1. (Simple Chatting System) (50 Points)

Refer to the programs **Socket Client** and **Socket Server** for **chatting** that you learned in the class (Theory course 9, slide 19-20, *Source Code*/*Client.java* and *Source Code*/*Server.java*). Write a program using Java Thread and Java Network to make **multiple clients communicate with each other.** The task of the socket server is to **listen the messages received from the clients** and **broadcast the messages to all the other clients**, and the socket clients also need to **listen the messages received from the socket server**. You can use either User Datagram Protocol or Transmission Control Protocol to implement it. The detail of the chatting system is shown as follows:

Details:

1. When the server is built up, it should output the initialization message as shown in Fig. 1(a).
2. When a client has connected to the server, the server should output the message in the form of "Connect to client: {*Client's IP Address*}: {*Client's local port*}". An example is shown in Fig. 1(b).
3. When a client is built up, it should output the initialized message in the form of "Client-> {*Client's IP Address*}: {*Client's local port*}" as shown in Fig. 2. User can type messages in the last line of console, and send it when enter a **line feed**.
4. When a message is received, client who receives the message will output it to the console in the form of "{*Client's IP Address*}:{*Client's local port*} : {*message*}". An example is shown in Fig. 3(a) and Fig. 3(b).
5. Empty message will not be sent, and an error message will be output. An example is shown in Fig. 4.
6. A chatting example is shown in Fig. 5.

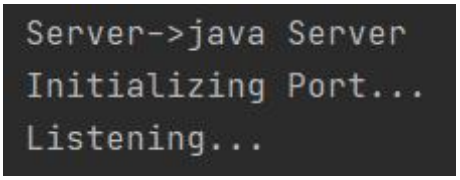


Fig. 1(a) The output of server when it is built up.

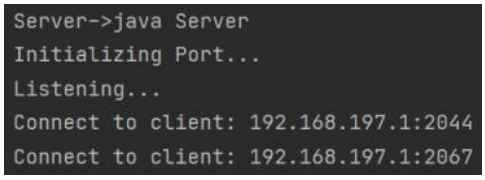


Fig. 1(b) The output of server when two clients connected.

Fig. 2 The output of clients when they are built up

Fig. 3(a) The output of client when it sends a message

Fig. 3(b) The output of client when it receives a message

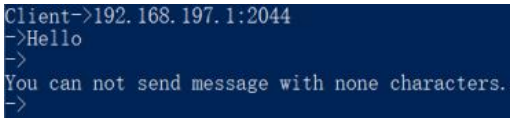


Fig. 4 Empty Message will not be sent

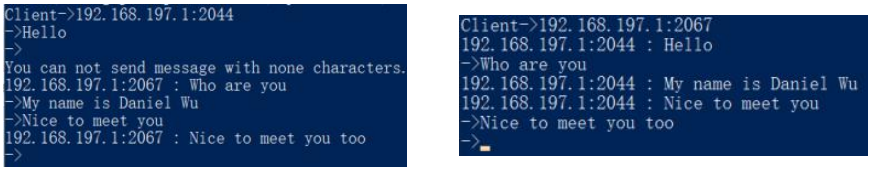
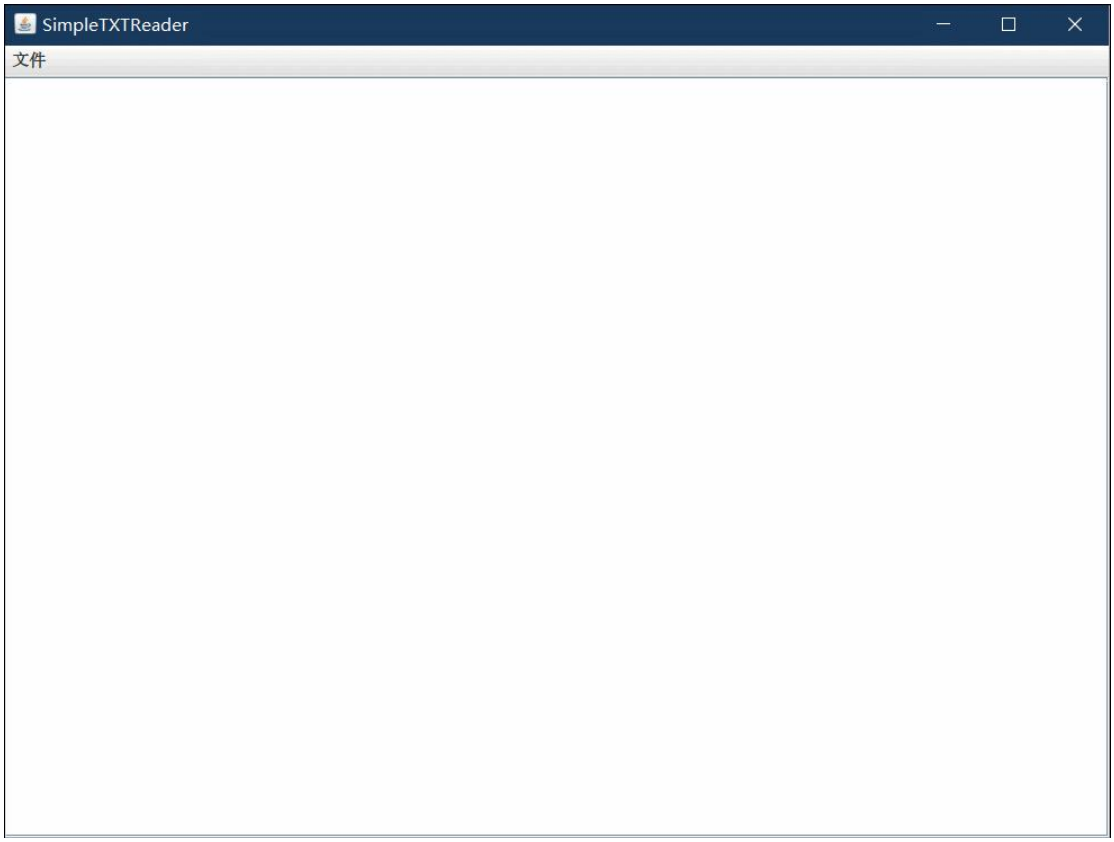


Fig. 5 A chatting Example

2. (Simple txt reader) (50 Points)

Write a simple txt reader. Its function is as follow.



Mention: If the GIF picture doesn't play, check the supplemental files. (Reader. gif).

Details:

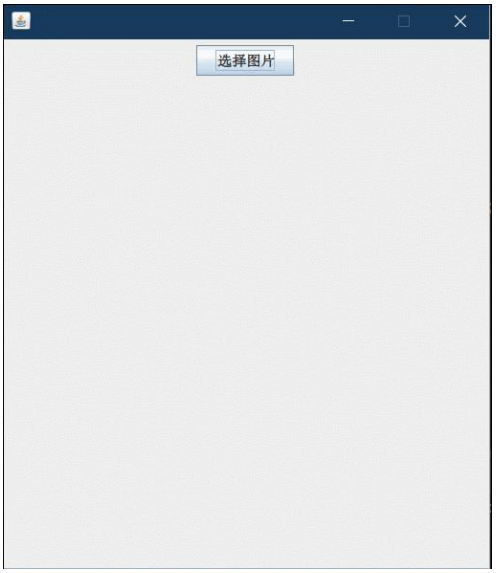
1. The initial size of the window is (800,600)
2. Text can be wrapped automatically
3. Text is not writable but readable only
4. Only vertical scroll bar, no horizontal scroll bar
5. A separating line is between the menu item "关闭" and "退出"
6. The menu item "关闭" is enabled only when a file is open
7. Only files with suffix ".txt" can be chosen to read
8. The default path of file chooser is the current path where program executes

Requirements:

- You should **implement** the program as shown above.
- **All details** should be implemented.

3. (Eight puzzle) (Bonus question: 20 Points)

Write a program to play the 8-puzzle game. It's function is as follow.



Mention: If the GIF picture doesn't play, check the supplemental files. (Puzzle.gif).

Details:

1. The window is not resizable and its size is (421, 480)
2. Only images with suffix ".jpg", ".jpeg" and ".png" can be chosen to play
3. Image should first be resized to (399, 399) and then cut into 3x3 with the bottom right part being set to a white image
4. Puzzle starts with random permutation
5. When press key ←, the image on the right of the white part should move left
6. When press key →, the image on the left of the white part should move right
7. When press key ↑, the image at the bottom of the white part should move up
8. When press key ↓, the image at the top of the white part should move down

Requirements:

- You should **implement** the program as shown above.
- **All details** should be implemented.