Gebze Technical University

Computer Engineering

CSE 443 - 2019 Fall

HOMEWORK 3 REPORT

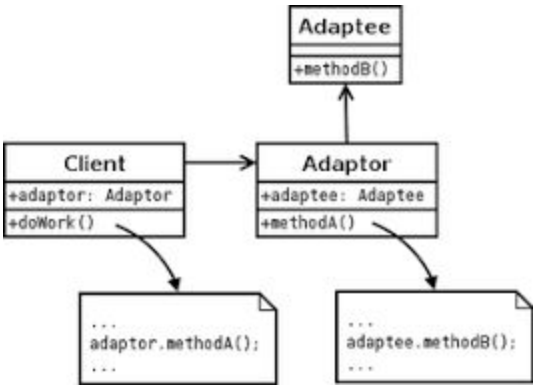
MUHAMMED OKUMUŞ

151044017

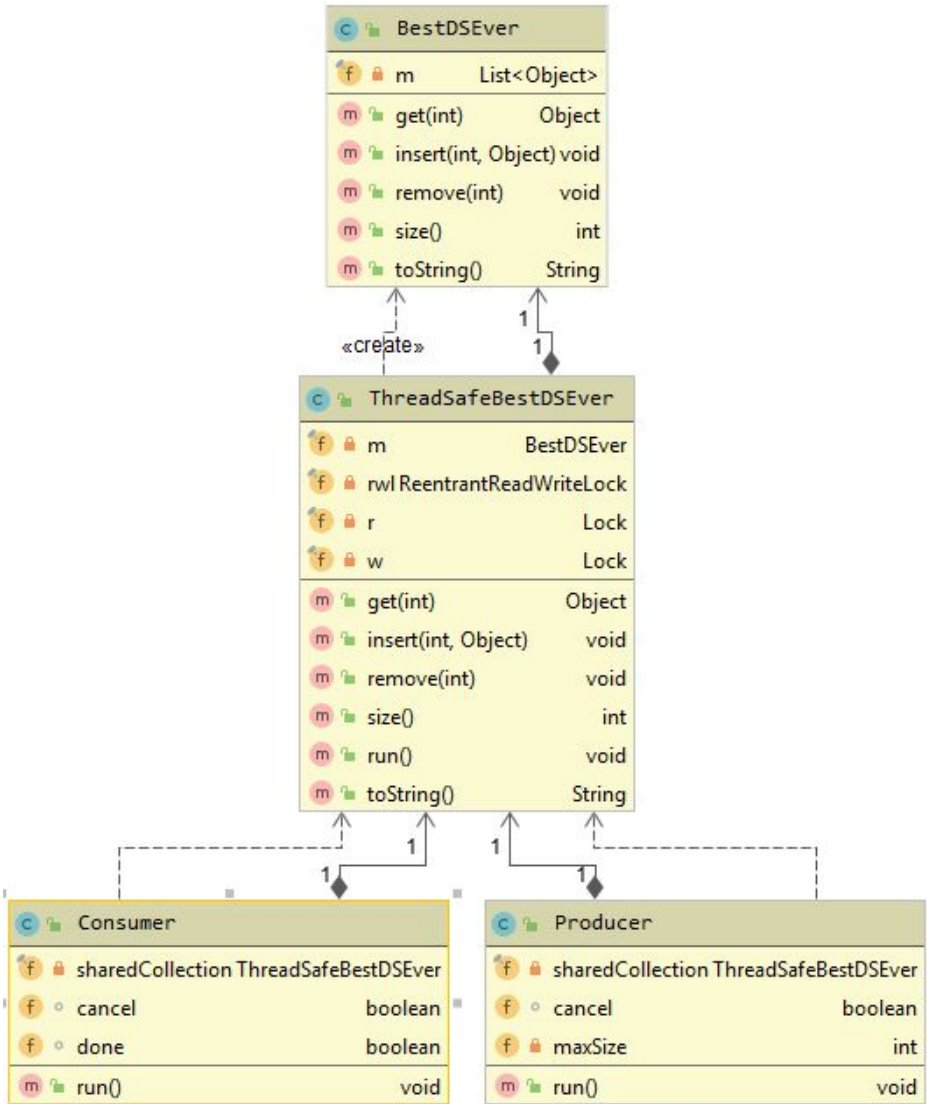
PART 1

We have a collection class here that needs to be thread safe without changing the class itself. This can be accomplished by applying the Adaptor Pattern. The adaptor class will wrap around the methods of the collection class. Since adaptors purpose in this case is to make the adaptee thread safe, methods will be wrapped with lock/unlock operations.

Adaptor Pattern Template →



Adaptor Pattern Implemented →



To test the multi thread safety of the design, Consumer and Producer class are implemented. They both implement Java's Runnable Interface and hold a reference to the adaptor class (ThreadSafeBestDSever).

Consumer class holds two boolean fields. One of them is 'done' field. This field is treated as a thread exit flag. This flag is risen just before the Consumer class is unable to access the shared collection. When this flag is risen, our GUI thread 'scroller' is scroll are text area and lowers the done flag. This flag could be in Producer class to, it effect the overall goal.

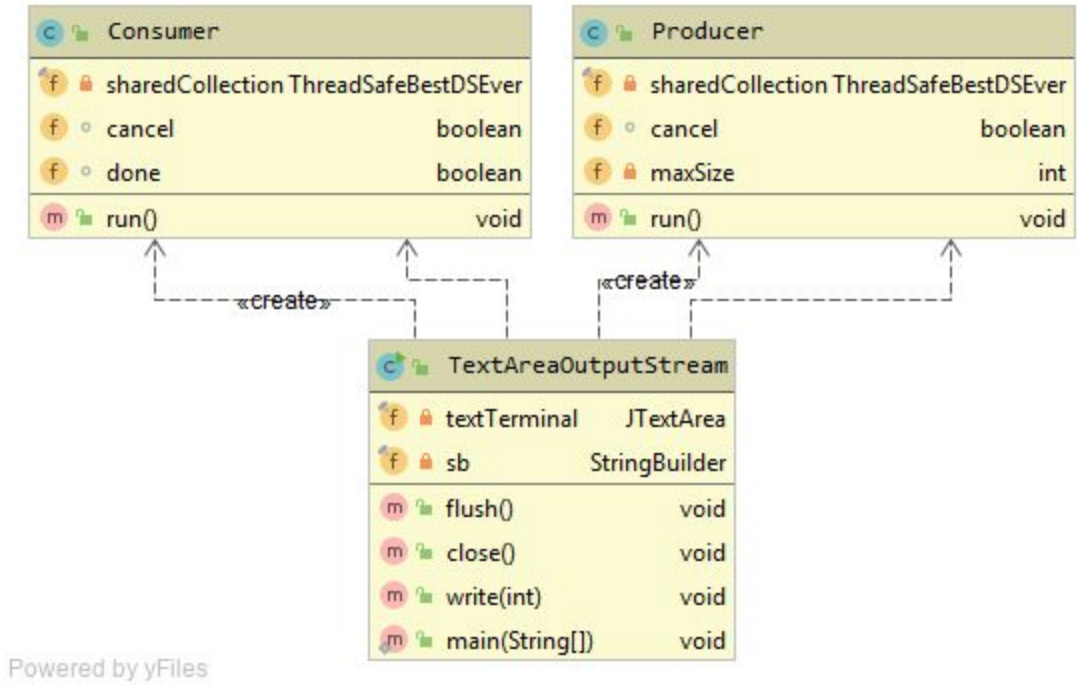
```
// BUTTON: START
// Start threads ******************************
buttonStart.addActionListener(e -> {
    producer.start();
    consumer.start();
    // Setup scroll pane for auto scrolling
    Thread scroller = new Thread(() -> {
        while(true) {
            synchronized (myCollection) {
                while(c.done){
                    textTerminal.setCaretPosition(textTerminal.getDocument().getLength());
                    c.done = false;
                }
                myCollection.notify();
            }
        }
    });
    scroller.start();
    buttonStart.setEnabled(false);
});
```

As seen in the code snippet, it's used as a control flag for the scroll thread. Note that scroller class also uses the same shared collection(myCollection) object -that producer and consumer uses- as its synchronization control object.

Other control flag is called 'cancel' which both Producer and Consumer class have. This flag is raised after the Stop button is pressed. It simply a one time turn off switch for the threads. Threads cannot be killed but their control variables can be changed.

```
@Override
public void run() {
    while(!cancel) {
        done = true;
        synchronized (sharedCollection) {
            while(sharedCollection.size() == 0) {
                try {
```

Testing GUI



The GUI class is called TestAreaOutputStream. It extends the Java's OutputStream class to redirect it into a JTextArea. This way we can create an executable file that looks like a terminal window for easier demonstration. This GUI has 3 buttons which are Start, Stop and Exit. Start button starts 3 separate threads for the consumer, producer and scroller. Stop button rises the cancel flag for the threads, this does not immediately stops the active threads, it just marks there cancel field for next time get cpu time they won't keep looping on the shared collection. Exit button makes a System.exit(1) system call. **(.jar file and gif in project folder)**