Questions

Bookmark this page

Message passing

1/1 point (graded)

Leif Noad just started a new job working for a stock trading company:

```
public interface Trade {
    public int numShares();
    public String stockName();
}

public class TradeWorker implements Runnable {
    private final Queue<Trade> tradesQueue;

    public TradeWorker(Queue<Trade> tradesQueue) {
        this.tradesQueue = tradesQueue;
    }

    public void run() {
        while (true) {
            Trade trade = tradesQueue.poll();
            TradeProcessor.handleTrade(trade.numShares(), trade.stockName());
        }
    }
}

public class TradeProcessor {
    public static void handleTrade(int numShares, String stockName) {
        /* ... process the trade ... takes a while ... */
    }
}
```

What are `TradeWorker`s?

- [ ] separate threads
- [x] objects that can be passed to the thread constructor
- [ ] producers in the producer/consumer pattern
- [x] consumers in the producer/consumer pattern

✔

Explanation

`TradeWorker` implements `Runnable`. An instance of `TradeWorker` is not a separate thread all by itself, but we can create a new thread that will execute its `run` method.
The worker is reading items off a shared queue, so it is a consumer in producer/consumer.

Submit | Show Answer

Answers are displayed within the problem

Mistakes were made

1/1 point (graded)

Suppose we have several `TradeWorker`s processing trades off the same shared queue.

Notice that we are *not using* `BlockingQueue`! Workers call `poll` to retrieve items from the queue.

Which of the following can happen?

- [x] trades are not processed by the `TradeProcessor` in the same order they were in on the queue
- [x] a single trade can be processed multiple times

- [x] we can crash with a `NullPointerException`

✔

Explanation

Not a good situation.

If order was important, having multiple concurrent consumers was not a good idea. Their calls to `handleTrade` can be interleaved arbitrarily, since they are on different threads.

Because we did not use a concurrency-safe queue, different workers can call `poll` and end up taking the *same trade* off the queue, processing it multiple times. And `poll` is *not blocking*, so if the queue is empty, `poll` returns `null`. Whoops! `NullPointerException`.

Submit

Show Answer

Answers are displayed within the problem

Previous Next