Questions

Bookmark this page

Rep invariant

1/1 point (graded)

```
/** An immutable squaring result message. */
public class SquareResult {
    private final int input;
    private final int output;

    /** Make a new result message.
     *  @param input input number
     *  @param output square of input */
    public SquareResult(int input, int output) {
        this.input = input;
        this.output = output;
    }

    @Override public String toString() {
        return input + "^2 = " + output;
    }

    private void checkRep() {
      assert /*REP_INVARIANT*/;
    }
}
```

Write the rep invariant of SquareResult, as an expression that could be used in the assert statement in checkRep(). Use the minimum number of characters you can without any method calls in your answer.

/*REP_INVARIANT*/

input*input==output ✔

Submit

Show Answer

✔ Correct (1/1 point)

Code review

1/3 points (graded)

```
/** Squares integers. */
public class Squarer {

    private final BlockingQueue<Integer> in;
    private final BlockingQueue<SquareResult> out;
    // Rep invariant: in, out != null

    /** Make a new squarer.
     *  @param requests queue to receive requests from
     *  @param replies queue to send replies to */
    public Squarer(BlockingQueue<Integer> requests,
                   BlockingQueue<SquareResult> replies) {
        this.in = requests;
        this.out = replies;
    }

    /** Start handling squaring requests. */
    public void start() {
        new Thread(new Runnable() {
            public void run() {
                while (true) {
                    // TODO: we may want a way to stop the thread
                    try {
                        // block until a request arrives
                        int x = in.take();
                        // compute the answer and send it back
                        int y = x * x;
                        out.put(new SquareResult(x, y));
                    } catch (InterruptedException ie) {
                        ie.printStackTrace();
                    }
                }
            }
        }).start();
    }
}
```

The code above undergoes a code review and produces the following comments. Evaluate the comments.

"The Squarer constructor shouldn't be putting references to the two queues directly into its rep; it should make defensive copies."

- ○ True
- ◉ False ✔

Explanation
Yes, Squarer does indeed share its (mutable) input and output queues with its client, and it's supposed to. Making a defensive copy would defeat the purpose, because the client needs to put messages into the input queue after the Squarer object has been created and its thread has started up. This is a limited form of shared mutable data that message-passing with blocking queues must use.

"Squarer.start() has an infinite loop in it, so the thread will never stop until the whole process is stopped."

- ◉ True ✔
- ○ False

Explanation
The while (true) loop will never exit. We'll see how to fix that in the next section.

"Squarer" can have only one client using it, because if multiple clients put requests in its input queue, their results will get mixed up in the result queue.

- ◉ True ✔
- ○ False

Explanation
Yes, if multiple clients are putting requests in a single Squarer's input queue, their results will get mixed up. A client won't be able to just take() the next result from the output queue, because it might belong to a different client.

Submit

Show Answer

Answers are displayed within the problem