

Міністерство освіти і науки України
Національний університет «Львівська політехніка»



Звіт

до лабораторної роботи №2

З дисципліни: «Кросплатформенні засоби програмування»

На тему: « КЛАСИ ТА ПАКЕТИ»

Виконав:

Студент групи КІ-301

Осадчук В.В.

Прийняв:

Майдан М.В.

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 18. Сканер

ScannerApp.java

```
package ki_301.osadchuk.Lab2;
import java.io.*;

public class ScannerApp {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner scanner = new Scanner();
        scanner.connectPowerCordConnector();
        scanner.connectUSB();
        scanner.turnOffColoredScan();
        scanner.putSmthOnTable();
        scanner.turnOnStartButton();
        scanner.scanning();
        scanner.disconnectPowerCordConnector();
        scanner.turnOnColoredScan();
        scanner.putSmthOnTable();
        scanner.turnOnStartButton();
        scanner.scanning();
        scanner.dispose();
    }
}
```

Scanner.java

```
package ki_301.osadchuk.Lab2;
import java.io.*;

public class Scanner{
    private ScannerButton scannerButton;
    private ScannerTablet scannerTablet;
    private ScannerPort scannerPort;
    private PrintWriter fout;
```

```

/**
 * Constructor
 * @throws FileNotFoundException
 */
public Scanner() throws FileNotFoundException
{
    scannerButton = new ScannerButton();
    scannerTablet = new ScannerTablet();
    scannerPort = new ScannerPort();
    fout = new PrintWriter(new File("Log.txt"));
}

/**
 * Constructor
 * @throws FileNotFoundException
 */
public Scanner(boolean colored) throws FileNotFoundException
{
    scannerButton = new ScannerButton(colored);
    scannerTablet = new ScannerTablet();
    scannerPort = new ScannerPort();
    fout = new PrintWriter(new File("Log.txt"));
}

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

/**
 * Method implements turning on Scanner
 */
public void turnOnScanner(){
    if(scannerPort.get_powerCordConnector()) {
        scannerButton.set_Power(true);
        if (scannerButton.get_Power()) {
            System.out.print("The scanner is on\n");
            fout.print("The scanner is on\n");
            fout.flush();
        } else {
            System.out.print("The scanner isn't on\n");
            fout.print("The scanner isn't on\n");
            fout.flush();
        }
    }
}

/**
 * Method implements turning off Scanner
 */
public void turnOffScanner(){
    if(!scannerPort.get_powerCordConnector()) {
        scannerButton.set_Power(false);
        if (!scannerButton.get_Power()) {
            System.out.print("The scanner isn't on\n");
            fout.print("The scanner isn't on\n");
            fout.flush();
        }
    }
}

```

```

} else {
System.out.print("The scanner is on\n");
fout.print("The scanner is on\n");
fout.flush();
}
}
}

/**
 * Method implements turn on colored scanning
 */
public void turnOnColoredScan(){
scannerButton.set_Colored(true);
if(scannerButton.get_Colored()) {
System.out.print("The colored scan is on\n");
fout.print("The colored scan is on\n");
fout.flush();
}else{
System.out.print("The colored scan isn't on\n");
fout.print("The colored scan isn't on\n");
fout.flush();
}
}

/**
 * Method implements turn off colored scanning
 */
public void turnOffColoredScan(){
scannerButton.set_Colored(false);
if(scannerButton.get_Colored()) {
System.out.print("The colored scan is on\n");
fout.print("The colored scan is on\n");
fout.flush();
}else{
System.out.print("The colored scan isn't on\n");
fout.print("The colored scan isn't on\n");
fout.flush();
}
}

/**
 * Method implements turn on Start button
 */
public void turnOnStartButton() {
scannerButton.set_Start(true);
if(scannerButton.get_Start()) {
System.out.print("The start button is pressed\n");
fout.print("The start button is pressed\n");
fout.flush();
}else{
System.out.print("The start button isn't pressed\n");
fout.print("The start button isn't pressed\n");
fout.flush();
}
}

/**
 * Method implements turn off Start button

```

```

*/
public void turnOffStartButton(){
scannerButton.set_Start(false);
if(scannerButton.get_Start()) {
System.out.print("The start button is pressed\n");
fout.print("The start button is pressed\n");
fout.flush();
}else{
System.out.print("The start button isn't pressed\n");
fout.print("The start button isn't pressed\n");
fout.flush();
}
}

/**
 * Method implements putting something on scanner table
 * to scan it
 */
public void putSmtHOnTable(){
scannerTablet.set_Tablet(true);
System.out.print("Something put on table\n");
fout.print("Something put on table\n");
fout.flush();
}

/**
 * Method implements check if something is on the table
 */
public boolean canWeScan(){
return scannerTablet.get_Tablet();
}

/**
 * Method implements connecting USB
 */
public void connectUSB(){
scannerPort.set_USB(true);
if(scannerPort.get_USB()){
System.out.print("The USB is connected\n");
fout.print("The USB is connected\n");
fout.flush();
}else{
System.out.print("The USB isn't connected\n");
fout.print("The USB isn't connected\n");
fout.flush();
}
}

/**
 * Method implements disconnecting USB
 */
public void disconnectUSB(){
scannerPort.set_USB(false);
if(scannerPort.get_USB()){
System.out.print("The USB isn't disconnected\n");
fout.print("The USB isn't disconnected\n");
fout.flush();
}else{
System.out.print("The USB is disconnected\n");
fout.print("The USB is disconnected\n");
}
}

```

```
fout.flush();
}
}

/**
 * Method implements connecting Power Cord
 */
public void connectPowerCordConnector() {
scannerPort.set_powerCordConnector(true);
if(scannerPort.get_powerCordConnector()){
System.out.print("The Power Cord is connected\n");
fout.print("The Power Cord is connected\n");
fout.flush();
}else{
System.out.print("The Power Cord isn't connected\n");
fout.print("The Power Cord isn't connected\n");
fout.flush();
}
}

/**
 * Method implements disconnecting Power Cord
 */
public void disconnectPowerCordConnector() {
scannerPort.set_powerCordConnector(false);
if(scannerPort.get_powerCordConnector()){
System.out.print("The Power Cord isn't disconnected\n");
fout.print("The Power Cord isn't disconnected\n");
fout.flush();
}else{
System.out.print("The Power Cord is disconnected\n");
fout.print("The Power Cord is disconnected\n");
fout.flush();
}
}

/**
 * Method implements scanning
 */
public void scanning(){
if(scannerPort.get_powerCordConnector()){
if(scannerButton.get_Power()){
if(scannerButton.get_Start()){
if(scannerTablet.get_Tablet()){
if(scannerButton.get_Colored()){
System.out.print("Colored scanning ...\n");
System.out.print("Scanned\n");
fout.print("Colored scanning ...\n");
fout.print("Scanned\n");
fout.flush();
}else {
System.out.print("White\\Black scanning ...\n");
System.out.print("Scanned\n");
fout.print("White\\Black scanning ...\n");
System.out.print("Scanned\n");
fout.flush();
}
}else{
System.out.print("Nothing to scan try again\n");
fout.print("Nothing to scan try again\n");
}
```

```

fout.flush();
}
}else{
System.out.print("Start button isn't pressed\n");
fout.print("Start button isn't pressed\n");
fout.flush();
}
}else{
System.out.print("Power button isn't plugged in\n");
fout.print("Power button isn't plugged in\n");
fout.flush();
}
}else{
System.out.print("Power isn't plugged in\n");
fout.print("Power isn't plugged in\n");
fout.flush();
}
}
}

class ScannerButton{
private boolean isStart;
private boolean isColored;
private boolean isPower;

/**
 * Constructor default
 */
public ScannerButton(){
isStart = false;
isColored = false;
isPower = true;
}

/**
 * Constructor with three parameters
 */
public ScannerButton(boolean setStart, boolean setColored, boolean setPower){
isStart = setStart;
isColored = setColored;
isPower = setPower;
}

/**
 * Constructor with one parameter
 */
public ScannerButton(boolean setColored){
isColored = setColored;
}

/**
 * Method sets start button
 */
public void set_Start(boolean setStart){
isStart = setStart;
}

/**
 * Method sets colored button
 */

```

```
public void set_Colored(boolean setColored){  
    isColored = setColored;  
}
```

```
/**  
 * Method sets power button  
 */  
public void set_Power(boolean setPower){  
    isPower = setPower;  
}
```

```
/**  
 * Method sets start button  
 */  
public boolean get_Start(){  
    return isStart;  
}
```

```
/**  
 * Method get colored button  
 */  
public boolean get_Colored(){  
    return isColored;  
}
```

```
/**  
 * Method get power button  
 */  
public boolean get_Power(){  
    return isPower;  
}  
}
```

```
class ScannerTablet{  
    private boolean isOnTablet;
```

```
/**  
 * Constructor default  
 */  
public ScannerTablet(){  
    isOnTablet = false;  
}
```

```
/**  
 * Method sets tablet  
 */  
public void set_Tablet(boolean sOnTablet){  
    isOnTablet = sOnTablet;  
}
```

```
/**  
 * Method get tablet  
 */  
public boolean get_Tablet(){  
    return isOnTablet;  
}  
}
```



```

class ScannerPort{
private boolean USB;
private boolean powerCordConnector;

/**
 * Constructor default
 */
public ScannerPort(){
USB = false;
powerCordConnector = false;
}

/**
 * Method sets USB connection
 */
public void set_USB(boolean sUSB){
USB = sUSB;
}

/**
 * Method sets Cord Connector connection
 */
public void set_powerCordConnector(boolean sPowerCordConnector){
powerCordConnector = sPowerCordConnector;
}

/**
 * Method get USB connection
 */
public boolean get_USB(){
return USB;
}

/**
 * Method get Cord connection
 */
public boolean get_powerCordConnector(){
return powerCordConnector;
}
}

```

```

<terminated> ScannerApp [Java Application]
The Power Cord is connected
The USB is connected
The colored scan isn't on
Something put on table
The start button is pressed
White\Black scanning ...
Scanned
The Power Cord is disconnected
The colored scan is on
Something put on table
The start button is pressed
Power isn't plugged in

```

Log.txt: Блокнот

Файл Редагування Формат Вигляд Довідка

```

The Power Cord is connected
The USB is connected
The colored scan isn't on
Something put on table
The start button is pressed
White\Black scanning ...
Scanned
The Power Cord is disconnected
The colored scan is on
Something put on table
The start button is pressed
Power isn't plugged in

```

Відповіді на КЗ:

1. [public] class Назва Класу {

[Методи]

[Змінні]

[Поля]

[Конструктори]

}

2. [СпецифікаторДоступу] Тип назваМетоду([параметри]) [throws класи] {

[Тіло методу] [return [значення]];

}

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

4. Використати [final], тобто/наприклад `private final int i;`

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

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

Якщо поле не ініціалізується жодним з цих способів, то йому присвоюється значення за замовчуванням.

6. [СпецифікаторДоступу] НазваКласу([параметри]) {

Тіло конструктора

}

7. `package НазваПакету{.НазваПідпакету};`

8. Клас може використовувати всі класи з власного пакету і всі загальнодоступні класи з інших пакетів. Доступ до класів з інших пакетів можна отримати двома шляхами:

1. вказуючи повне ім'я пакету перед іменем кожного класу

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. Статичний імпорт дозволяє не вживати явно назву класу при звертанні до статичного поля або методу класу.

10. Файл, каталоги повинні бути строго структурованими. Чітка ієрархія, назви пакетів та підпакетів повинні співпадати з назвами каталогів де вони розміщуються.

Висновок: на цій лабораторній роботі я ознайомився з процесом розробки класів та пакетів мовою Java.