Sample Solution for Homework 3

Problem 1 AMP, p.66, Exercise 24: Sequential Consistency and Linearizability (12 Points)

- Fig. 3.13: This history is linearizable. A sequential execution of the history's events that is a linearization of the history is: r.write(1), r.read(1), r.write(2), r.read(2). Since linearizability implies sequential consistency, the history is also sequentially consistent.
- Fig. 3.14: This history is linearizable. A sequential execution of the history's events that is a linearization of the history is: r.write(2), r.write(1), r.read(1), r.read(1). Again, this implies that the history is also sequentially consistent.

Problem 2 AMP, p.67, Exercise 27: (7 Points)

Consider an object `q` of class `IQueue<Integer>` and an execution of two threads T_0 and T_1, where T_0 executes `q.enq(0)`, while T_1 executes first `q.enq(1)` and then `q.deq()`. Now consider the following interleaving of the three calls. First, T_0 executes `q.enq(0)` up to but excluding line 10. Then, T_1 executes first `q.enq(1)` and then `q.deq()`. After, the call to `q.deq()` returns, T_0 proceeds executing `q.enq(0)` until the call returns, too. Before the call to `deq` we have that `q.head == 0` and `q.items[0] == null`. Hence, this call will throw an `EmptyException`. There is no sequential execution of the queue that will produce this behavior, i.e., throwing `EmptyException` after a preceding call to `q.enq(1)`. Hence, `IQueue` is not linearizable.

Problem 3 AMP, p.67, Exercise 28: (6 Points)

Yes, the method `reader` may potentially divide by zero. The Boolean `v` is declared `volatile`. This means that any sequence of read and write operations on `v` will be sequentially consistent. However, the Java memory model does not guarantee sequential consistency of non-volatile variables, even if they are used together with volatile variables. In particular, the read of `x` on line 10 might return 0, even though this read requires `v == true`. That is, `v == true` does not imply that the assignment to `x` on line 5 has already taken effect.