7. The `loadCommands` method calls the `notifyAll` method.
8. Both `Robot` threads are unblocked and attempt to reacquire the `RobotController` object lock.
9. Because only one thread can acquire the lock, assume the lock is obtained by the first `Robot` thread. The second `Robot` thread does not acquire the lock until the first `Robot` thread releases it.
10. The first `Robot` thread processes the commands for the robot, sets the `commands` variable to `null`, and releases the lock.
11. The second `Robot` thread acquires the lock and attempts to process the commands for the robot.
12. Because the `commands` variable is `null`, the code fails with a `NullPointerException`.

The problem with this code is that it failed to recheck the value of the `commands` variable before proceeding. Because all threads are awakened on a call to `notifyAll`, they all eventually reacquire the object lock in an undetermined order (see PRAXIS 53). When they reacquire the lock and begin execution, they start at the line of code immediately following the call to the `wait` method. When a thread is awakened, it must recheck the condition on which it was waiting. This is because it might not be the first thread to run and the condition could have changed.

In the previous example, the first thread that runs changes the value of the condition variable. The first thread sets the `commands` field to `null`. Because the code does not recheck the value of the `commands` variable, the second thread fails. Whenever code is waiting on a particular condition, it should do so inside of a loop or a spin lock. The correct implementation of the `run` method of the `Robot` class is as follows:

```
public void run() {
  byte[] cmds;
  while(true) {
    synchronized(controller) {
      while (commands == null) {                              //1
        try {
          controller.wait(); }
        catch(InterruptedException e){}  //Exception is ignored
      }                                  //purposefully.
      cmds = new byte[commands.length];
      for (int i=0; i<commands.length; i++)
        cmds[i] = commands[i];
```