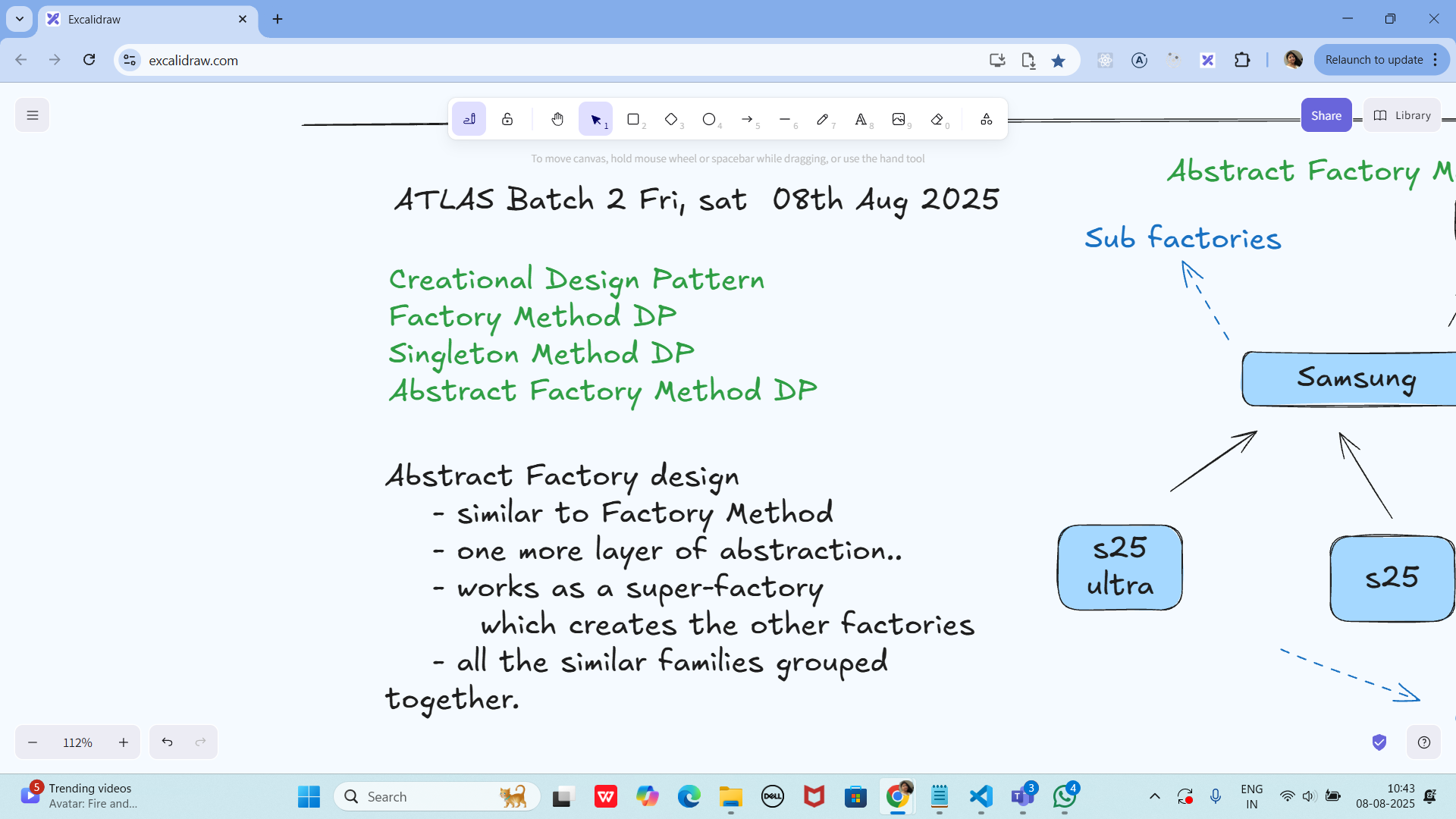
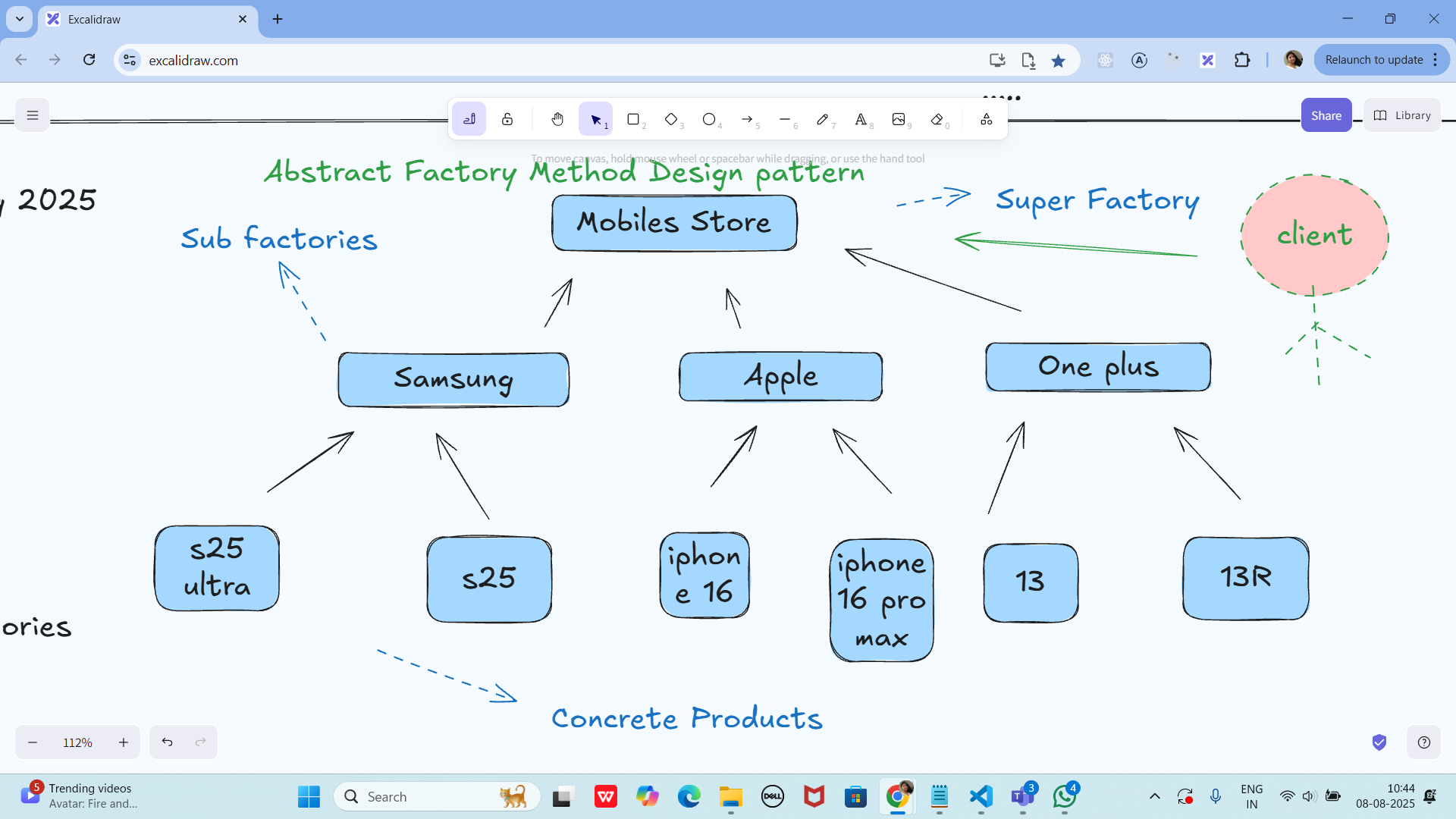
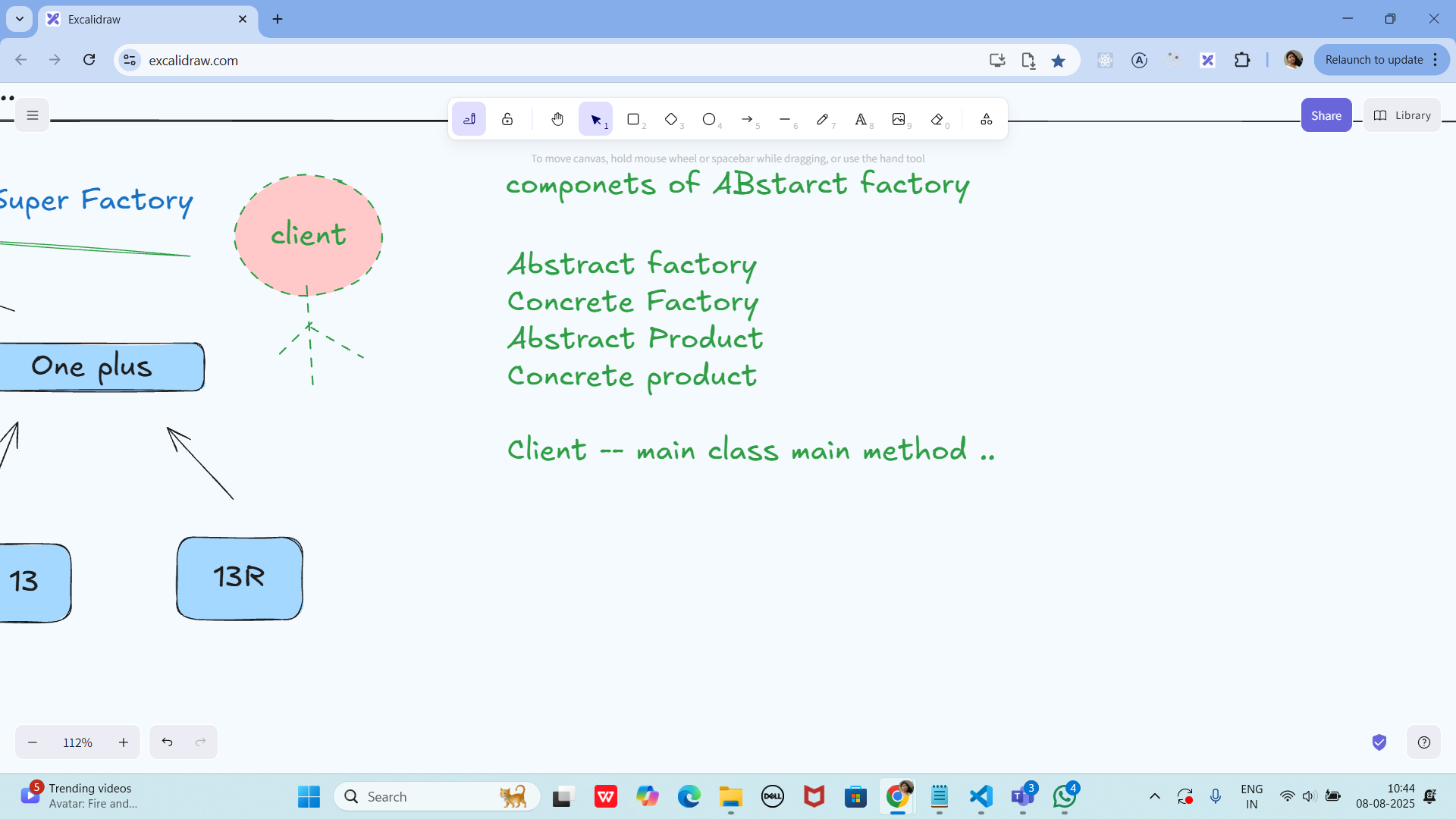
Day 24 - 08th Aug 2025

Abstract Factory Design pattern







Task 01:

Apple.java

package Demo\_Codes.Module\_03\_OOAD.AbstractFactoryDP.AbstractFactoryDpPack;

// create samsung and oneplus classes..

public class Apple {

private Apple() {

public static Mobile getMobile(String model) {

if(Object.equals(model, "iphone16")) {

return new Mobile("Here is your iphone 16");

}

else if (Object.equals(model, "iphone16MaxPro")) {

return new Mobile("Here is your iphone 16 Max Pro");

}

return new NoMobile();

}

}

}

Mobile.java

package Demo\_Codes.Module\_03\_OOAD.AbstractFactoryDP.AbstractFactoryDpPack;

public class Mobile {

String desc;

public Mobile(String model){

this.desc = model;

}

public void getDesc() {

System.out.println("this.desc");

}

}

MobileStore.java

package Demo\_Codes.Module\_03\_OOAD.AbstractFactoryDP.AbstractFactoryDpPack;

// super factory

public class MobileStore {

private MobileStore() {

System.out.println("hello welcome to the world of Mobile");

}

public static Mobile getMobile(String model) {

if(Object.equals(brand, "Apple")) {

System.out.println("Here are your Apple Models");

retrun Apple.getMobile(model);

} // else if (Object.equals(brand, "Samsung")) {

// System.out.println("Here are your Samsung Models");

// retrun Samsung.getMobile(model); }

return new NoMobile();

}

}

NoMobile.java

package Demo\_Codes.Module 03 OOAD.AbstractFactoryDP.AbstractFactoryDpPack;

public class NoMobile extends Mobile{

public NoMobile() {

super("sorry invalid model");

}

}

ClientAbstractFactory.java

package Demo\_Codes.Module 03 OOAD.AbstractFactoryDP.AbstractFactoryDpPack;

public class ClientAbstractFactory {

public static void main(String[] args){

Mobile Mobj = MobileStore.getMobile("Apple","iphone16");

Mobj.getDesc();

System.out.println("...");

}

}

Builder method Design pattern

Task 02:

// product Component

class Laptop {

int memory;

int storage;

// graphic card,processor...

Laptop() {....}

getters()..

}

public interface LaptopBuilder {

LaptopBuilder buildMemory(int memory);

LaptopBuilder buildStorage(int storage);

Laptop build();

}

public class LaptopConcreteBuilder implements LaptopBuilder

{

private Laptop laptop;

public LaptopConcreteBuilder() {

this.laptop = new Laptop();

}

@Override

public LaptopBuilder buildMemory(int memory) {

laptop.setMemory(memory);

return this;

}

@Override

public LaptopBuilder buildStorage(int storage) {

laptop.setStorage(storage);

return this;

}

@Override

public LaptopBuilder build() {

return laptop;

}

}

//Director (optional)

public class LaptopDirector {

private LaptopBuilder laptopBuilder;

LaptopDirector(LaptopBuilder laptopBuilder) {

this.laptopBuilder = laptopBuilder;

}

public Laptop constructLaptop() {

return laptopBuilder

.buildMemory

.buildStorage

.build();

}

}

public class ClientBuildMethodDP {

psvm (String....) {

LaptopBuilder lbobj = new LaptopConcreteBuilder();

LaptopDirector dir = new LaptopDirector(lbobj);

Laptop lobj = dir.constructLaptop();

sout(lobj);

}

}

Task 03:

package com.example;

//Prototype interface

public interface Colors {

Colors clone();

String getName();

void setName(String name);

}

package com.example;

public class BlackConcreteprototype implements Colors{

private String name;

public BlackConcreteprototype() {

System.out.println(" BlackConcreteprototype constructor is called");

}

public BlackConcreteprototype(String name) {

this.name = name;

}

public Colors clone() {

return new BlackConcreteprototype(this.name);

}

public String getName() {

return name;

}

public void setName(String name){

this.name = name;

}

}

package com.example;

public class WhiteConcretePrototype implements Colors{

private String name;

public WhiteConcretePrototype() {

System.out.println(" WhiteConcretePrototype constructor is called");

}

public WhiteConcretePrototype(String name) {

this.name = name;

}

public Colors clone() {

return new WhiteConcretePrototype(this.name);

}

public String getName() {

return name;

}

public void setName(String name){

this.name = name;

}

}

package com.example;

// client code of Prototype method Design pattern

public class Main {

public static void main(String[] args) {

Colors BlackPrototypeObj = new BlackConcreteprototype("Black Color");

Colors WhitePrototypeObj = new WhiteConcretePrototype("white color");

Colors clonedBlackObj = BlackPrototypeObj.clone();

Colors clonedWhiteObj = WhitePrototypeObj.clone();

clonedBlackObj.setName("dark color");

clonedWhiteObj.setName("light color");

System.out.println("black color is " + clonedBlackObj.getName());

System.out.println("while color is "+ clonedWhiteObj.getName());

}

}

Prototype Method Design pattern

prototype interface -- Colors

Concrete prototype --- class

WhiteConcretePrototype

constructor()

constructor(String...)

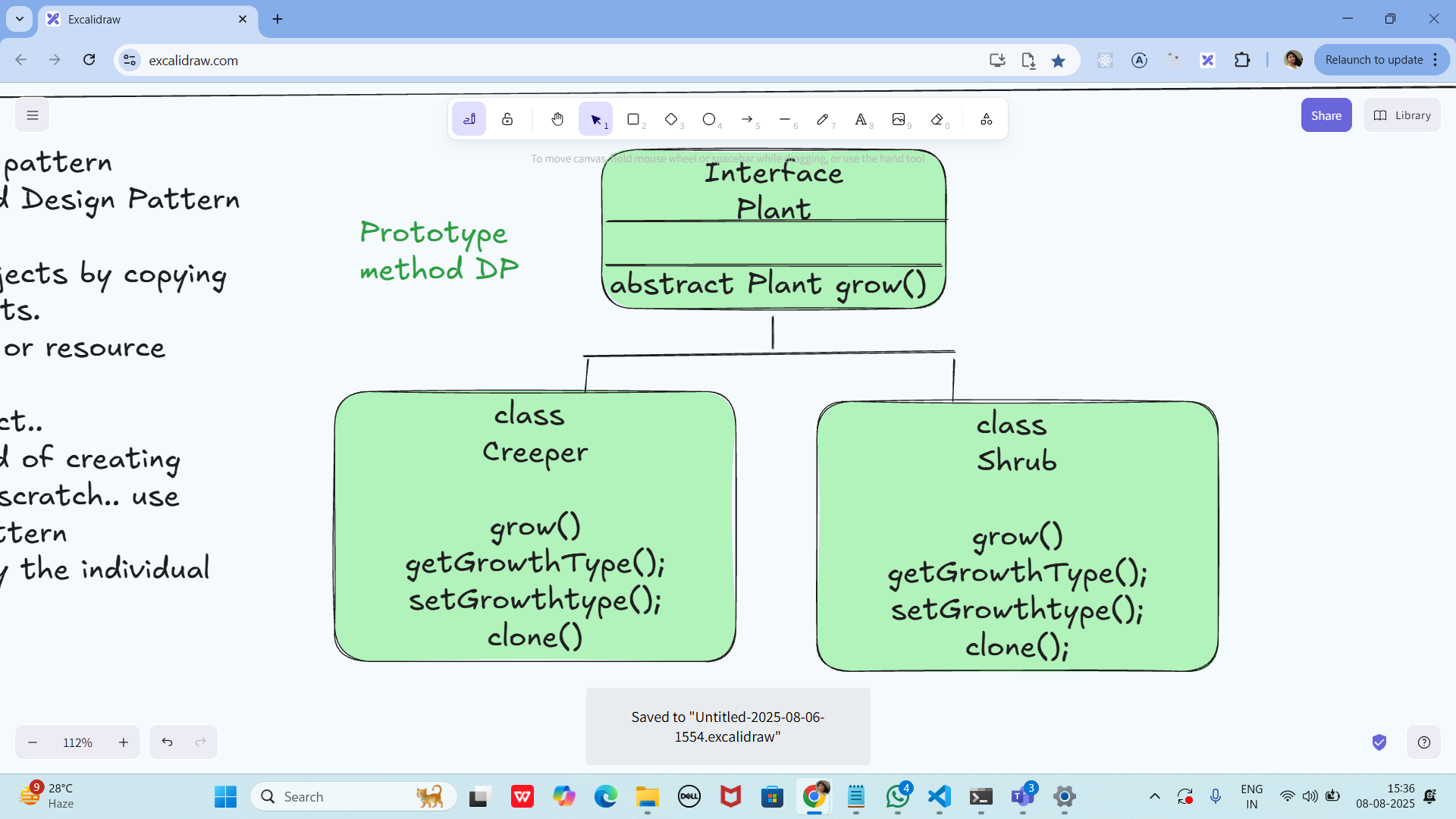
clone()

BlackConcreteprototype

client code

Home tasks 01

Develop the below prototype method DP..



Info Box

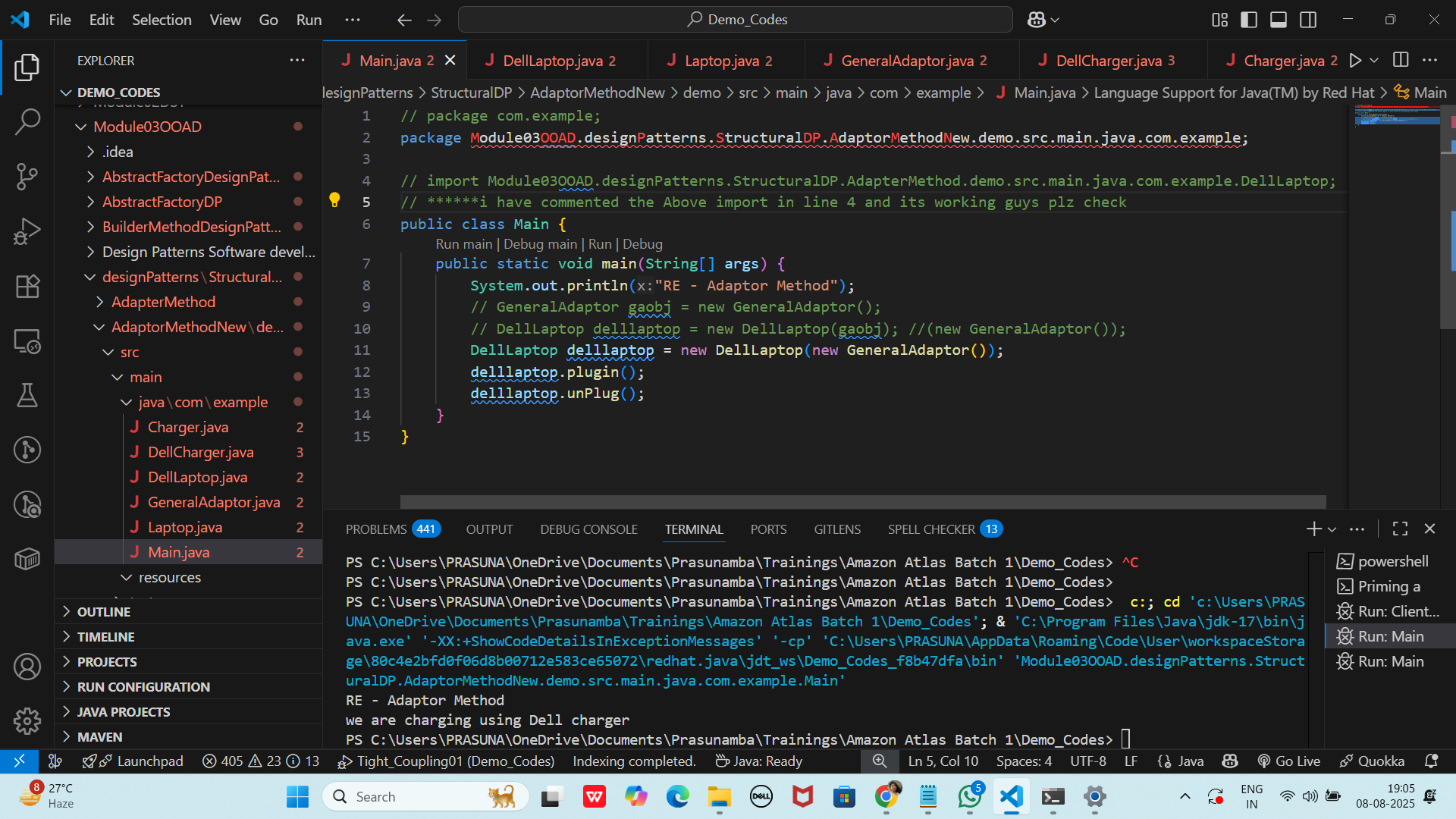
Excalidraw updated at 10.43

<https://excalidraw.com/#json=PfNv_D1GXzAH0yPsXLqss,ab_5Dl0GZWRrpJtLtyHOVg>

Code for reference

<https://drive.google.com/drive/folders/1LwhNov1s1-vHzF9GPAObLSnP9kAvipmw?usp=sharing>

// \*\*\*\*\*\*i have commented the Above import in line 4 and its working guys plz check



// \*\*\*\*\*\*i have commented the Above import in line 4 and its working guys plz check

<https://excalidraw.com/#json=ZpErQ-rSmDqNvoGQ_lz2y,jKEkwwFeT5D-IKsaMs4WGA>

Updated excalidraw - 9.45 am 9th Aug 2025