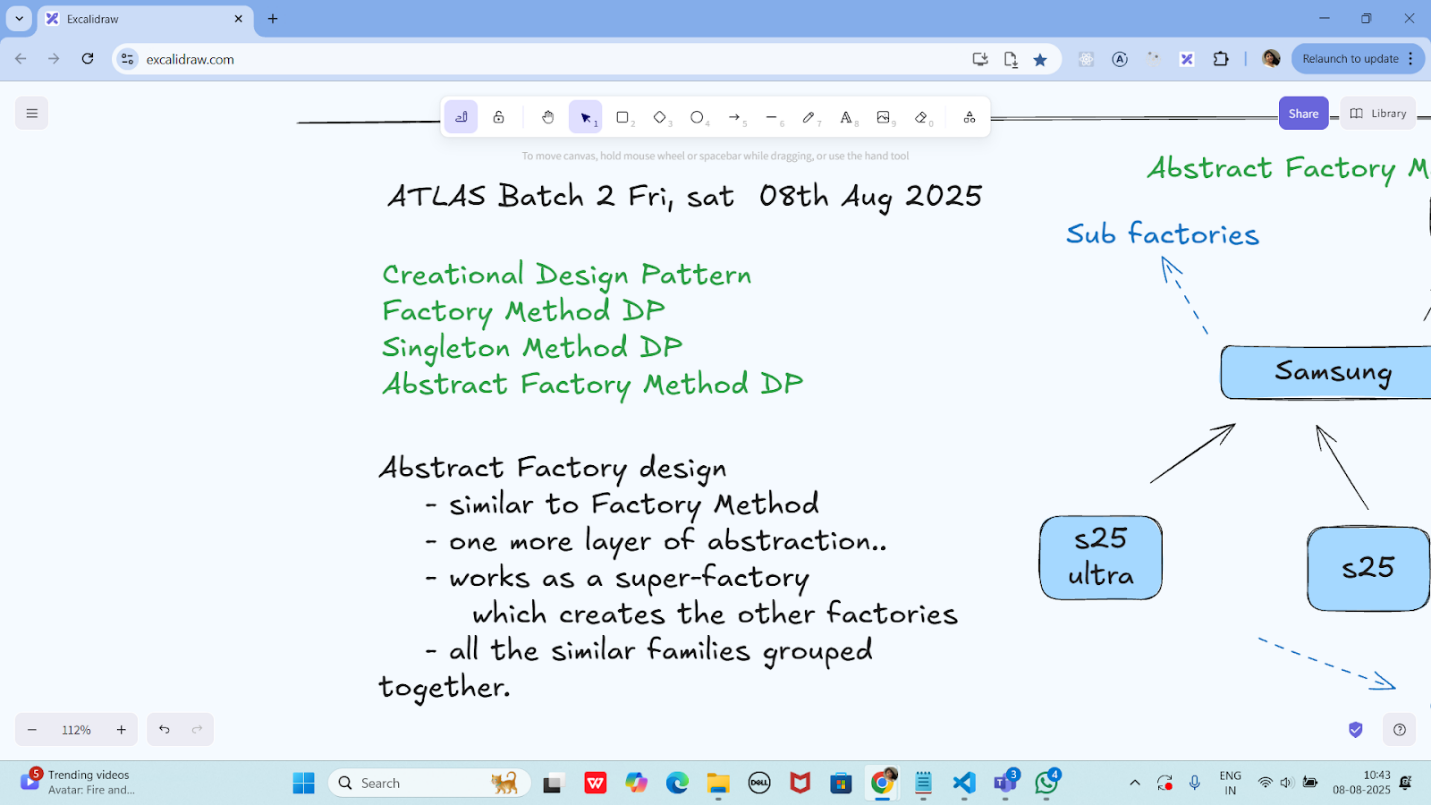
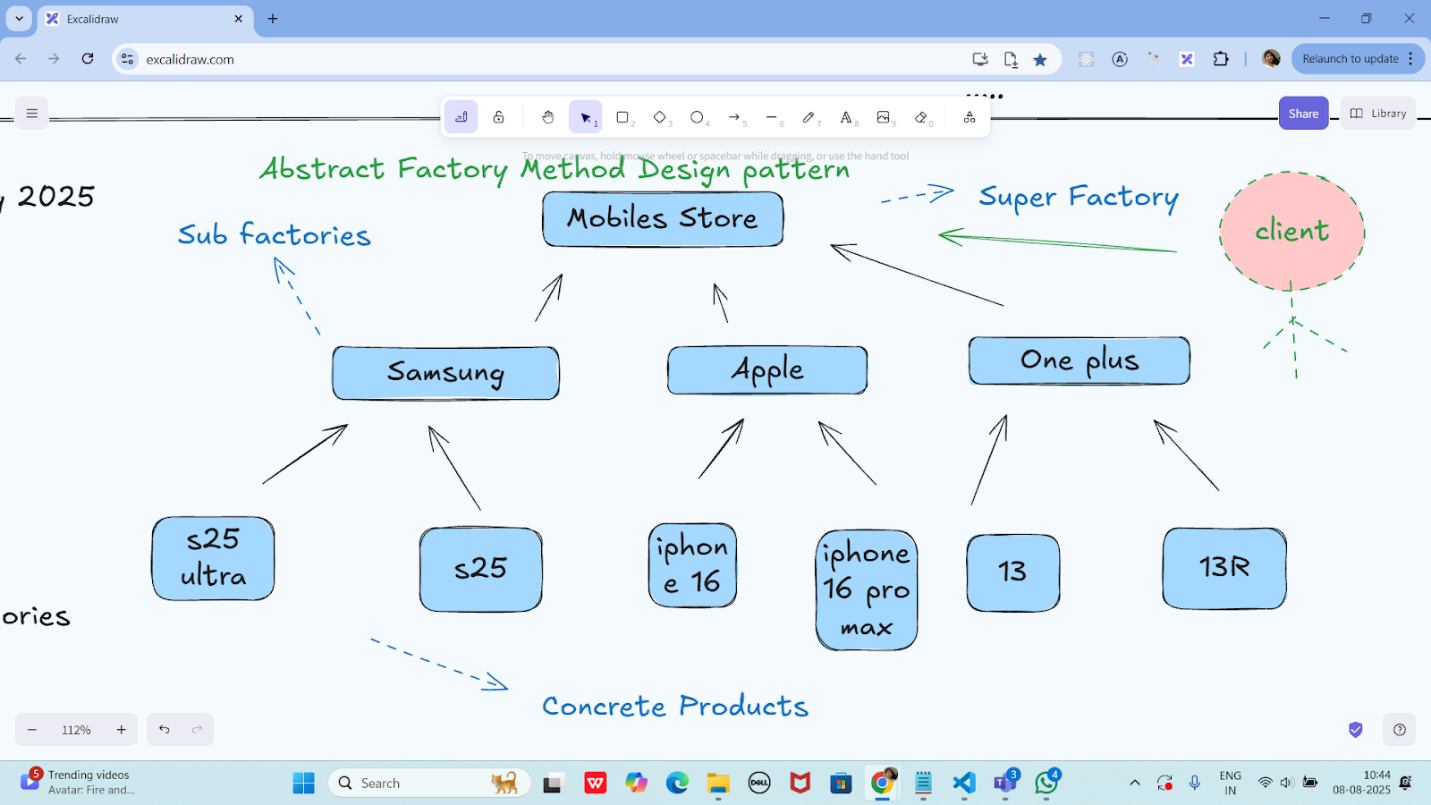
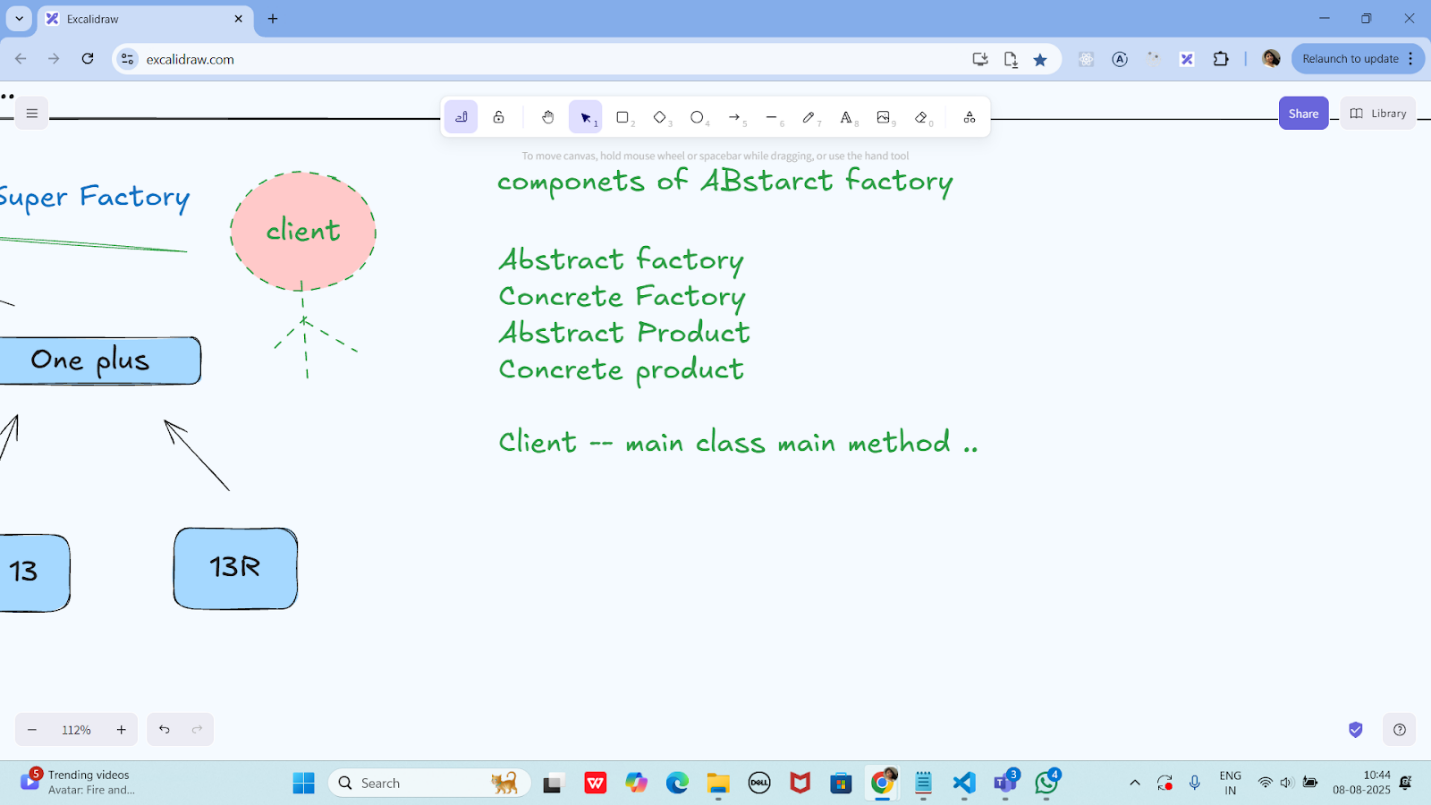
Day 24 - 08th Aug 2025

Abstract Factory Design pattern







Task 01:

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

import java.util.Objects;

public class Apple {

   private Apple() {

       // Prevent instantiation

   }

   public static Mobile getMobile(String model) {

       if (Objects.*equals*(model, "iphone16")) {

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

       } else if (Objects.*equals*(model, "iphone16MaxPro")) {

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

       }

       return new NoMobile();

   }

}

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("...");

   }

}

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);

   }

}

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

import java.util.Objects;

public class MobileStore {

   private MobileStore() {

       System.*out*.println("Hello, welcome to the world of Mobile");

   }

   public static Mobile getMobile(String brand, String model) {

       if (Objects.*equals*(brand, "Apple")) {

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

           return Apple.*getMobile*(model);

       }

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

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

       //     return Samsung.getMobile(model);

       // }

       return new NoMobile();

   }

}

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

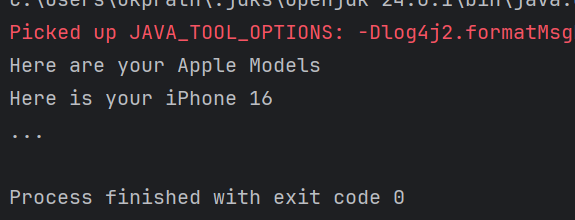
public class NoMobile extends Mobile {

   public NoMobile() {

       super("Sorry, invalid model");

   }

}



Hometask01

Adding samsung and oneplus in task 1

package hometask;

import java.util.Objects;

public class Apple {

   private Apple() {

       // Prevent instantiation

   }

   public static Mobile getMobile(String model) {

       if (Objects.*equals*(model, "iphone16")) {

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

       } else if (Objects.*equals*(model, "iphone16MaxPro")) {

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

       }

       return new NoMobile();

   }

}

package hometask;

public class Mobile {

   String desc;

   public Mobile(String model) {

       this.desc = model;

   }

   public void getDesc() {

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

   }

}

package hometask;

import java.util.Objects;

public class MobileStore {

   private MobileStore() {

       System.*out*.println("Hello, welcome to the world of Mobile");

   }

   public static Mobile getMobile(String brand, String model) {

       if (Objects.*equals*(brand, "Apple")) {

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

           return Apple.*getMobile*(model);

       } else if (Objects.*equals*(brand, "Samsung")) {

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

           return Samsung.*getMobile*(model);

       } else if (Objects.*equals*(brand, "OnePlus")) {

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

           return OnePlus.*getMobile*(model);

       }

       return new NoMobile();

   }

}

package hometask;

public class NoMobile extends Mobile {

   public NoMobile() {

       super("Sorry, invalid model");

   }

}

package hometask;

import java.util.Objects; // ✅ add this

public class OnePlus {

   private OnePlus() {

       // Prevent instantiation

   }

   public static Mobile getMobile(String model) {

       if (Objects.*equals*(model, "OnePlus12")) {

           return new Mobile("Here is your OnePlus 12");

       } else if (Objects.*equals*(model, "OnePlus12R")) {

           return new Mobile("Here is your OnePlus 12R");

       }

       return new NoMobile();

   }

}

package hometask;

import java.util.Objects;

public class Samsung {

   private Samsung() {

       // Prevent instantiation

   }

   public static Mobile getMobile(String model) {

       if (Objects.*equals*(model, "S24")) {

           return new Mobile("Here is your Samsung S24");

       } else if (Objects.*equals*(model, "S24Ultra")) {

           return new Mobile("Here is your Samsung S24 Ultra");

       }

       return new NoMobile();

   }

}

package hometask;

public class ClientAbstractFactory {

   public static void main(String[] args) {

       Mobile m1 = MobileStore.*getMobile*("Apple", "iphone16");

       m1.getDesc();

       Mobile m2 = MobileStore.*getMobile*("Samsung", "S24");

       m2.getDesc();

       Mobile m3 = MobileStore.*getMobile*("OnePlus", "OnePlus12");

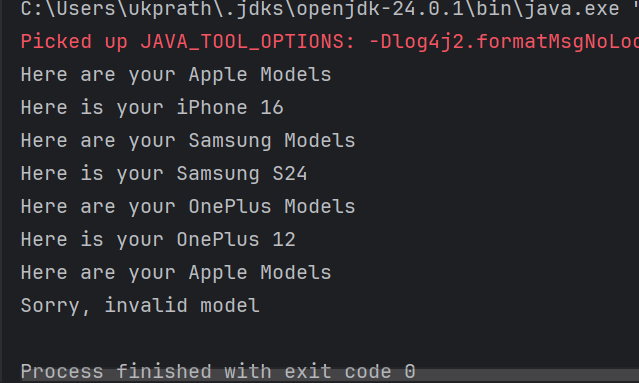
       m3.getDesc();

       Mobile m4 = MobileStore.*getMobile*("Apple", "iphone20"); // invalid

       m4.getDesc();

   }

}



**Builder method Design pattern**

**Task 02:**

package Task02.BuilderDesignPattern;

class Laptop {

   private int memory;

   private int storage;

   // Could add: graphicCard, processor, etc.

   public Laptop() {

       // Default constructor

   }

   // Getters and Setters

   public int getMemory() {

       return memory;

   }

   public void setMemory(int memory) {

       this.memory = memory;

   }

   public int getStorage() {

       return storage;

   }

   public void setStorage(int storage) {

       this.storage = storage;

   }

   @Override

   public String toString() {

       return "Laptop{" +

               "memory=" + memory +

               "GB, storage=" + storage +

               "GB}";

   }

}

package Task02.BuilderDesignPattern;

interface LaptopBuilder {

   LaptopBuilder buildMemory(int memory);

   LaptopBuilder buildStorage(int storage);

   Laptop build();

}

package Task02.BuilderDesignPattern;

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 Laptop build() {

       return laptop;

   }

}

package Task02.BuilderDesignPattern;

class LaptopDirector {

   private LaptopBuilder laptopBuilder;

   public LaptopDirector(LaptopBuilder laptopBuilder) {

       this.laptopBuilder = laptopBuilder;

   }

   public Laptop constructLaptop() {

       return laptopBuilder

               .buildMemory(16)   // Example: 16GB RAM

               .buildStorage(512) // Example: 512GB SSD

               .build();

   }

}

package Task02.BuilderDesignPattern;

public class ClientBuildMethodDP {

   public static void main(String[] args) {

       LaptopBuilder lbobj = new LaptopConcreteBuilder();

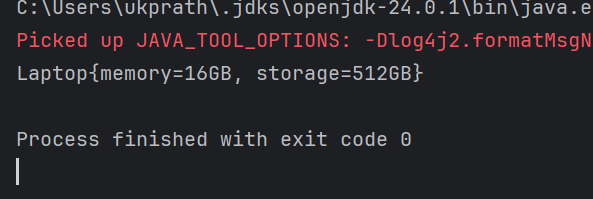
       LaptopDirector dir = new LaptopDirector(lbobj);

       Laptop lobj = dir.constructLaptop();

       System.*out*.println(lobj);

   }

}



Task 03:

package Task03;

public interface Colors {

   Colors clone();

   String getName();

   void setName(String name);

}

package Task03;

public class BlackConcretePrototype implements Colors {

   private String name;

   public BlackConcretePrototype() {

       System.*out*.println("BlackConcretePrototype constructor is called");

   }

   public BlackConcretePrototype(String name) {

       this.name = name;

   }

   @Override

   public Colors clone() {

       return new BlackConcretePrototype(this.name);

   }

   @Override

   public String getName() {

       return name;

   }

   @Override

   public void setName(String name) {

       this.name = name;

   }

}

package Task03;

public class WhiteConcretePrototype implements Colors {

   private String name;

   public WhiteConcretePrototype() {

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

   }

   public WhiteConcretePrototype(String name) {

       this.name = name;

   }

   @Override

   public Colors clone() {

       return new WhiteConcretePrototype(this.name);

   }

   @Override

   public String getName() {

       return name;

   }

   @Override

   public void setName(String name) {

       this.name = name;

   }

}

package Task03;

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");

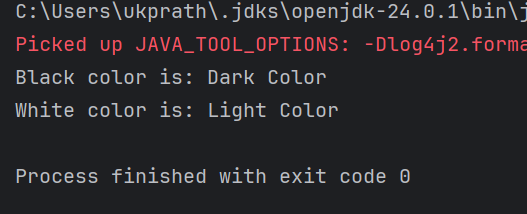
       clonedWhiteObj.setName("Light Color");

       System.*out*.println("Black color is: " + clonedBlackObj.getName());

       System.*out*.println("White color is: " + clonedWhiteObj.getName());

   }

}



Prototype Method Design pattern

    prototype interface -- Colors

    Concrete prototype --- class

        WhiteConcretePrototype

            constructor()

            constructor(String...)

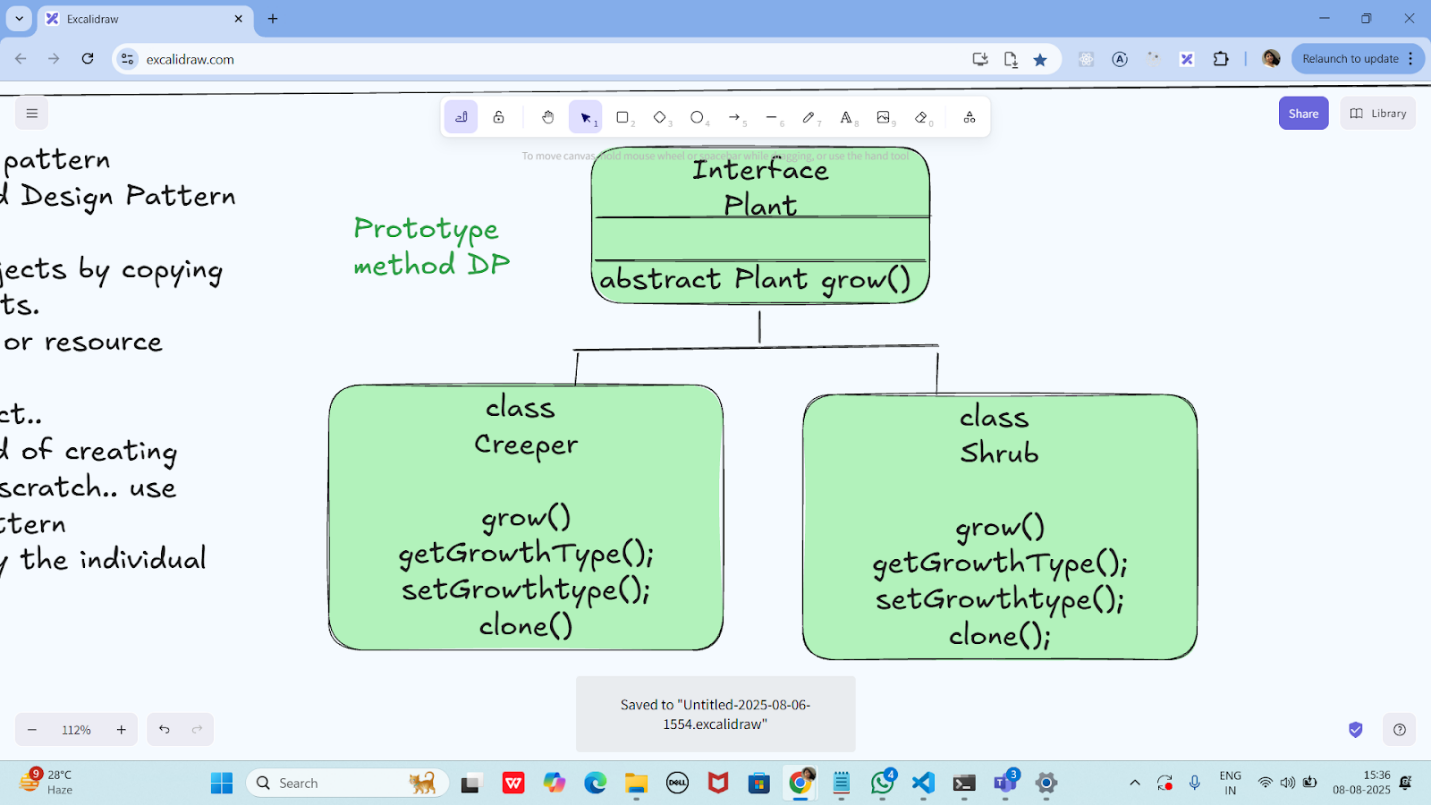
            clone()

        BlackConcreteprototype

    client code

Home tasks 01

Develop the below prototype method DP..



package prototype\_Home\_task;

public interface Plant extends Cloneable {

   Plant grow();

   String getGrowthType();

   void setGrowthType(String growthType);

   Plant clone();

}

package prototype\_Home\_task;

public class Creeper implements Plant {

   private String growthType;

   public Creeper() {

       System.*out*.println("Creeper constructor is called");

   }

   public Creeper(String growthType) {

       this.growthType = growthType;

   }

   @Override

   public Plant grow() {

       System.*out*.println("Creeper grows by spreading on the ground or climbing.");

       return this;

   }

   @Override

   public String getGrowthType() {

       return growthType;

   }

   @Override

   public void setGrowthType(String growthType) {

       this.growthType = growthType;

   }

   @Override

   public Plant clone() {

       return new Creeper(this.growthType);

   }

}

package prototype\_Home\_task;

public class Shrub implements Plant {

   private String growthType;

   public Shrub() {

       System.*out*.println("Shrub constructor is called");

   }

   public Shrub(String growthType) {

       this.growthType = growthType;

   }

   @Override

   public Plant grow() {

       System.*out*.println("Shrub grows as a small to medium-sized woody plant.");

       return this;

   }

   @Override

   public String getGrowthType() {

       return growthType;

   }

   @Override

   public void setGrowthType(String growthType) {

       this.growthType = growthType;

   }

   @Override

   public Plant clone() {

       return new Shrub(this.growthType);

   }

}

package prototype\_Home\_task;

public class Main {

   public static void main(String[] args) {

       // Original prototypes

       Plant creeperPrototype = new Creeper("Climbing Plant");

       Plant shrubPrototype = new Shrub("Bushy Plant");

       // Cloning from prototypes

       Plant clonedCreeper = creeperPrototype.clone();

       Plant clonedShrub = shrubPrototype.clone();

       // Changing clone names to prove independence

       clonedCreeper.setGrowthType("Ground Spreader");

       clonedShrub.setGrowthType("Flowering Shrub");

       // Showing original and clone info

       System.*out*.println("\nOriginal Creeper: " + creeperPrototype.getGrowthType());

       System.*out*.println("Cloned Creeper: " + clonedCreeper.getGrowthType());

       System.*out*.println("\nOriginal Shrub: " + shrubPrototype.getGrowthType());

       System.*out*.println("Cloned Shrub: " + clonedShrub.getGrowthType());

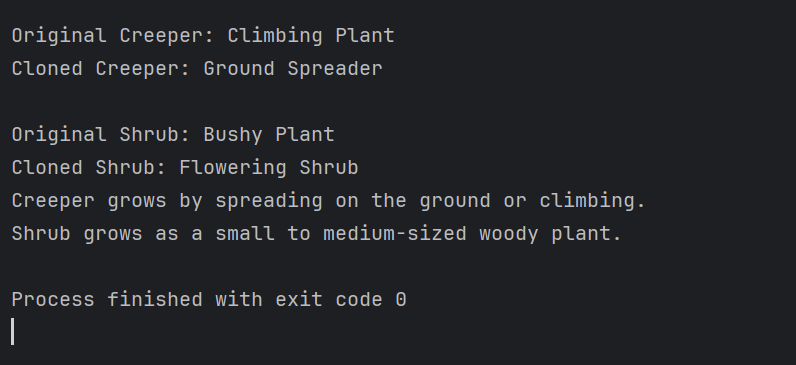
       // Growth actions

       clonedCreeper.grow();

       clonedShrub.grow();

   }

}



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>