**Exercise 1**

1. A Petri net C = <P,T,I,O> consists of:
   1. A finite set P of places
   2. A finite set T of transitions
   3. An input function, I
   4. An output function, O

Petri nets are directed graphs with two kinds of vertices — places and transitions — together with a marking — a function that indicates how many tokens are currently in every given place. Transitions are enabled if they have at least one token in every input place. Firing an enabled transition leads to a new marking by removing one token from every input and adding one to every output place.

1. Petri nets can be used to model concurrency problems. Transitions can represent competing processes, and places can represent resources, with tokens (markings) indicating the availability of a resource. But a process may also correspond to a subnet, with places representing the state of a process. Tokens can then represent control flow, or data flow, or synchronization conditions.
2. The reachability set R(C,m) of a net C is the set of all markings mʹ reachable from initial marking m.
3. Any bounded Petri net with a finite reachability set, can be modeled by a FSP.
4. A net C with initial marking m is safe if places always hold at most 1 token.  
   A marked net is (k-)bounded if places never hold more than k tokens.  
   A marked net is conservative if the number of tokens is constant.
5. Adding almost any feature to nets (such as zero-testing), will make them Turing-complete.

**Exercise 2**

1. P = {v,w,x,y,z}  
   T = {a,b,c,d}  
   I(a) = {v,w}, I(b) = {w,x}, I(c) = {y}, I(d) = {z}  
   O(a) = {y}, O(b) = {z}, O(c) = {v,w}, O(d) = {w,x}  
   m = {y,y,y,w,x,x}
2. P = {a,b,c,d}  
   T = {x,y}  
   I(x) = {a,b}, I(y) = {c,d}  
   O(x) = {c,d}, O(y) = {b}  
   m = {a,a,b}
3. It’s 2 bounded. Not Safe. Not Conservative. This petri net will be deadlocked after all tokens in “a” is consumed. Transitions are not live, because it will be deadlocked.

**Exercise 3**

Model can be found in Database.pr file. I used two different locks for read and write. Machines can read as much as they want, but only one of them can write at the same time. This is not a strong consistency like linearizability. As a result, this implementation can cause dirty reads. However, this will perform faster than linearizability.

**Exercise 4**

1. These two classes have constructors with fairness parameter.
2. A daemon thread is a thread that does not prevent the JVM from exiting when the program finishes but the thread is still running. Purpose of creating a daemon thread is to execute some parallel job that main threads are not dependent on it. Before starting a Thread, you should use setDaemon method to make that Thread run in Daemon mode.