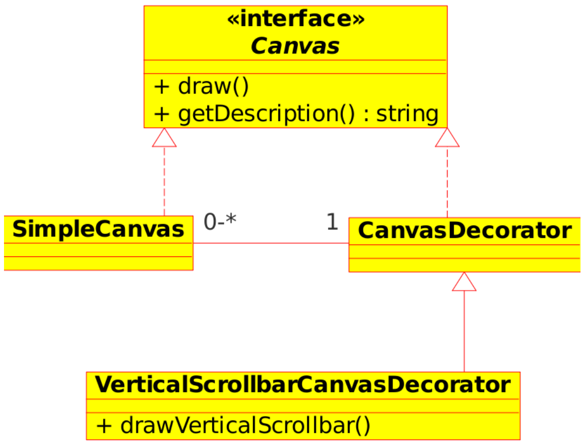
Module Title: Informatic 2C - Software Engineering
Exam Diet (Dec/April/Aug): December 2012
Brief notes on answers:

1. (a) Bookwork. An abstract class can provide implementations for some operations; an interface cannot.

   Marking guide:
   1 for each definition, 1 for the difference

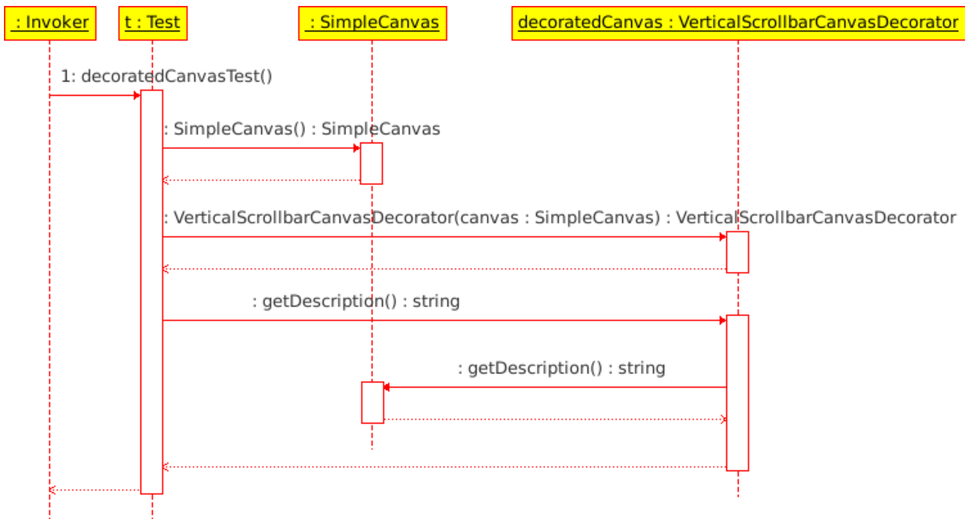
   (b) Application of knowledge. See:


«interface»
Canvas
+ draw()
+ getDescription() : string
SimpleCanvas
0-*
1
CanvasDecorator
VerticalScrollbarCanvasDecorator
+ drawVerticalScrollbar()

   (note this question and next: Canvas-¿Graph, VerticalScrollbar-¿XAxisScale, getDescription-¿getSlope, string-¿float)

   Marking guide:
   1 each for: the basic classes; the abstract class; the interface; conformance to an interface; association; multiplicity; specialisation; operations. (Where there are several instances of a concept, e.g. for operations, all must be correctly shown to get that mark.). Abstract class should be annotated as such or in italics.

   (c) Application of knowledge. See:


: Invoker
t : Test
: SimpleCanvas
decoratedCanvas : VerticalScrollbarCanvasDecorator
1: decoratedCanvasTest()
: SimpleCanvas() : SimpleCanvas
: VerticalScrollbarCanvasDecorator(canvas : SimpleCanvas) : VerticalScrollbarCanvasDecorator
: getDescription() : string
: getDescription() : string

   Marking guide:

   1 each for:

(i). the box and lifeline of t:Test;
(ii). appropriate new message to a SimpleGraph;
(iii). the box and lifeline of the new SimpleGraph;
(iv). appropriate new message (with argument) to decoratedGraph:XAxisScaleDecorator;
(v). box and lifeline of decoratedGraph:XAxisScaleDecorator;
(vi). correct placement of the new objects further down the page than t:Test;
(vii). getDescription message from t:Test to decoratedGraph;
(viii). getDescription message from decoratedGraph to the unnamed SimpleGraph

(d) A more problem-solving kind of application of knowledge.

GraphWithXAxisScale would specialise Graph, overriding its getDescription operation appropriately (1 mark). Our test method would directly create a GraphWithXAxisScale (1 mark) and invoke its getDescription (1 mark).

(e) Problem-solving (unless they actually did as I recommended and read around the pattern catalogies, in which case they could recognise this as the classic example of the Decorator pattern, described in Wikipedia, GoF etc.). To be marked generously: getting the idea and making points approximating the following is OK.

A graph might require x-axis scaling, y-axis scaling, neither or both (1 mark). The naive solution will require a number of classes that increases exponentially with the number of features that may or may not be present (1 mark). Even though in this case the amount of overriding code in each class will be small, this will be error-prone and hard to maintain (1 mark).

2. (a) (i). PUBLIC – Software engineers shall act consistently with the public interest.
   (ii). CLIENT AND EMPLOYER – Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
   (iii). PRODUCT – Software engineers shall ensure that their products and related modications meet the highest professional standards possible.
   (iv). JUDGMENT – Software engineers shall maintain integrity and independence in their professional judgment.

   All of the above are relevant. In favour: loss of financial data is serious for the individual, more than just a commercial decision; product/company will lose reputation for integrity if anyone loses financial data; commercial and emotional factors should now cloud professional judgement. Mitigating: the bug is obscure so the risk is actually very small.

   (b) It tells you that it is completely ethical to use the information. This information was made public by your competitor. There is no public interest argument, you are not undermining professional standards. It is a competitive market, information is valuable, companies should guard it.

   (c) CMMI is suitable. ISO9000 is not suitable because all it does is specify what sort of documents and procedures a company should follow in its quality control. It doesn't specify any particular level of product quality, or any method for improving product quality.

   (d)
   - Establish and maintain control over all of your inputs.
   - Lock down your environment.
   - Assume that external components can be subverted, and your code can be read by anyone.
   - Use industry-accepted security features instead of inventing your own.

   (e) Problem-solving.

   The component should make sure that any special characters that will be interpreted by the browser are escaped or HTML-encoded. If this is not done, then carefully constructed input could include commands that the browser will execute, such as javascript.

   (f) ROCOF - Rate Of ocurrence Of Failure. The number of times per minute/hour/day the system is unavailable. Mean uptime also acceptable.

(g) Find out what the licensing terms of the library are. If it is GPL, it can't be used. If it is LGPL, it can be used. If it is some other open-source license, then you need to check if it can be included in proprietary software.

(h) Verification - are we building the software right? Validation - are we building the right software? Junit testing helps with verification in that it can verify that the software does what was intended if the test suite is comprehensive. It helps with validation if the test suite was designed in collaboration with the customer.