Assignment 7

(total 90 points)

Exercise 1: Thread

(50 points)

1. Defines a `MyStack` class, including following attributes and methods:

   Attributes:

- An array of char type with 5 elements .
- Other variables are defined as needed.

  Methods:

- `void push(char c)`: push a character in an array **(10 points)**
- `char pop()`: get a character from an array **(10 points)**

2. Defines two thread classes `Producer` and `Consumer`, `Producer` thread stores 26 English letters in the array of `MyStack` objects, `Consumer` thread extracts the 26 letters from the array of the same object. **(20 points, each class 10 points)**
3. When all five elements in the array have values, the producer thread needs to wait for the consumer thread to take the letter, and when the consumer thread takes one letter, the producer thread can be notified by the consumer thread. On the other hand, when there are no letters in the array, the consumer thread has to wait for the producer thread to put the letter, and after the producer puts a letter , it can notify the consumer thread to take the letter.
4. Define a test class, create an object for two thread classes, and start both threads at the same time. **(10 points)**

Tip: using `synchronized`, `wait()` and `notify()` to synchronize two threads.

Exercise 2: Network

(40 points, each class 20 points)

Write a Java Client / Server Application using TCP protocol, which meets the following requirements:

1. Write the client class named `EchoClient`**,** Which is a GUI application implemented with `JFrame`, the `JFrame` includes a `JLabel` , `JTextField` , `JButton` and `JTextArea`, The appearance looks like the figure 1.

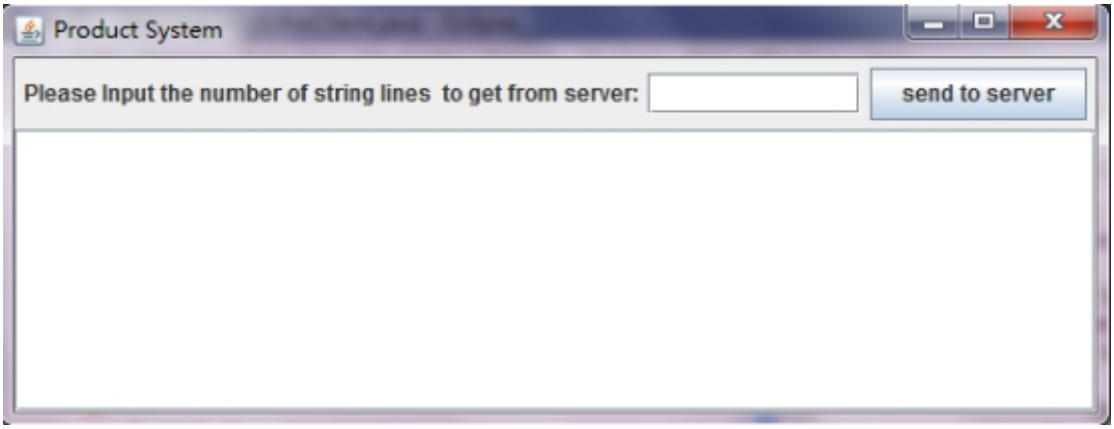

Figure 1

2. Write the server class named `EchoServer` , the server application will run on the server with IP xxx.xxx.xxx.xxx(for local computer, IP maybe 127.0.0.1) and listen on the network port number 4448. When server has started, display "**wait for client request**" on the console, like the figure 2.

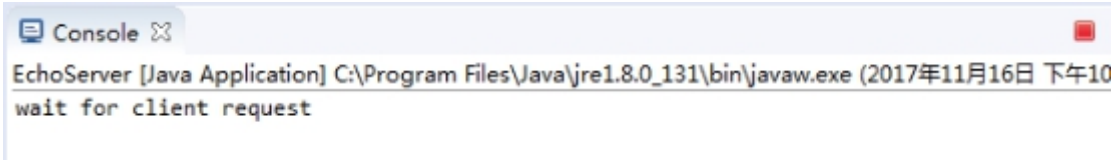

Figure 2

3. When user inputs the **integer number** in `JTextField` and click button " **send to server**" , the **number** will be sent to the network server. When the server accepts the number sent by client **,** the server will read **the number lines sentences** from **the file multiLines.txt** (according to **the number sent by client**) and merge the multiple sentences into **one string** , each sentence is separated by " **##"** , and send it back to client. If there are no more sentence to read from the file, then send the string "**No more sentences**" to client. (like figure 3, figure 4, figure 5)

   The content of file **multiLines.txt** is as follows:

```
This is a test1
This is a test2
This is a test3
This is a test4
This is a test5
This is a test6
This is a test7
This is a test8
This is a test9
This is a test10
```

4. When client received **the string** sent by server, then **appends** the string in the `JTextArea` (as shown in Figure 3) If the string include "No more sentences ", then display "**can not take sentences anymore**" at the end of `JTextArea` (as shown in figure 5)

After sent 2,4,5 to server separately, the results look like figure 3, figure 4, figure 5

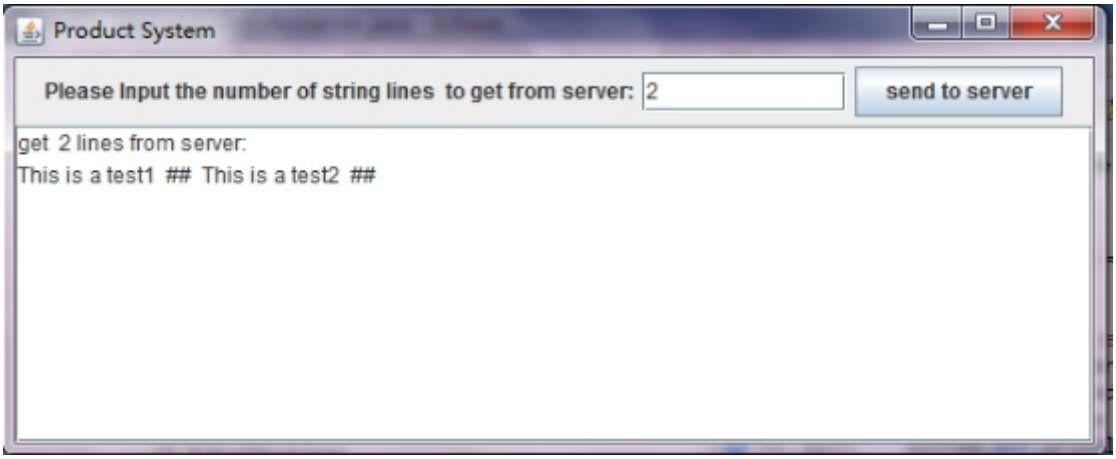

Figure 3

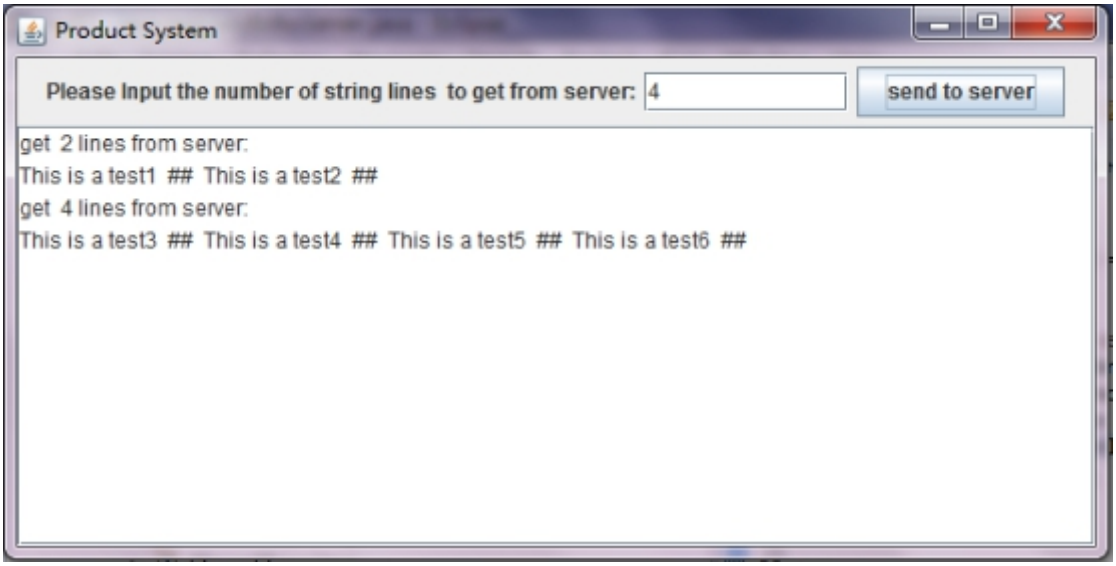

Figure 4

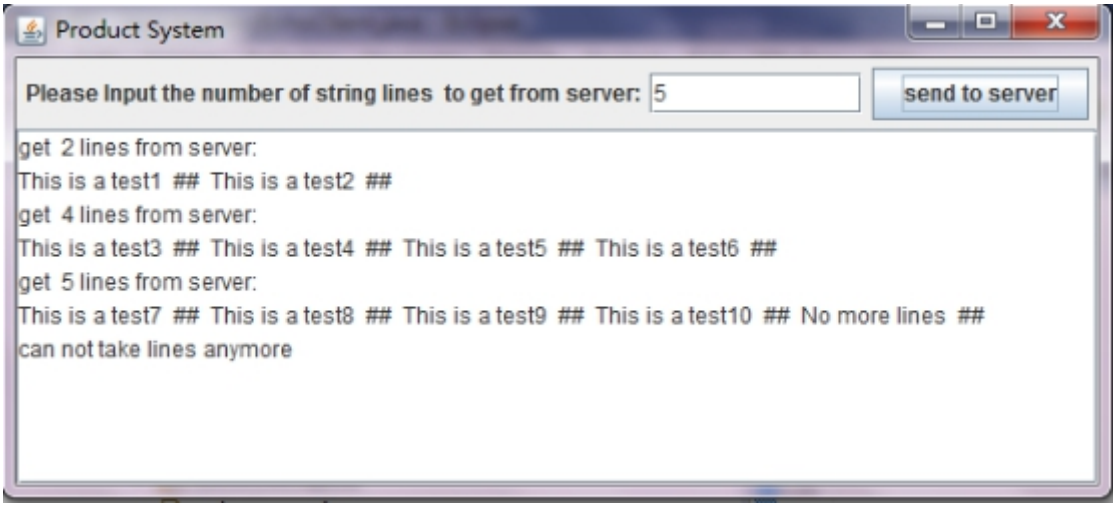

Figure 5