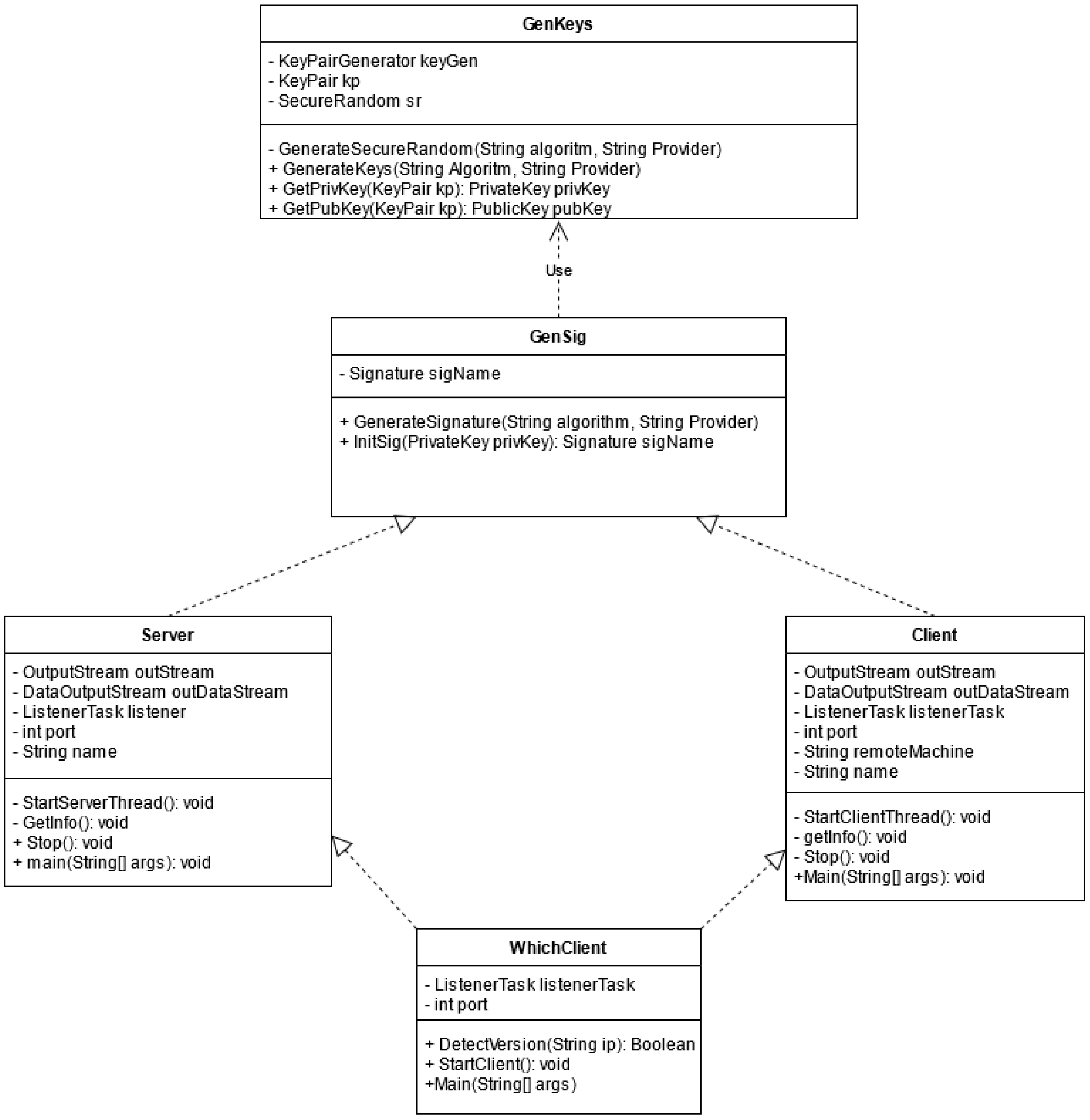
UML Diagram Write-Up

Introduction:

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


GenKeys
- KeyPairGenerator keyGen
- KeyPair kp
- SecureRandom sr
- GenerateSecureRandom(String algoritm, String Provider)
+ GenerateKeys(String Algoritm, String Provider)
+ GetPrivKey(KeyPair kp): PrivateKey privKey
+ GetPubKey(KeyPair kp): PublicKey pubKey
Use
GenSig
- Signature sigName
+ GenerateSignature(String algorithm, String Provider)
+ InitSig(PrivateKey privKey): Signature sigName
Server
- OutputStream outStream
- DataOutputStream outDataStream
- ListenerTask listener
- int port
- String name
- StartServerThread(): void
- GetInfo(): void
+ Stop(): void
+ main(String[] args): void
Client
- OutputStream outStream
- DataOutputStream outDataStream
- ListenerTask listenerTask
- int port
- String remoteMachine
- String name
- StartClientThread(): void
- getInfo(): void
- Stop(): void
+Main(String[] args): void
WhichClient
- ListenerTask listenerTask
- int port
+ DetectVersion(String ip): Boolean
+ StartClient(): void
+Main(String[] args)

[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.

WhichClient:

README:

The Client and Server classes are near identical aside from the following:

- The Server waits for the client to initiate a connection.
- The client needs to know the IP of the server it is connecting to.

It is my intention that either user is capable of being the server, depending purely on whoever starts the application fist. This class is designed to help see if the user needs to run a server, if their recipient has not loaded their application yet, or run a client, if the recipient has loaded the server first.

Private Variables:

listenerTask:

This object is going to listen to a reply from either a client or server.

port:

The integer value of the port the application will be running on.

Class Functions:

DetectVersion (Public):

This function will send a piece of data to the ip specified. The response from the application will set a Boolean value, true if the server is running and false if not. This will be used to determine which application the client loads.

StartClient (Public):

This will use the DetectVersion function and launch the corresponding Main function of either the Server, or the client.

Main (Public):

Used to start the application and run the proper commands.

Notes:

- Implement key generation and key exists check in WhichClient.
- Review Server/Client Classes with team to look at best implementation approach.