Questions

Bookmark this page

Factorial

1/1 point (graded)

Suppose `main` looks like this:

```
public static void main(String[] args) {
    new Thread(new Runnable() { // create a thread using an
        public void run() {     // anonymous Runnable
            computeFact(99);
        }
    }).start();
    computeFact(100);
}
```

Which of the following are possible interleavings?

- [✓] The call to computeFact(100) starts before the call to computeFact(99) starts
- [✓] The call to computeFact(99) starts before the call to computeFact(100) starts
- [✓] The call to computeFact(100) finishes before the call to computeFact(99) starts
- [✓] The call to computeFact(99) finishes before the call to computeFact(100) starts

✓

Explanation
All of these are possible.

Submit   Show Answer

Answers are displayed within the problem

PinballSimulator

3/3 points (graded)

Here's part of the pinball simulator example from the previous section:

```
public class PinballSimulator {

    private static PinballSimulator simulator = null;

    // ...

    public static PinballSimulator getInstance() {
/* 1 */      if (simulator == null) {
/* 2 */          simulator = new PinballSimulator();
             }
/* 3 */      return simulator;
    }
}
```

The code has a race condition that invalidates the invariant that only one simulator object is created.

Suppose two threads are running `getInstance()`. One thread is about to execute one of the numbered lines above; the other thread is about to execute the other. For each pair of possible line numbers, is it possible the invariant will be violated?

About to execute lines 1 and 3

[✓ Yes, it could be violated]

No, we're safe

✓

Explanation
The thread on line 3 has already assigned `simulator`, so the thread on line 1 will not enter the conditional. Right?
Unfortunately, that's not correct. As we saw in the last reading, Java doesn't guarantee that the assignment to simulator in one thread will be immediately visible in other threads; it might be cached temporarily. In fact, our reasoning is broken, and the invariant can still be violated.

About to execute lines 1 and 2

[✓ Yes, it could be violated]

No, we're safe

✓

Explanation
If the thread about to execute line 1 goes first, both threads are inside the conditional and will create new simulator objects.

About to execute lines 1 and 1

[✓ Yes, it could be violated]

No, we're safe

✓

Explanation
If both threads test the predicate before either thread assigns simulator, both will enter the conditional and create new simulator objects.

Submit

Show Answer

Answers are displayed within the problem

Confinement

1/1 point (graded)

In the following code, which variables are confined to a single thread?

```
public class C {
    public static void main(String[] args) {
        new Thread(new Runnable() {
            public void run() {
                threadA();
            }
        }).start();

        new Thread(new Runnable() {
            public void run() {
                threadB();
            }
        }).start();
    }

    private static String name = "Napoleon Dynamite";
    private static int cashLeft = 150;

    private static void threadA() {
        int amountA = 20;
        cashLeft = spend(amountA);
    }

    private static void threadB() {
        int amountB = 30;
        cashLeft = spend(amountB);
    }
    private static int spend(int amountToSpend) {
        return cashLeft - amountToSpend;
    }
}
```

- [✓] amountA
- [✓] amountB
- [✓] amountToSpend
- [ ] cashLeft
- [ ] name

✓

Explanation

amountA, amountB, and amountToSpend are all local variables, so they are automatically confined to the thread that is running that method call. cashLeft and name are static variables, so they are accessible to all three threads in this program (main, threadA and threadB), and not confined.

Submit

Show Answer

Answers are displayed within the problem

Previous Next