FALMOUTH UNIVERSITY

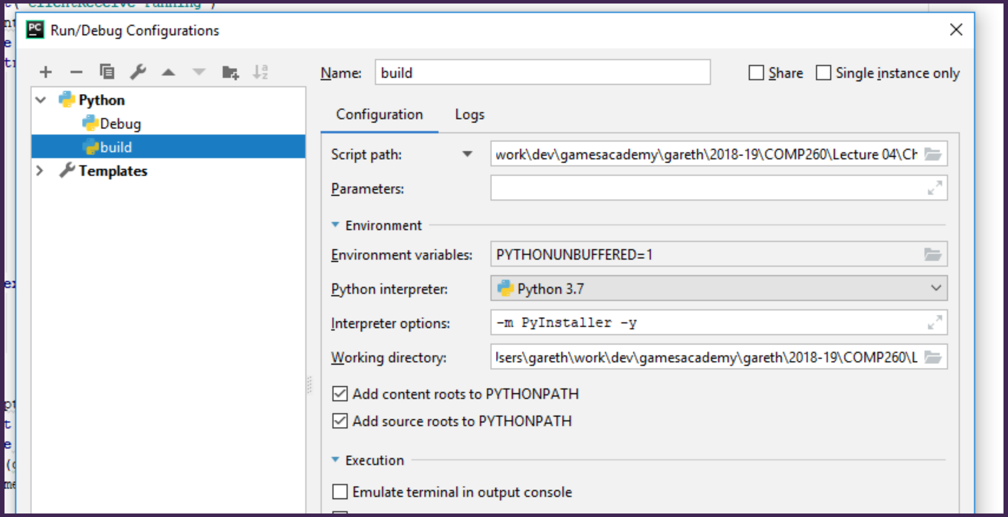
Lecture 4: Chat Service

COMP260: Distributed Systems
BSc (Hons) Computing for Games

- Today's session:
  - Demo
  - Chat service Spec
  - Chat service Design
  - Chat service key implementational issues

- Demo

- Demo
  - – For ease of use, I’ve added a build configuration that will make .exe of python
    - Need to install PyInstaller



Run/Debug Configurations

- Python
  - Debug
  - build
- Templates

Name: build

□ Share □ Single instance only

Configuration | Logs

Script path: work\dev\gamesacademy\gareth\2018-19\COMP260\Lecture 04\Ch

Parameters:

Environment

Environment variables: PYTHONUNBUFFERED=1

Python interpreter: Python 3.7

Interpreter options: `-m PyInstaller -y`

Working directory: lsers\gareth\work\dev\gamesacademy\gareth\2018-19\COMP260\L

☑ Add content roots to PYTHONPATH

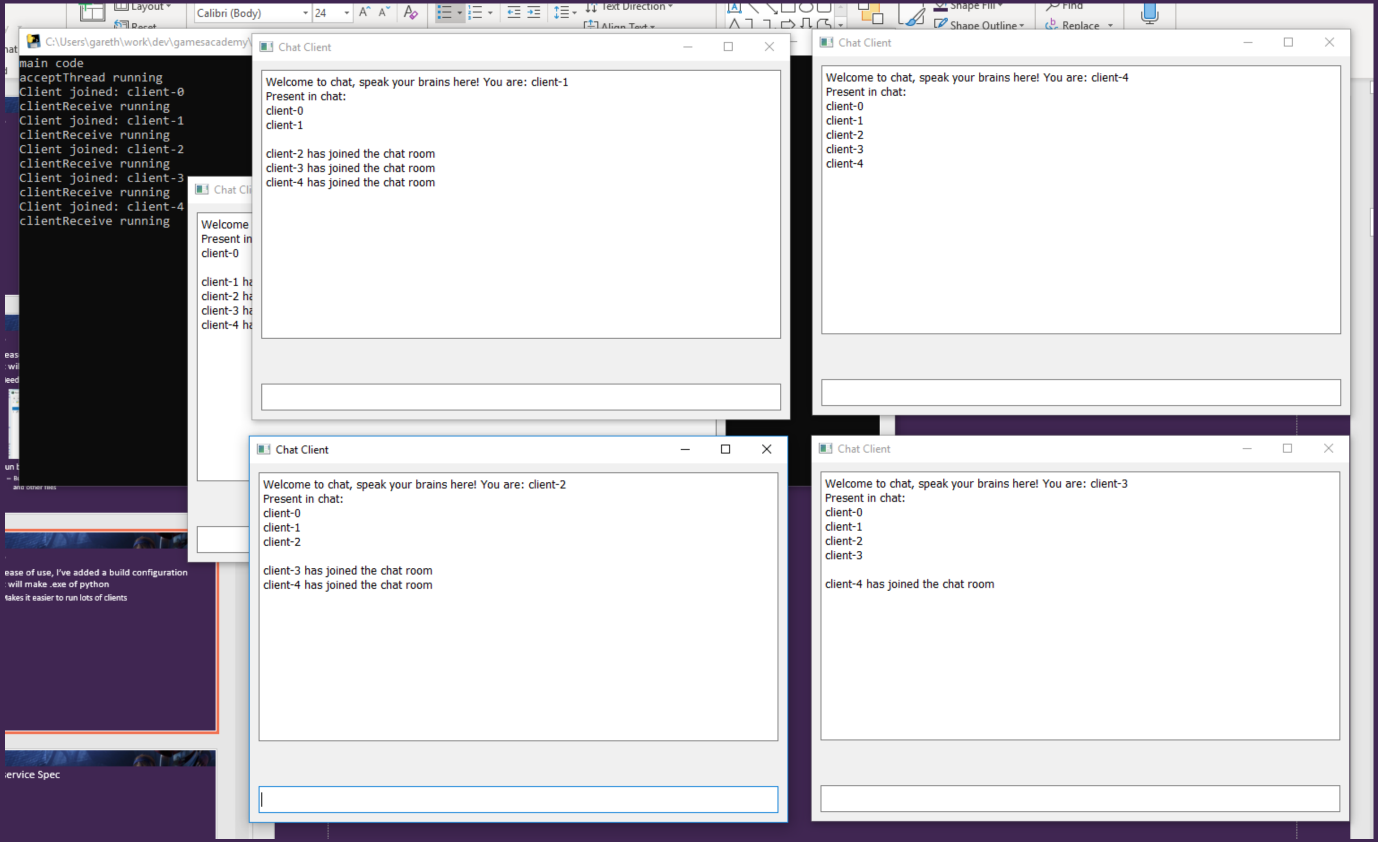
☑ Add source roots to PYTHONPATH

Execution

□ Emulate terminal in output console

- Run build configuration (don’t debug it)
  - – Build folder will be created containing .exe and associated .dll and other files

- Demo
  - For ease of use, I've added a build configuration that will make .exe of python
    - Makes it easier to run lots of clients



C:\Users\gareth\work\dev\gamesacademy\

```
main code
acceptThread running
Client joined: client-0
clientReceive running
Client joined: client-1
clientReceive running
Client joined: client-2
clientReceive running
Client joined: client-3
clientReceive running
Client joined: client-4
clientReceive running
```

Chat Client

```
Welcome
Present in
client-0

client-1 ha
client-2 ha
client-3 ha
client-4 ha
```

Chat Client

```
Welcome to chat, speak your brains here! You are: client-1
Present in chat:
client-0
client-1

client-2 has joined the chat room
client-3 has joined the chat room
client-4 has joined the chat room
```

Chat Client

```
Welcome to chat, speak your brains here! You are: client-4
Present in chat:
client-0
client-1
client-2
client-3
client-4
```

Chat Client

```
Welcome to chat, speak your brains here! You are: client-2
Present in chat:
client-0
client-1
client-2

client-3 has joined the chat room
client-4 has joined the chat room
```

Chat Client

```
Welcome to chat, speak your brains here! You are: client-3
Present in chat:
client-0
client-1
client-2
client-3

client-4 has joined the chat room
```

- Chat service Spec

- Chat service Spec

You have been hired to develop a ‘chat service’ that will consist of client and server applications that communicate using the IP protocol.
Users can run a graphical client application (Python & PyQt5) which will connect to the server. Once connected, any messages the user enters will be sent to the rest of the clients currently connected by way of the server. When a client joins or leaves the service, all other users will be informed.
The service will be robust in that client or server termination or unavailability will not adversely affect the remaining clients and/or server.

- Chat service Design

- Chat service Design
  - Step 1: Do some analysis
    - What happens if we perform a linguistic decomposition on the spec

You have been hired to develop a ‘chat service’ that will consist of client and server applications that communicate using the IP protocol.
Users can run a graphical client application (Python & PyQt5) which will connect to the server. Once connected, any messages the user enters will be sent to the rest of the clients currently connected by way of the server. When a client joins or leaves the service, all other users will be informed.
The service will be robust in that client or server termination or unavailability will not adversely affect the remaining clients and/or server.

- Chat service Design
  - Step 1: Do some analysis
    - What happens if we perform a linguistic decomposition on the spec
      - Objects (nouns), methods (verbs) , attributes (adjectives)

You have been hired to develop a ‘chat service’ that will consist of client and server applications that communicate using the IP protocol.
Users can run a graphical client application (Python & PyQt5) which will connect to the server. Once connected, any messages the user enters will be sent to the rest of the clients currently connected by way of the server. When a client joins or leaves the service, all other users will be informed.
The service will be robust in that client or server termination or unavailability will not adversely affect the remaining clients and/or server.

- Chat service Design
  - Step 1: Do some analysis
    - What happens if we perform a linguistic decomposition on the spec

You have been hired to develop a ‘chat service’ that will consist of client and server applications that communicate using the IP protocol.
Users can run a graphical client application (Python & PyQt5) which will connect to the server. Once connected, any messages the user enters will be sent to the rest of the clients currently connected by way of the server. When a client joins or leaves the service, all other users will be informed.
The service will be robust in that client or server termination or unavailability will not adversely affect the remaining clients and/or server.

- LD doesn’t guarantee a simple path, there is scope for ambiguity
  - » Connected as a verb
  - » Connected as an adjective

- Work to earn your salary mr. technical designer

- Chat service Design
  - Step 2: First pass class hierarchy

Class	Methods	Attributes
Chat service		Client Server
Client	Connect Terminate Join Leave Communicate(server)	Message
Server	Communicate(client) Terminate Send (message) Inform (user)	Message

- Chat service Design
  - Step 3: Add some sense

Class	Methods	Attributes
Chat service		Client Server

Client	Connect(server)
	Terminate
	~~Join~~
	~~Leave~~
	~~Communicate(server)~~
	SendMessage(server)
	ReceiveMessage(server)
	Type(message)
		Message
		Server socket

Server	~~Communicate(client)~~
	Terminate
	~~Send (message)~~
	~~Inform (user)~~
	SendMessage(clients)
	ReceiveMessage(clients)
	AddClient()
	LoseClient(client)
		Message
		container<clients>

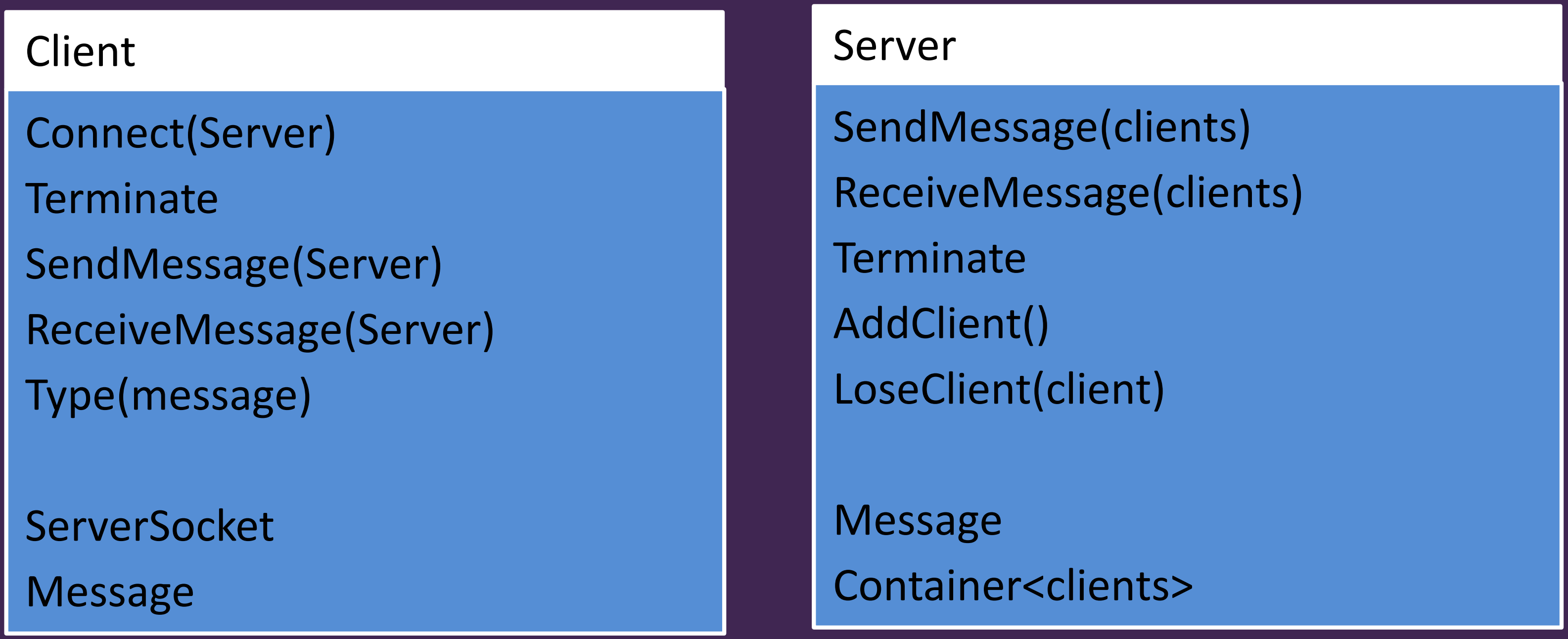
- Chat service Design
  - – Step 4: Create a class hierarchy

Class	Methods	Attributes
Chat service		Client Server

Client	Connect(server)
	Terminate
	SendMessage(server)
	ReceiveMessage(server)
	Type(message)
		Message
		Server socket

Server	AddClient()
	SendMessage(clients)
	ReceiveMessage(clients)
	LoseClient(client)
		Message
		container<clients>

- Chat service Design
  - – Step 4: Create a class hierarchy



Client

Connect(Server)
Terminate
SendMessage(Server)
ReceiveMessage(Server)
Type(message)

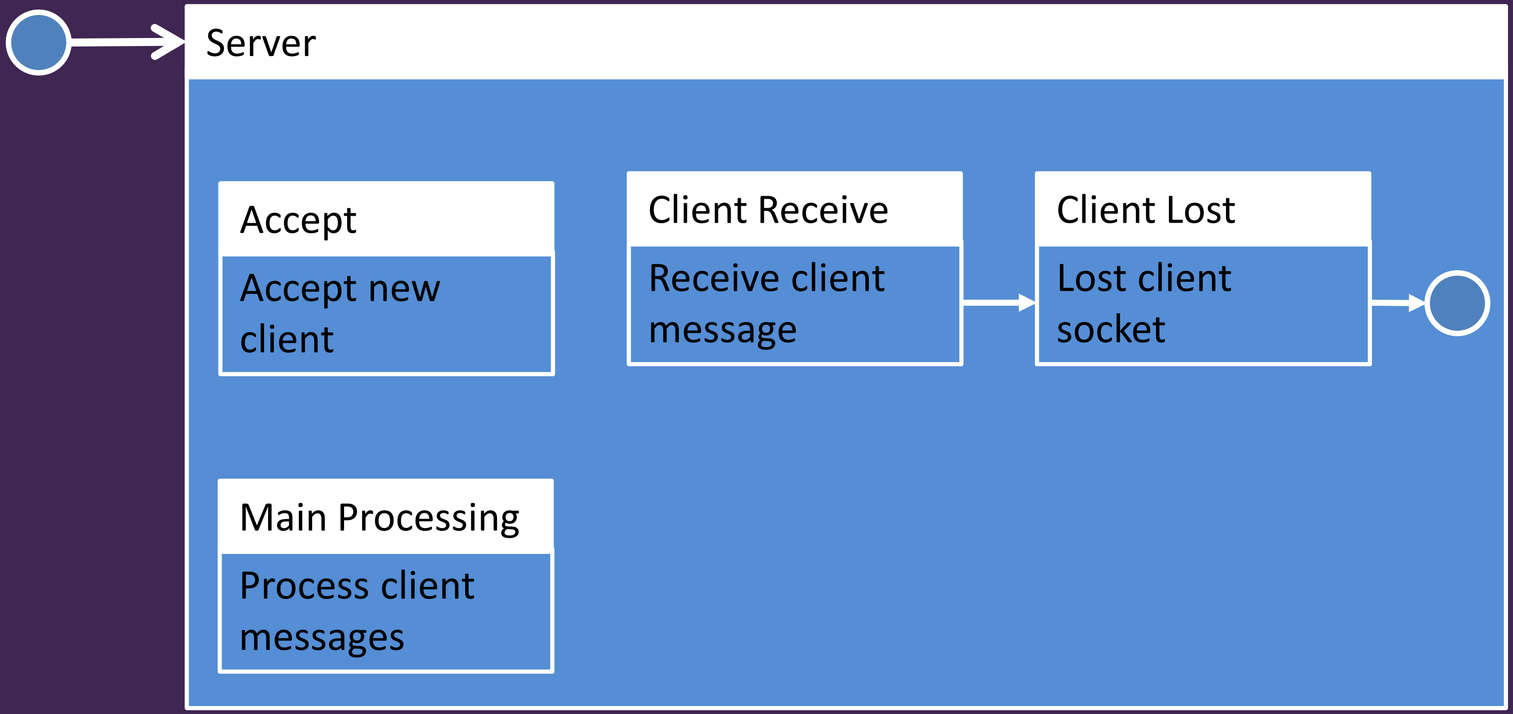
ServerSocket
Message

Server

SendMessage(clients)
ReceiveMessage(clients)
Terminate
AddClient()
LoseClient(client)

Message
Container<clients>

- Chat service Design
  - Step 5: Create state diagrams



Server

Accept

Accept new client

Client Receive

Receive client message

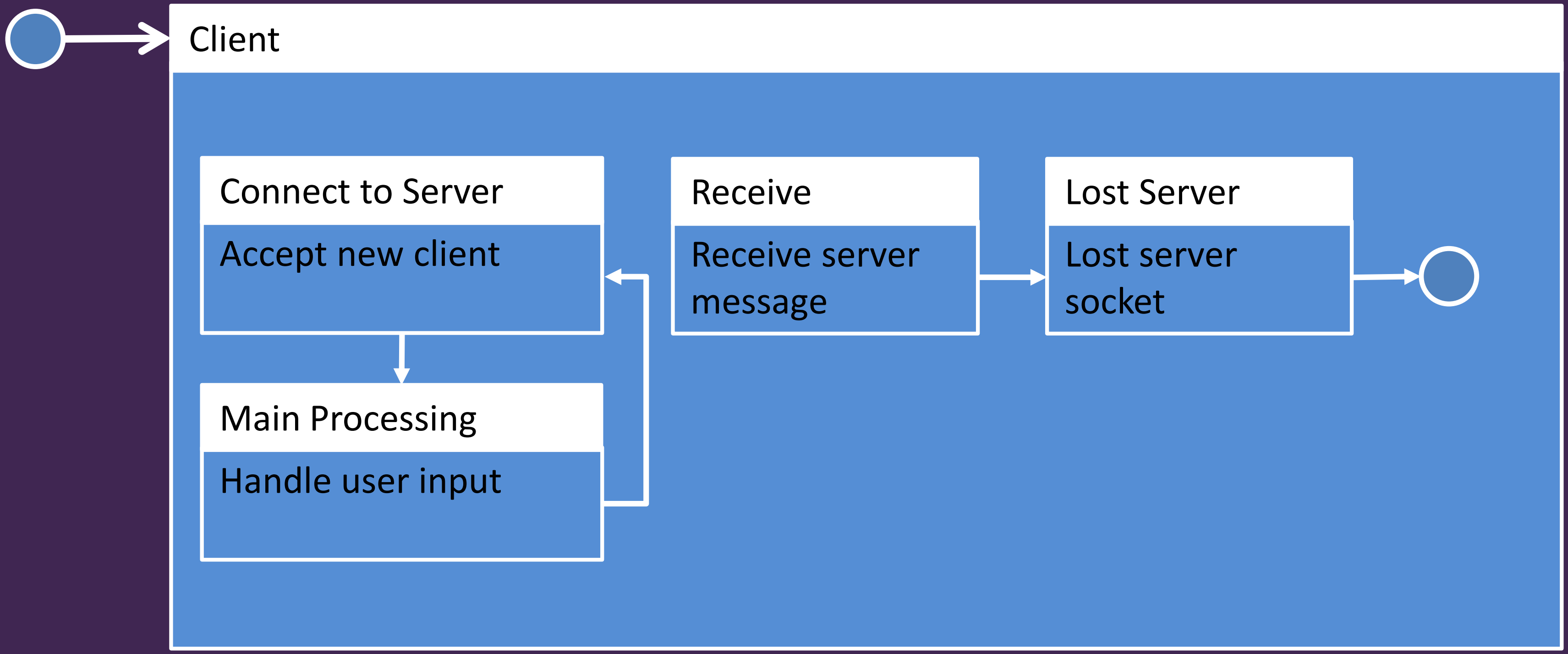
Client Lost

Lost client socket

Main Processing

Process client messages

- Chat service Design
  - Step 5: Create state diagrams



Client

Connect to Server

Accept new client

Main Processing

Handle user input

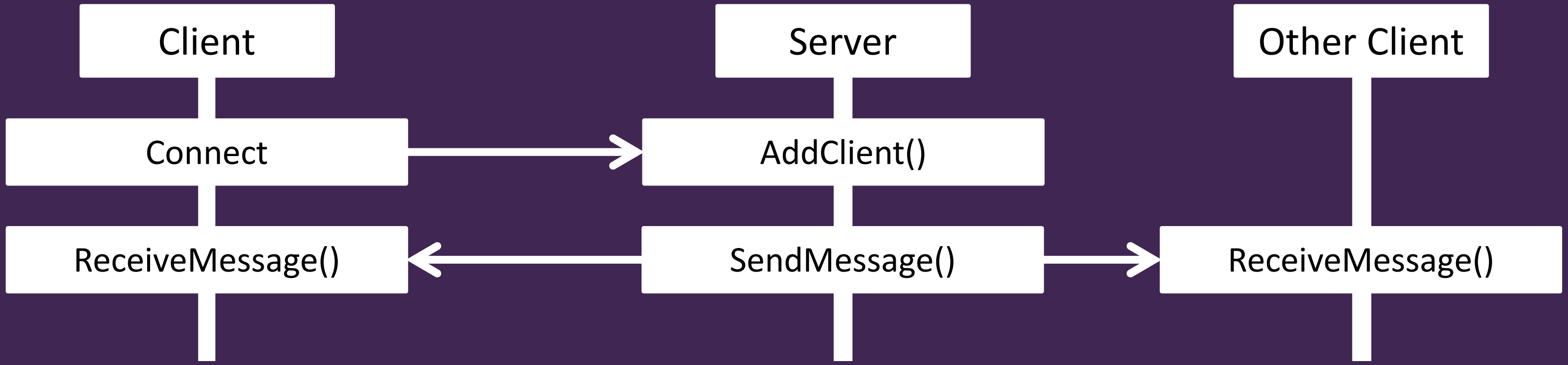
Receive

Receive server message

Lost Server

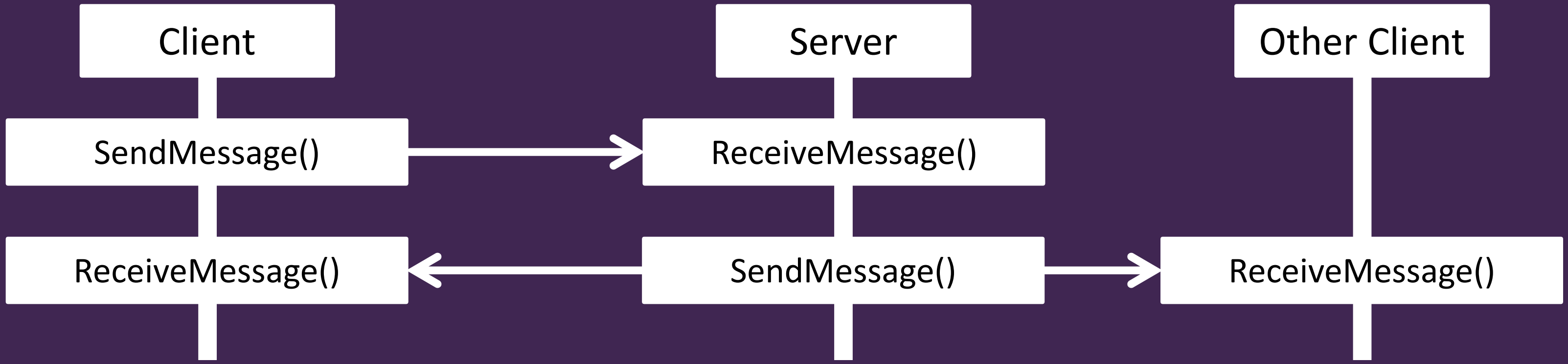
Lost server socket

- Chat service Design
  - Step 6: Create sequence diagrams
    - New client connects



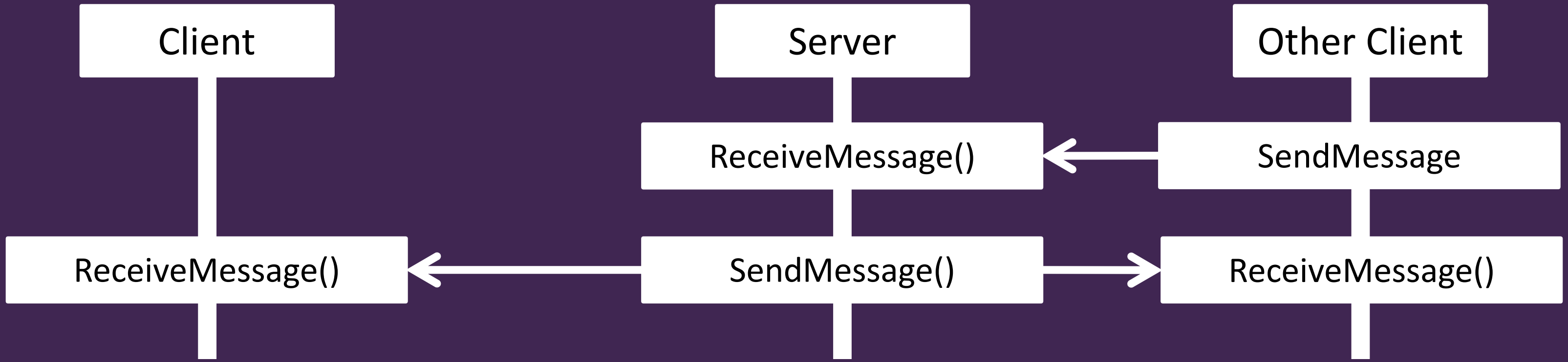
New client connects
-server receives connect request, adds to client container
-sends messages to all clients that a client has connected

- Chat service Design
  - Step 6: Create sequence diagrams
    - Client sends a message



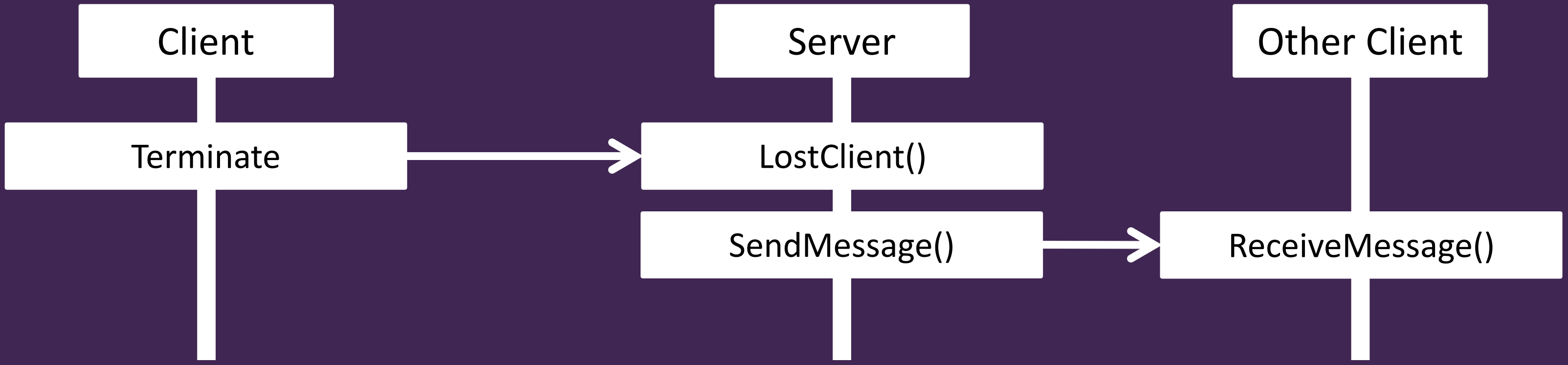
Client sends a message
-client sends a message to the server
-server receives it and sends it to all clients in the container<client>

- Chat service Design
  - Step 6: Create sequence diagrams
    - An other client sends a message



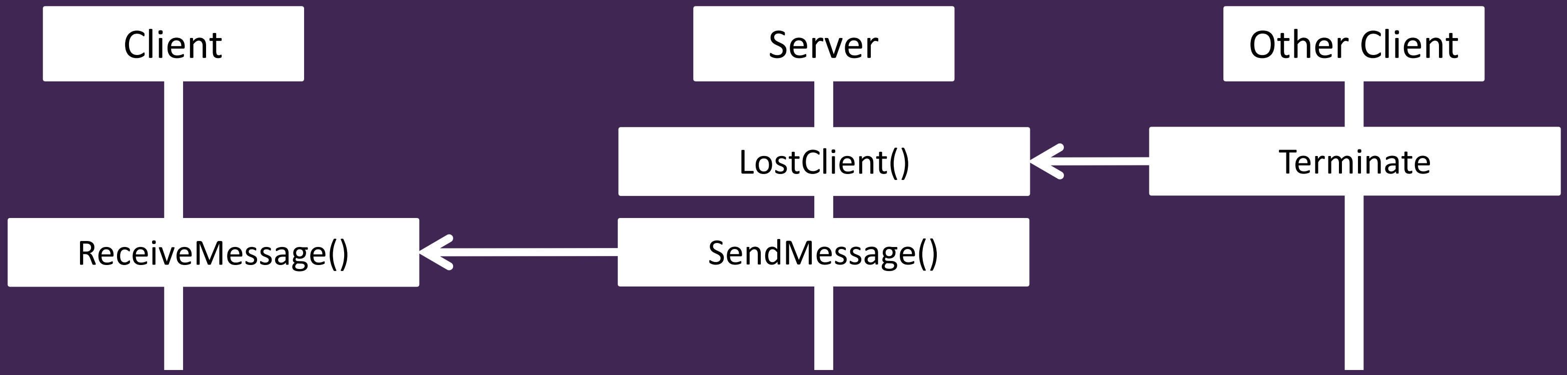
An other client sends a message
-other client sends a message to the server
-server receives it and sends it to all clients in the container<client>
-’our’ client receives it

- Chat service Design
  - Step 6: Create sequence diagrams
    - Client is lost



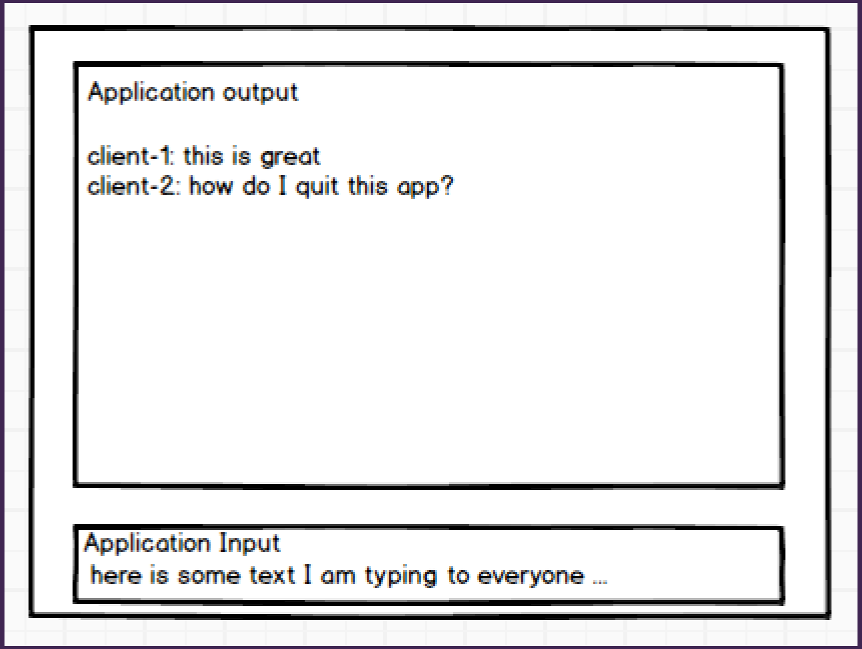
Client is lost
-Server receives lost client (from socket.recv() exception)
-Server removes it from container<client>
-Sends a lost client message to other clients
-other clients receive message and display it

- Chat service Design
  - Step 6: Create sequence diagrams
    - An other client is lost



An other Client is lost
-Server receives lost client (from socket.recv() exception) from other client
-Server removes it from container<client>
-Sends a lost client message to other clients
-other clients receive message and display it

- Chat service Design
  - Step 7: Client UI design
    - How to make / layout PyQT



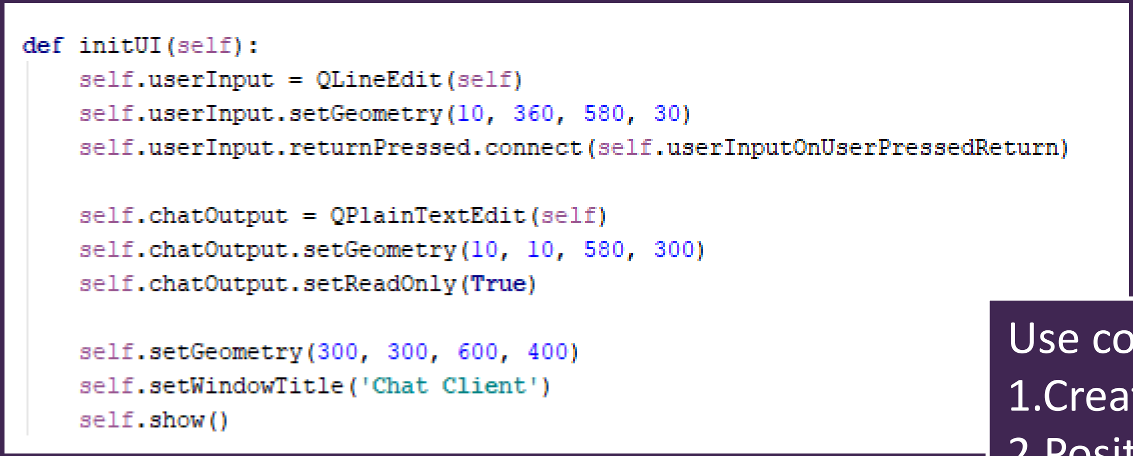
Application output

client-1: this is great
client-2: how do I quit this app?

Application Input
here is some text I am typing to everyone ...

Mockups are a great place to start
-whiteboard
-pen and paper
-Balsamiq
-Graph paper
-Powerpoint
-Photoshop / an.other art package
-etc

- Chat service Design
  - Step 7: Client UI design
    - How to make / layout PyQT
      - UI can be hard coded in Python (Widget.Init)



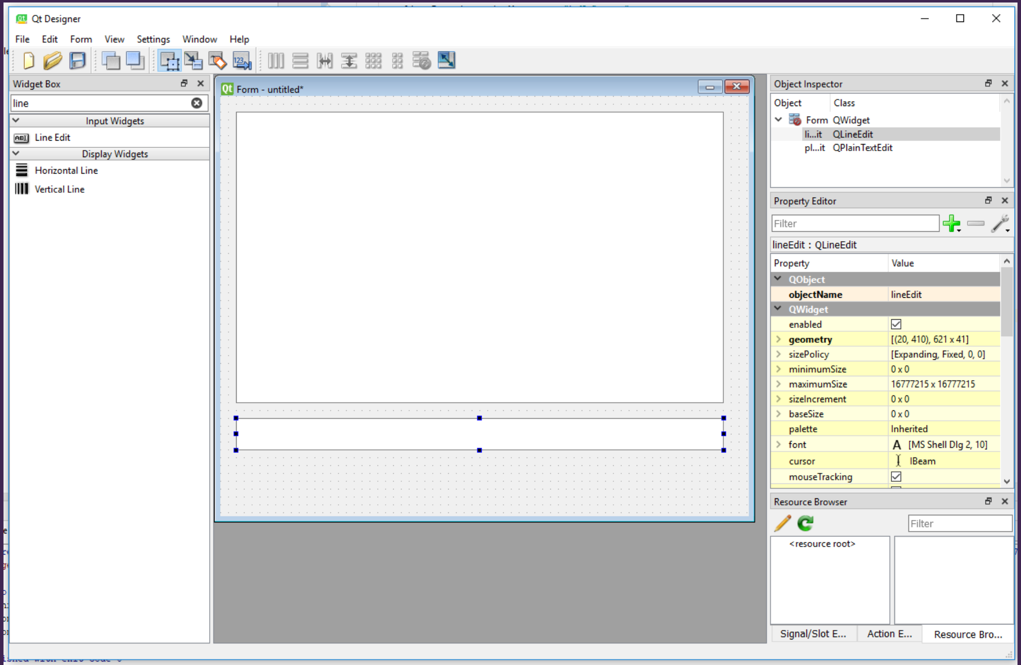
```
def initUI(self):
    self.userInput = QLineEdit(self)
    self.userInput.setGeometry(10, 360, 580, 30)
    self.userInput.returnPressed.connect(self.userInputOnUserPressedReturn)

    self.chatOutput = QPlainTextEdit(self)
    self.chatOutput.setGeometry(10, 10, 580, 300)
    self.chatOutput.setReadOnly(True)

    self.setGeometry(300, 300, 600, 400)
    self.setWindowTitle('Chat Client')
    self.show()
```

Use code to:
1.Create Qt components
2.Position and size them
3.Add callbacks for functionality
4.Set any widget-specific states

- Chat service Design
  - Step 7: Client UI design
    - How to make / layout PyQT
      - Use Qt Designer
        - » Python\Python37-32\Lib\site-packages\pyqt5_tools
        - » https://stackoverflow.com/questions/18429452/convert-pyqt-ui-to-python



Widget Box: line

Object	Class
Form	QWidget
li...it	QLineEdit
pl...it	QPlainTextEdit

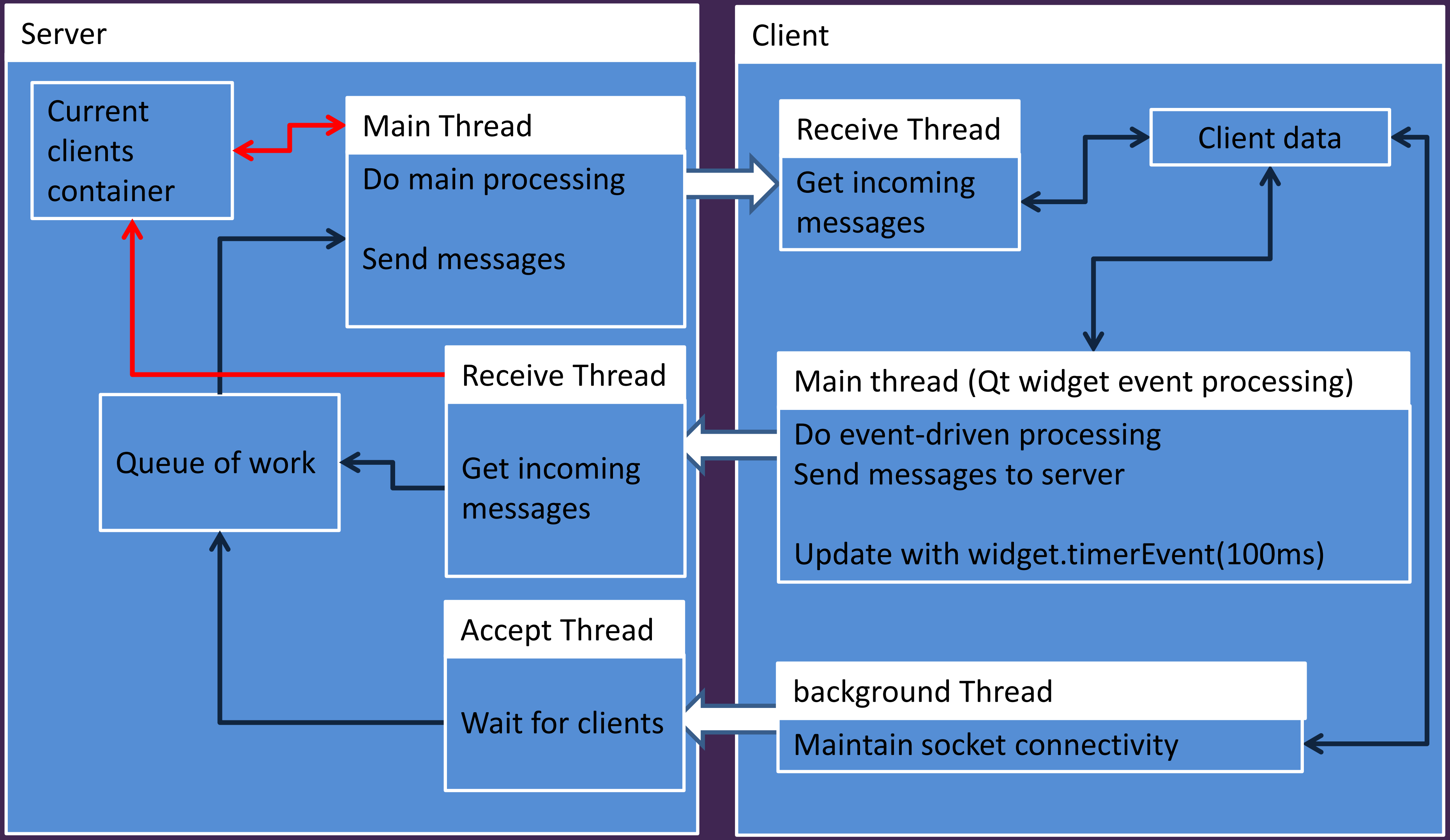
lineEdit : QLineEdit

Property	Value
QObject	
objectName	lineEdit
QWidget	
enabled	☑
geometry	[(20, 410), 621 x 41]
sizePolicy	[Expanding, Fixed, 0, 0]
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	Inherited
font	A [MS Shell Dlg 2, 10]
cursor	IBeam
mouseTracking	☑

- Chat service key implementational issues

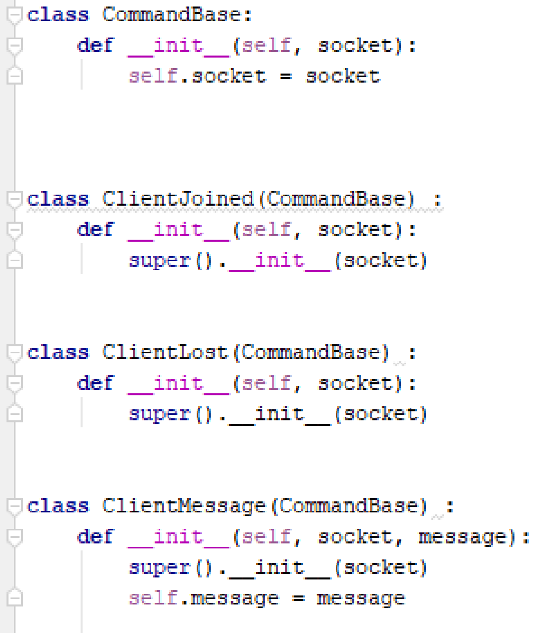
- Chat service key implementational issues
  - How do we implement this as threaded apps?

- Chat service key implementational issues
  - How do we implement this as a threaded app?



- Chat service key implementational issues
  - How do we implement this as a threaded app?
    - For the server, it makes sense to treat the app as all the threads pushing data into the main thread through a queue
      - This gives the server different types of ‘work’ to do which can be specialised by data and function

- Chat service key implementational issues
  - How do we implement this as a threaded app?
    - For the server, it makes sense to treat the app as all the threads pushing data into the main thread through a queue
      - This gives the server different types of ‘work’ to do which can be specialised by data and function



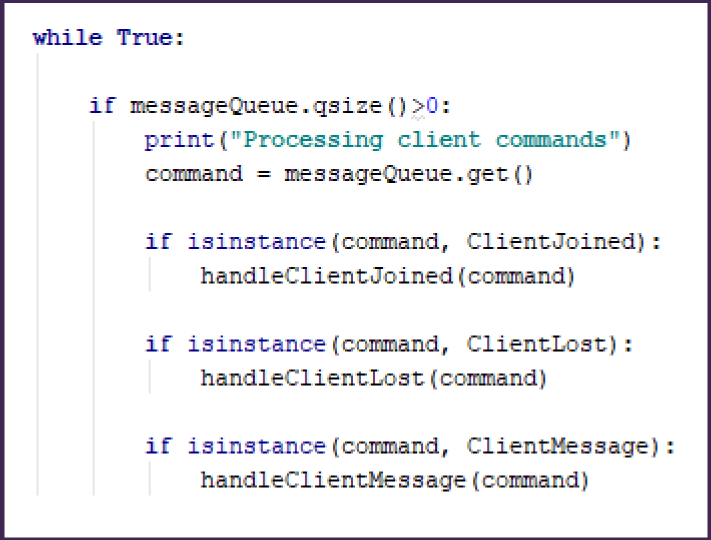
```
class CommandBase:
    def __init__(self, socket):
        self.socket = socket



class ClientJoined(CommandBase) :
    def __init__(self, socket):
        super().__init__(socket)


class ClientLost(CommandBase) :
    def __init__(self, socket):
        super().__init__(socket)


class ClientMessage(CommandBase) :
    def __init__(self, socket, message):
        super().__init__(socket)
        self.message = message
```



```
while True:

    if messageQueue.qsize()>0:
        print("Processing client commands")
        command = messageQueue.get()

        if isinstance(command, ClientJoined):
            handleClientJoined(command)

        if isinstance(command, ClientLost):
            handleClientLost(command)

        if isinstance(command, ClientMessage):
            handleClientMessage(command)
```

- Workshop
  - You will be able to explore and extend the functionality of the chat service in this weeks’ workshop

- Questions