Laboratory 6
Car-Sharing (1/2)

Agenda

Introduction

The topic we want to discuss today is Telematics.

Telecommuncation Informatics is the technology of sending, receiving and storing information via telecommunication devices in conjunction with affecting control on remote objects

At a very high level telematic systems comprise:

- A processing unit connected to the local system's control interface – serial port(s), data bus(es)
- A communication device (usually a cellular network device)
- A remote command and control center

Telematic systems have a large domain of applicability – automotive, utility distribution (gas/electric), road safety, intelligent transportation, military, etc. In Continental we implement Telematics solutions particularly designed for the automotive domain. Telematics Control Units enable cars to be connected to the outside world. Common applications offered through telematics systems are:

- Emergency warning systems: automated crash notification, emergency calls, roadside assistance
- Door locking/unlocking and stolen vehicle tracking
- Remote diagnostics to enable early detection of failures

The application we will implement today will be focused on another application made possible by telematics: carsharing.

Before we continue, please watch this video for a brief introduction :
https://www.youtube.com/watch?v=Q8EvQrKwjxY

Carsharing refers to automobile rental services intended to substitute for private vehicle ownership. Their vehicles are generally located in residential areas, priced by the hour, with convenient (generally automated) pick-up and drop-off procedures. This makes occasional use of an automobile affordable, even for low-income households, and by allowing households to reduce their vehicle ownership it provides an incentive to reduce driving and rely more on alternative modes. Carsharing services are typically developed by private companies, cooperatives or by software applications that facilitate peer-to-peer carsharing.

We are particularry interested in peer-to-peer carsharing. This model allows the user to rent a vehicle in a fully automated way by use of his smartphone, without the involvement of private companies. The process usually involves a mobile application which provides a communication environment for car owners and people who want to rent a car.
With the integration of Telematics modules in modern cars, there is no need for the client to meet the owner of the car because unlocking and starting the car can be done with the use of the mobile phone.

Preconditions

Software to install:

Python : https://www.python.org/downloads/windows/ (please select Latest Python 3 Release)

Pycharm: https://www.jetbrains.com/pycharm/download/#section=windows

Already available items:

a) a script called carsharingClient.py which implements a TCP client

b) a script called carsharingServer.py which implements a TCP server

c) a json file cars.json containing list of authorized cars

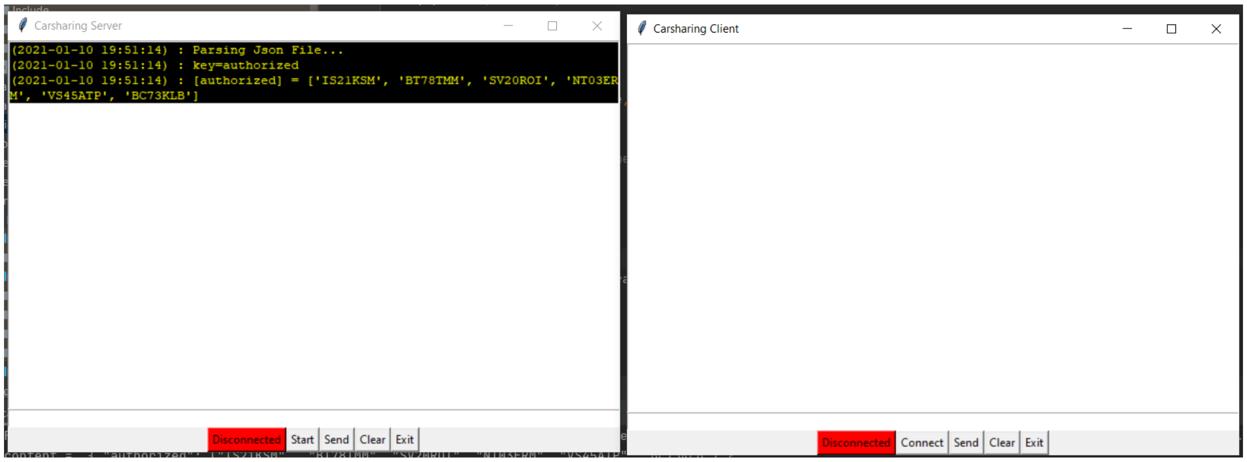
Hints

In order to run client/server, you can run the following command.

```
py carsharingClient.py 127.0.0.1 -p 65432

py carsharingServer.py 127.0.0.1 -p 65432
```

Address and port are specified as command line parameters. If both scrips are run successfully you should be able to see two windows as shown in the image below.



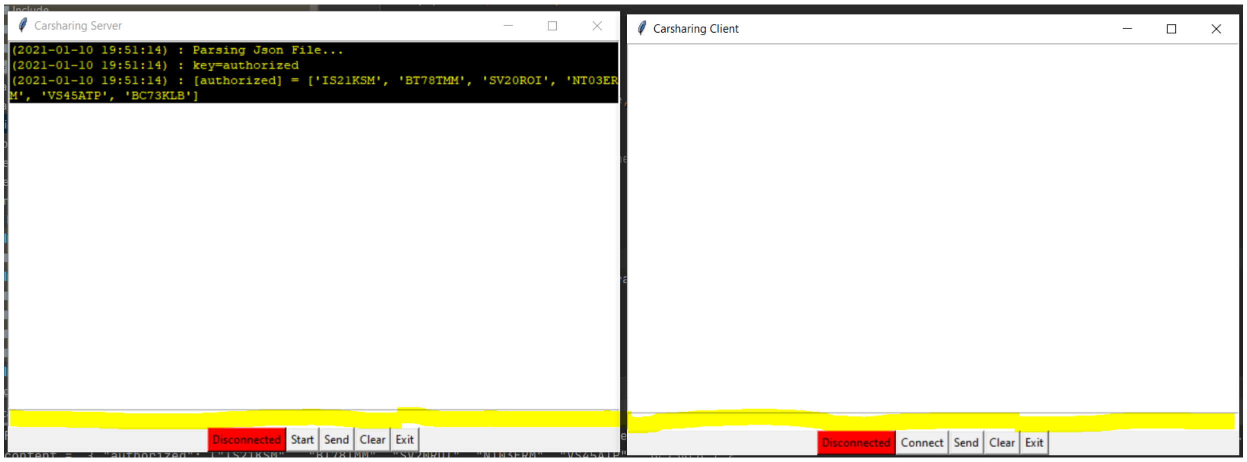
```
(2021-01-10 19:51:14) : Parsing Json File...
(2021-01-10 19:51:14) : key=authorized
(2021-01-10 19:51:14) : [authorized] = ['IS21KSM', 'BT78TMM', 'SV20ROI', 'NT03ER
M', 'VS45ATP', 'BC73KLB']
```

Figure 1: Client & Server – graphical view

Both the client and the server have the following elements:

- text box: all messages server/client are printed

- input box: collects user input



```
(2021-01-10 19:51:14) : Parsing Json File...
(2021-01-10 19:51:14) : key=authorized
(2021-01-10 19:51:14) : [authorized] = ['IS21KSM', 'BT78TMM', 'SV20ROI', 'NT03ER
M', 'VS45ATP', 'BC73KLB']
```

Figure 2: Input box (highlighted with yellow) → For commands entered by user

- buttons:

    connect (client only): client connects to the server

    start (server only): server starts listening for tcp connection

    send: attempts sending the data from input box to server/client

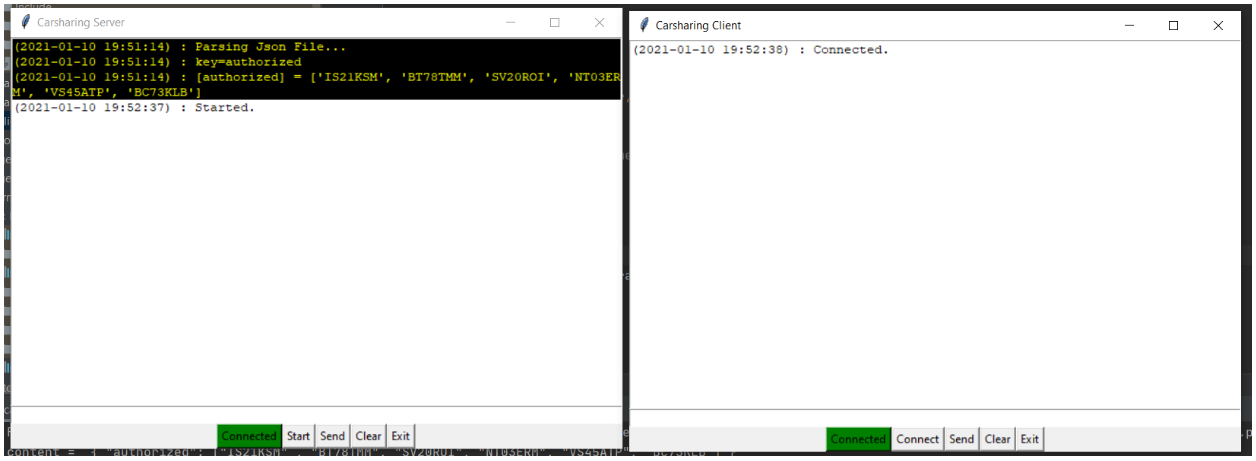
    clear: clears text box (messages shown)

    exit: exit application

Steps to connect client to server:

1. Server -> Start
2. Client -> Connect

You should see the red button turning green.



Carsharing Server

```
(2021-01-10 19:51:14) : Parsing Json File...
(2021-01-10 19:51:14) : key=authorized
(2021-01-10 19:51:14) : [authorized] = ['IS21KSM', 'BT78TMM', 'SV20ROI', 'NT03ER
M', 'VS45ATP', 'BC73KLB']
(2021-01-10 19:52:37) : Started.
```

Carsharing Client

```
(2021-01-10 19:52:38) : Connected.
```

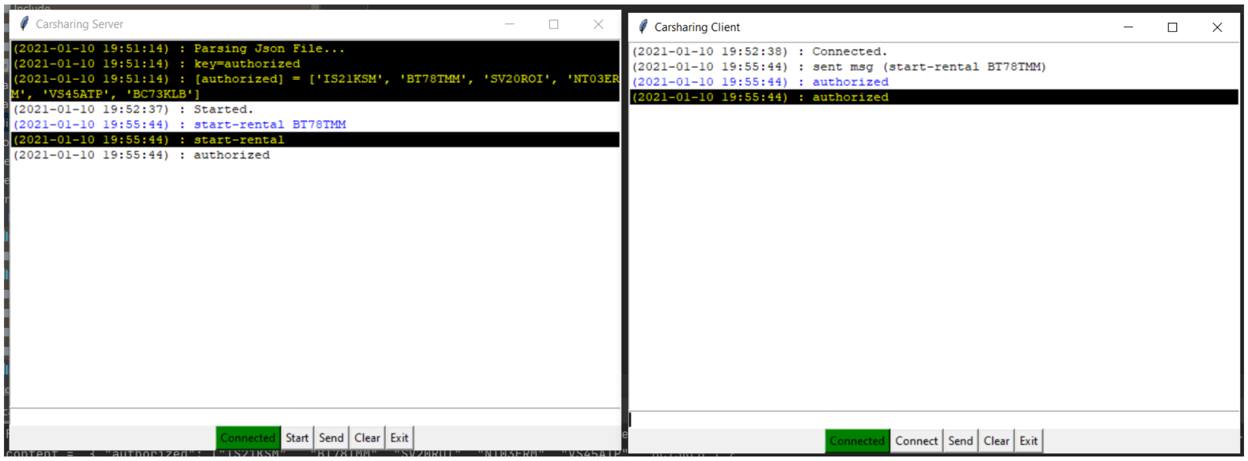
Figure 3: Client & Server – successful connection

At startup, the server is parsing the file clients.json and saves data into class member info:

```
{ "authorized": ["IS21KSM" , "BT78TMM", "SV20ROI", "NT03ERM", "VS45ATP",
"BC73KLB"] }
```

Please be aware of:

1. Received messages are colored with blue (in client / server).



Carsharing Server

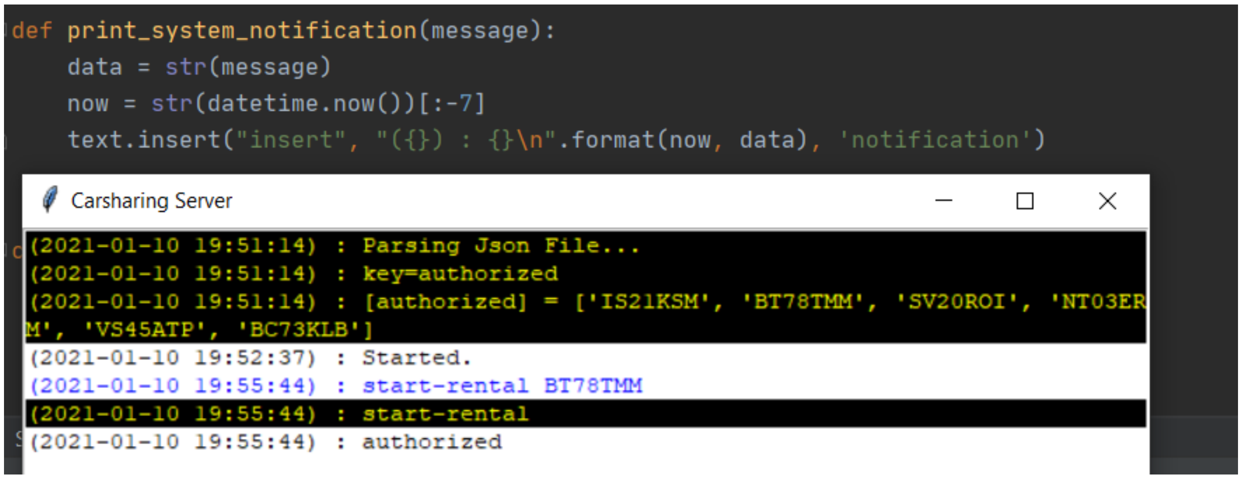
```
(2021-01-10 19:51:14) : Parsing Json File...
(2021-01-10 19:51:14) : key=authorized
(2021-01-10 19:51:14) : [authorized] = ['IS21KSM', 'BT78TMM', 'SV20ROI', 'NT03ER
M', 'VS45ATP', 'BC73KLB']
(2021-01-10 19:52:37) : Started.
(2021-01-10 19:55:44) : start-rental BT78TMM
(2021-01-10 19:55:44) : start-rental
(2021-01-10 19:55:44) : authorized
```

Carsharing Client

```
(2021-01-10 19:52:38) : Connected.
(2021-01-10 19:55:44) : sent msg (start-rental BT78TMM)
(2021-01-10 19:55:44) : authorized
(2021-01-10 19:55:44) : authorized
```

Figure 4: Client sending request to start rental to server

2. Notifications are colored with a black background.



```
def print_system_notification(message):
    data = str(message)
    now = str(datetime.now())[:-7]
    text.insert("insert", "({}) : {}\n".format(now, data), 'notification')
```

```
(2021-01-10 19:51:14) : Parsing Json File...
(2021-01-10 19:51:14) : key=authorized
(2021-01-10 19:51:14) : [authorized] = ['IS21KSM', 'BT78TMM', 'SV20ROI', 'NT03ER
M', 'VS45ATP', 'BC73KLB']
(2021-01-10 19:52:37) : Started.
(2021-01-10 19:55:44) : start-rental BT78TMM
(2021-01-10 19:55:44) : start-rental
(2021-01-10 19:55:44) : authorized
```

Figure 5: Notifications – Display

Tasks

Exercise 1

Please fill in the necessary code in method parse_json from class Server.

The authorized car_ids (from the json file) must be added to list self.auth_clients.

Exercise 2

The client shall request “start-rental <car_id>” to server.

The server will check if <car_id> is authorized for rental. This will be done by checking whether the <car_id> is in self.auth_clients.

If it is, then server adds <car_id> to self. rentals_in_progress and sends “authorized” to client.

If it isn’t, then server sends “not-authorized” to client.

Note: You must only modify method handle_message from class Server:

```
if command.startswith('start-rental'):
    print_system_notification('start-rental')
```

Exercise 3

The client shall request “end-rental <car_id>” to server.

The server shall check if the car was rented. This will be done by checking whether the <car_id> is in self. rentals_in_progress.

If yes, then it will respond with “end-rental success”.

If not, then it will respond with “end-rental error not-found”.

Note: You must only modify method handle_message from class Server:

```
if command.startswith('end-rental'):
    print_system_notification('end-rental')
```

Evaluation

Score:

a) 2 point(s)

b) 2 point(s)

c) 2 point(s)