Inf1OP Advanced Programming Exam

About this exam

You have two hours. There are 7 questions. You are **not** expected to be able to do all the questions in the time available. Do as many as you can, bearing in mind that you will get more credit for a complete answer to one question than for fragmentary answers to several. Marking will not be by percentage but on a scale Incomplete (because nobody fails who was invited to sit this!), Pass, Merit, Distinction. As a rough guide, you can expect to pass if you submit good, full answers to two questions. Questions might not be of equal difficulty, and definitely don't require the writing of equal amounts of code – that's partly why I want to mark on a very coarse scale.

You are not required to submit JUnit tests, but you are encouraged to do so: there is a bit of bonus credit for submitting tests that save me effort in the marking, in that they should be passed by any correct solution (whether or not they are passed by yours!). If you do so, use JUnit4, and use the standard naming convention: e.g. tests for class Question1 should be submitted in class Question1Test.

You must use Java 7.

Files

Download files you will need using command `getpapers`.

In this exam, files may be submitted with any names, but where the question does specify a file name, please use it. Later submissions override earlier submissions of files with the same name. If you find you have submitted a file, and later decide you do not want me to consider any file of that name, please make this clear either by submitting a file called README that tells me what to consider for each question, or by overriding your submission with a blank file.

If it might not be obvious to me which files correspond to which questions, please make that clear in a README file which you also submit.

There are no basic tests, but code that fails to compile will still get no credit. Be sure to submit everything you want me to look at! Use the examsubmit command as usual, e.g.

```
examsubmit Question1.java
```

Caution!

This paper is informal and unofficial: it has been written in a hurry, and has not been through the usual scrutiny process for official exams. It is likely that you will find bugs and inclarities in the questions. Sorry!

On the positive side, your submissions will be marked by a human. So when you encounter bugs and inclarities, feel free to interpret questions as you wish, and include explanatory comments in your code.

For the same reason, unlike in the official Inf1OP exams, *style is significant*: you might get more credit for a submission with well-structured code, appropriate choice of names and a consistent indentation style than for one that is harder for me to read.

1. *Language basics*

   This question is adapted from examples in the (recommended) book "Java Puzzlers" by Joshua Bloch and Neal Gafter

   (a) Consider the file Question1.java, which contains a method `isOdd` which is supposed to take an integer and return true if and only if the argument is odd. This method is buggy. Add a standard `main` method to the class which demonstrates the bug. Add a method `isTrulyOdd` which fixes the bug. The code of your method should be as similar to that of `isOdd` as you can manage.

   (b) Consider the following loop, in which `i` is, so far, undeclared:

   ```
   while (i == i+1) {}
   ```

   Still in class Question1.java, write a method `public static void q1b()` which contains that line, exactly as it is, preceded by a definition of `i`, *such that the while loop continues for ever.*

   (c) Same thing, for the loop

   ```
   while (i != i) {}
   ```

   Write your solution in a method `public static void q1c()` of Question1.java.

   Submit your answers in a file called Question1.java

2. *Language basics*

   Without using anything from the JDK other than the basic Java language itself (no import statements or identifier starting `java.`; in particular, do not use any collection class in your answer!), implement your own version of the standard ArrayList class. Call your version MyArrayList; unlike the real ArrayList, your class should not be declared to extend any other class or implement any interface. Take care that your MyArrayList can grow and shrink as required and does not leak memory. Performance is not crucial for credit, but makes a good thing to think about if you want extra challenge...

   You do not need to implement all the methods (and that would be boring): full marks will be awarded if you correctly implement: a constructor; add; get; indexOf; remove. (Where the standard ArrayList contains more than one version of any of these, you choose.)

   If you can make your implementation use generics like the real one does, then do – but almost all marks can also be obtained from an implementation that only works with one fixed type of list item.

3. *This question is still in Java 7 – no using Java 8!*

   Consider the following Haskell function (taken from Lecture 13 of Inf1FP):

   ```
   elem :: Eq a => a -> [a] -> Bool
   elem x ys = foldr (||) False (map (x ==) ys)
   ```

   (a) Implement a Java class `Question3` with one method `public static boolean elem (Object x, Object[] ys)` whose behaviour is like that of the Haskell code. Use the `equals` method of `Object` to replace Haskell's `==`. *Internally, your method must work as similarly to the Haskell code as you can manage. For example, point out which part of your code corresponds to `map` and which to `foldr`.*

   (b) Next, in the same class, implement a method `public static boolean elem2 (Object x, Object[] ys)` with the same behaviour, but this time implemented in a way which seems to you to be reasonable in Java.

(c) Next, still in the same class, implement the same behaviour in a third way, also reasonable in Java, this time in `public static boolean elem3 (Object x, Object[] ys)`.

(d) Aim to make your `elem2` and `elem3` each optimal in some dimension; for example, one might be the shortest way to write this behaviour in Java and one might be the fastest, for large arrays. Explain in comments, and if you can, justify any performance claims you make by adding a `main` method that demonstrates.

4. *String processing*

A student id has the familiar form: it begins with s, followed by seven digits. The s may be lower or upper case. Write a class called `Question4` containing a method called `findStudentIds`. Your method takes as input a String, which may be very large, and extracts from it all the student ids. Note that a student id must form a complete word: this7688263 is not a student id, neither is S9999999999999999999999999. Your method must return a String which gives the student ids it found, comma separated, in the same order they were found in the input string. E.g. given

"s9782647 and S9782647 and especially s0000001 but never s876."

it must return "s9782647,S9782647,s0000001"

Preserve the case of the 's', as shown, and note the commas.

5. *I/O*

One of your downloaded files is `io.txt`, a sample file containing astronomical data. Study this file and take it as a definition (by example) of the format your program should handle. Each line comprises a string which identifies an star, an integer (the star's Flamsteed designation), and a piece of newly measured scientific data which is a double. Write a program in `Question5.java` that takes a (perhaps relative) path name to a file on the command line. Your progam should expect this file to exist, be readable, and contain data in the format exemplified by `io.txt`. Your program must locates the file, parse its input, print a human-readable line for each line of data, and summarise the scientific data, as indicated in the following example output. This is what you should output when the sample file `io.txt` is specified on the command line:

```
Star Polaris has Flamsteed 1 and data 78.3422
Star BetaUMi has Flamsteed 7 and data 2.05E-11
Star GammaUMi has Flamsteed 13 and data 7340000.0
There were 3 records and the average data was 2446692.78 to 2dp.
```

Most of the credit can be obtained just by writing a program that behaves properly on the given sample file. There is some additional credit available for sensible choices about aspects of behaviour that this question has not specified, e.g., what should happen in the event that the input file does not exist, is not readable, does not contain data in the expected format, etc.

6. *Threads and concurrency. This question is adapted, by kind permission of Prof. K. V. S. Prasad, from an exercise used in the course Concurrent Programming of Chalmers University of Technology and Gothenburg University*

Consider the downloaded file Counter.java. Its programmer intended that each of its two threads should increment the same counter 100,000 times, giving the counter an eventual value of 200,000. However, it contains a concurrency-related bug.

Compile the program, and run it repeatedly. You will (probably!) find that it does not always print 200000; sometimes it prints a number less than that, because of the bug.

(a) Fix the bug; that is, make an appropriate *small* modification to the program so that it will always print 200000. Take care not to slow the program's execution more than you must. Put a comment in your code to explain why your fix works. Submit your fixed version as Counter.java.

(b) Next, you must demonstrate what is the least value the program can in principle print, for any interleaving of the threads. By inserting one or more delays, you will make a version of the program that typically prints this least value.

Go back to the original, unfixed Counter.java.

i. Make a copy of this as class Counter2 in Counter2.java.

ii. Uncomment the commented out lines. At this stage the program will not compile; it will give an error message something like "exception InterruptedException is never thrown in body of corresponding try statement".

iii. Now experiment with adding one or more lines like

```
if (id== ?? && i== ??) Thread.sleep(??)
```

replacing ?? with appropriate values, wherever you think it might be helpful to do so. (If you do this in the right place, it will cure the compilation problem mentioned above, because Thread.sleep can throw InterruptedException.) You may also, if you wish, replace one line of the program by two lines that between them have the same effect, so that you can insert a delay between the two lines.

iv. Make no other modification to the program.

v. Once you are confident that your code prints the smallest value it can print, subject to the rules above, add a comment near `// Print the counter` to state what value you expect to be printed, and submit your code as Counter2.java.

7. *Especially for people who haven't found what they wanted in this paper*

Write an excellent question for an advanced programming exam, which you give me permission to use or adapt in next year's exam if I repeat this experiment. Submit your question either as plain text as `Question7.txt` or as LaTeX as `Question7.tex`. Submit a correct answer to your question in appropriate file(s).

This question attracts credit like all the others, but only once, i.e. there's no extra credit for submitting more than one question and answer!