Further Java Ticklet 2*

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 implement a custom security manager for a new chat client called `SafeChatClient`. Your new version of the chat client should allow *only* the following network and file system actions for any code your program downloads:

- Connect to `www.cam.ac.uk` on port 80 (i.e. web traffic).
- Read (but not modify) the home directory of the user running the chat client.

Start the exercise on Chime here https://www.cl.cam.ac.uk/teaching/current/FJava/ticklet2star then clone the ticklet2star repository to your local machine. Then copy your implementation of `ChatClient` from Ticklet 2 into the class called `SafeChatClient` inside the package `uk.ac.cam.your-crsid.fjava.tick2star`. Take a copy of the source code for `uk.ac.cam.cl.fjava.messages.DynamicObjectInputStream` and use it as a starting point for `SafeObjectInputStream` in your Ticklet 2* package. Your task for this Ticklet is to modify the code for `SafeChatClient` and `SafeObjectInputStream` so that any code downloaded and invoked by the client can only perform the file system and network actions as described as above.

Hints 'n' tips

- Replace the use of `DynamicObjectInputStream` with `SafeObjectInputStream` inside `SafeChatClient` and check your code works in an identical way to `ChatClient` by connecting to `java-1b.cl.cam.ac.uk` on port 15004. The class which is downloaded from the server will print `DONE` if it has worked successfully.
- By default, the JVM runs without an instance of the `SecurityManager` object. You'll need one in order to be able to define security policies for the class loader. Therefore your first task is to create an instance of `SecurityManager` and add it to the virtual machine (hint: try `System.setSecurityManager`). Your chat client should *not* work with the default `SecurityManager`, since the default policy does not permit class loaders.
- You should use the following `String`, representing a URL, as the source of the security policy: `"http://www.cl.cam.ac.uk/teaching/current/FJava/all.policy"` (hint: try `System.setProperty("java.security.policy", ...)`). Your implementation of `SafeChatClient` should now run as before (i.e. with the security flaws). Why?
- Rewrite the `addClass` method in `SafeObjectInputStream` to use `java.security.SecureClassLoader` instead of `java.lang.ClassLoader`. The class `SecureClassLoader` provides a definition of the method `defineClass` which takes a fifth argument of type `java.security.ProtectionDomain`. (You will need to create an instance of `ProtectionDomain` which supports only the permissions the downloaded class should be given. Apple users may find that additional permissions are needed and should investigate the need for read and write permissions within `apple.*` as well as `java.version`)
- The home directory of the current user can be found using the following expression `System.getProperty("user.home")`.
- Test your implementation of `SafeChatClient` by connecting to `java-1b.cl.cam.ac.uk` on port 15004. The new class file should load and execute and the "wholesome content" should download, but the "dubious content" should fail to download. Similarly, any modifications to the filesystem should fail.

Submission

When you are satisfied you have completed everything, you should commit all outstanding changes and push these to the Chime server. On the Chime server, check that the latest version of your files are in the repository, and once you are happy schedule your code for testing. You can resubmit as many times as you like and there is no penalty for re-submission. If, after waiting one hour, you have not received a final response you should notify `ticks1b-admin@cl.cam.ac.uk` of the problem. You should submit a version of your code which successfully passes the automated checks by the deadline, so don't leave it to the last minute!