Further Java Ticklet 1*

In order to gain a star in the mark sheet you must complete this exercise. Completing the exercise does not gain you any credit in the examination. In this exercise you will write a client which connects to a server and sends and receive images in JPEG[1] format. An example client is shown in Figure 1, “An example image chat client”.

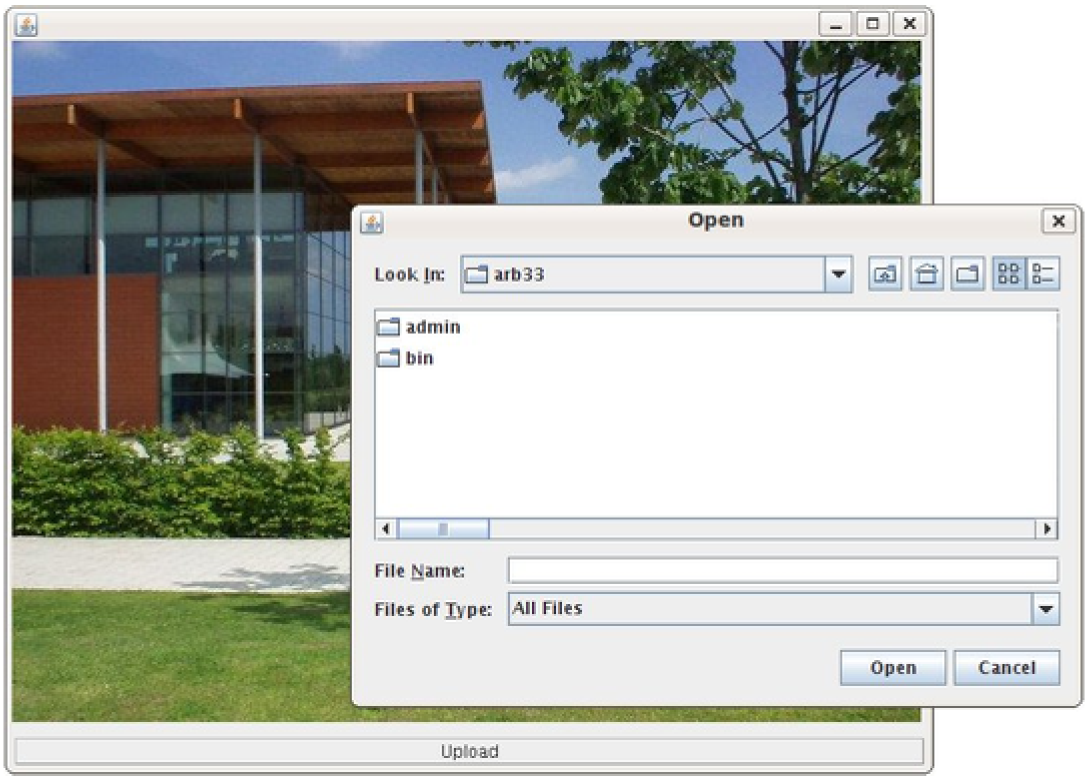

Figure 1. An example image chat client

Your program should have the following features:

- Your solution should contain a class called `ImageChatClient` which extends `JFrame` inside a package called `uk.ac.cam.your-crsid.fjava.tick1star`.
- Your implementation of `ImageChatClient` must contain a constructor which takes two arguments; the first argument should accept the name of the server as a `String` and the second argument the port number to connect to, expressed as an `int`.
- When an object of type `ImageChatClient` is instantiated it should create a Swing GUI with at least two components, an AWT `Canvas` object to render images sent by the server, and an AWT `Button` with the text label "Upload". You may add additional GUI components if you wish.
- Your implementation of `ImageChatClient` should use an additional thread in a similar manner to `StringChat` you wrote for Ticklet 1. One thread should listen on a `Socket` object connected to the server and draw any new image sent by the server to the `Canvas` object; the other thread should listen for user input.
- If the `Button` object is clicked, you should create and show an instance of the `JFileChooser` class. In the case where a user selects a suitable JPEG file using the dialog and selects the "Open" button, your program should send the bytes representing the JPEG file to the server over a `Socket` object; otherwise your program should do nothing.
- Clicking on the close button in the top right-hand corner of the Swing window created by the `ImageChatClient` class should close the application.

[1]http://en.wikipedia.org/wiki/JPEG

Hints 'n' tips

- You may test your implementation by connecting to `java-1b.cl.cam.ac.uk` on port number 15002.
- The data sent by the server is a sequence of JPEG images sent as a stream of bytes. You will need to design a method of working out when you have received all the data representing a single image. One solution is to inspect the stream for the JPEG format End Of Image (EOI) marker, which in hexadecimal is `0xffD9`. (In your code you can check whether consecutive bytes have the values `-1` and `-39` to detect the end of an image in the stream of bytes; why?)
- Avoid using the `ImageIO.read(InputStream is)` method to directly read data from a `Socket` object since the read method (erroneously in this case!) assumes that only a single image exists in a given stream of data. Instead you should read the data from the `Socket` object into a small buffer, perform some manual checks to detect whether you have reached the end of the JPEG image or not (e.g. look for the EOI marker), before writing the data out to a second stream which you can feed into `ImageIO.read(InputStream is)`. You will probably find a small array of type `byte` and the `ByteArrayInputStream` and `ByteArrayOutputStream` classes useful for this.

Submission

Start by joining the Ticklet 1* project then cloning the Ticklet 1* repository from the Chime server: https://www.cl.cam.ac.uk/teaching/current/FJava/ticklet1star. Complete the implementation of `ImageChatClient` as described above, commit all changes to your local repository and push your changes to the Chime server. On the Chime server, schedule testing for this exercise as you did for Ticklet 1. If, after waiting an hour, you have not received a response, please send an email to `ticks1b-admin@cl.cam.ac.uk`.