الجمهورية العربية السورية

وزارة التعليم العالي والبحث العملي

جامعة تشرين – اللاذقية

كلية الهمك

قسم هندسة الاتصالات والالكترونيات

السنة الخامسة

Second Network Programming Homework

إعداد الطالبتين:

ضحى خالد زنجرلي 2431

دعاء محمد جمعه زينب 2499

إشراف الدكتور المهندس:

مهند عيسى

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

```
import socket
import threading

HOST = '0.0.0.0'
PORT = 9999

accounts = {
    "Doha": 1050,
    "Doaa": 5050
}

def handle_client(conn, addr):
    print(f"Connected by {addr}")
    while True:
        data = conn.recv(1024).decode()
        if not data:
            break

        account_number, operation, amount = data.split()
        if account_number not in accounts:
            conn.sendall("Invalid account number".encode())
            continue

        try:
            amount = float(amount)
        except ValueError:
            conn.sendall("Invalid amount".encode())
            continue

        if operation == "check_balance":
            balance = accounts[account_number]
            conn.sendall(f"Your balance is: {balance}".encode())
        elif operation == "deposit":
            accounts[account_number] += amount
            conn.sendall(f"Deposit successful. New balance:
{accounts[account_number]}".encode())
        elif operation == "withdraw":
            if accounts[account_number] < amount:
                conn.sendall("Insufficient funds".encode())
            else:
                accounts[account_number] -= amount
                conn.sendall(f"Withdrawal successful. New balance:
{accounts[account_number]}".encode())
```

```
        else:
            conn.sendall("Invalid operation".encode())

    conn.close()
    print(f"Client {addr} disconnected")

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    print(f"Server listening on {HOST}:{PORT}")
    while True:
        conn, addr = s.accept()
        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start()
```

كود السيرفر: حتى نجعل السيرفر يخدم عدد كبير من المستخدمين بنفس الوقت يجب الاستفادة من المودل threading، تم تعيين IP السيرفر على 0.0.0.0 من أجل نخديم أي عنوان بالشبكة، ورقم المنفذ على 8000. خزنت الحسابات في dictionary له الاسم accounts، اعتمدت على رقم الحساب في التخزين بحيث جعلت رقم الحساب هو المفتاح والمبلغ المالي هو القيمة المقابلة.

بتعرف التابع handle_client(conn, addr) أتعامل مع اتصالات العملاء بحيث مررت له سوكيت العميل وهو البارمتر conn وعنوان العميل addr. استقبل معلومات العميل باستخدام:

```
data = conn.recv(1024).decode()
```

وثم عن طريق تعريف المتحولات رقم الحساب ونوع العملية المرادة وإجمالي القيمة المضافة أو المسحوبة أستطيع فصل هذه البيانات باستخدام data.split().

من ثم حسب العملية التي يرد القيام بها العميل أستطيع تنفيذ ما يريد كما يلي:

```
try:
        amount = float(amount)
    except ValueError:
        conn.sendall("Invalid amount".encode())
        continue
```

```
        if operation == "check_balance":
            balance = accounts[account_number]
            conn.sendall(f"Your balance is: {balance}".encode())
        elif operation == "deposit":
            accounts[account_number] += amount
                        conn.sendall(f"Deposit successful. New balance: {accounts[account_number]}".encode())
        elif operation == "withdraw":
            if accounts[account_number] < amount:
                conn.sendall("Insufficient funds".encode())
            else:
                accounts[account_number] -= amount
                        conn.sendall(f"Withdrawal successful. New balance: {accounts[account_number]}".encode())
        else:
            conn.sendall("Invalid operation".encode())


    conn.close()
    print(f"Client {addr} disconnected")
```

بإنشاء الأوبجيكت (الغرض) thread أستطيع جعل السيرفر يتعامل مع المستخدمين بنفس الوقت بإمرار التابع target=handle_client ومن ثم args=(conn, addr).

تشغيل السيرفر:

```
Server listening on 0.0.0.0:9999
Connected by ('127.0.0.1', 2064)
Client ('127.0.0.1', 2064) disconnected
Connected by ('127.0.0.1', 2066)
Client ('127.0.0.1', 2066) disconnected
```

```
Enter your account Name: Doha
Enter operation (check_balance, deposit, withdraw): deposit
Enter amount: 1230
Deposit successful. New balance: 2280.0
Enter your account Name: Doha
Enter operation (check_balance, deposit, withdraw): check_balance
Your balance is: 2280.0
Enter your account Name:
```

```python
import socket

HOST = '127.0.0.1'
PORT = 9999

while True:
    account_number = input("Enter your account Name: ")
    operation = input("Enter operation (check_balance, deposit, withdraw):
")
    if operation in ("deposit", "withdraw"):
        amount = float(input("Enter amount: "))
    data = f"{account_number} {operation} {amount}".encode()

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT))
        s.sendall(data)
        response = s.recv(1024).decode()
        print(response)
```

Question 2: Simple Calendar App using Kivy:

```
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.label import Label
from kivy.uix.gridlayout import GridLayout
from datetime import datetime
import calendar

class DayOfWeekApp(App):
    def build(self):
        root_layout = BoxLayout(orientation='vertical')

        day = datetime.today().strftime('%A')
        year = datetime.today().year
        calendar_month = datetime.today().strftime('%B %Y')

        day_label = Label(text=f"Today is {day}", font_size=30)
        year_label = Label(text=f"Current Year: {year}", font_size=22)
        calendar_label = Label(text=f"Calendar: {calendar_month}",
font_size=22)

        root_layout.add_widget(day_label)
        root_layout.add_widget(year_label)
        root_layout.add_widget(calendar_label)

        calendar_table = self.create_calendar_table()
        root_layout.add_widget(calendar_table)

        return root_layout

    def create_calendar_table(self):
        calendar_table = GridLayout(cols=7, spacing=5, padding=10,
size_hint_y=None, height=400)
        calendar_table.bind(minimum_height=calendar_table.setter('height')
)

        day_names = ['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat']
        for day in day_names:
            day_label = Label(text=day, size_hint_y=None, height=40)
            calendar_table.add_widget(day_label)

        today = datetime.today()
        calendar_matrix = calendar.monthcalendar(today.year, today.month)
```

```
        for week in calendar_matrix:
            for day in week:
                if day == 0:
                    day_label = Label(text='', size_hint_y=None,
height=40)
                else:
                    day_label = Label(text=str(day), size_hint_y=None,
height=40)
                calendar_table.add_widget(day_label)

        return calendar_table

if __name__ == '__main__':
    DayOfWeekApp().run()
```

هذا الكود هو تطبيق بسيط باستخدام مكتبة Kivy لعرض يوم الأسبوع وتقويم شهري بتنسيق جميل:

1. from kivy.app import App: يستورد الكائن الرئيسي لتطبيق Kivy.

2. from kivy.uix.boxlayout import BoxLayout: يستورد تخطيط مربعي لترتيب العناصر في صفوف وأعمدة.

3. from kivy.uix.label import Label: يستورد ويستخدم العنصر Label لعرض النص.

4. from kivy.uix.gridlayout import GridLayout: يستورد تخطيط الجدول لترتيب العناصر في شبكة.

5. from datetime import datetime: يستورد وقت التشغيل لاستخدام التواريخ والأوقات.

6. import calendar: يستورد مكتبة التقويم لإنشاء التقويمات.

الكلاس DayOfWeekApp :

7. ::class DayOfWeekApp(App) يعرف الكلاس الرئيسي للتطبيق.

8. ::def build(self) يعرف الوظيفة التي يتم استدعاؤها لإنشاء وتكوين واجهة المستخدم.

9. :root_layout = BoxLayout(orientation='vertical') ينشئ تخطيطًا مربعيًا لترتيب العناصر بشكل رأسي.

10 .تحديد معلومات اليوم والسنة والشهر الحالي.

11 .إنشاء عناصر Label لعرض معلومات اليوم والسنة والشهر.

12 .إضافة العناصر إلى التخطيط الرئيسي.

13: calendar_table = self.create_calendar_table() .ينشئ جدول التقويم باستخدام الميثود create_calendar_table.

14 .إضافة جدول التقويم إلى التخطيط الرئيسي.

15: return root_layout .يعيد التخطيط الرئيسي كجزء من واجهة المستخدم.

الآن لنشرح الوظيفة:create_calendar_table

16 ::def create_calendar_table(self) .تعريف وظيفة لإنشاء جدول التقويم.

. calendar_table = GridLayout(cols=7, spacing=5, padding=10, 17
size_hint_y=None, height=400): ينشئ جدولًا مكونًا من 7 أعمدة ويحدد البعد والتباعد.

. calendar_table.bind(minimum_height=calendar_table.setter('height')): 18
يربط ارتفاع الجدول بحد أدنى ممكن للحفاظ على الحجم.

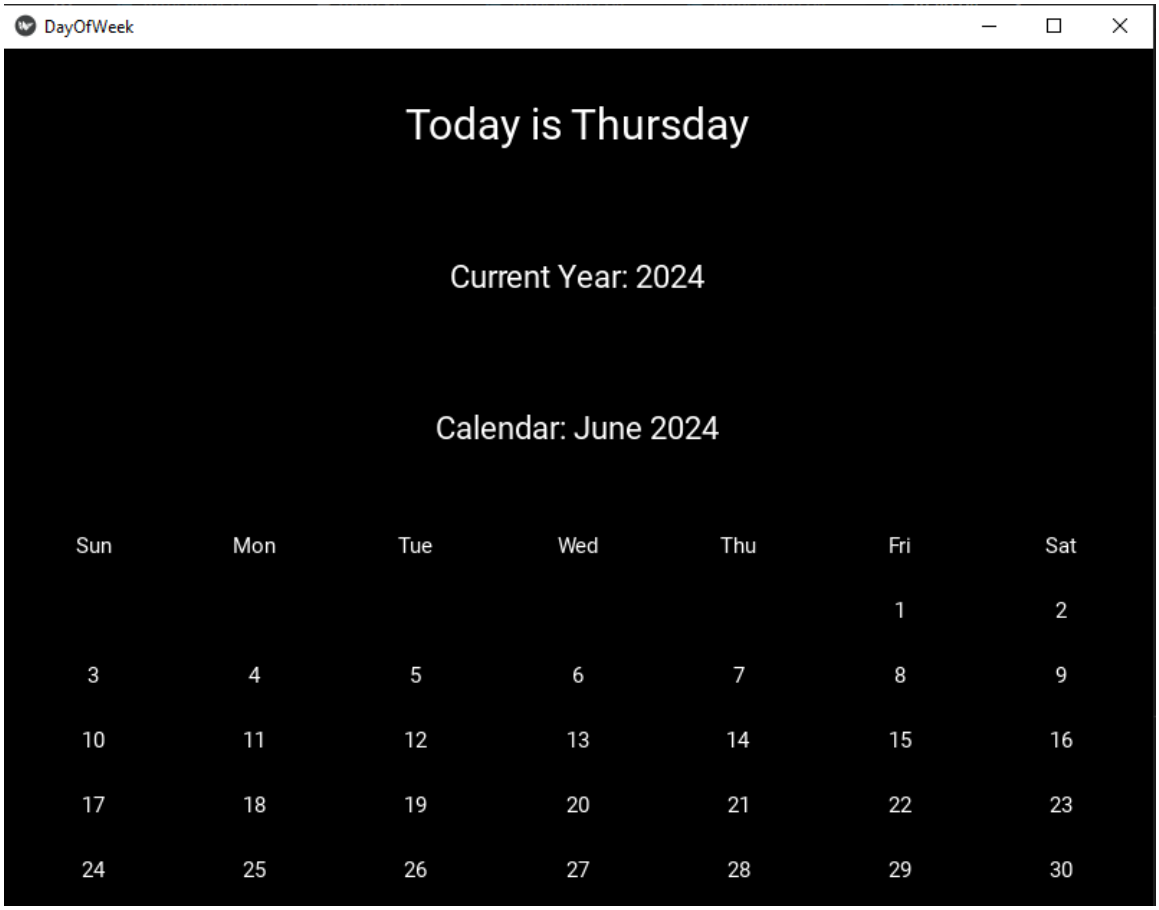
19 .إنشاء تسميات الأيام (Sun, Mon, Tue إلخ وإضافتها إلى الجدول.

20 .إنشاء مصفوفة تقويم للشهر الحالي وملء الجدول بأيام الشهر الحالي.

21 :return calendar_table يعيد الجدول كجزء من واجهة المستخدم.

22 :if __name__ == '__main__'
يتحقق مما إذا كان البرنامج يشغل مباشرة من الملف أو مستوردًا كمكتبة.

23 ()DayOfWeekApp().run: تشغيل التطبيق.


DayOfWeek
Today is Thursday
Current Year: 2024
Calendar: June 2024
Sun Mon Tue Wed Thu Fri Sat
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30