Міністерство освіти і науки України
Національний університет «Львівська політехніка»

Кафедра ЕОМ

Звіт

до лабораторної роботи № 9

З дисципліни: «Кросплатформенні засоби програмування»

На тему: **«ОСНОВИ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ У PYTHON»**

Варіант 16

Виконав: ст.гр. КІ-301
Онисько М.М.
Прийняв: доц.
Майдан М.В.

Львів 2023

Мета: оволодіти навиками реалізації парадигм об’єктно-орієнтованого програмування використовуючи засоби мови Python.

Завдання:
1. Написати та налагодити програму на мові Python згідно варіанту. Програма має
задовольняти наступним вимогам:
• класи програми мають розміщуватися в окремих модулях в одному пакеті;
• точка входу в програму (main) має бути в окремому модулі;
• мають бути реалізовані базовий і похідний класи предметної області згідно варіанту;
• програма має містити коментарі.
2. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її
виконання та фрагменту згенерованої документації та завантажити його у ВНС.
4. Дати відповідь на контрольні запитання.

Варіант 16:

Базовий клас:

16. Аудіоплеєр

Похідний клас:

16. Диктофон

Виконання роботи:

Код програми:

AudioPlayerApp.py

```
from AudioPlayer import AudioPlayer, Playlist, Song, Battery
from AudioRecorder import AudioRecorder

audioplayer = AudioPlayer("IPhone", "IPod 4", 99)
playlistOfBroke = Playlist("Плейлист броука")
song1 = Song("Скрябін - Говорили і курили", "3:31")
song2 = Song("Kanye West - Ok Ok", "3:24")
song3 = Song("The Prodigy - Breathe", "5:36")
playlistOfBroke.addSong(song1)
playlistOfBroke.addSong(song2)
playlistOfBroke.addSong(song3)
print("1.додати плейлист броук\n")
audioplayer.addPlaylist(playlistOfBroke)
playlistNirvana = Playlist("Nirvana")
```

```
song4 = Song("Nirvana - Heart-Shaped Box", "4:41")
song5 = Song("Nirvana - Something In The Way", "3:52")
song6 = Song("Nirvana - Smells Like Teen Spirit", "5:01")
playlistNirvana.addSong(song4)
playlistNirvana.addSong(song5)
playlistNirvana.addSong(song6)
print("2.додати плейлист нірвана\n")
audioplayer.addPlaylist(playlistNirvana)
print("3.вибрати плейлист броук\n")
audioplayer.setPlaylist(playlistOfBroke)
print("4.вибір пісні2\n")
audioplayer.setSong(song2)
print("5.кнопка плей\n")
audioplayer.playback()
print("6.кнопка плей\n")
audioplayer.playback()
print("7.інформація плеєра\n")
audioplayer.displayInfo()
print("8.список пісень\n")
audioplayer.displaySongs()
print("9.вибрати плейлист нірвана\n")
audioplayer.setPlaylist(playlistNirvana)
print("10.список пісень\n")
audioplayer.displaySongs()
print("11.видалення пісні\n")
audioplayer.removeSong(song4)
print("12.список пісень\n")
audioplayer.displaySongs()
print("13.видалення плейлиста\n")
audioplayer.removePlaylist(playlistNirvana)
print("14.вибрати плейлист броук\n")
audioplayer.setPlaylist(playlistOfBroke)
print("15.вибір пісні1\n")
audioplayer.setSong(song1)
print("16.інформація плеєра\n")
audioplayer.displayInfo()
print("17.Наступна пісня\n")
audioplayer.skipToNextSong()
print("18.Наступна пісня\n")
audioplayer.skipToNextSong()
print("19.Наступна пісня\n")
audioplayer.skipToNextSong()
print("20.Наступна пісня\n")
audioplayer.skipToNextSong()
print("21.Попередня пісня\n")
audioplayer.skipToPreviousSong()
print("22.Попередня пісня\n")
audioplayer.skipToPreviousSong()
print("23.Попередня пісня\n")
audioplayer.skipToPreviousSong()
print("24.Попередня пісня\n")
audioplayer.skipToPreviousSong()
print("25.Попередня пісня\n")
```

```
audioplayer.skipToPreviousSong()
audioplayer.isCharging()
audioplayer.chargeBattery()
audioplayer.isCharging()
audioplayer.chargeBattery()
audioplayer.isCharging()
audioplayer.dischargeBattery(10)
audioplayer.dischargeBattery(100)
audioplayer.getCurrentLevel()
audiorecorder = AudioRecorder("IPhone", "IPod 4", 99)
song1 = Song("Zapys 1")
audiorecorder.startRecording(song1)
audiorecorder.waitingAbout(song1, 130)
audiorecorder.stopRecording(song1)
audiorecorder.saveRecording(song1)
song2 = Song("Zapys 2")
audiorecorder.startRecording(song2)
audiorecorder.waitingAbout(song2, 130)
audiorecorder.stopRecording(song2)
audiorecorder.saveRecording(song2)
audiorecorder.displayInfo()
audiorecorder.displaySongs()
audiorecorder.connectHeadset()
audiorecorder.disconnectHeadset()
```

AudioPlayer.py

```
class AudioPlayer:
    log_file = open("log_file.txt", "w", encoding="utf-8")

    def __init__(self, brand="невідомий", model="невідома", charge=100):
        self.brand = brand
        self.model = model
        self.playlists = []
        self.playlist = Playlist()
        self.battery = Battery(charge)
        self.song = Song()


    def write_to_log(self, message):
    # Запишіть повідомлення у файл
        print(message, file=self.log_file)
        self.log_file.flush()

    def addPlaylist(self, playlist):
        self.playlists.append(playlist)
        print("Додано плейлист " + playlist.playlistName + "\n")
        self.write_to_log("Додано плейлист " + playlist.playlistName + "\n")
```

```
    def setPlaylist(self, playlist):
        for tmpPlaylist in self.playlists:
            if tmpPlaylist == playlist:
                self.playlist = playlist
        print("Вибрано плейлист " + playlist.playlistName + "\n")
        self.write_to_log("Вибрано плейлист " + playlist.playlistName + "\n")

    def setSong(self, song):
        for tmpSong in self.playlist.songs:
            if tmpSong == song:
                self.song = song
        print("Вибрано пісню " + song.songName + "\n")
        self.write_to_log("Вибрано пісню " + song.songName + "\n")

    def playback(self):
        if self.song.playback():
            print("Грає пісня " + self.song.songName + "\n")
            self.write_to_log("Грає пісня " + self.song.songName + "\n")
        else:
            print("Призупинена пісня " + self.song.songName + "\n")
            self.write_to_log("Призупинена пісня " + self.song.songName + "\n")

    def removePlaylist(self, playlist):
        i = self.playlists.index(playlist)
        self.playlists.pop(i)
        print("Видалено плейлист " + playlist.playlistName + "\n")
        self.write_to_log("Видалено плейлист " + playlist.playlistName + "\n")

    def skipToNextSong(self):
        i = self.playlist.songs.index(self.song)
        if 0 <= i < len(self.playlist.songs) - 1:
            self.song = self.playlist.songs[i + 1]
        else:
            self.song = self.playlist.songs[0]
        print("Тепер грає пісня: " + self.song.songName)
        self.write_to_log("Тепер грає пісня: " + self.song.songName)

    def skipToPreviousSong(self):
        i = self.playlist.songs.index(self.song)
        if i > 0:
            self.song = self.playlist.songs[i - 1]
        elif i == 0:
            lastIndex = len(self.playlist.songs) - 1
            self.song = self.playlist.songs[lastIndex]
        print("Тепер грає пісня: " + self.song.songName)
        self.write_to_log("Тепер грає пісня: " + self.song.songName)

    def displayInfo(self):
        print("Марка аудіопрогравача: " + self.brand +
              "\nМодель аудіопрогравача: " + self.model +
              "\nПлейлисти:\n ")
        self.write_to_log("Марка аудіопрогравача: " + self.brand +
              "\nМодель аудіопрогравача: " + self.model +
```

```
              "\nПлейлисти:\n ")
        for playlist in self.playlists:
            print(playlist.playlistName)
            self.write_to_log(playlist.playlistName)
        print("\n")
        self.write_to_log("\n")

    def displaySongs(self):
        print("Пісні плейлиста '" + self.playlist.playlistName + "':\n")
        for song in self.playlist.songs:
            print(song.songName)
            self.write_to_log(song.songName)
        print("\n")
        self.write_to_log("\n")

    def removeSong(self, song):
        self.playlist.removeSong(song)

    def getCurrentLevel(self):
        print("Поточна батарея:" + str(self.battery.getCurrentLevel()) + "\n")
        self.write_to_log("Поточна батарея:" +
str(self.battery.getCurrentLevel()) + "\n")

    def isCharging(self):
        if self.battery.isCharging():
            print("Батарея заряджається\n")
            self.write_to_log("Батарея заряджається\n")
        else:
            print("Батарея не заряджається\n")
            self.write_to_log("Батарея не заряджається\n")

    def chargeBattery(self):
        if self.battery.chargeBattery():
            if self.battery.getCurrentLevel() < 100:
                print("Батарея почала заряджатись\n")
                self.write_to_log("Батарея почала заряджатись\n")
            else:
                print("Батарея не потребує зарядки\n")
                self.write_to_log("Батарея не потребує зарядки\n")
        else:
            print("Батарея перестала заряджатись\n")
            self.write_to_log("Батарея перестала заряджатись\n")

    def dischargeBattery(self, energyConsumed):
        print("Заряд батареї зменшився")
        self.write_to_log("Заряд батареї зменшився")
        t = self.battery.dischargeBattery(energyConsumed)
        if t <= 0:
            print("Батарея повністю розряджена\n")
            self.write_to_log("Батарея повністю розряджена\n")
        else:
            print("Стан батареї:" + str(t) + "\n")
            self.write_to_log("Стан батареї:" + str(t) + "\n")
```

```
    def dispose(self):
    # Закрити файл при завершенні роботи
        self.log_file.close()



class Song:
    def __init__(self, songName="невідома", songLength="00:00"):
        self.songName = songName
        self.songLength = songLength
        self.isPlayed = False

    def playback(self):
        if not self.isPlayed:
            self.isPlayed = True
            return self.isPlayed
        else:
            self.isPlayed = False
            return self.isPlayed


class Playlist:
    def __init__(self, playlistName="невідома"):
        self.playlistName = playlistName
        self.songs = []

    def addSong(self, song):
        self.songs.append(song)

    def removeSong(self, song):
        self.songs.remove(song)


class Battery:
    def __init__(self, currentLevel=100):
        self.currentLevel = currentLevel
        self.charging = False
        self.energyConsumed = 0

    def getCurrentLevel(self):
        return self.currentLevel

    def isCharging(self):
        return self.charging

    def chargeBattery(self):
        if not self.charging:
            self.charging = True
            return self.charging
        else:
            self.charging = False
            return self.charging
```

```
    def dischargeBattery(self, energyConsumed):
        self.currentLevel -= energyConsumed
        if self.currentLevel <= 0:
            self.currentLevel = 0
        return self.currentLevel
```

AudioRecorder.py

```
from AudioPlayer import AudioPlayer, Playlist, Song

class AudioRecorder(AudioPlayer):
    def __init__(self, brand="невідомий", model="невідома", charge=100):
        super().__init__(brand, model, charge)
        self.fileName = ""
        self.time = 0
        self.playlistRec = Playlist("Recordered")
        self.check = False
        self.connectedHeadset = False

    @staticmethod
    def formatTime(time):
        minutes = time // 60
        seconds = time % 60
        return "{}:{:02d}".format(minutes, seconds)

    def startRecording(self, song):
        print("Початок запису аудіо")
        self.check = True

    def stopRecording(self, song):
        print("Зупинка запису аудіо")

    def saveRecording(self, song):
        self.check = False
        self.playlistRec.addSong(song)
        print("Пісню '" + song.songName + "' додано до папки 'Recordered'.")

    def waitingAbout(self, song, time):
        if self.check == False:
            print("НЕ МОЖНА")
        else:
            self.time = time
            formattedTime = self.formatTime(time)
            print("Пісня '" + song.songName + "' записується " + formattedTime)

    def displaySongs(self):
        print("Пісні плейлиста '" + self.playlistRec.playlistName + "':\n")
        for song in self.playlistRec.songs:
            print(song.songName)
```

```
        print("\n")

    def connectHeadset(self):
        if self.connectedHeadset == False:
            self.connectedHeadset = True
            print("Навушники підключено")
        else:
            print("Навушники вже підключено!")

    def disconnectHeadset(self):
        if self.connectedHeadset == True:
            self.connectedHeadset = False
            print("Навушники відключено")
        else:
            print("Навушники вже відключено!")
```

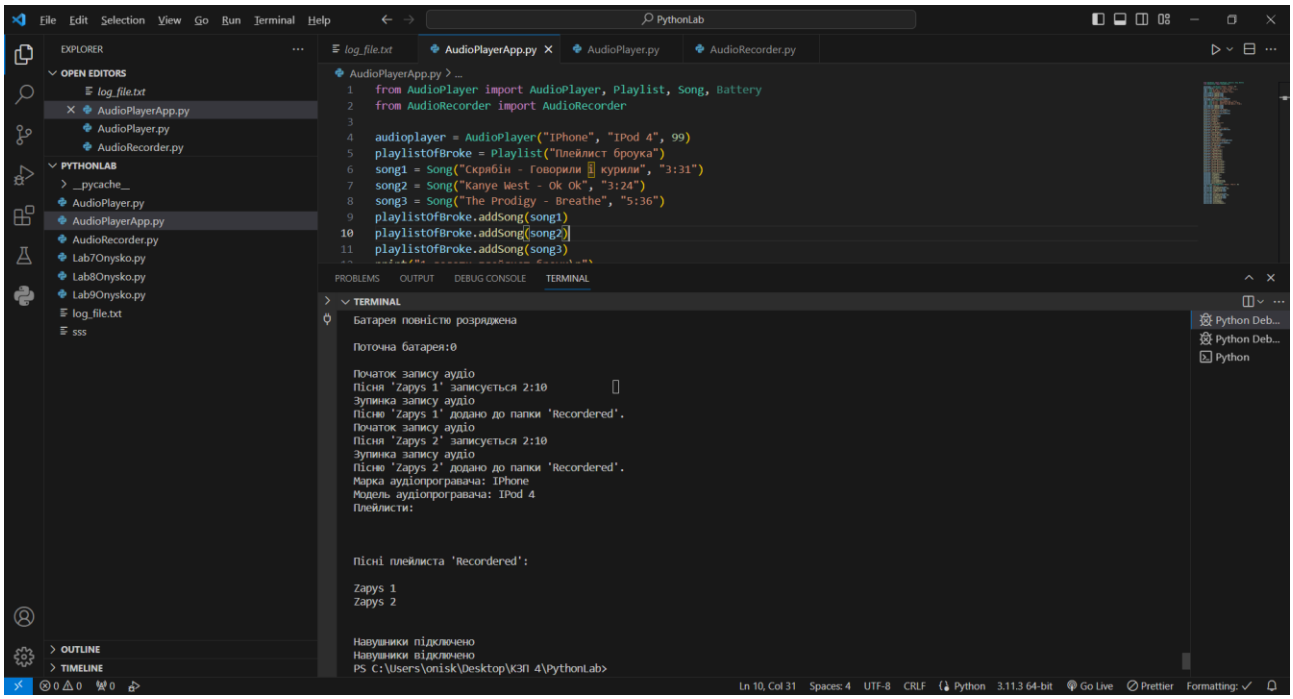

Рис.1 Код та вивід програми

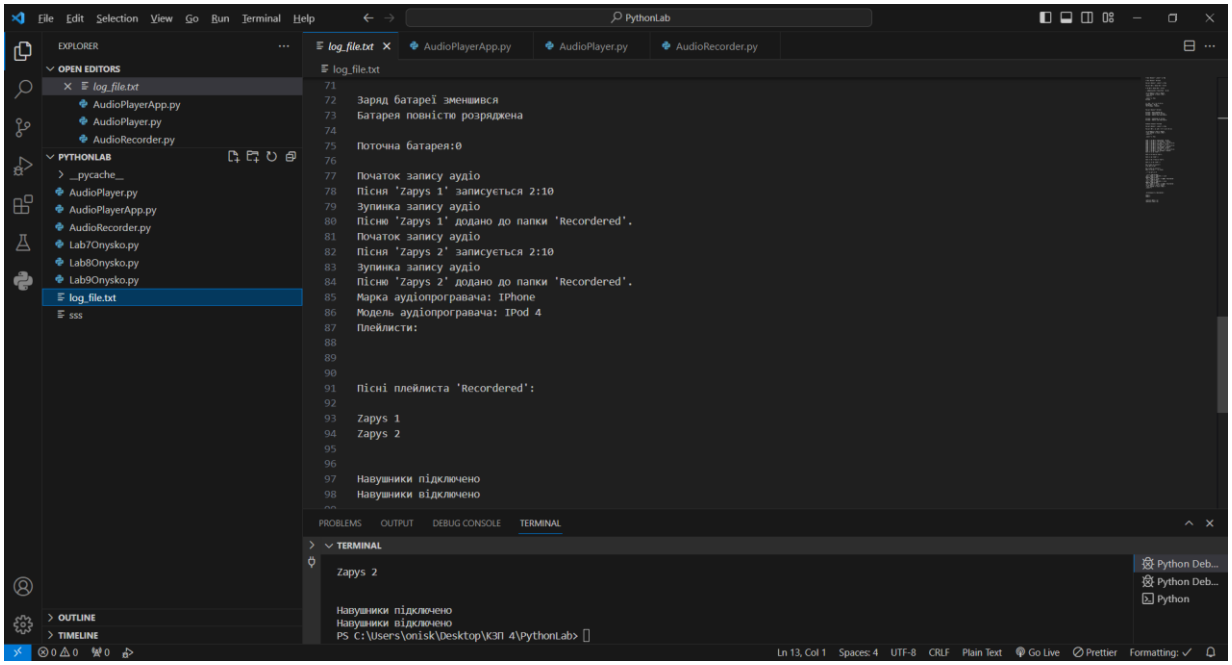

Рис.2 Запис у файл

Висновок:
У ході виконання завдання була розроблена програма на мові Python використовуючи розділення класів програми на окремі модулі в одному пакеті. Було освоєно навички роботи з об'єктно-орієнтованим програмуванням в мові Python, а також вміння структурувати код за допомогою пакетів та модулів.

Контрольні питання:
1. Що таке модулі?
Модулі - це файли, що містять Python код. Вони використовуються для організації коду в більш зручний та структурований спосіб.
2. Як імпортувати модуль?
import module_name
3. Як оголосити клас?
class MyClass:
    # тіло класу
4. Що може міститися у класі?
- атрибути (змінні, які мають значення);
- методи (функції, які виконують дії);
- конструктор (init метод, що викликається при створенні об'єкта класу);
- інші властивості та методи.

5. Як називається конструктор класу?
Конструктор класу оголошується за допомогою методу __init__
6. Як здійснити спадкування?
class ChildClass(ParentClass):
    # тіло класу
7. Які види спадкування існують?

Існують одинарне спадкування (клас успадковується від одного батьківського класу) та множинне спадкування (клас успадковується від кількох батьківських класів).
8. Які небезпеки є при множинному спадкуванні, як їх уникнути?

Небезпеки при множинному спадкуванні:
Колізії імен: Коли різні батьківські класи мають методи чи атрибути з однаковими іменами, можуть виникнути конфлікти.
Складність зміни: Зміни в одному з батьківських класів можуть вплинути на всіх його спадкоємців.

Як уникнути:
Використовуйте композицію: Замість множинного спадкування використовуйте композицію, коли один клас містить екземпляр іншого.
9. Що таке класи-домішки?
Класи-домішки (Mixin classes) - це класи, які не призначені для самостійного використання, але додають функціональність іншим класам шляхом спадкування.
10. Яка роль функції super() при спадкуванні?
super() у функції __init__ класу-нащадка використовується для виклику методів батьківського класу і передачі параметрів.