Syrian Arab Republic

Lattakia – Tishreen University

Department of Communication and electrical engineering

5 th , Network Programming : Homework No1

الجمهورية العربية السورية

وزارة التعليم العالي والبحث العلمي

جامعة تشرين

كلية الهندسة الميكانيكية والكهربائية

قسم هندسة الاتصالات والالكترونيات

السنة الخامسة

وظيفة مقرر برمجة الشبكات الثانية

Second Network Programming Homework

إعداد الطالبة: ربا إبراهيم أحمد

الرقم الجامعي:

2402

إشراف الدكتور المهندس:

مهند عيسى

Question 1: Bank ATM Application with TCP Server/Client and Multi-threading

```
import socket
import threading
import json

accounts = {
    'Robee1': {'password': 'password1', 'balance': 1500},
    'Robee2': {'password': 'password2', 'balance': 3000}
}

def handle_client(client_socket):
    while True:
        received_data = client_socket.recv(1024).decode('utf-8')
        if not received_data:
            break

        data = json.loads(received_data)
        command = data['command']
        username = data['username']
        password = data['password']

        if username in accounts and accounts[username]['password'] ==
password:
            if command == 'balance':
                response = {'status': 'success', 'balance':
accounts[username]['balance']}
            elif command == 'deposit':
                amount = data['amount']
                accounts[username]['balance'] += amount
                response = {'status': 'success', 'balance':
accounts[username]['balance']}
            elif command == 'withdraw':
                amount = data['amount']
                if accounts[username]['balance'] >= amount:
                    accounts[username]['balance'] -= amount
                    response = {'status': 'success', 'balance':
accounts[username]['balance']}
                else:
                    response = {'status': 'error', 'message':
'Insufficient funds'}
            else:
                response = {'status': 'error', 'message': 'Invalid
command'}
        else:
```

```
            response = {'status': 'error', 'message': 'Authentication
failed'}

        client_socket.send(json.dumps(response).encode('utf-8'))

    client_socket.close()

def start_server():
    server_ip = '0.0.0.0'
    server_port = 65432

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((server_ip, server_port))
    server_socket.listen()

    print(f'Server is listening on {server_ip}:{server_port}')

    while True:
        client_socket, client_address = server_socket.accept()
        print(f'Accepted connection from {client_address}')
        client_handler = threading.Thread(target=handle_client,
args=(client_socket,))
        client_handler.start()

if __name__ == '__main__':
    start_server()
```

شرح كود السيرفر: في البداية قمت باستدعاء جميع المودل التي أحتاجها في الكود وأهمها threading لجعل السيرفر يتعامل مع عدة عملاء بنفس الوقت. في السيرفر قمت بتخزين الحسابات ضمن قاموس له الاسم accounts واعتبرت أن كل حساب هو قاموس بحد ذاته بحيث يكون المفتاح هو اسم الحساب مثل Robee1 والقيم هي كلمة السر والنقود المودعة الكلية في الحساب.

حتى أتعامل مع العملاء Clients قمت بتعريف التابع handle_client() الذي يقوم بالتعامل مع اتصالات العملاء، طبعاً البارمتر المرر للتابع هو مقبس العميل client_socket نستقبل البيانات من العميل على أساس client_socket ونقوم بفك ترميزها بـ decode('utf-8'). لسهولة التعامل مع البيانات المستقبلة وكونها تتكون من عدة قيم قمت بتعريف متحول data وباستخدام json.loads حملت البيانات عن طريق إمرار البارمتر received_data وهو البيانات المستقبلة من العميل.

لسهولة التعامل وحتى أحصل على الأوامر واسم الحساب وكلمة المرور عرفت ثلاثة متغيرات هي: command وusername وpassword، قمت بالحصول على قيمها من المتغير data بإمرار المفتاح المناسب طبعاً.

ثم بعمليات تحقق من هوية المستخدم مثل:

if username in accounts and accounts[username]['password'] == password:

قمت بالتحقق من هوية كل عميل، وبدأت بعدها بإظهار الأوامر له.

في تابع بدء تشغيل السيرفر start_server قمت بتعيين عنوان IP للسيرفر لأجعله يتعامل مع أي عنوان ممكن، ومن ثم البورت (رقم المنفذ).

بواسطة:

client_handler = threading.Thread(target=handle_client,
args=(client_socket,)

client_handler.start()

قمت بإنشاء أوبجيكت من الكلاس Thread ومررت له التابع الذي يتعامل مع اتصالات العملاء وطبعا البارمتر المهم جداً مقبس العميل، ومن ثم بدأت التشغيل.

أكواد العملاء:

```
import socket
import json

def bank_client():
    server_ip = '127.0.0.1'
    server_port = 65432

    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((server_ip, server_port))

    username = input('Enter username: ')
    password = input('Enter password: ')

    while True:
        command = input('Enter command (balance/deposit/withdraw/exit): ')
```

```
        if command in ['deposit', 'withdraw']:
            amount = float(input('Enter amount: '))
        else:
            amount = 0

        data = {
            'command': command,
            'username': username,
            'password': password,
            'amount': amount
        }
        client_socket.send(json.dumps(data).encode('utf-8'))

        response = json.loads(client_socket.recv(1024).decode('utf-8'))
        if response['status'] == 'success':
            print(f"Balance: {response['balance']}")
        else:
            print(f"Error: {response['message']}")

        if command == 'exit':
            break

    client_socket.close()

if __name__ == '__main__':
    bank_client()
```

قمت في كود العميل للبنك: بإجراءات مثل إدخال الاسم وكلمة المرور للتحقق من كون العميل فعلاً موجود أم لا وهل يحق له القيام بالعمليات.

ومن ثم عن طريق:

```
data = {
            'command': command,
            'username': username,
            'password': password,
            'amount': amount
        }
```

قمت بعرض الأوامر المتاحة للعميل.

Question 2: contact book program

هذا البرنامج عبارة عن قائمة اتصالات الكترونية (دفتر للأسماء والأرقام المقابلة):

```
contacts = {}

def display_menu():
    print("nContact Book Menu:")
    print("1. Add Contact")
    print("2. Search Contact")
    print("3. Display All Contacts")
    print("4. Exit")


def add_contact():
    name = input("Enter the contact name: ")
    phone = input("Enter the contact phone number: ")
    if name in contacts:
        print("This contact name already exists. Please use a different
name.")
    else:
        contacts[name] = phone
        print("Contact added successfully!")


def search_contact():
    name = input("Enter the contact name: ")
    if name in contacts:
        print("Phone number:", contacts[name])
    else:
        print("Contact not found!")


def display_contacts():
    if len(contacts) == 0:
        print("No contacts found!")
    else:
        print("All contacts:")
        for name, phone in contacts.items():
            print("Name:", name, "| Phone number:", phone)


def main():
    while True:
        display_menu()
        choice = input("Enter your choice: ")
```

```
        if choice == "1":
            add_contact()
        elif choice == "2":
            search_contact()
        elif choice == "3":
            display_contacts()
        elif choice == "4":
            print("Exiting the program...")
            break
        else:
            print("Invalid choice! Try again.")


if __name__ == "__main__":
    main()
```

اعتمدت في هذا الكود على ما يلي:

نحن نعلم أن كل دفتر أرقام يتكون من أسماء الأشخاص وأرقامهم المقابلة، لذلك كان الخيار الأفضل هو تعريف قاموس له الاسم contacts حتى أخزن هذه البيانات. حتى أجعل التعامل ديناميكاً ولسهولة تنظيم الكود قمت بتعريف تابع صغير يقوم بعمل منفصل ويتعامل مع البيانات بشكل مختلف.

في البداية التابع display_menu: هذا التابع لعرض الخيارات المتاحة لمستخدم البرنامج مثل:

إضافة جهة اتصال، بحث عن جهة اتصال، عرض كل جهات الاتصال، والخروج من البرنامج وإنهاء التنفيذ.

لذلك من الأفضل بناء توابع تقوم بالعمل المطلوب: التابع add_contact يقوم التابع عند استدعائه بطلب اسم المستخدم ورقمه طبعاً. من الأفضل التعامل مه هذه البيانات على أنها سلاسل محرفية تجنباً لحدوث الأخطاء.

التابع search_contact يقوم بالبحث عن المستخدم من خلال اسمه.

التابع display_contacts يقوم بعرض جهات الاتصال.

لإنهاء تنفيذ البرنامج من التابع: main

```
elif choice == "4":
            print("Exiting the program...")
            break
        else:
            print("Invalid choice! Try again.")
```

عندما يدخل المستخدم 4 الموافق ل exit عن طريق break أنهي تنفيذ البرنامج.