About the COMP213 Exam

The written examination for COMP213 consists of 5 questions, each worth 25 marks; you are asked to answer four questions (if you answer all five, your grade will come from your four answers with the highest marks). Questions might ask you to: write a Maude specification of an abstract data type described in words; implement a Maude specification in Java; implement generic classes, interfaces and methods in Java and describe how generics are implemented; describe specific Java features such as Exceptions, the method-call stack, concurrency and synchronization, scope modifiers, static and final modifiers, and anonymous classes; and to discuss more theoretical concepts such as thread-safety, deadlock, class invariants, and abstract data types. In addition, you should also be able to demonstrate knowledge of some of the data structures covered in the module, such as arrays, vectors, (doubly) linked lists and trees, and an awareness of the advantages and disadvantages of these.

Revision Questions

1. The Abstract Data Type of Count Lists is a list of pairs; each pair in the list has an integer (which we call the ‘value’) and another integer (which we call the ‘count’); the idea is that the ‘count’ integer records how often the value occurs in the list. An example application might be to read in a sequence of integer values, perhaps recording the grades of individuals in a class, e.g.,

\[2, 7, 7, 5, 9, 3, 6, 7, 2, 5\]

and, for each grade value, record how often that grade occurs in the given data. For the example data above, we should have the following Count List (where pairs are grouped in brackets):

\[(2, 2), (3, 1), (5, 2), (6, 1), (7, 3), (9, 1)\]

Note that grades that don’t occur in the data, such as 4, are not listed.

As an Abstract Data Type, Count Lists have one constant, representing the empty list (no data); and the following operations:

- **add** a Pair (consisting of two integer values, a value and its count) to the list;
- **insert** a given integer value into the list (if it already occurs in the list, increment the count value; otherwise, add a new pair with the given value and a count of 1); and
- **getCount** — this takes an integer value as argument and returns the count of how often that value occurs in the list (0 if it doesn’t occur at all in the list).
Count Lists make use of a subsidiary Abstract Data Type of Pairs: this Abstract Data Type has three operations:

- a constructor that takes two integers and returns the Pair built from those values;
- `getValue`, returning the value component of the Pair; and
- `getCount`, returning the count.

(a) Specify the Abstract Data Type of Pairs in Maude. [6 marks]
(b) Specify Count Lists in Maude. [11 marks]
(c) Write a term in Maude that corresponds to the Count List recording all the data in (1) above. [3 marks]
(d) Sketch why the equations in your specification of Count Lists would allow you to conclude that the value 5 occurs twice in that list. [5 marks]

2. Consider the following Maude specification of Count Lists: (insert your solution to Question 1 here! — a real exam question would give the specification and would also have an informal description of Count Lists).

(a) Give a Java implementation of the sort Pair. [4 marks]
(b) Why would an implementation of Count Lists using linked lists be more efficient than one using java.util.Vector? [2 marks]
(c) Give a Java implementation of Count Lists using linked lists. Marks will be awarded for correctly implementing:
   i. the linked-list structure; [4 marks]
   ii. a constructor corresponding to the empty list; [1 mark]
   iii. the method `add`; [2 marks]
   iv. the method `insert`; [4 marks]
   v. the method `getCount`. [4 marks]
(d) Briefly say why your implementation of `getCount` behaves in the way specified by the equations in the Maude specification. [4 marks]