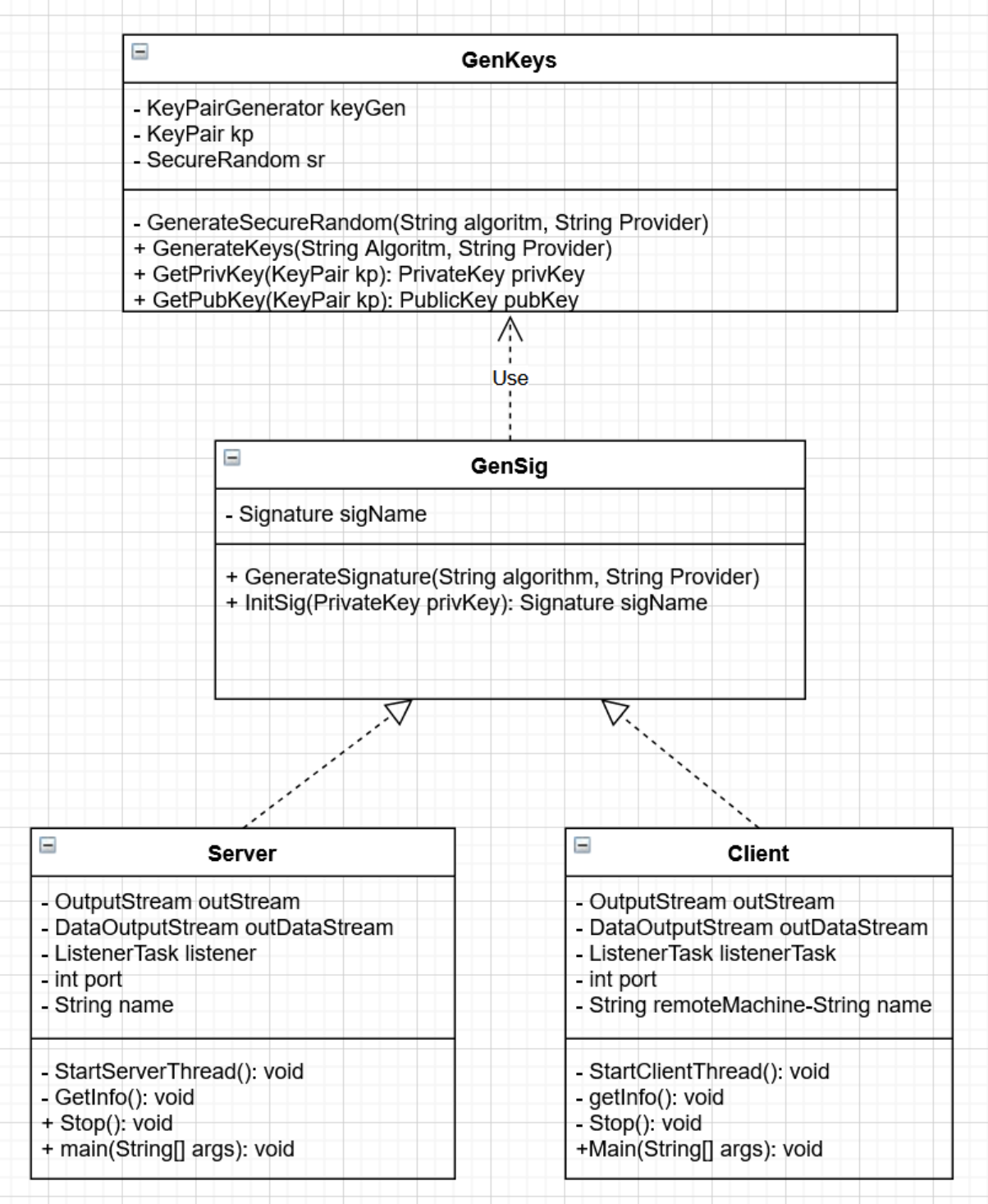
UML Diagram Write-Up

Introduction:

This document is meant to serve as an aid in understanding the UML being used for the project.



[Figure 1.1 – The Current UML Revision]

GenKeys:

This class will be responsible for the generation of the user's keys that will be used to authenticate and encrypt their data.

Private Variables:

KeyGen:

This is a KeyPairGenerator Object that is used to generate keys. It will be used to populate the KeyPair object.

kp:

This is a KeyPair Object that is used to store the KeyPair that will be generated. It stores both a Public and Private Key value.

sr:

This is a SecureRandom Object that is used for key generation. It will be used to store a securely generated random number suitable for encryption.

Class Functions:

GenerateSecureRandom (Private):

This function is void and will generate a SecureRandom object to populate the sr variable. It takes in an algorithm string, such as "SHA1PRNG" and a provider string, such as "SUN".

GenerateKeys (Public):

This function is void. First, it will instantiate the KeyPairGenerator object using it's "getInstance" function, this will use the algorithm String, such as "DSA" and provider String, such as "SUN".

The KeyPairGenerator will then be initialised using it's "intitialize" function where it will take in a hard-coded bit-size and SecureRandom object (This will involve calling the GenerateSecureRandom Function).

It will then populate the kp using the KeyPairGenerator object's "generateKeyPair" function.

GetPriv/PubKey (Public):

This pair of function simply return the corresponding key using the KeyPair Object's "getPrivate" and "getPublic" functions.

GenSig:

Private Variables:

sigName:

This is a Signature Object. It will be used to sign data. Each user will have their own unique signature.

Class Functions:

GenerateSignature (Public):

This will use the sigName's "getInstance" function which will use the algorithm String, such as "SHA1withDSA" and provider String "SUN".

InitSig (Public):

This function will initialise sigName using the "initSign" function which will take the user's private key as a parameter.

Server/Client:

README:

These functions are nigh on identical. While some may pose the question, why not Client-Client? The answer to this is based off the current model researched. The Client and Server are interchangeable, the server is not centralised. Whichever user loads the program first will act as the ‘server’ for that chat session. This will become more clear as you get to the end of this section.