Concurrency Bad Threads

1. Compile and run BadThreads.java:

```
public class BadThreads {

  static String message;

  private static class CorrectorThread

      extends Thread {

      public void run() {

          try {

              sleep(1000);

          } catch (InterruptedException e) {}

          // Key statement 1:

          message = "Mares do eat oats.";

      }

  }

public static void main(String args[])

      throws InterruptedException {


      (new CorrectorThread()).start();

      message = "Mares do not eat oats.";

      Thread.sleep(2000);

      // Key statement 2:

      System.out.println(message);

  }

}
```

2. The application should print out "Mares do eat oats."

- Is it guaranteed to always do this? No.
- If not, why not?

The result is not guaranteed to always work. The code has to be written so that one statement executes before the other so there is reliability in how the steps are performed.

3. Would it help to change the parameters of the two invocations of Sleep?

This is a double edged sword. Whether you do or do not there is no guarantee that the changes will not switch between which of the statements is returned. There is no guarantee that the actions in one thread will be visible to any other thread.

4. How would you guarantee that all changes to the message will be visible in the main thread?

You have to make the message visible by the main thread either by referencing the CorrectorThread and invoking a "join" on the instance of the thread. You can use an object to encapsulate the message with synchronized methods.