Exercises on Multithreading

1. Give an example of a multi-threaded program and describe some of the tasks that are carried out by different threads in the program.
2. Explain what is meant by the term *race condition*.
3. Describe how new threads are created, started and stopped in Java.
4. There are two basic ways of creating threads in Java: one involves extending a superclass, the other involves implementing an interface. Briefly describe the two methods. Which is generally preferable and why?
5. The Thread class has *two* relationships with the Runnable interface. Draw a UML class diagram that shows the relationship between the Thread class and the Runnable interface.
6. Explain the meanings of the terms *interference* and *critical region* with regards to a multi-threaded program.
7. Explain how *synchronization* and *locks* can be used to avoid interference.
8. Why would you declare a method as being synchronized? What object does a synchronized method acquire a lock on?
9. Suppose I want to make sure that only the current thread accesses the object, `obj`, when I execute the following two lines:

   ```
   obj.methodA();
   obj2.methodB(obj);
   ```

   How can this be accomplished in Java?
10. Explain the difference between server-side and client-side synchronization. What are the advantages and disadvantages of each type of synchronization?
11. What static method is used to get the current thread in Java?
12. What static method is used to pause the current thread in Java? What kind of exception does this method throw?
13. In Java, how do you make the current thread wait until some other specified thread (call it *t*) has died?
14. Explain how a thread can be stopped in Java before it has finished executing. What happens when a thread is stopped by an instruction from some other thread?
15. Explain the function of the *interrupt status flag* in Java.
16. What method do we use in Java to determine if a thread, *t*, is still running?
17. What is a *guarded block*?
18. What does the term *busy waiting* mean with reference to a guarded block?
19. How can the wait and notifyAll methods be used to avoid busy waiting when creating guarded blocks in Java?