МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

**Лабораторна робота № 3**

з дисципліни « ООП на Java »

*назва дисципліни*

на тему: «ВИКОРИСТАННЯ ІНТЕРФЕЙСІВ У МОВІ JAVА»

Виконав: студент 2 курсу групи № 622п

освітньої програми

121 інженерія програмного забезпечення

(шифр і назва ОП)

Зайченко Ярослав Ігорович

(прізвище й ініціали студента)

Прийняв: доцент

Симбірський Генадій Дмитрович

(посада, науковий ступінь, прізвище й ініціали)

Кількість балів:

Харків – 2023

**ЗМІСТ**

[Постановка завдання 3](#_Toc158672321)

[Зміст звіту 3](#_Toc158672322)

[Порядок виконання роботи 3](#_Toc158672323)

[Виконання роботи 5](#_Toc158672324)

[Висновок 10](#_Toc158672325)

[Контрольні питання 11](#_Toc158672326)

Мета роботи: отримання практичних навичок із використання інтерфейсів при проектуванні об'єктно-орієнтованих програмних додатків мовою Java.

Постановка завдання

Розробити та реалізувати мовою Java програмний додаток для моделювання властивостей, поведінки та взаємодії об'єктів. Предметною областю для цього додатка є об'єкти, наведені у п.3 цієї роботи. Для заданої предметної області створити ієрархію класів.

У додатку продемонструвати використання механізмів створення та реалізації інтерфейсів для додання нової поведінки до об'єктів класів, успадкування інтерфейсів, а також використання інтерфейсів в якості посилального (об'єктного) типу та поліморфізму.

Порядок виконання роботи

1. Вивчити теоретичний матеріал, пов'язаний з темою даної лабораторної роботи, використовуючи конспект лекцій, літературу, що рекомендується, і документацію, що входить до складу пакету Java SE JDK.
2. Розробити концептуальну модель для заданої категорії об'єктів у вигляді словесного опису їх основних властивостей та поведінки.
3. Спроектувати абстрактний клас, у якому буде визначено найбільш загальні характерні властивості заданої категорії об'єктів.
4. Визначити найбільш характерні риси поведінки заданої категорії об'єктів і подати їх у вигляді набору методів, визначених в абстрактному класі.
5. Розробити підкласи для об'єктів, які успадковують клас узагальненого об'єкта (абстрактний) та реалізують поведінку (методи). Ці методи мають бути задекларовані у заздалегідь розроблених абстрактних класах або суперкласах.
6. Спроектувати два інтерфейси, в методах яких визначити на абстрактному рівні додаткову поведінку для деяких підкласів.
7. У розроблених підкласах реалізувати додаткову поведінку за допомогою спроектованих інтерфейсів
8. Відобразити структурні відносини між спроектованими класами та інтерфейсами за допомогою діаграми класів UML.
9. Реалізувати програмний код класів та інтерфейсів на мові Java. При реалізації програмного коду програми обов'язково використовувати коментарі та дескриптори документування для опису класів, конструкторів, методів та полів.
10. Розробити об'єкти для підкласів, для яких викликатимуться методи, що змінюють властивості та поведінку цих об'єктів. Продемонструвати роботу методів інтерфейсів, реалізованних у підкласах та їх поліморфного використання.
11. Запустити розроблену програму на виконання за допомогою середовища Intellij IDEA.
12. Протестувати розроблений додаток.
13. Отримати документацію на розроблений додаток за допомогою утиліти javadoc.exe.
14. Зробити висновки щодо роботи та відповісти на контрольні запитання.
15. Оформити звіт з лабораторної роботи відповідно до вимог, що пред'являються.
16. Захистити роботу.

Завдання для варіанту 10:

Принтери (матричні, струменеві, лазерні, світлодіодні). Властивості: тип моделі, формат паперу, вага, ціна, швидкість друку, режим роботи, тип інтерфейсу, стан. Поведінка: визначення та зміна властивостей.

Виконання роботи

Загальні відомості

Мова програмування – Java

Операційна система – Windows 11 Pro

Процесор – Apple Silicon M1 Pro with 6 CPU cores

Тип компілятора – IntelliJ IDEA 2024

Концептуальна модель об'єктів

Властивості:

* Тип моделі: тип транспортного засобу (наприклад, "автобус", "трамвай", "тролейбус").
* Фірма-виробник: компанія, яка виготовила принтер.
* Номер: унікальний ідентифікатор принтера.
* Маршрут: маршрут, за яким рухається принтер.
* Місткість: кількість пасажирів, які можуть перевозитися принтером.
* Закріплений оператор: водій або оператор, який керує принтером.

Поведінка:

* Визначення та зміна властивостей: методи для встановлення та зміни властивостей принтера, таких як тип моделі, фірма-виробник, номер, маршрут, місткість та закріплений оператор.
* Визначення стану: методи для визначення поточного стану принтера, наприклад, чи працює він, скільки паперу залишилося, чи є з'єднання з комп'ютером, і так далі.

Абстрактний клас, у якому визначено найбільш загальні характерні властивості заданої категорії об'єктів

Створений абстрактний клас з загальними характерними властивостями та поведінкою поданий в Додатку А.

Розроблені підкласи з об'єктами, які успадковують абстрактний клас та реалізують поведінку.

Створені класи принтерів матричний, світлодіодний струменевий та лазерний подані в Додатках Б, В, Г, Д.

У розроблених підкласах реалізувано додаткову поведінку за допомогою спроектованих інтерфейсів. Додаток Е.

Структурні відносини між спроектованими класами за допомогою діаграми класів UML.

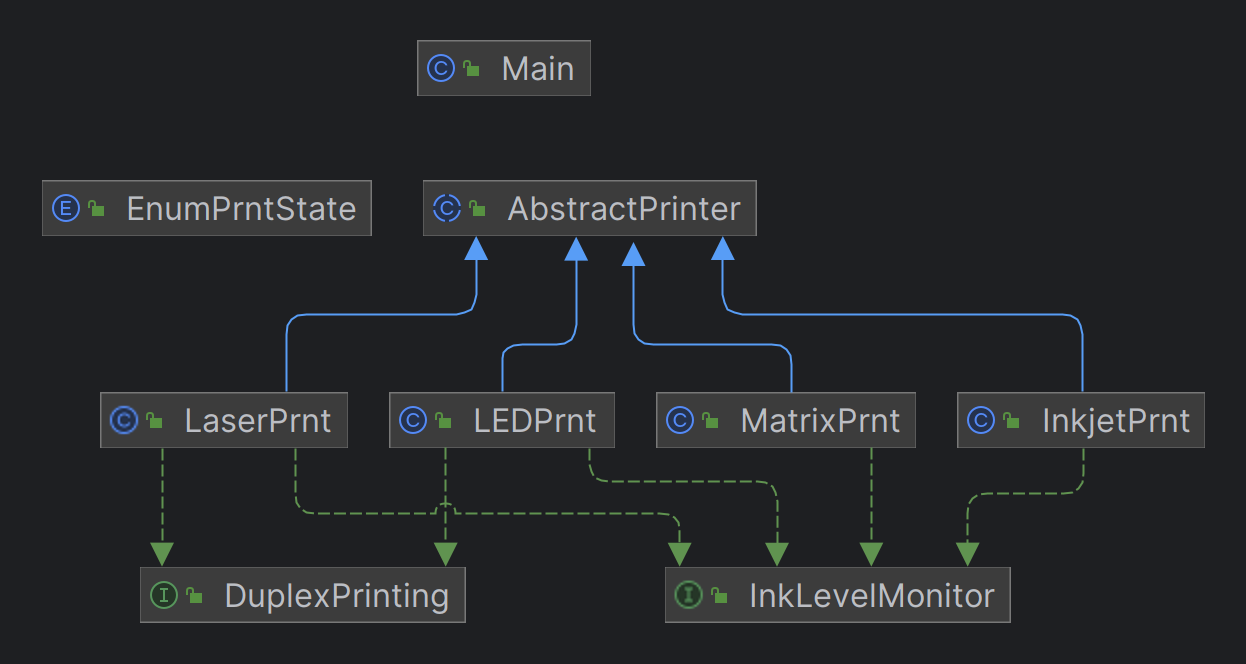


Рисунок 1 – згенерована діаграма у середовищі розробки

Реалізувати програмний код класів на мові Java.

При написанні коду були виконані умови: абстрактний клас з характерними властивостями та рисами поведінки. Підкласи об’єктів, які успадковують абстрактний клас та реалізовують додаткову поведінку об’єкту зі створених інтерфейсів. Фрагменти фунціоналу коду описані коментарями. Програмний код класів можна переглянути в додатках А-З.

Тестування програми

Тестувати будемо правильними значеннями. Всі тестування були виконані за допомогою введення даних в методи файлі *main*. Результат виведення в консоль поданий нижче.

Model Type: Matrix Model 1

Paper Format: A4

Weight: 5.5

Price: 100.0

Print Speed: 30

Mode: Normal

Interface Type: COM PORT

State: Connected; Ready

Connected; Ready

Initial Ink Level: 100.0%

Is Ink Low? false

..::\* AaBbCcDdEeFf 1234567890 !@#$%^&\*()\_+ \*::..

After printing, Ink Level: 96.4%

Is Ink Low? false

After reloading ink, Ink Level: 100.0%

Is Ink Low? false

After disconnecting, Printer State: Disconnected; Offline

After error, Printer State: Stopped; Error

After connecting, Printer State: Connected; Ready

Model Type: LED Model 1

Paper Format: A4

Weight: 4.0

Price: 150.0

Print Speed: 50

Mode: High Quality

Interface Type: Wi-Fi

State: Connected; Ready

Connected; Ready

Initial Toner Level: 100.0%

Is Toner Low? false

◍◍ AaBbCcDdEeFf 1234567890 !@#$%^&\*()\_+ ◍◍

◍◍ AaBbCcDdEeFf 1234567890 !@#$%^&\*()\_+ ◍◍+\_)(\*&^%$#@! 0987654321 fFeEdDcCbBa

After printing, Toner Level: 78.4%

Is Toner Low? false

After reloading toner, Toner Level: 100.0%

Is Toner Low? false

After disconnecting, Printer State: Disconnected; Offline

After error, Printer State: Stopped; Error

After connecting, Printer State: Connected; Ready

Model Type: Laser Model 1

Paper Format: A4

Weight: 6.0

Price: 200.0

Print Speed: 40

Mode: High Quality

Interface Type: USB

State: Connected; Ready

Connected; Ready

Initial Ink Level: 100.0%

Is Ink Low? false

**AaBbCcDdEeFf 1234567890 !@#$%^&\*()\_+**

**AaBbCcDdEeFf 1234567890 !@#$%^&\*()\_++\_)(\*&^%$#@! 0987654321 fFeEdDcCbBaA**

After printing, Ink Level: 56.79999999999999%

Is Ink Low? false

After reloading ink, Ink Level: 100.0%

Is Ink Low? false

After disconnecting, Printer State: Disconnected; Offline

After error, Printer State: Stopped; Error

After connecting, Printer State: Connected; Ready

Model Type: Inkjet Model 1

Paper Format: A4

Weight: 4.0

Price: 150.0

Print Speed: 20

Mode: Normal

Interface Type: Wi-Fi

State: Connected; Ready

Connected; Ready

Initial Ink Level: 100.0%

Is Ink Low? false

AaBbCcDdEeFf 1234567890 !@#$%^&\*()\_+

Model Type: Laser Model 1

Paper Format: A4

Weight: 6.0

Price: 200.0

Print Speed: 40

Mode: High Quality

Interface Type: USB

State: Connected; Ready

After printing, Ink Level: 71.2%

Is Ink Low? false

After reloading ink, Ink Level: 100.0%

Is Ink Low? false

After disconnecting, Printer State: Disconnected; Offline

After error, Printer State: Stopped; Error

After connecting, Printer State: Connected; Ready

Process finished with exit code 0

Генерація документації на розроблений додаток

Для цього необхідно скористатись командою **javadoc**. В даному випадку вона буде написана таким чином: ***javadoc -d docs Main.java***. Результат виконання команди для створення документації зображено на рисунку 5.

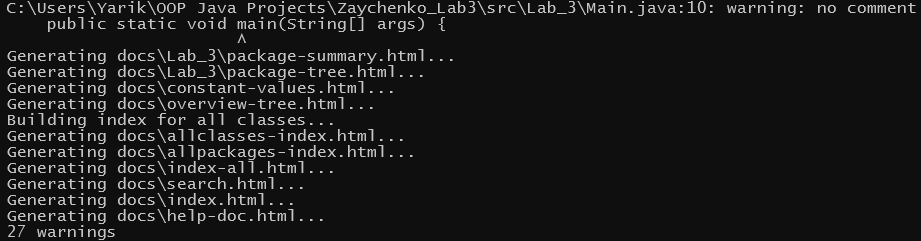
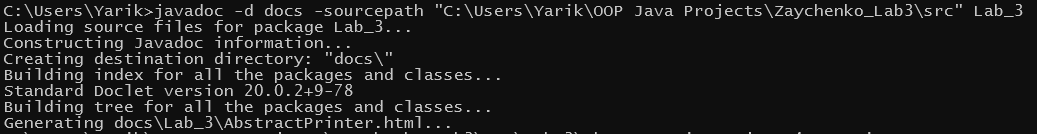


Рисунок 5 – створення документації додатку стандартними командами JRE



Рисунок 6 – отримані файли документації на додаток

Висновок

В ході виконання лабораторної роботи було проведено дослідження та практична реалізація концепції об'єктно-орієнтованого програмування (ООП) на мові Java. Спочатку було вивчено теоретичний матеріал, пов'язаний з темою лабораторної роботи, використовуючи конспект лекцій, рекомендовану літературу та документацію, що входить до складу пакету Java SE JDK.

Після теоретичного осмислення теми було розроблено концептуальну модель для заданої категорії об'єктів у вигляді словесного опису їх основних властивостей та поведінки. На основі цієї моделі було спроектовано абстрактний клас, у якому було визначено загальні характеристики для об'єктів цієї категорії.

Далі, були визначені основні риси поведінки об'єктів та представлені у вигляді набору методів, визначених у абстрактному класі. З метою розширення функціональності та забезпечення можливості реалізації додаткової поведінки було спроектовано два інтерфейси, методи яких було визначено на абстрактному рівні.

Підкласи були розроблені для успадкування від абстрактного класу та реалізації методів, визначених у ньому, а також для реалізації додаткової поведінки через інтерфейси. Для відображення структурних відносин між класами та інтерфейсами була побудована діаграма класів UML.

Наприкінці, було реалізовано програмний код класів та інтерфейсів на мові Java з використанням коментарів та дескрипторів документування для опису їх структури та функціоналу. Додатково було розроблено об'єкти для підкласів та продемонстровано їх роботу за допомогою методів інтерфейсів, реалізованих у підкласах, з використанням поліморфізму.

Нарешті, розроблена програма була запущена на виконання за допомогою середовища Intellij IDEA та протестована, що підтвердило правильність функціонування розробленого додатку. Документація на розроблений додаток була згенерована за допомогою утиліти javadoc.exe, що дозволить зберігати та підтримувати зрозумілу та зроблену згідно стандартів документацію для подальшого розвитку проекту.

Контрольні питання

1. З якою метою застосовуються інтерфейси у мові Java? Інтерфейси у мові Java застосовуються для опису специфікації поведінки об'єктів без їх конкретної реалізації.
2. Який синтаксис має оголошення інтерфейсу у мові Java? ключове слово interface, його назва, фігурні дужки, а в них оголошення методів та/або полів.
3. Які модифікатори можливо використовувати для оголошення полів інтерфейсу? Для оголошення полів інтерфейсу можна використовувати модифікатори **public**, **static**, **final**.
4. Які модифікатори можливо використовувати для оголошення методів інтерфейсу? Для оголошення методів інтерфейсу можна використовувати лише модифікатори public та abstract. За замовчуванням методи інтерфейсу є абстрактними.
5. Яку кількість інтерфейсів може успадковувати інтерфейс? Інтерфейс може успадковувати будь-яку кількість інших інтерфейсів.
6. Яку кількість інтерфейсів може реалізувати клас? Клас може реалізувати будь-яку кількість інтерфейсів.
7. З яким модифікатором повинен оголошуватися клас у якому визначені методи, які не мають реалізації? Клас, у якому визначені методи, що не мають реалізації, повинен бути оголошений з модифікатором **abstract**.
8. Що може зберігатися в змінній, яка має тип інтерфейсу? В змінній, яка має тип інтерфейсу, може зберігатися будь-який об'єкт, який реалізує цей інтерфейс.

*Додаток А*Машинний лістинг файлу **Main.java**

package Lab\_3;

import Lab\_3.EnumPrntState;

import Lab\_3.PrintersTypesClasses.InkjetPrnt;

import Lab\_3.PrintersTypesClasses.LEDPrnt;

import Lab\_3.PrintersTypesClasses.LaserPrnt;

import Lab\_3.PrintersTypesClasses.MatrixPrnt;

public class Main {

public static void main(String[] args) {

String testDoc = "AaBbCcDdEeFf 1234567890 !@#$%^&\*()\_+";

try {

MatrixPrnt matrixPrinter = new MatrixPrnt("Matrix Model 1", "A4", 5.5, 100.0, 30,

"Normal", "COM PORT", EnumPrntState.READY.getEnumState());

System.out.println(matrixPrinter.getProperties());

System.out.println(matrixPrinter.getState());

System.out.println("Initial Ink Level: " + matrixPrinter.getInkLevel());

System.out.println("Is Ink Low? " + matrixPrinter.isLowInk());

System.out.println(matrixPrinter.print(testDoc)); // Друк документа

System.out.println("After printing, Ink Level: " + matrixPrinter.getInkLevel());

System.out.println("Is Ink Low? " + matrixPrinter.isLowInk());

matrixPrinter.reloadInk(); // Перезавантаження чорнила

System.out.println("After reloading ink, Ink Level: " + matrixPrinter.getInkLevel());

System.out.println("Is Ink Low? " + matrixPrinter.isLowInk());

matrixPrinter.disconnect(); // Відключення принтера

System.out.println("After disconnecting, Printer State: " + matrixPrinter.getState());

matrixPrinter.error(); // Стан помилки

System.out.println("After error, Printer State: " + matrixPrinter.getState());

matrixPrinter.connect(); // Підключення принтера

System.out.println("After connecting, Printer State: " + matrixPrinter.getState());

LEDPrnt ledPrinter = new LEDPrnt("LED Model 1", "A4", 4.0, 150.0, 50,

"High Quality", "Wi-Fi", EnumPrntState.READY.getEnumState());

System.out.println(ledPrinter.getProperties());

System.out.println(ledPrinter.getState());

System.out.println("Initial Toner Level: " + ledPrinter.getInkLevel());

System.out.println("Is Toner Low? " + ledPrinter.isLowInk());

System.out.println(ledPrinter.print(testDoc)); // Друк документа

ledPrinter.enableDuplexPrinting();

System.out.println(ledPrinter.print(testDoc)); // Друк документа

System.out.println("After printing, Toner Level: " + ledPrinter.getInkLevel());

System.out.println("Is Toner Low? " + ledPrinter.isLowInk());

ledPrinter.reloadInk(); // Перезавантаження тонера

System.out.println("After reloading toner, Toner Level: " + ledPrinter.getInkLevel());

System.out.println("Is Toner Low? " + ledPrinter.isLowInk());

ledPrinter.disconnect(); // Відключення принтера

System.out.println("After disconnecting, Printer State: " + ledPrinter.getState());

ledPrinter.error(); // Стан помилки

System.out.println("After error, Printer State: " + ledPrinter.getState());

ledPrinter.connect(); // Підключення принтера

System.out.println("After connecting, Printer State: " + ledPrinter.getState());

LaserPrnt laserPrinter = new LaserPrnt("Laser Model 1", "A4", 6.0, 200.0, 40,

"High Quality", "USB", EnumPrntState.READY.getEnumState());

System.out.println(laserPrinter.getProperties());

System.out.println(laserPrinter.getState());

System.out.println("Initial Ink Level: " + laserPrinter.getInkLevel());

System.out.println("Is Ink Low? " + laserPrinter.isLowInk());

System.out.println(laserPrinter.print(testDoc)); // Друк документа

laserPrinter.enableDuplexPrinting();

System.out.println(laserPrinter.print(testDoc)); // Друк документа

System.out.println("After printing, Ink Level: " + laserPrinter.getInkLevel());

System.out.println("Is Ink Low? " + laserPrinter.isLowInk());

laserPrinter.reloadInk(); // Перезавантаження чорнила

System.out.println("After reloading ink, Ink Level: " + laserPrinter.getInkLevel());

System.out.println("Is Ink Low? " + laserPrinter.isLowInk());

laserPrinter.disconnect(); // Відключення принтера

System.out.println("After disconnecting, Printer State: " + laserPrinter.getState());

laserPrinter.error(); // Стан помилки

System.out.println("After error, Printer State: " + laserPrinter.getState());

laserPrinter.connect(); // Підключення принтера

System.out.println("After connecting, Printer State: " + laserPrinter.getState());

InkjetPrnt inkjetPrinter = new InkjetPrnt("Inkjet Model 1", "A4", 4.0, 150.0, 20,

"Normal", "Wi-Fi", EnumPrntState.READY.getEnumState());

System.out.println(inkjetPrinter.getProperties());

System.out.println(inkjetPrinter.getState());

System.out.println("Initial Ink Level: " + inkjetPrinter.getInkLevel());

System.out.println("Is Ink Low? " + inkjetPrinter.isLowInk());

System.out.println(inkjetPrinter.print(testDoc)); // Друк документа

System.out.println("After printing, Ink Level: " + inkjetPrinter.getInkLevel());

System.out.println("Is Ink Low? " + inkjetPrinter.isLowInk());

inkjetPrinter.reloadInk(); // Перезавантаження чорнила

System.out.println("After reloading ink, Ink Level: " + inkjetPrinter.getInkLevel());

System.out.println("Is Ink Low? " + inkjetPrinter.isLowInk());

inkjetPrinter.disconnect(); // Відключення принтера

System.out.println("After disconnecting, Printer State: " + inkjetPrinter.getState());

inkjetPrinter.error(); // Стан помилки

System.out.println("After error, Printer State: " + inkjetPrinter.getState());

inkjetPrinter.connect(); // Підключення принтера

System.out.println("After connecting, Printer State: " + inkjetPrinter.getState());

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

}

*Додаток Б*Машинний лістинг файлу **EnumPrntState.java**

package Lab\_3;

public enum EnumPrntState {

READY("Connected; Ready"), // Готовий

PRINTING("Printing"), // Друк

ERROR("Stopped; Error"), // Помилка

OFFLINE("Disconnected; Offline"); // Відключений

// Змінна для зберігання назви стану

private final String state;

// Конструктор

EnumPrntState(String state) {

this.state = state;

}

// Метод для отримання назви стану

public String getEnumState() {

return state;

}

}

*Додаток В*Машинний лістинг файлу **AbstractPrinter.java**

package Lab\_3;

// Абстрактний клас для принтерів

public abstract class AbstractPrinter {

protected String modelType;

protected String paperFormat;

protected double weight;

protected double price;

protected int printSpeed;

protected String mode;

protected String interfaceType;

protected String state;

// Конструктор

public AbstractPrinter(String modelType, String paperFormat, double weight, double price, int printSpeed,

String mode, String interfaceType, String state) {

this.modelType = modelType;

this.paperFormat = paperFormat;

this.weight = weight;

this.price = price;

this.printSpeed = printSpeed;

this.mode = mode;

this.interfaceType = interfaceType;

this.state = state;

}

// Методи для визначення та зміни властивостей

// зміна властивостей

public void changeProperties(String modelType, String paperFormat, double weight, double price,

int printSpeed, String mode, String interfaceType, String state) {

this.modelType = modelType;

this.paperFormat = paperFormat;

this.weight = weight;

this.price = price;

this.printSpeed = printSpeed;

this.mode = mode;

this.interfaceType = interfaceType;

this.state = state;

}

// отримання властивостей

public String getProperties() {

StringBuilder properties = new StringBuilder();

properties.append("Model Type: ").append(modelType).append("\n");

properties.append("Paper Format: ").append(paperFormat).append("\n");

properties.append("Weight: ").append(weight).append("\n");

properties.append("Price: ").append(price).append("\n");

properties.append("Print Speed: ").append(printSpeed).append("\n");

properties.append("Mode: ").append(mode).append("\n");

properties.append("Interface Type: ").append(interfaceType).append("\n");

properties.append("State: ").append(state).append("\n");

return properties.toString();

}

// визначення стану принтера

public String getState() {

return state;

}

// Дії принтера

public abstract String print(String document); // Метод для друку документа

// Метод для підключення принтера

public void connect() {

this.state = EnumPrntState.READY.getEnumState();

}

// Метод для відключення принтера

public void disconnect() {

this.state = EnumPrntState.OFFLINE.getEnumState();

}

public void error() {

this.state = EnumPrntState.ERROR.getEnumState();

} // Метод для відображення помилки друку

public static final String RESET = "\033[0m"; // Повернення до стандартних налаштувань

}

}

*Додаток Г*Машинний лістинг файлу **MatrixPrnt.java**

package Lab\_3.PrintersTypesClasses;

import Lab\_3.\*;

// Підклас для матричного принтера

public class MatrixPrnt extends AbstractPrinter implements InkLevelMonitor {

public MatrixPrnt(String modelType, String paperFormat, double weight, double price, int printSpeed,

String mode, String interfaceType, String state) {

super(modelType, paperFormat, weight, price, printSpeed, mode, interfaceType, state);

this.reloadInk();

}

@Override

public String print(String document) {

if(isLowInk()) throw new RuntimeException("Unable to print. Low Ink");

decreaseInkLevel(document);

return "..::\* " + document + " \*::..";

}

private double inkLevel;

@Override

public String getInkLevel() {

return inkLevel + "%";

}

@Override

public boolean isLowInk() {

return inkLevel < 10;

}

@Override

public void decreaseInkLevel(String doc) {

int SymbCount = doc.length();

double oneSymbInk=0.1;

double inkSpend = SymbCount\*oneSymbInk;

inkLevel-=inkSpend;

}

@Override

public void reloadInk() {

this.inkLevel = 100;

}

}

*Додаток Ґ*Машинний лістинг файлу **LEDPrnt.java**

package Lab\_3.PrintersTypesClasses;

import Lab\_3.AbstractPrinter;

public class LEDPrnt extends AbstractPrinter implements DuplexPrinting, InkLevelMonitor {

public LEDPrnt(String modelType, String paperFormat, double weight, double price, int printSpeed, String mode, String interfaceType, String state) {

super(modelType, paperFormat, weight, price, printSpeed, mode, interfaceType, state);

this.reloadInk();

}

@Override

public String print(String document) {

if(isLowInk()) throw new RuntimeException("Unable to print. Low Ink");

decreaseInkLevel(document);

String res ="";

res+= "◍◍ " + document + " ◍◍";

if (duplexPrintingEnabled)

res +=DPrnt(document);

return res;

}

@Override

public String DPrnt(String doc) {

String res = "";

for (int i = doc.length() - 1; i >= 0; i--) {

res += doc.charAt(i);

}

return res;

}

private boolean duplexPrintingEnabled = false;

@Override

public void enableDuplexPrinting() {

duplexPrintingEnabled = true;

}

private double inkLevel;

@Override

public String getInkLevel() {

return inkLevel + "%";

}

@Override

public boolean isLowInk() {

return inkLevel < 15;

}

@Override

public void decreaseInkLevel(String doc) {

int SymbCount = doc.length();

double oneSymbInk = 0.3;

double inkSpend = SymbCount \* oneSymbInk;

inkLevel -= inkSpend;

}

@Override

public void reloadInk() {

inkLevel = 100;

}

}

*Додаток Д*Машинний лістинг файлу **LaserPrnt.java**

package Lab\_3.PrintersTypesClasses;

import Lab\_3.AbstractPrinter;

public class LaserPrnt extends AbstractPrinter implements DuplexPrinting, InkLevelMonitor {

public LaserPrnt(String modelType, String paperFormat, double weight, double price, int printSpeed, String mode, String interfaceType, String state) {

super(modelType, paperFormat, weight, price, printSpeed, mode, interfaceType, state);

this.reloadInk();

}

public static final String BOLD = "\033[1m"; // Жирний текст

@Override

public String print(String document) {

if(isLowInk()) throw new RuntimeException("Unable to print. Low Ink");

decreaseInkLevel(document);

String res = "";

res +=BOLD+document+RESET;

if (duplexPrintingEnabled)

res+=DPrnt(document);

return res;

}

private boolean duplexPrintingEnabled = false;

@Override

public void enableDuplexPrinting() {

duplexPrintingEnabled = true;

}

@Override

public String DPrnt(String doc) {

decreaseInkLevel(doc);

String res = "";

for (int i = doc.length() - 1; i >= 0; i--) {

res += doc.charAt(i);

}

return BOLD+res+RESET;

}

private double inkLevel;

@Override

public String getInkLevel() {

return inkLevel+"%";

}

@Override

public boolean isLowInk() {

return inkLevel < 20;

}

@Override

public void decreaseInkLevel(String doc) {

int SymbCount = doc.length();

double oneSymbInk=0.4;

double inkSpend = SymbCount\*oneSymbInk;

inkLevel-=inkSpend;

}

@Override

public void reloadInk() {

inkLevel=100;

}

}

*Додаток Е*Машинний лістинг файлу **InkLevelMonitor.java**

package Lab\_3.PrintersTypesClasses;

// Інтерфейс для моніторингу рівня чорнила або тонера

public interface InkLevelMonitor {

String getInkLevel(); // Метод для отримання рівня чорнила або тонера (у відсотках)

boolean isLowInk(); // Метод для перевірки, чи є рівень чорнила або тонера низьким

void decreaseInkLevel(String doc); // Метод для зменшення рівня чорнила на величину inkSpend

void reloadInk(); // Заправка чорнил

}

*Додаток Є*Машинний лістинг файлу **InkjetPrnt.java**

package Lab\_3.PrintersTypesClasses;

import Lab\_3.AbstractPrinter;

public class InkjetPrnt extends AbstractPrinter implements InkLevelMonitor{

public InkjetPrnt(String modelType, String paperFormat, double weight, double price, int printSpeed, String mode, String interfaceType, String state) {

super(modelType, paperFormat, weight, price, printSpeed, mode, interfaceType, state);

this.reloadInk();

}

public static final String YELLOW = "\033[0;33m"; // Жовтий

@Override

public String print(String document) {

if(isLowInk()) throw new RuntimeException("Unable to print. Low Ink");

decreaseInkLevel(document);

return YELLOW+document+RESET;

}

private double inkLevel;

@Override

public String getInkLevel() {

return inkLevel+"%";

}

@Override

public boolean isLowInk() {

return inkLevel<30;

}

@Override

public void decreaseInkLevel(String doc) {

int SymbCount = doc.length();

double oneSymbInk=0.8;

double inkSpend = SymbCount\*oneSymbInk;

inkLevel-=inkSpend;

}

@Override

public void reloadInk() {

inkLevel=100;

}

}

*Додаток Ж*Машинний лістинг файлу **DuplexPrinting.java**

package Lab\_3.PrintersTypesClasses;

// Інтерфейс для підтримки дуплексного друку

public interface DuplexPrinting {

void enableDuplexPrinting(); // Метод для увімкнення дуплексного друку

String DPrnt(String doc); // метод зворотнього друку

}