

Міністерство освіти і науки України
Національний університет «Львівська політехніка»

Кафедра ЕОМ



Звіт
до лабораторної роботи № 2
з дисципліни: «Кросплатформні засоби програмування»
на тему: «Класи та пакети»
Варіант №13

Виконав:
ст. гр. КІ-203
Панченко Д. В.

Прийняв:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів 2023

Мета: ознайомитися з процесом розробки класів та пакетів мовою Java.

Завдання:

1. Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну область згідно варіанту. Програма має задовольняти наступним вимогам:

- програма має розміщуватися в пакеті Група.Прізвище.Lab2;
- клас має містити мінімум 3 поля, що є об'єктами класів, які описують складові частини предметної області;
- клас має містити кілька конструкторів та мінімум 10 методів;
- для тестування і демонстрації роботи розробленого класу розробити клас-драйвер;
- методи класу мають вести протокол своєї діяльності, що записується у файл;
- розробити механізм коректного завершення роботи з файлом (не надіятися на метод `finalize()`);
- програма має володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленої програми.

3. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.

4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.

5. Дати відповідь на контрольні запитання.

Варіант завдання: 13 – Телефон.

Вихідний код програми:

Файл Phone.java:

```
/**
 * lab 2 package
 */
package KI_306.Panchenko.Lab2;
import java.io.*;
/**
 * Class <code>Phone</code> implements phone
 */

public class Phone {
    private String nameDevice;
    private boolean isOn;
    private int batteryLevel;
    private String numberOfThisPhone;
    private boolean bluetooth;
    private double balance;
    private String connectedDevices;
    private int loudness;
    private PrintWriter fout;
```

```

/**
 * Constructor
 * @throws FileNotFoundException is exception
 */
public Phone() throws FileNotFoundException
{
    nameDevice = "This phone";
    isOn = false;
    batteryLevel = 0;
    numberOfThisPhone = "+380933333333";
    bluetooth = false;
    balance = 0;
    connectedDevices = "";
    loudness = 0;
    fout = new PrintWriter(new File("Log.txt"));
}

/**
 * Constructor
 * @param nameDevice The name of this phone
 * @param batteryLevel This phone's battery level
 * @param numberOfThisPhone The number of this phone
 * @param balance This phone's balance
 * @throws FileNotFoundException is exception
 */
public Phone(String nameDevice, int batteryLevel, String numberOfThisPhone, double
balance) throws FileNotFoundException
{
    this.nameDevice = nameDevice;
    isOn = false;
    this.batteryLevel = batteryLevel;
    this.numberOfThisPhone = numberOfThisPhone;
    this.balance = balance;
    bluetooth = false;
    connectedDevices = "";
    loudness = 0;
    fout = new PrintWriter(new File("Log.txt"));
}

/**
 * Method for turning on a phone
 */
public void turnOn() {
    System.out.println(" \n");
    fout.print(" \n");
    isOn = true;
    System.out.println("Телефон " + nameDevice + " увімкнено \n");
    fout.print("Телефон " + nameDevice + " увімкнено \n");
}

/**
 * Method for turning off a phone
 */
public void turnOff() {
    System.out.println(" \n");
    fout.print(" \n");
    isOn = false;
    bluetooth = false;
    connectedDevices = "";
    System.out.println("Телефон " + nameDevice + " вимкнено \n");
}

```

```

        fout.print("Телефон " + nameDevice + " вимкнено \n");
    }

    /**
     * Method for check the battery charge level
     */
    public void checkBattery() {
        System.out.println(" \n");
        fout.print(" \n");
        System.out.println("Заряд батареї: " + batteryLevel + " % \n");
        fout.print("Заряд батареї: " + batteryLevel + " % \n");
    }

    /**
     * Method for charging a phone battery
     * @param time The battery charging time in minutes
     */
    public void chargeBattery(int time) {
        System.out.println(" \n");
        fout.print(" \n");
        if (batteryLevel + time/2 >= 100) {
            batteryLevel = 100;
        } else { batteryLevel += time/2;}
        System.out.println("Батарея заряджалась " + time + " хвилин \n" + "Заряд
батареї: " + batteryLevel + " % \n");
        fout.print("Батарея заряджалась " + time + " хвилин \n" + "Заряд батареї: "
+ batteryLevel + " % \n");
    }

    /**
     * Method for sending an SMS message
     * @param phoneNumber Recipient's phone number
     * @param message Message text
     */
    public void sendSMS(String phoneNumber, String message) {
        System.out.println(" \n");
        fout.print(" \n");
        if (isOn) {
            System.out.println("Відправляємо SMS на номер " + phoneNumber + ": " +
message + "\n" + "SMS успішно надіслано \n");
            fout.print("Відправляємо SMS на номер " + phoneNumber + ": " + message +
"\n" + "SMS успішно надіслано \n");
        } else {
            System.out.println("Телефон вимкнено, щоб надіслати SMS ввімкніть його
\n");
            fout.print("Телефон вимкнено, щоб надіслати SMS ввімкніть його \n");
        }
    }

    /**
     * Method for calling the number
     * @param phoneNumber The phone number to call
     */
    public void call(String phoneNumber) {
        System.out.println(" \n");
        fout.print(" \n");
        if (isOn) {
            System.out.println("Дзвінок на номер " + phoneNumber + "... \n");
            fout.print("Дзвінок на номер " + phoneNumber + "... \n");
        } else {
            System.out.println("Телефон вимкнено, щоб здійснити дзвінок ввімкніть його
\n");
            fout.print("Телефон вимкнено, щоб здійснити дзвінок ввімкніть його \n");
        }
    }

```

```

}

/**
 * Method to change device name
 * @param newName New name
 */
public void changeNameDevice(String newName) {
    System.out.println(" \n");
    fout.print(" \n");
    if (isOn) {
        nameDevice = newName;
        System.out.println("Назву цього пристрою змінено, нова назва - " + newName +
"\n");
        fout.print("Назву цього пристрою змінено, нова назва - " + newName + "\n");
    } else {
        System.out.println("Телефон вимкнено, щоб змінити ім'я ввімкніть його
\n");
        fout.print("Телефон вимкнено, щоб змінити ім'я ввімкніть його \n");
    }
}

/**
 * Method for changing number of this phone
 * @param newNumber New number of this phone
 */
public void changeNumberOfThisPhone(String newNumber) {
    System.out.println(" \n");
    fout.print(" \n");
    if (isOn) {
        numberOfThisPhone = newNumber;
        System.out.println("Номер телефону цього пристрою змінено, новий номер - "
+ newNumber + "\n");
        fout.print("Номер телефону цього пристрою змінено, новий номер - " +
newNumber + "\n");
    } else {
        System.out.println("Телефон вимкнено, щоб змінити номер ввімкніть
його \n");
        fout.print("Телефон вимкнено, щоб змінити номер ввімкніть його \n");
    }
}

/**
 * Method for check the balance
 */
public void checkBalance() {
    System.out.println(" \n");
    fout.print(" \n");
    if (isOn) {
        System.out.println("Стан балансу: " + balance + " грн \n");
        fout.print("Стан балансу: " + balance + " грн \n");
    } else {
        System.out.println("Телефон вимкнено, щоб перевірити баланс
ввімкніть його \n");
        fout.print("Телефон вимкнено, щоб перевірити баланс ввімкніть його
\n");
    }
}

/**
 * Method for replenishing the balance
 * @param sum The balance replenishment amount
 */
public void replenishBalance (int sum) {
    System.out.println(" \n");

```

```

        fout.print(" \n");
    if (isOn) {
        balance += sum;
        System.out.println("Баланс поповнено на " + sum + " грн \n" + "Поточний
баланс: " + balance + " грн \n");
        fout.print("Баланс поповнено на " + sum + " грн \n" + "Поточний баланс: "
+ balance + " грн \n");
    } else {
        System.out.println("Телефон вимкнено, щоб змінити баланс ввімкніть
його \n");
        fout.print("Телефон вимкнено, щоб змінити баланс ввімкніть
його \n");
    }
}

/**
 * Method for turning on a Bluetooth
 */
public void bluetoothOn() {
    System.out.println(" \n");
    fout.print(" \n");
    if (isOn) {
        bluetooth = true;
        System.out.println("Bluetooth увімкнено " + "\n");
        fout.print("Bluetooth увімкнено " + "\n");
    } else {
        System.out.println("Телефон вимкнено, щоб керувати Bluetooth ввімкніть
його \n");
        fout.print("Телефон вимкнено, щоб керувати Bluetooth ввімкніть його \n");
    }
}

/**
 * Method for turning off a Bluetooth
 */
public void bluetoothOff() {
    System.out.println(" \n");
    fout.print(" \n");
    if (isOn) {
        bluetooth = false;
        connectedDevices = "";
        System.out.println("Bluetooth вимкнено, усі під'єднані пристрої від'єднано
\n");
        fout.print("Bluetooth вимкнено, усі під'єднані пристрої від'єднано \n");
    } else {
        System.out.println("Телефон вимкнено, щоб керувати Bluetooth ввімкніть
його \n");
        fout.print("Телефон вимкнено, щоб керувати Bluetooth ввімкніть його \n");
    }
}

/**
 * Method for connect the device
 * @param newDevice The device we want to connect
 */
public void connectDevice(String newDevice) {
    System.out.println(" \n");
    fout.print(" \n");
    if (isOn) {
        if (bluetooth) {
            if (connectedDevices == "") {
                connectedDevices = newDevice;
            }
            else {

```

```

        connectedDevices += ", " + newDevice;
    }
    System.out.println("Під'єднано новий пристрій: " + newDevice + " \n" +
"Під'єднані пристрої: " + connectedDevices + " \n");
    fout.print("Під'єднано новий пристрій: " + newDevice + " \n" +
"Під'єднані пристрої: " + connectedDevices + " \n");
    }
    else {
        System.out.println("Bluetooth вимкнено, щоб під'єднати пристрій
ввімкніть його \n");
        fout.print("Bluetooth вимкнено, щоб під'єднати пристрій ввімкніть
його \n");
    }

    } else {
        System.out.println("Телефон вимкнено, щоб керувати Bluetooth ввімкніть
його \n");
        fout.print("Телефон вимкнено, щоб керувати Bluetooth ввімкніть його \n");
    }
}

/**
 * Method for disconnect the device
 * @param disDevice The device we want to disconnect
 */
public void disconnectDevice(String disDevice) {
    System.out.println(" \n");
    fout.print(" \n");
    if (isOn) {
        if (bluetooth) {
            if (connectedDevices.contains(", " + disDevice)) {
                connectedDevices = connectedDevices.replace(", " + disDevice,
"");
                System.out.println("Від'єднано пристрій: " + disDevice + "
\n" + "Під'єднані пристрої: " + connectedDevices + " \n");
                fout.print("Від'єднано пристрій: " + disDevice + " \n" +
"Під'єднані пристрої: " + connectedDevices + " \n");
            }
            else {
                if (connectedDevices.contains(disDevice)) {
                    connectedDevices = connectedDevices.replace(disDevice,
"");
                    System.out.println("Від'єднано пристрій: " + disDevice
+ " \n" + "Під'єднані пристрої: " + connectedDevices + " \n");
                    fout.print("Від'єднано пристрій: " + disDevice + " \n" +
"Під'єднані пристрої: " + connectedDevices + " \n");
                }
                else {
                    System.out.println("Пристрій: " + disDevice + " не було
під'єднано \n" + "Під'єднані пристрої: " + connectedDevices + " \n");
                    fout.print("Пристрій: " + disDevice + " не було
під'єднано \n" + "Під'єднані пристрої: " + connectedDevices + " \n");
                }
            }
        }
        else {
            System.out.println("Bluetooth вимкнено, а отже усі пристрої
від'єднано \n");
            fout.print("Bluetooth вимкнено, а отже усі пристрої від'єднано \n");
        }
    }
    else {
        System.out.println("Телефон вимкнено, щоб керувати Bluetooth ввімкніть його \n");
    }
}

```

```

        fout.print("Телефон вимкнено, щоб керувати Bluetooth ввімкніть його \n"); }
    }

    /**
     * Method for displaying information about a phone
     */
    public void info() {
        System.out.println(" \n");
        fout.print(" \n");
        if (isOn) {
            System.out.println("Назва пристрою: " + nameDevice + " \n");
            fout.print("Назва пристрою: " + nameDevice + " \n");
            System.out.println("Номер телефону: " + numberOfThisPhone + " \n");
            fout.print("Номер телефону: " + numberOfThisPhone + " \n");
            System.out.println("Заряд батареї: " + batteryLevel + " % \n");
            fout.print("Заряд батареї: " + batteryLevel + " % \n");
            System.out.println("Баланс: " + balance + " грн \n");
            fout.print("Баланс: " + balance + " грн \n");
            if (bluetooth) {
                System.out.println("Bluetooth: увімкнено\n");
                fout.print("Bluetooth: увімкнено\n");
            }
            else {
                System.out.println("Bluetooth: вимкнено\n");
                fout.print("Bluetooth: вимкнено\n");
            }
            System.out.println("Під'єднані пристрої: " + connectedDevices + " \n");
            fout.print("Під'єднані пристрої: " + connectedDevices + " \n");
            System.out.println("Гучність: " + loudness + "/15 \n");
            fout.print("Гучність: " + loudness + "/15 \n");
        }
        else {
            System.out.println("Телефон вимкнено, щоб переглянути інформацію про нього - ввімкніть його \n");
            fout.print("Телефон вимкнено, щоб переглянути інформацію про нього - ввімкніть його \n");
        }
    }

    /**
     * Method to change loudness
     * @param clicks Number of clicks
     * @param button more or less buttons
     */
    public void changeLoudness(int clicks, String button) {
        System.out.println(" \n");
        fout.print(" \n");
        if (isOn) {
            if (button == "more") {
                for (int i = 0; i < clicks; i++) {
                    loudness ++;
                    if (loudness > 15) { loudness = 15;}
                    System.out.println("Гучність - " + loudness + "\n");
                    fout.print("Гучність - " + loudness + "\n");
                }
            }
            else {
                if (button == "less") {
                    for (int i = 0; i < clicks; i++) {
                        loudness --;
                        if (loudness < 0) { loudness = 0;}
                        System.out.println("Гучність - " + loudness + "\n");
                        fout.print("Гучність - " + loudness + "\n");
                    }
                }
            }
        }
    }

```



```

        }
    }
    else {
        System.out.println("Такої кнопки немає \n");
        fout.print("Такої кнопки немає \n");
    }
}

} else {
    System.out.println("Телефон вимкнено, щоб змінювати гучність ввімкніть його \n");
    fout.print("Телефон вимкнено, щоб змінювати гучність ввімкніть його \n");
}
}

/**
 * Method releases used recourses
 */
public void dispose()
{ fout.close(); }

}

```

Файл PhoneApp.java:

```

/**
 * lab 2 package
 */
package KI_306.Panchenko.Lab2;
import static java.lang.System.out;
import java.io.*;
/**
 * Phone Application class implements main method for
 * Phone class abilities demonstration
 */
public class PhoneApp
{
    /**
     * @param args is arguments
     * @throws FileNotFoundException is exception
     */
    public static void main(String[] args) throws FileNotFoundException
    {
        // TODO Auto-generated method stub
        Phone myPhone = new Phone("SamsungA10", 30, "+0987654321", 120);

        myPhone.call("+0912345678");

        myPhone.turnOn();

        myPhone.call("+0912345678");

        myPhone.checkBattery();

        myPhone.chargeBattery(60);

        myPhone.checkBalance();
    }
}

```

```

myPhone.replenishBalance(40);

myPhone.connectDevice("Наушники");

myPhone.bluetoothOn();

myPhone.connectDevice("Наушники");

myPhone.changeLoudness(5, "more");

myPhone.changeLoudness(2, "less");

myPhone.disconnectDevice("Годинник");

myPhone.disconnectDevice("Наушники");

myPhone.sendSMS("0912345678", "Привіт");

myPhone.changeNameDevice("Мій пристрій");

myPhone.info();

myPhone.turnOff();

myPhone.dispose();

}
}

```

Результат виконання програми:

Log: Блокнот
Файл Редагування Формат Вигляд Довідка

Телефон вимкнено, щоб здійснити дзвінок ввімкніть його

Телефон SamsungA10 увімкнено

Дзвінок на номер +0912345678...

Заряд батареї: 30 %

Батарея заряджалась 60 хвилин

Заряд батареї: 60 %

Стан балансу: 120.0 грн

Баланс поповнено на 40 грн

Поточний баланс: 160.0 грн

Bluetooth вимкнено, щоб під'єднати пристрій ввімкніть його

Bluetooth увімкнено

Під'єднано новий пристрій: Наушники

Під'єднані пристрої: Наушники

Гучність - 1

Гучність - 2

Гучність - 3

Гучність - 4

Гучність - 5

Рд 1, ствп 1
100%
UNIX (LF)
UTF-8

Log: Блокнот
Файл Редагування Формат Вигляд Довідка

Гучність - 2

Гучність - 3

Гучність - 4

Гучність - 5

Гучність - 4

Гучність - 3

Пристрій: Годинник не було під'єднано

Під'єднані пристрої: Наушники

Від'єднано пристрій: Наушники

Під'єднані пристрої:

Відправляємо SMS на номер 0912345678: Привіт

SMS успішно надіслано

Назву цього пристрою змінено, нова назва - Мій пристрій

Назва пристрою: Мій пристрій

Номер телефону: +0987654321

Заряд батареї: 60 %

Баланс: 160.0 грн

Bluetooth: увімкнено

Під'єднані пристрої:

Гучність: 3/15

Телефон Мій пристрій вимкнено

Рд 1, ствп 1
100%
UNIX (LF)
UTF-8

Фрагмент згенерованої документації:

Phone

PhoneApp

D:/Github_repos/CPPT_LABS/LAB_02/doc/KI_306/Panchenko/Lab2/Phone.html

PACKAGECLASSUSE TREELINKSHELP

SUMMARY: NESTED | FIELD | CONSTR | METHODDETAIL: FIELD | CONSTR | METHOD

SEARCH

Package KI_306.Panchenko.Lab2

Class Phone

java.lang.Object[Ⓜ]
KI_306.Panchenko.Lab2.Phone

public class Phone
extends Object[Ⓜ]

Class Phone implements phone

Constructor Summary

Constructors

Constructor	Description
Phone()	Constructor
Phone(String [Ⓜ] nameDevice, int batteryLevel, String [Ⓜ] numberofThisPhone, double balance)	Constructor

Method Summary

All MethodsInstance MethodsConcrete Methods

Modifier and Type	Method	Description
void	bluetoothOff()	Method for turning off a Bluetooth
void	bluetoothOn()	Method for turning on a Bluetooth
void	call(String [Ⓜ] phoneNumber)	Method for calling the number
void	changeLoudness(int clicks, String [Ⓜ] button)	Method to change loudness
void	changeNameDevice(String [Ⓜ] newName)	Method to change device name
void	changeNumberOfThisPhone(String [Ⓜ] newNumber)	Method for changing number of this phone
void	chargeBattery(int time)	Method for charging a phone battery
void	checkBalance()	Method for check the balance
void	checkBattery()	Method for check the battery charge level
void	connectDevice(String [Ⓜ] newDevice)	Method for connect the device
void	disconnectDevice(String [Ⓜ] disDevice)	Method for disconnect the device
void	dispose()	Method releases used resources
void	info()	Method for displaying information about a phone
void	replenishBalance(int sum)	Method for replenishing the balance
void	sendsMS(String [Ⓜ] phoneNumber, String [Ⓜ] message)	Method for sending an SMS message
void	turnOff()	Method for turning off a phone
void	turnOn()	Method for turning on a phone

Methods inherited from class java.lang.Object[Ⓜ]

equals[Ⓜ], getClass[Ⓜ], hashCode[Ⓜ], notify[Ⓜ], notifyAll[Ⓜ], toString[Ⓜ], wait[Ⓜ], wait[Ⓜ], wait[Ⓜ]

PACKAGE **CLASS** USE TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH

Package KI_306.Panchenko.Lab2

Class PhoneApp

java.lang.Object[Ⓜ]
KI_306.Panchenko.Lab2.PhoneApp

public class PhoneApp
extends Object[Ⓜ]

Phone Application class implements main method for Phone class abilities demonstration

Constructor Summary

Constructors

Constructor	Description
PhoneApp()	

Method Summary

All Methods **Static Methods** **Concrete Methods**

Modifier and Type	Method	Description
static void	main(String [Ⓜ] [] args)	

Methods inherited from class java.lang.Object[Ⓜ]

equals[Ⓜ], getClass[Ⓜ], hashCode[Ⓜ], notify[Ⓜ], notifyAll[Ⓜ], toString[Ⓜ], wait[Ⓜ], wait[Ⓜ], wait[Ⓜ]

Constructor Details

PhoneApp

```
public PhoneApp()
```

Method Details

main

```
public static void main(StringⓂ[] args)
    throws FileNotFoundExceptionⓂ
```

Parameters:
args - is arguments

Throws:
FileNotFoundException[Ⓜ] - is exception

Відповіді на контрольні запитання:

1. Синтаксис визначення класу.

Синтаксис оголошення простого класу в мові Java має наступний вигляд:

```
[public] class НазваКласу
{
    [конструктори]
    [методи]
    [поля]
}
```

2. Синтаксис визначення методу.

Синтаксис оголошення методу наступний:

```
[СпецифікаторДоступу] [static] [final] Тип назваМетоду([параметри]) [throws класи]
{
    [Тіло методу]
    [return [значення]];
}
```

3. Синтаксис оголошення поля.

Синтаксис оголошення поля наступний:

[СпецифікаторДоступу] [static] [final] Тип НазваПоля [= ПочатковеЗначення];

Приклад оголошення поля:

```
private int i;
```

4. Як оголосити та ініціалізувати константне поле?

Приклад оголошення константного поля:

```
private final int i;
```

Приклад явної ініціалізації поля str константою при оголошенні:

```
public class StartClass
{
    ...
    private String str = "Hello";
}
```

5. Які є способи ініціалізації полів?

Ініціалізацію полів при створенні об'єкту можна здійснювати трьома способами:

- у конструкторі;
- явно при оголошенні поля;
- у блоці ініціалізації (виконується перед виконанням конструктора).

6. Синтаксис визначення конструктора.

Синтаксис оголошення конструктора:

```
[СпецифікаторДоступу] НазваКласу([параметри])
{
    Тіло конструктора
}
```

7. Синтаксис оголошення пакету.

Синтаксис оператора package:

```
package НазваПакету { .НазваПідпакету };
```

8. Як підключити до програми класи, що визначені в зовнішніх пакетах?

Доступ до класів з інших пакетів можна отримати двома шляхами:

1. вказуючи повне ім'я пакету перед іменем кожного класу, наприклад:
`java.util.Date today = new java.util.Date();`
2. використовуючи оператор `import`, що дозволяє підключати як один клас так і всі загальнодоступні класи пакету, позбавляючи необхідності записувати імена класів з вказуванням повної назви пакету перед ними.

9. В чому суть статичного імпорту пакетів?

Починаючи з Java SE 5.0 у мову додано можливість імпортувати статичні методи і поля класів. Для цього при підключенні пакету слід вжити ключове слово `static` та вказати назву пакету, або назву пакету класу та статичного методу чи поля, які ви хочете підключити:

```
import static НазваПакету { .НазваПідпакету } .НазваКласу.
НазваСтатичногоМетодуАбоПоля;
import static НазваПакету { .НазваПідпакету } .*;
```

10. Які вимоги ставляться до файлів і каталогів при використанні пакетів?

Використання пакетів вимагає, щоб файли і каталоги проекту та їх ієрархія були строго структурованими. Так назви пакету і його підпакетів мають співпадати з назвами каталогів, де вони розміщуються. Назви загальнодоступних класів мають співпадати з назвами файлів, де вони розміщуються. Ієрархія каталогів і файлів проекту має співпадати з ієрархією пакетів. Після компіляції ієрархія каталогів, де містяться файли класів, співпадає з ієрархією каталогів проекту.

Висновок: Я ознайомився з процесом розробки класів та пакетів мовою Java.