Advances in Programming Languages

Lecture 20: Exam Review

Ian Stark

School of Informatics
The University of Edinburgh

Friday 2 December 2016
Semester 1 Week 11

Outline

Outline

Approaches to Assessment

Formative Assessment — "Assessment for Learning"

Aims to help learning by providing feedback to both student and teacher about what areas need more work. Happens while learning a topic; ideally provides room for students to take risks, explore, attempt challenges, and even fail, without risking their course grade.

Summative Assessment — "Assessment of Learning"

Aims to provide information on what a student now knows or can do; measuring performance at the end of a topic. This is the test which determines course marks and grades.

These contrasting aims mean that formative and summative assessment often involve different kinds of task, and certainly different working environments.

Approaches to Assessment

Criterion-Referenced Assessment

Evaluating student performance against pre-set outcomes and targets: can you do this, or how many of these things can you do?

Norm-Referenced Assessment

Evaluating student performance against other students in the class, year, or some wider group: which students are better than average, and which worse?

This may involve "grading to the curve".

Approaches to Assessment

Outcome

Performance on a task, demonstration of a skill, display of reasoning.

Process

Continuous evaluation of coursework, participation, attendance, grade-point averages over an extended period.

Outline

Examinable Material

- Material in lectures (green-banded slides only; not guest lectures)
- Exercises set as homework (only that explicitly set as homework; not all those miscellaneous references)
- That's it.

Exam Format

The format of the exam is the same from year to year, with a fixed rubric.

> Even so, you should read the rubric carefully on the day.

The paper has three questions, and you should choose and answer exactly two.

> I strongly recommend that you read all three questions before choosing which two to answer.

This is a "closed-book" exam: you may not take in any textbooks, printouts, notes, or other supporting material. Calculators are not permitted, and there will not be any questions that require them.

Grades

University of Edinburgh — Extended Common Marking Scheme

Mark (%)	Grade	Description	Honours Class	MSc Level
70-100	A	Excellent	I	Distinction
60-69	B	Very Good	II.1	Merit
50-59	C	Good	II.2	Pass
40-49	D	Undergraduate pass	III	Diploma, no MSc
30-39	E	Marginal Fail	Fail	Fail
20-29	F	Clear Fail		
10-19	G	Bad Fail		
0-9	H	Very Bad Fail		

Exam Technique

- Read the question.
- Answer the question.
- Make sure your answer exactly the details asked for in the question.
- Note that marking in APL is *positive* — marks are added for good things done, not taken away for things omitted.
- Look at the mark counts for an indication of how much or how little is required.
- Look at the mark counts to check how you are using time.
- READ THE QUESTION.

Exam Timing

This course has an exam during the April/May diet, at the end of the academic year, in common with other undergraduate honours courses in Informatics.

For such courses there is a period of between four weeks and four months between the end of lectures and the exam itself. This has the following impact:

- You need to recall material from the course a significant period after originally studying it, not just immediately after lectures end.

  *Being able to recall and apply knowledge well after the course has completed is an important outcome; it also engages the **spacing effect** where revisiting material helps learning.*

- You need to demonstrate ability across several different areas at the same time, with different exams covering material from different semesters.

  None of these courses exist in isolation, and working ably across different areas, combining knowledge, is a key skill for the effective application of Informatics.

What Kinds of Things are Assessed?

Subject area content obviously varies dramatically between different courses and their examinations, but some factors remain steady and most exams try to assess a range of skills.

Here is one listing of different elements that questions aim to stimulate and assess:

• Knowledge	Do you know the thing?
• Understanding	Do you know you know the thing?
• Ability to explain	Can you tell me the thing?
• Application of knowledge	Can you use the thing?
• Judgement	Can you tell which thing to use when?

In most cases questions and parts of questions will call on more than one of these; and there are also many other ways to classify learning and assessment goals.

Outline

2014/15 Question 1

```
public class FullName {                    // Class to represent a person's full name,
                                           // made up of their first name and last name.
    private String first = "", last = "";

    public static void copy(FullName p, FullName q) {     // Copy the contents of one
        q.first = p.first;                                // fullname into another
        q.last = p.last;
    }

    public static void safe_copy(FullName p, FullName q) {
        synchronized(p){              // Claim first fullname p
            synchronized(q){          // Claim second fullname q
                copy(p,q);            // Copy across contents
            }
        }
    }
...
```

2014/15 Question 1

(a) Describe what it means for methods in Java to be *thread safe*.

(b) The copy method is not thread safe. Explain why, showing fragments of code and their execution to demonstrate how this can be a problem in practice.

(c) The method safe_copy is a wrapper around copy that is intended to be thread safe. However, it is still problematic for use in concurrent code, as it may cause *deadlock*.

  (i) Describe what it means for threaded concurrent code to *deadlock*.

  (ii) Explain why safe_copy may cause deadlock, and give code fragments demonstrating how this can be a problem in practice.

2014/15 Question 1

(a) Methods are *thread safe* if they can be invoked in different threads at the same time without returning incorrect results or leading to inconsistent data.

(b) The method copy is not thread safe because if called simultaneously from two different threads the two executions may interfere with each other to give the wrong final results.

For example, suppose that we have three FullName objects

a = {"Alice", "Apple"} b = {"Bob", "Bean"} c = {"Carol", "Code"}

and make two calls to copy from concurrently-executing threads running on a single processor. The next slide shows an interleaved execution that gives the following result.

a = {"Alice","Apple"} b = {"Alice","Apple"} c = {"Bob","Apple"}

Here the full name a has been copied to b, but c has been corrupted and contains a full name not previously present in any of the objects.

2014/15 Question 1 : Interleaved execution of two copy calls

a	b	c	copy(a,b)	copy(b,c)
{"Alice","Apple"}	{"Bob","Bean"}	{"Carol","Code"}		
				c. first =b.first;
{"Alice","Apple"}	{"Bob","Bean"}	{"Bob","Code"}		
			b. first =a.first;	
{"Alice","Apple"}	{"Alice","Bean"}	{"Bob","Code"}		
			b.last=a.last;	
{"Alice","Apple"}	{"Alice","Apple"}	{"Bob","Code"}		
				c.last=b.last;
{"Alice","Apple"}	{"Alice","Apple"}	{"Bob","Apple"}		

2014/15 Question 1

(c)(i) A *deadlock* of concurrent code occurs when two or more threads are all unable to proceed because they are all waiting for one of the others to complete. In particular, this may occur when each holds a resource required by another; and this resource is very often a lock of some kind.

(c)(ii) The safe_copy method can cause deadlock because it acquires two locks in succession, and if called from two threads concurrently then each may hold a lock needed by the other.

For example, given the FullName objects used previously, suppose two threads make concurrent calls safe_copy(a,b) and safe_copy(b,a). The next slide shows one possible interleaving of their execution. At this end of this, each thread holds one of the two locks it requires, but cannot acquire the other. No values are copied at all, and the two threads are in deadlock.

2014/15 Question 1 : Interleaved execution of two safe_copy calls

a	b	Thread 1	Thread 2
{"Alice","Apple"}	{"Bob","Bean"}	safe_copy(a,b)	safe_copy(b,a)
		synchronized(a)	
Lock on 'a' now held by thread 1			
			synchronized(b)
	Lock on 'b' now held by thread 2		
		synchronized(b) blocked, waiting on thread 2	
			synchronized(a) blocked, waiting on thread 1
		DEADLOCK	

2014/15 Question 2(a)

(a) The following are three variations on the idea of *polymorphism* in programming languages.

(i) Subtype polymorphism.

(ii) Parametric polymorphism.

(iii) Ad-hoc polymorphism.

For each of these give a brief explanation of what it is, and give an example.

Note: Each example can be in any programming language — Haskell, Java, Scala, or whatever you think appropriate — but you must say which language it is. You can use different languages for each example if you think that will help your explanations.

2014/15 Question 2(a)

(i) If type S is a subtype of T then *subtype polymorphism* means that any method acting on a T can operate equally well on an S.

For example, in Java String is a subtype of Object: every String has all the properties required of an Object, and any method acting on an Object can also operate on a String.

(ii) With *parametric polymorphism* a single piece of code can again be run on values of different types, but carries out the same action for each.

The following Rust function exhibits parametric polymorphism: applied to any pair of two elements, it always exchanges them.

```
fn swap<T,U>(p:(T,U)) -> (U,T) {
  match p { (x,y) => (y,x) }
}
```

2014/15 Question 2(a)

(iii) With *ad-hoc polymorphism*, an operation may be applied to values of many different types, but with a different action on each.

For example, the following Java was used in the lecture slides, showing ad-hoc polymorphism: the drawing operation will do different things for different shapes.

```
Shape[] shapeArray;

...

for (Shape s : shapeArray)    // For every shape in the array ...
{ s.draw(); }                 // ... invoke its "draw" method.

...
```

2014/15 Question 2(b)

(b) Suppose we have a dependently-typed lambda calculus which includes types $\mathbf{Int}$ of integers, $\mathbf{Num}$ of non-negative integers, and $\mathbf{Matrix}\ \mathbf{n}\ \mathbf{m}$ of integer matrices with $\mathbf{n}$ rows and $\mathbf{m}$ columns, for $\mathbf{n}, \mathbf{m} : \mathbf{Num}$. One possible operation in the language is to generate an identity matrix, with the following type.

$$\mathbf{identity} \quad : \quad \forall \mathbf{n} : \mathbf{Num}\ .\ \mathbf{Matrix}\ \mathbf{n}\ \mathbf{n}$$

(i) Give a suitable dependent type for the operation of matrix addition $\mathbf{add}$.

(ii) Give a suitable dependent type for matrix multiplication $\mathbf{mult}$.

(iii) Use some or all of $\mathbf{identity}$, $\mathbf{add}$, and $\mathbf{mult}$ to write out a term that computes twice the $\mathbf{5} \times \mathbf{5}$ identity matrix.

2014/15 Question 2(b)

The types and term are as follows.

$$\mathbf{add} \;:\; \forall \mathbf{n} : \mathbf{Num} \,.\, \forall \mathbf{m} : \mathbf{Num} \,.\, \mathbf{Matrix}\,\mathbf{n}\,\mathbf{m} \to \mathbf{Matrix}\,\mathbf{n}\,\mathbf{m} \to \mathbf{Matrix}\,\mathbf{n}\,\mathbf{m}$$

$$\mathbf{mult} \;:\; \forall \mathbf{n} : \mathbf{Num} \,.\, \forall \mathbf{m} : \mathbf{Num} \,.\, \forall \mathbf{p} : \mathbf{Num} \,.\, \mathbf{Matrix}\,\mathbf{n}\,\mathbf{m} \to \mathbf{Matrix}\,\mathbf{m}\,\mathbf{p} \to \mathbf{Matrix}\,\mathbf{n}\,\mathbf{p}$$

$$\mathbf{add}\;\mathbf{5}\;\mathbf{5}\;(\mathbf{identity}\;\mathbf{5})\;(\mathbf{identity}\;\mathbf{5}) \quad = \quad \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

Finally. . . !

Thank You

. . . for your attention, interest and participation in this course.

I wish you all the best in the exam, your degree, and your future enjoyment of programming languages.

Course Enhancement Questionnaires

Please complete the online feedback survey for APL. It's anonymous, and I read every submission.

Surveys through MyEd

http://www.inf.ed.ac.uk/teaching/take-surveys
Select **Advances in Programming Languages**

OR

Direct link

http://ed.evasys.co.uk/evasys/online.php?pswd=88L1N