Day 24

Employee ID: 201933938

Login ID: iamasif

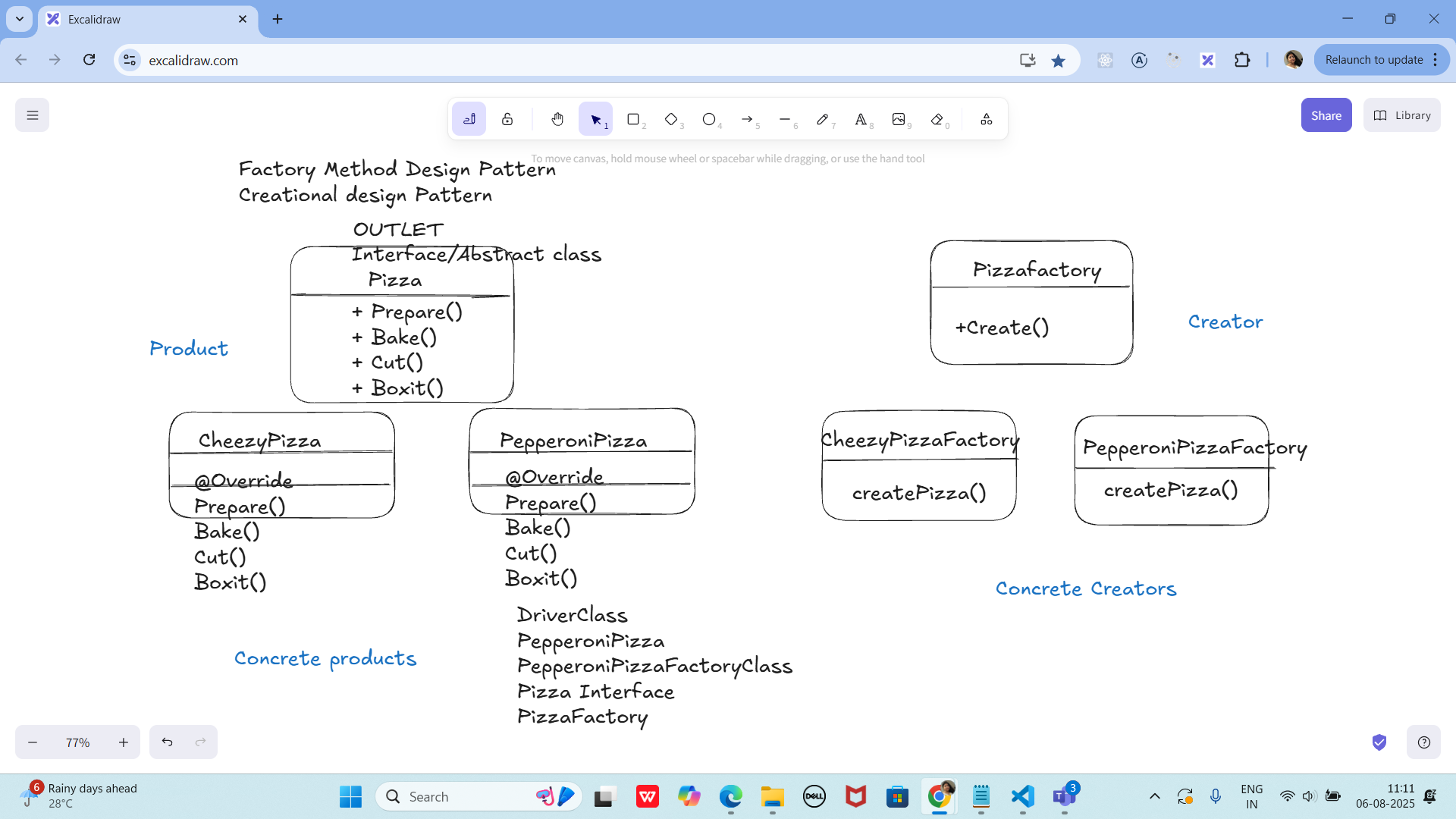
Name: Shaik Asif

**Design patters:**

1. Creational Design Patterns:

types of Creational design Patterns:

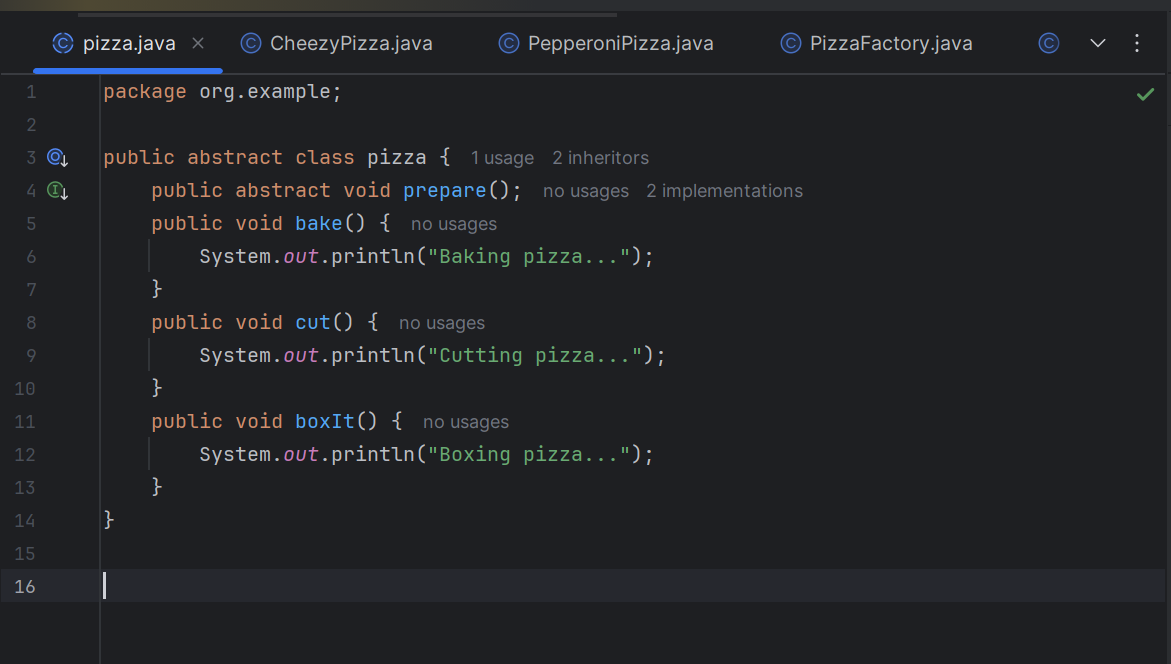
1. Factory Method
2. Singleton Method
3. Abstract Factory Method
4. Builder Method
5. Prototype Method



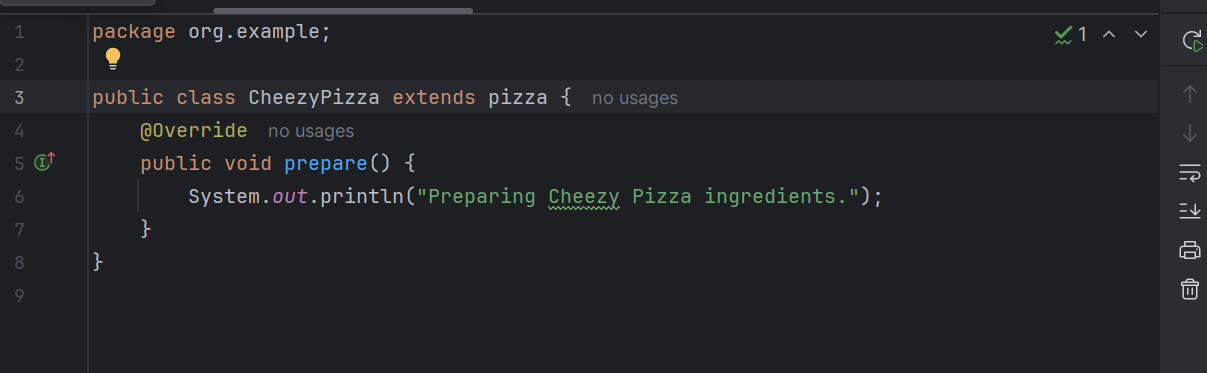
Task 01:

On the basis of above SS try to create a Fcational Method Design Pattern program

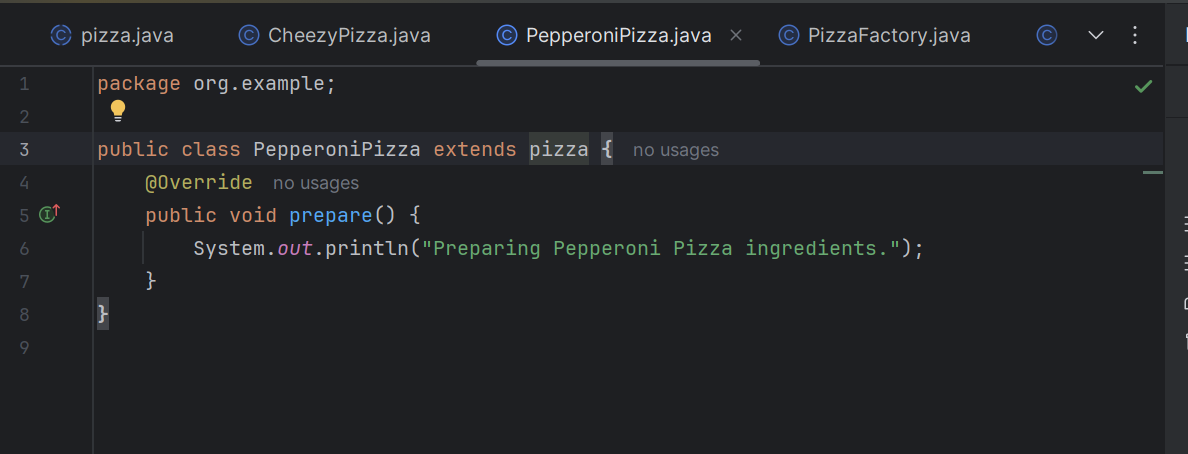
**Pizza cls**



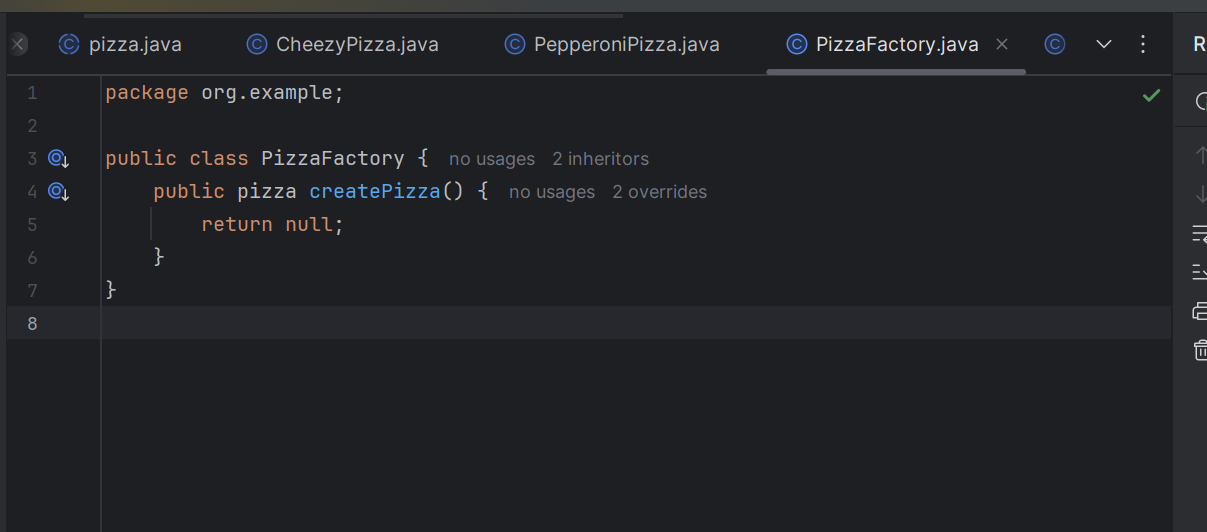
**CheezyPizza**

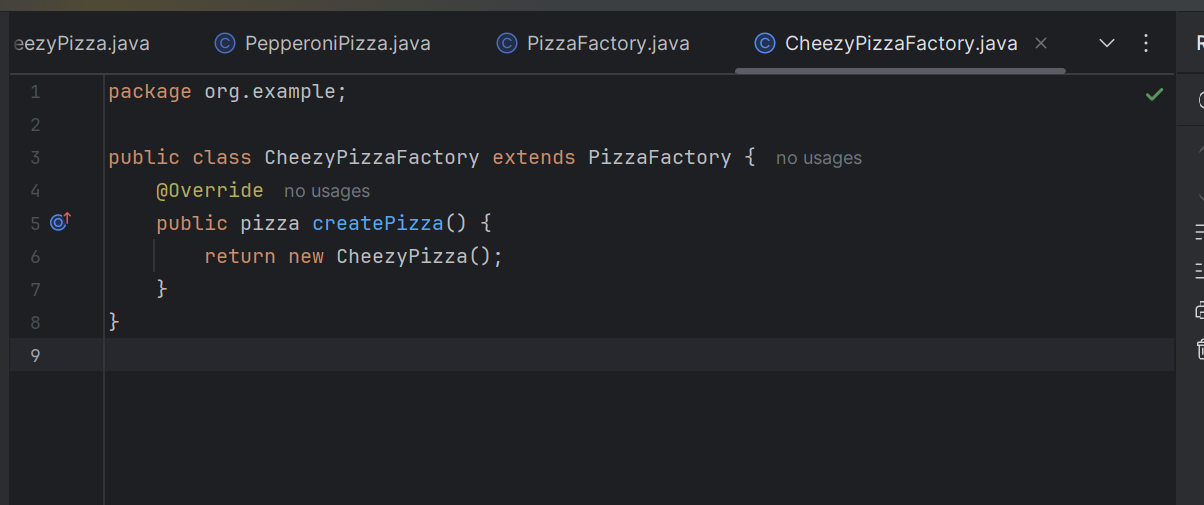


**PepperoniPizza**



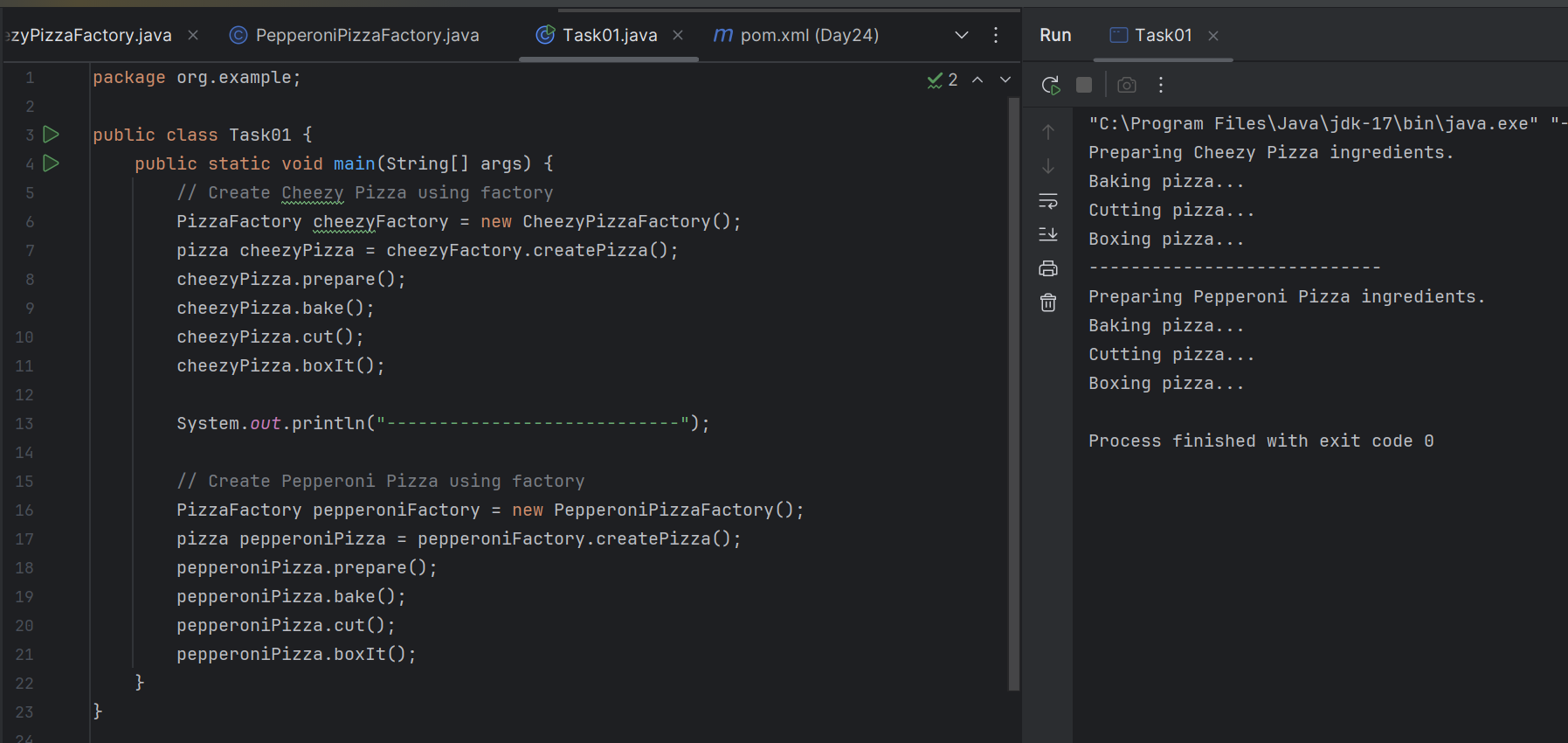
**PizzaFactory**







**Main**



Excalidraw doc - updated as of 06th Aug 2025 at 12.55.

<https://excalidraw.com/#json=lNVBffaZ5rqFz0EguLS2O,Q045iriTq36O-xpkRuMALw>

Batch 1 search for ..

Task 02:

package Demo\_Codes.Module;

import java.io.IOException;

import java.io.InputStream;

import java.util.Properties;

public class SingletonDPConfigManager {

private static SingletonDPConfigManager ConfigManagerInstance;

private Properties properties;

private SingletonDPConfigManager() {

properties = new Properties();

try (InputStream input = getClass().getClassLoader().getResourceAsStream("config.prop")) {

if (input == null) {

System.out.println("we cant find config properties"); // error

return;

}

properties.load(input);

}catch (IOException ioexp) {

ioexp.printStackTrace();

}

}

public String getProperty(String key) {

return properties.getProperty(key);

}

public static synchronized SingletonDPConfigManager getConfigManagerInstance() {

if (ConfigManagerInstance == null) {

ConfigManagerInstance = new SingletonDPConfigManager();

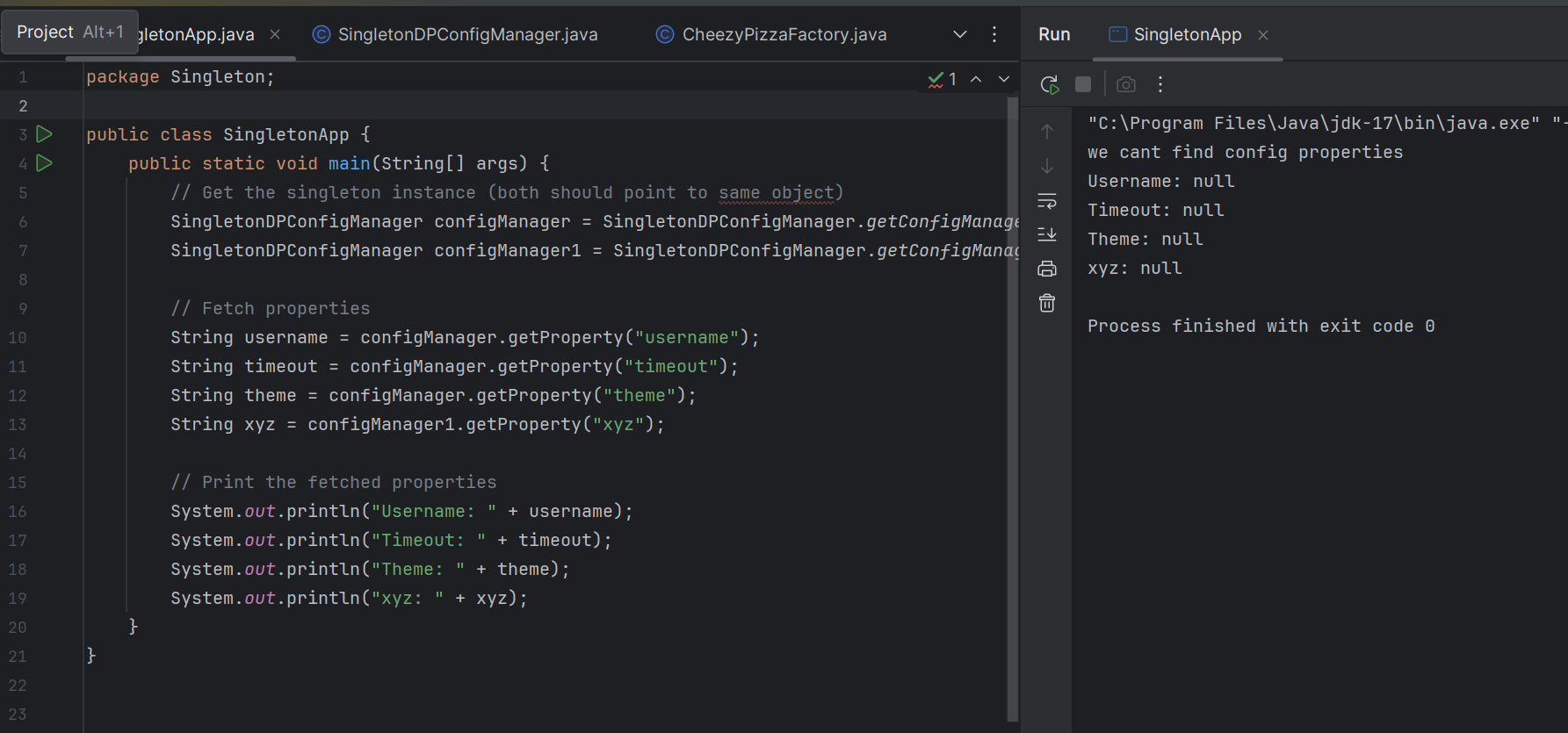
}

return ConfigManagerInstance;

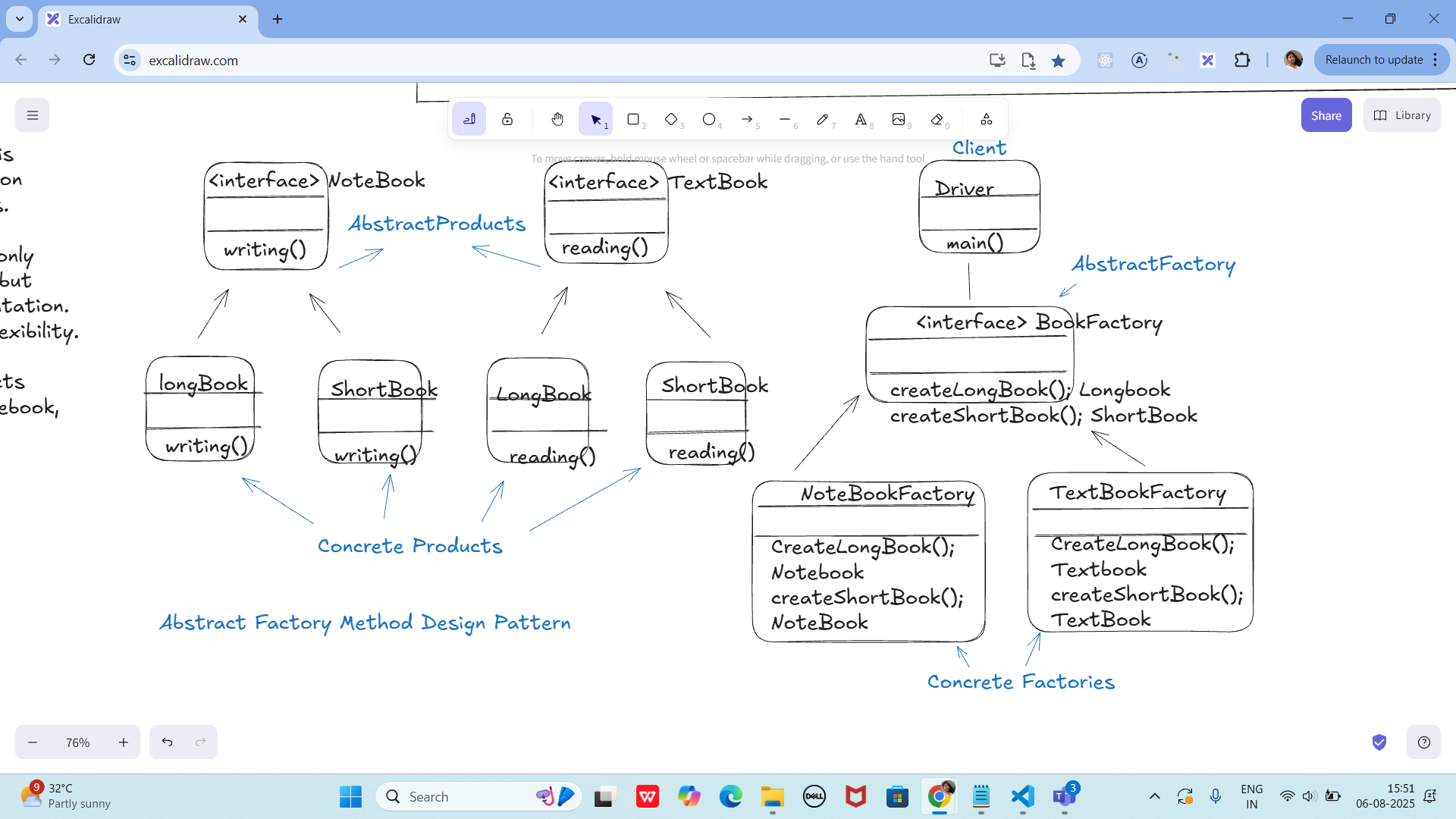
}

}

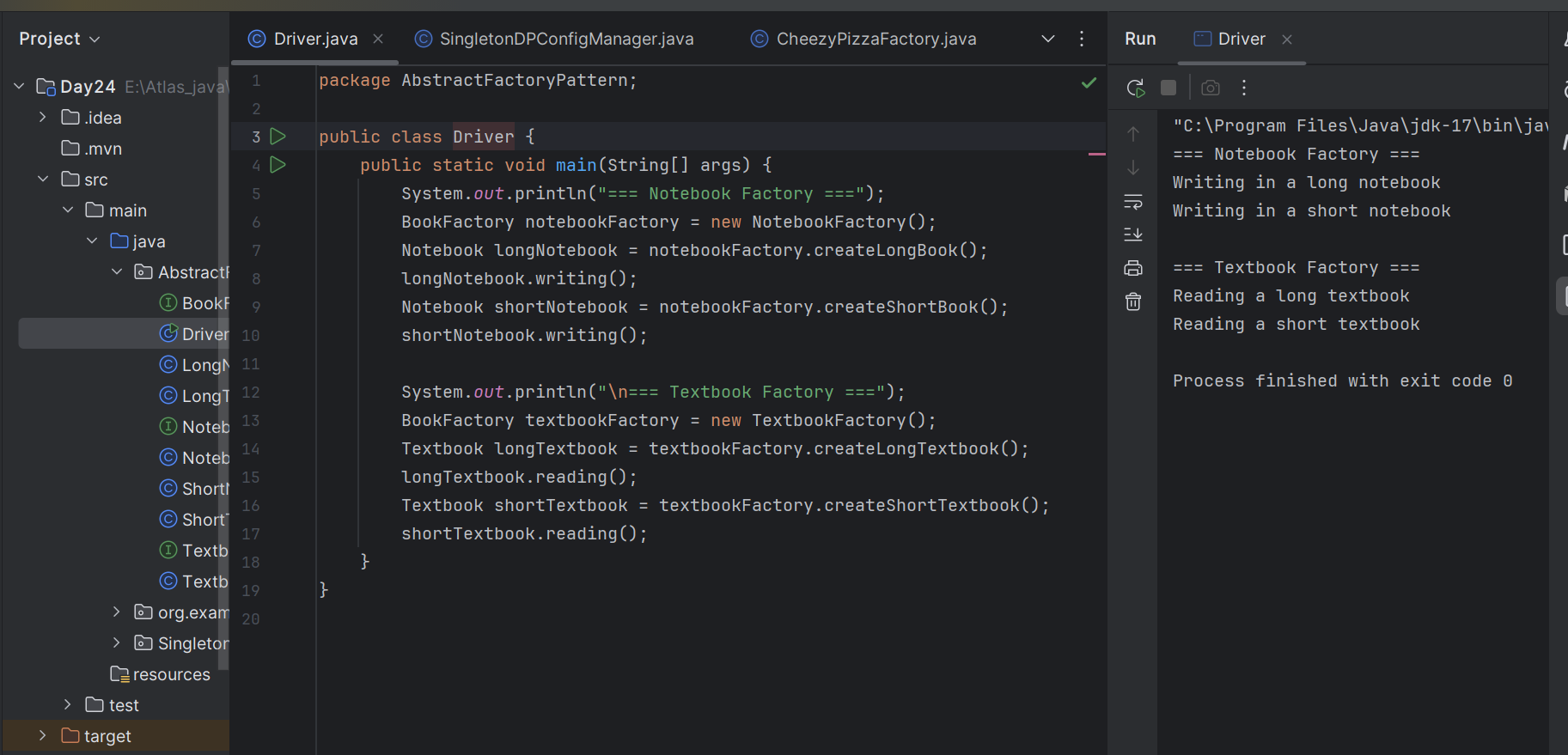
**OutPut**



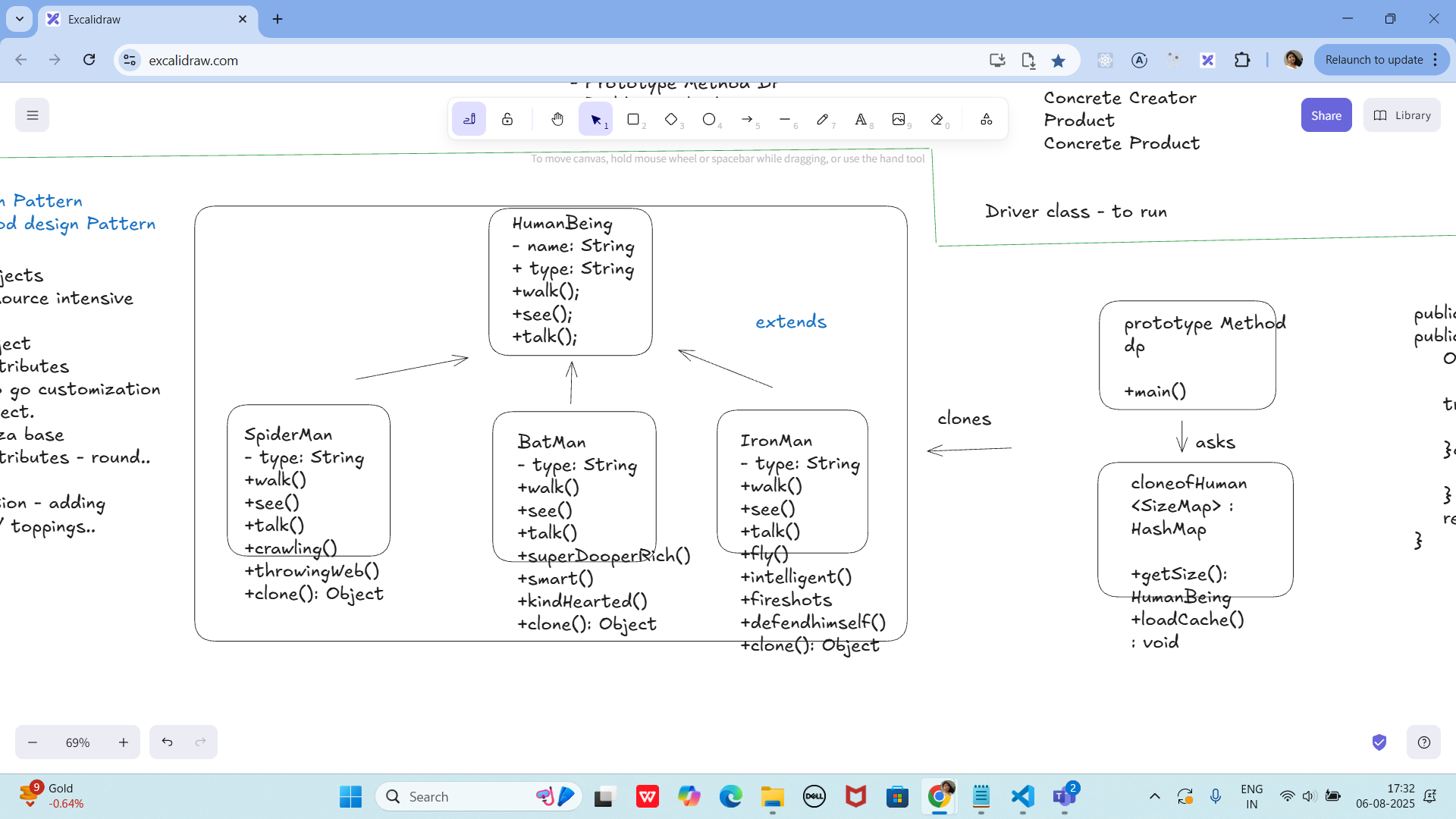
**Task 03:**



**OUTPUT:**

****

**Task4:**

****

**HumanBeing.java:**

**package PrototypeDesignPattern;**

**public abstract class HumanBeing implements Cloneable {**

**private String name;**

**protected String type;**

**abstract void walk();**

**abstract void see();**

**abstract void talk();**

**public String getType() {**

**return type;**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**@Override**

**public Object clone() {**

**Object clone = null;**

**try {**

**clone = super.clone();**

**} catch (CloneNotSupportedException e) {**

**e.printStackTrace();**

**}**

**return clone;**

**}**

**}**

**SpiderMan.java:**

**package PrototypeDesignPattern;**

**public class SpiderMan extends HumanBeing {**

**public SpiderMan() {**

**this.type = "SpiderMan";**

**}**

**@Override**

**void walk() {**

**System.*out*.println("SpiderMan walks quickly.");**

**}**

**@Override**

**void see() {**

**System.*out*.println("SpiderMan sees everything with spider sense.");**

**}**

**@Override**

**void talk() {**

**System.*out*.println("SpiderMan talks witty.");**

**}**

**void crawling() {**

**System.*out*.println("SpiderMan crawls on walls.");**

**}**

**void throwingWeb() {**

**System.*out*.println("SpiderMan throws web.");**

**}**

**}**

**BatMan.java:**

**package PrototypeDesignPattern;**

**public class BatMan extends HumanBeing {**

**public BatMan() {**

**this.type = "BatMan";**

**}**

**@Override**

**void walk() {**

**System.*out*.println("BatMan walks stealthily.");**

**}**

**@Override**

**void see() {**

**System.*out*.println("BatMan sees with detective skills.");**

**}**

**@Override**

**void talk() {**

**System.*out*.println("BatMan talks in a deep voice.");**

**}**

**void superDooperRich() {**

**System.*out*.println("BatMan is super rich!");**

**}**

**void smart() {**

**System.*out*.println("BatMan is smart.");**

**}**

**void kindHearted() {**

**System.*out*.println("BatMan is kind-hearted.");**

**}**

**}**

**IronMan.java:**

**package PrototypeDesignPattern;**

**public class IronMan extends HumanBeing {**

**public IronMan() {**

**this.type = "IronMan";**

**}**

**@Override**

**void walk() {**

**System.*out*.println("IronMan walks with power.");**

**}**

**@Override**

**void see() {**

**System.*out*.println("IronMan sees with advanced HUD.");**

**}**

**@Override**

**void talk() {**

**System.*out*.println("IronMan talks sarcastically.");**

**}**

**void fly() {**

**System.*out*.println("IronMan flies with jet boots.");**

**}**

**void intelligent() {**

**System.*out*.println("IronMan is highly intelligent.");**

**}**

**void fireShots() {**

**System.*out*.println("IronMan fires shots from suit.");**

**}**

**void defendHimself() {**

**System.*out*.println("IronMan defends himself with armor.");**

**}**

**}**

**cloneofHuman.java:**

**package PrototypeDesignPattern;**

**import java.util.Hashtable;**

**public class cloneofHuman {**

**private static Hashtable<String, HumanBeing> *sizeMap* = new Hashtable<>();**

**public static HumanBeing getSize(String humanId) {**

**HumanBeing cachedHuman = *sizeMap*.get(humanId);**

**return (HumanBeing) cachedHuman.clone();**

**}**

**public static void loadCache() {**

**SpiderMan spiderMan = new SpiderMan();**

**spiderMan.setName("Peter Parker");**

***sizeMap*.put("1", spiderMan);**

**BatMan batMan = new BatMan();**

**batMan.setName("Bruce Wayne");**

***sizeMap*.put("2", batMan);**

**IronMan ironMan = new IronMan();**

**ironMan.setName("Tony Stark");**

***sizeMap*.put("3", ironMan);**

**}**

**}**

**prototypeMethod.java (Driver class with main):**

**package PrototypeDesignPattern;**

**public class prototypeMethod {**

**public static void main(String[] args) {**

**cloneofHuman.*loadCache*();**

**HumanBeing clonedSpiderMan = cloneofHuman.*getSize*("1");**

**System.*out*.println("Human : " + clonedSpiderMan.getType() + ", Name : " + clonedSpiderMan.getName());**

**clonedSpiderMan.walk();**

**clonedSpiderMan.see();**

**clonedSpiderMan.talk();**

**if (clonedSpiderMan instanceof SpiderMan) {**

**((SpiderMan) clonedSpiderMan).crawling();**

**((SpiderMan) clonedSpiderMan).throwingWeb();**

**}**

**HumanBeing clonedBatMan = cloneofHuman.*getSize*("2");**

**System.*out*.println("\nHuman : " + clonedBatMan.getType() + ", Name : " + clonedBatMan.getName());**

**clonedBatMan.walk();**

**clonedBatMan.see();**

**clonedBatMan.talk();**

**if (clonedBatMan instanceof BatMan) {**

**((BatMan) clonedBatMan).superDooperRich();**

**((BatMan) clonedBatMan).smart();**

**((BatMan) clonedBatMan).kindHearted();**

**}**

**HumanBeing clonedIronMan = cloneofHuman.*getSize*("3");**

**System.*out*.println("\nHuman : " + clonedIronMan.getType() + ", Name : " + clonedIronMan.getName());**

**clonedIronMan.walk();**

**clonedIronMan.see();**

**clonedIronMan.talk();**

**if (clonedIronMan instanceof IronMan) {**

**((IronMan) clonedIronMan).fly();**

**((IronMan) clonedIronMan).intelligent();**

**((IronMan) clonedIronMan).fireShots();**

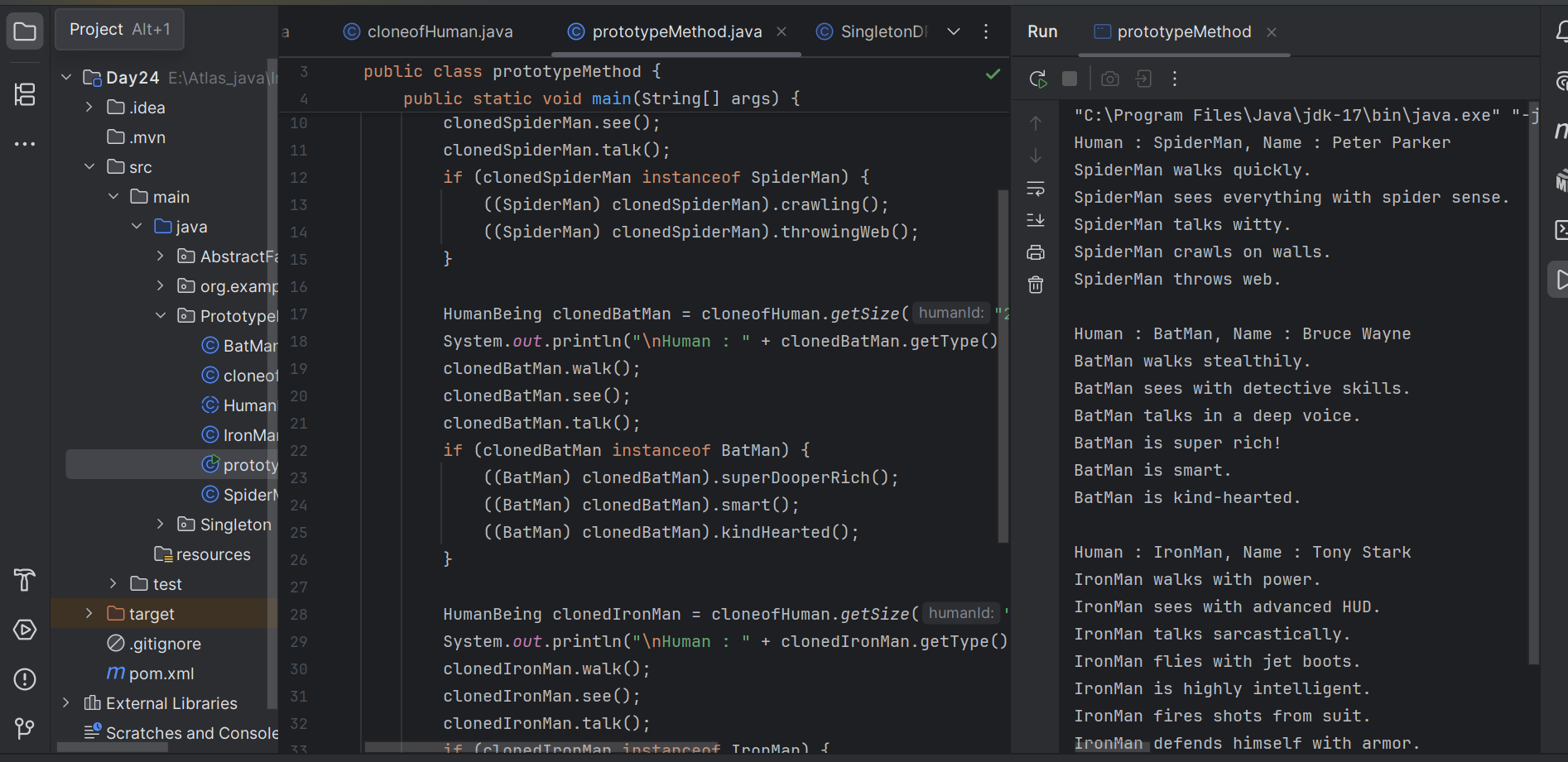
**((IronMan) clonedIronMan).defendHimself();**

**}**

**}**

**}**

**OUTPUT:**

****

================================================================================================================================================

**Home task 01:**

class SingletonDemo {

private static SingletonDemo instance; // lodinids.. //pass/ pin no

private SingletonDemo() {

System.out.println("initiating the singleton");

}

public static SingletonDemo getInstance() {

if (instance == null) {

instance = new SingletonDemo();

System.out.println("in get instance");

}

return instance;

}

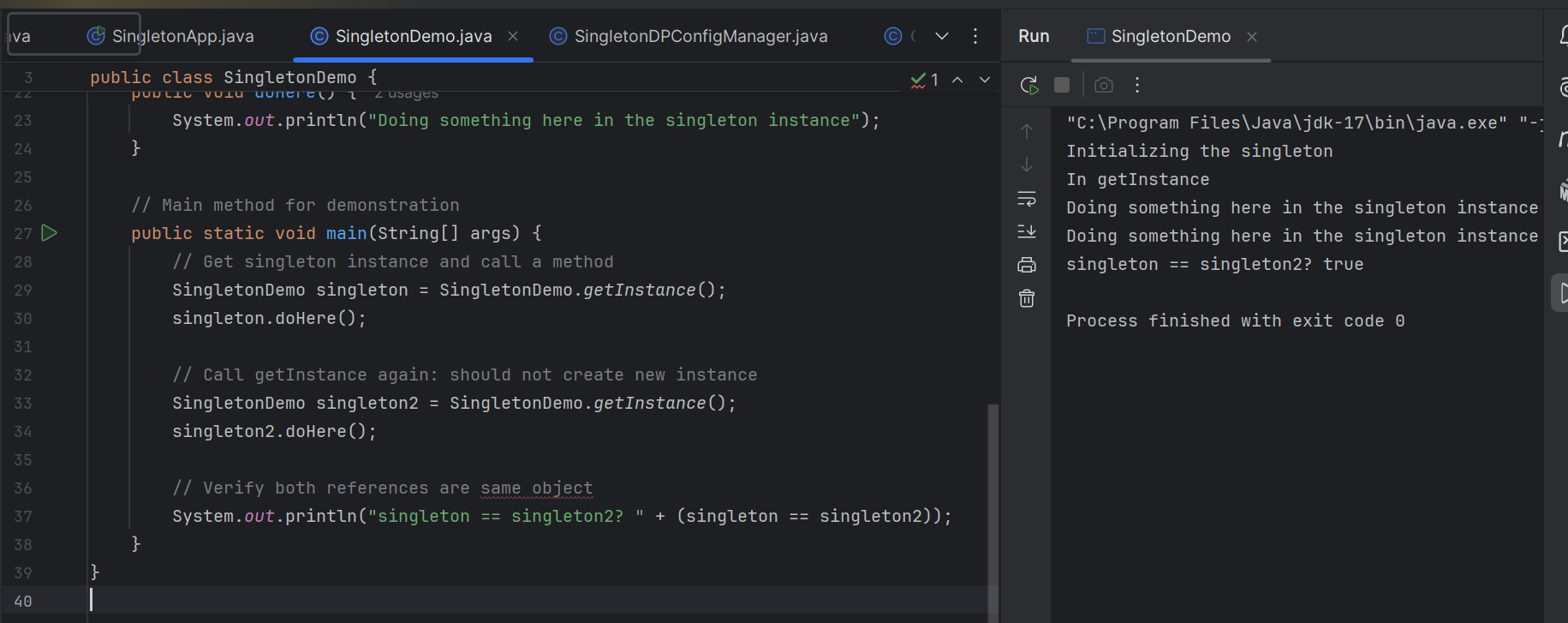
public static void doHere() {

System.out.println("doing here some thing");

}

}

**OutPut**



**Home Task 2:**

public class SingletonDP {

public static void main(String[] args) {

SingletonDemo.getInstance().doHere();

// SingletonDemo obj = new SingletonDemo();

// obj.doHere();

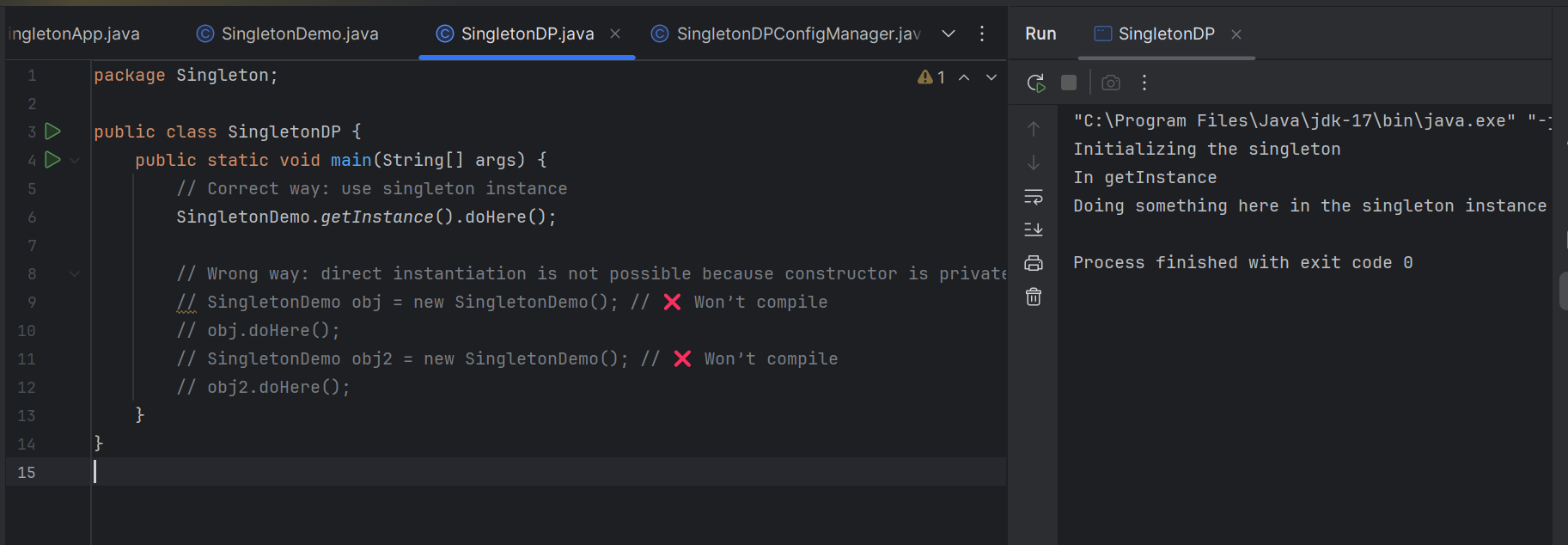
// SingletonDemo obj2 = new SingletonDemo();

// obj2.doHere();

}

}

**OutPut**



================================================================================================================================================