Day 23 - 02nd Aug 2025

JUNIT testcases - atleast one or two if working pla raise your hands.

TestCase01 file

.

………………………………………………………………………………………..

TASK 01

Tags Annotation:

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.Tags;

import org.junit.jupiter.api.Assertions.assertEquals;

class TestCase02 {

@Test

@Tags("firstPriority")

void testMethod01() {

}

@Test

@Tag("firstPriority")

void runTestcase02() {

}

@Tags("fastTag")

void testMethod03() {

}

@Test

@Tag("slowTag")

void runTestcase04() {

}

}

Pom.xml:

<project xmlns="[http://maven.apache.org/POM/4.0.0"](http://maven.apache.org/POM/4.0.0%22)

xmlns:xsi="[http://www.w3.org/2001/XMLSchema-instance"](http://www.w3.org/2001/XMLSchema-instance%22)

xsi:schemaLocation="<http://maven.apache.org/POM/4.0.0> [https://maven.apache.org/xsd/maven-4.0.0.xsd">](https://maven.apache.org/xsd/maven-4.0.0.xsd%22%3E)

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>  
<artifactId>junit5demo</artifactId>  
<version>0.0.1-SNAPSHOT</version>

<dependencies>  
<dependency>  
<groupId>org.junit.jupiter</groupId>  
<artifactId>junit-jupiter</artifactId>  
<version>5.10.0</version>  
<scope>test</scope>  
</dependency>  
</dependencies>  
<build>  
<plugins>  
<plugin>  
<groupId>org.apache.maven.plugins</groupId>  
<artifactId>maven-surefire-plugin</artifactId>  
<version>3.1.2</version>  
<configuration>  
<includeTags>firstPriority,fastTag</includeTags>  
<excludeTags>slowTag</excludeTags>  
</configuration>  
</plugin>  
</plugins>  
</build>

</project>

—----------------------------------------------------------------------------------------------------------------------

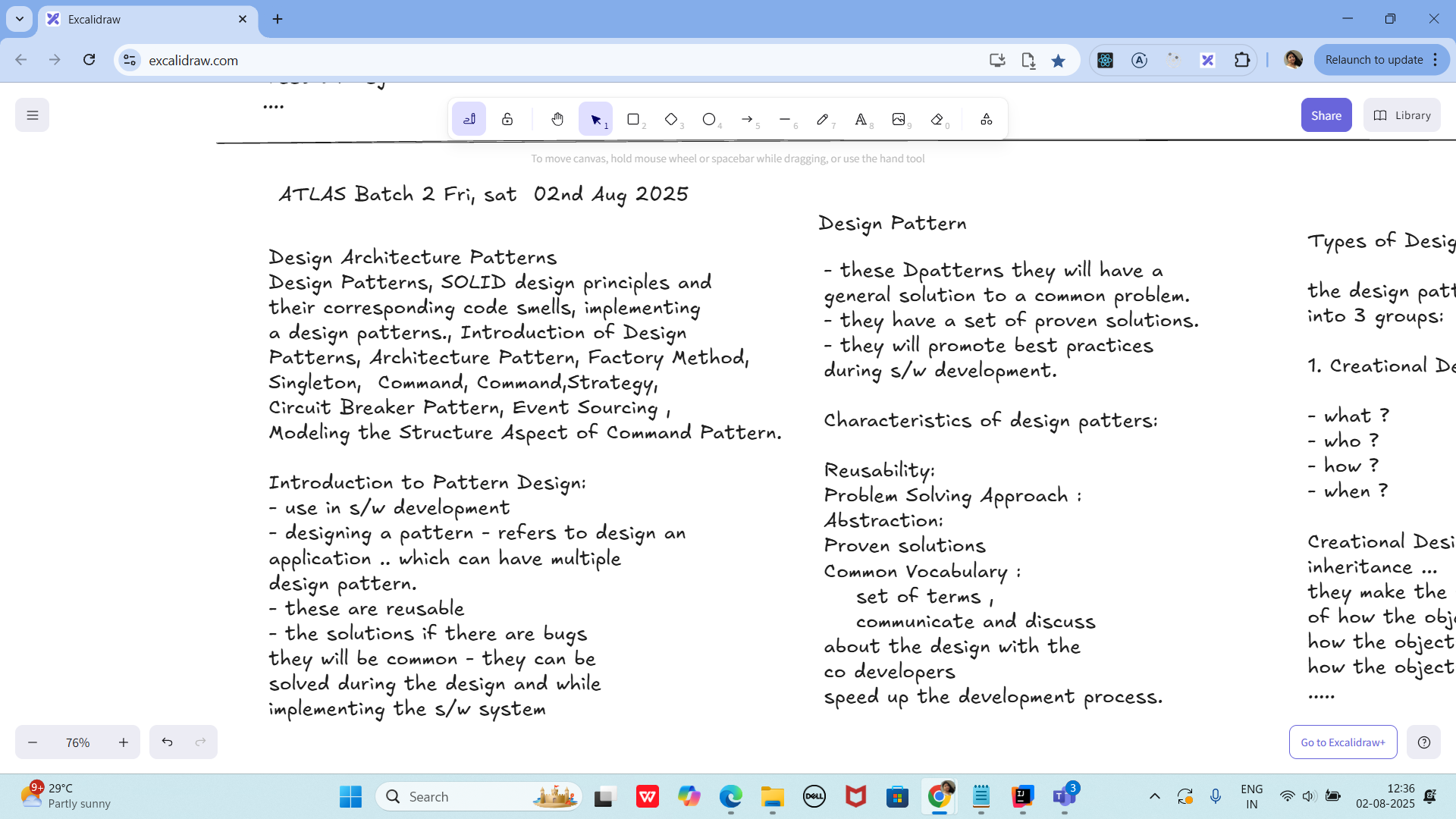
**Design patterns:**

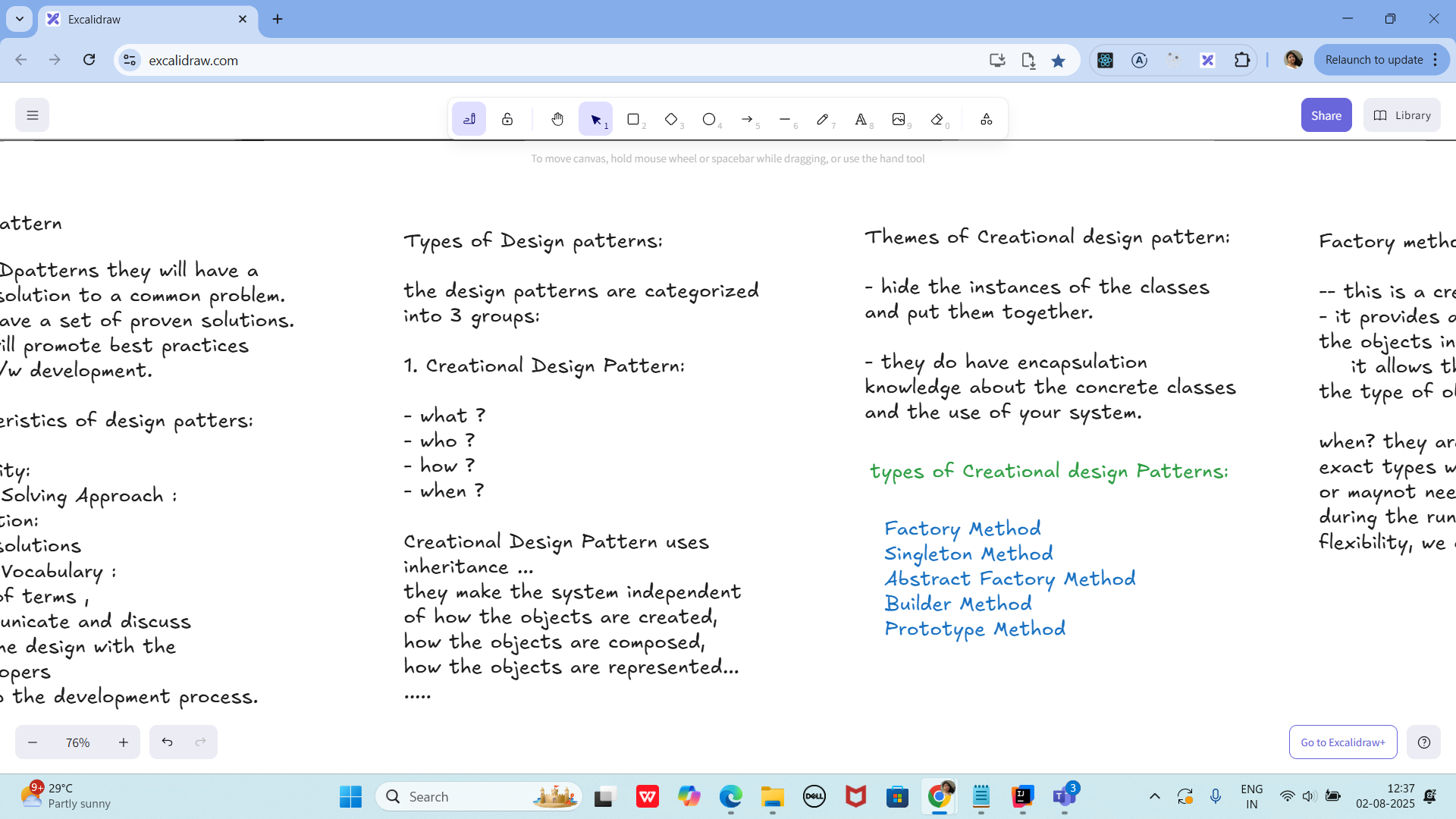
1. Creational Design Patterns:

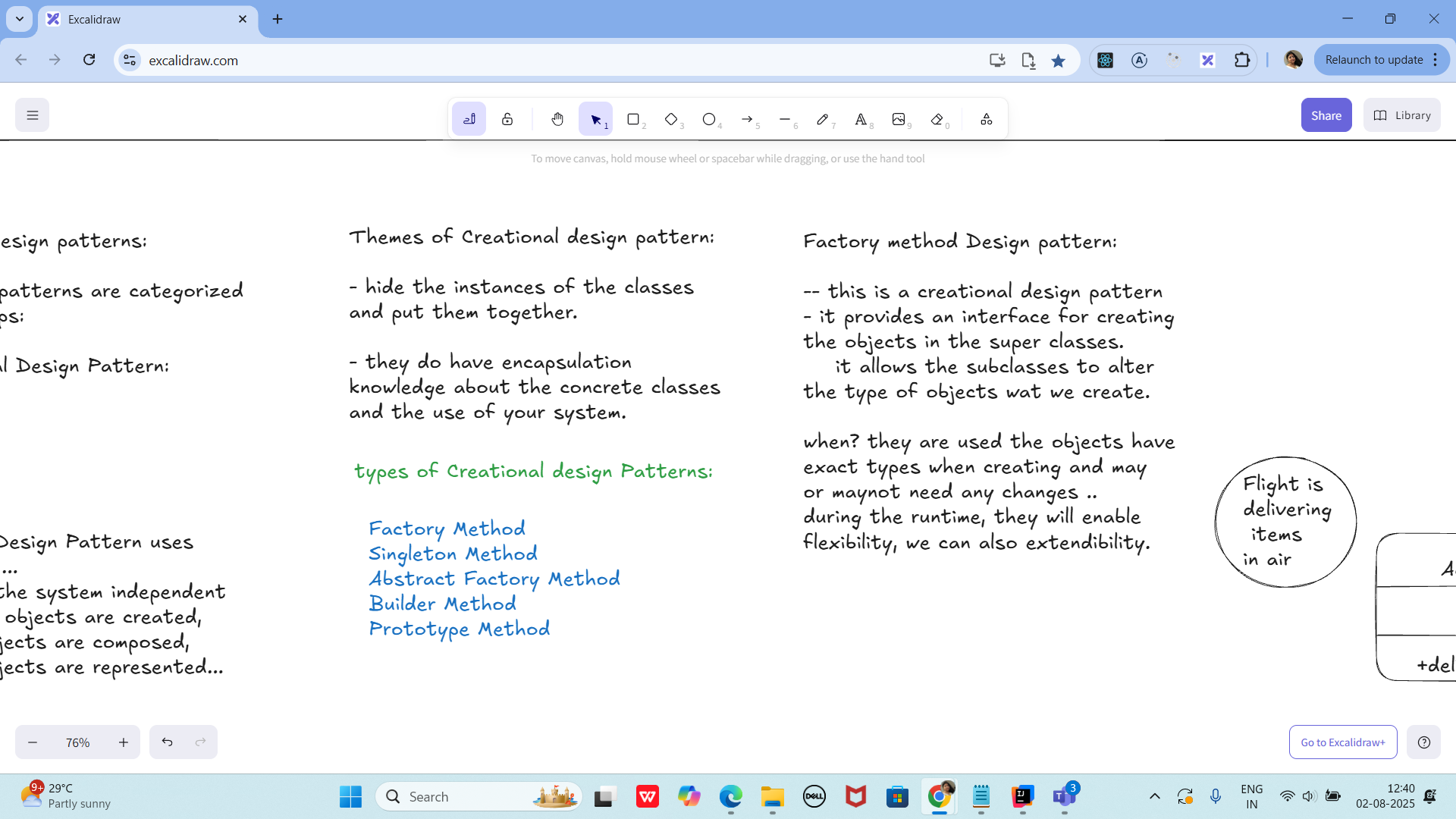
types of Creational design Patterns:

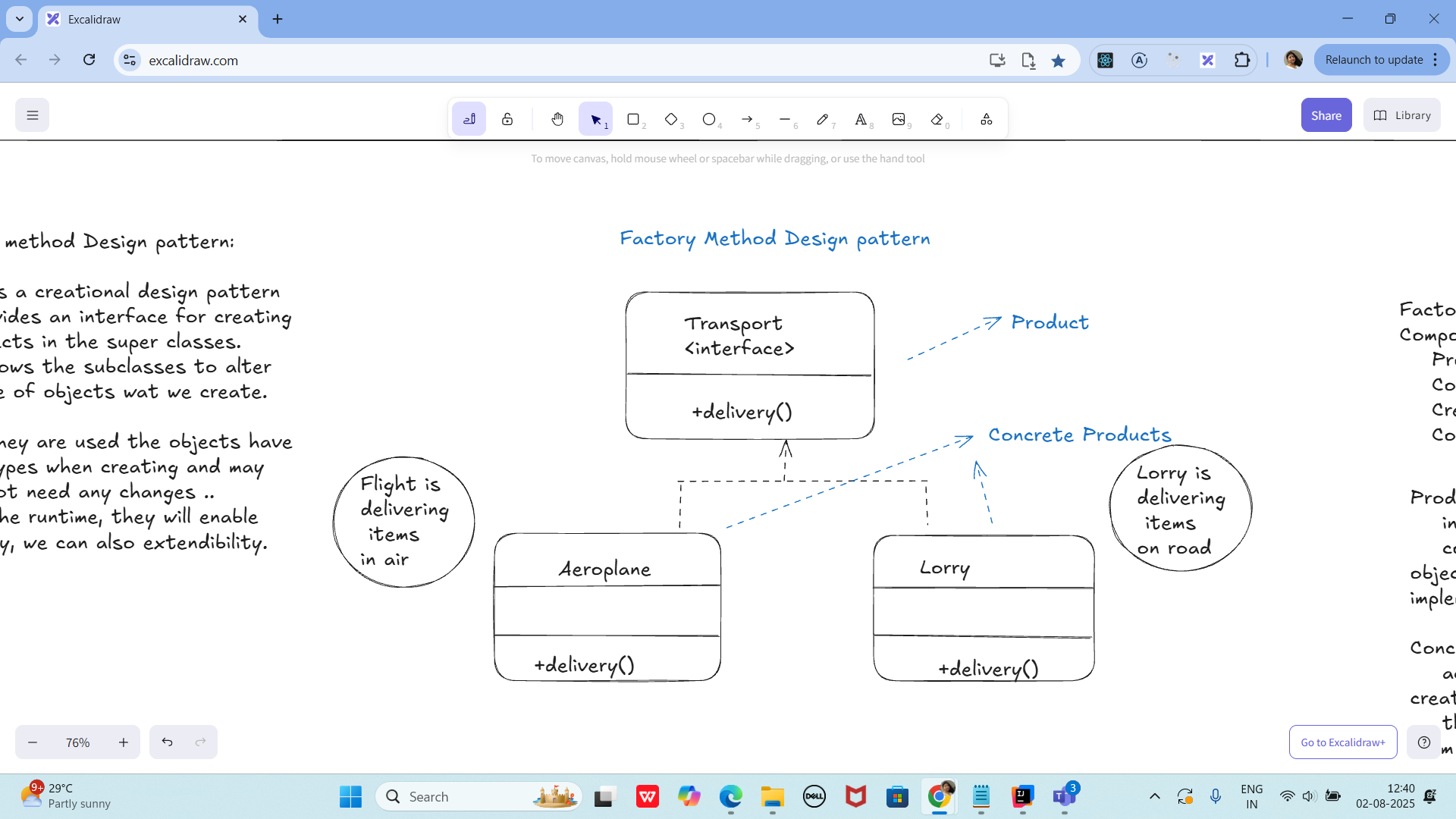
1. Factory Method
2. Singleton Method
3. Abstract Factory Method
4. Builder Method
5. Prototype Method

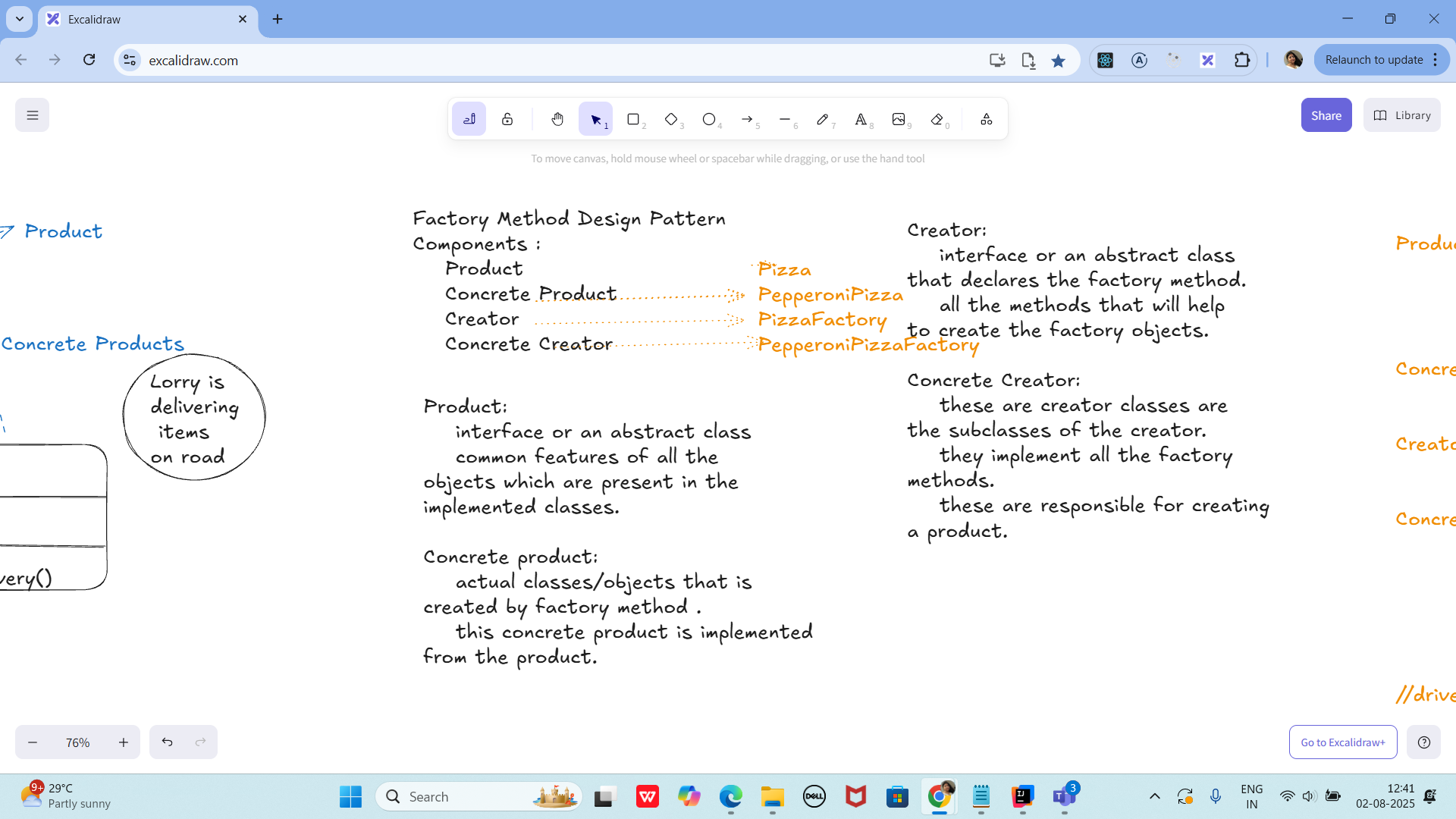
**Factory Method Design Pattern**

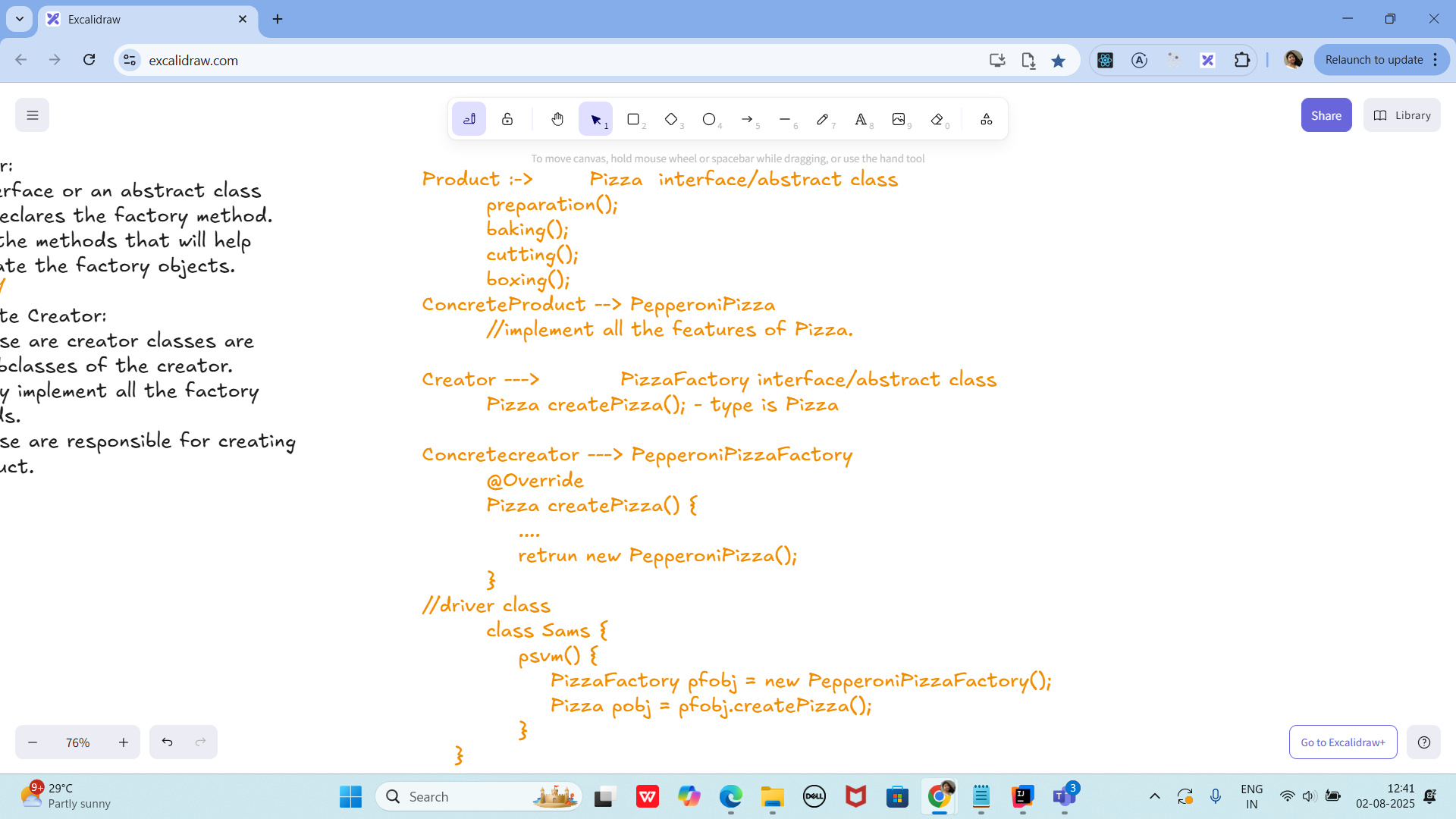












Task 02:

Product :-> Pizza interface/abstract class

preparation();

baking();

cutting();

boxing();

ConcreteProduct --> PepperoniPizza

//implement all the features of Pizza.

Creator ---> PizzaFactory interface/abstract class

Pizza createPizza(); - type is Pizza

Concretecreator ---> PepperoniPizzaFactory

@Override

Pizza createPizza() {

....

retrun new PepperoniPizza();

}

//driver class

class Sams {

psvm() {

PizzaFactory pfobj = new PepperoniPizzaFactory();

Pizza pobj = pfobj.createPizza();

}

}

**Answer**:

interface Pizza {

void preparation(); void baking(); void cutting(); void boxing();

}

class PepperoniPizza implements Pizza {

public void preparation(){ System.out.println(&quot;prep pepperoni&quot;); }

public void baking(){ System.out.println(&quot;bake 220C&quot;); }

public void cutting(){ System.out.println(&quot;cut 8 slices&quot;); }

public void boxing(){ System.out.println(&quot;box standard&quot;); }

}

interface PizzaFactory { Pizza createPizza(); }

class PepperoniPizzaFactory implements PizzaFactory {

public Pizza createPizza(){ return new PepperoniPizza(); }

}

class Sams {

public static void main(String[] args){

PizzaFactory pf = new PepperoniPizzaFactory();

Pizza p = pf.createPizza();

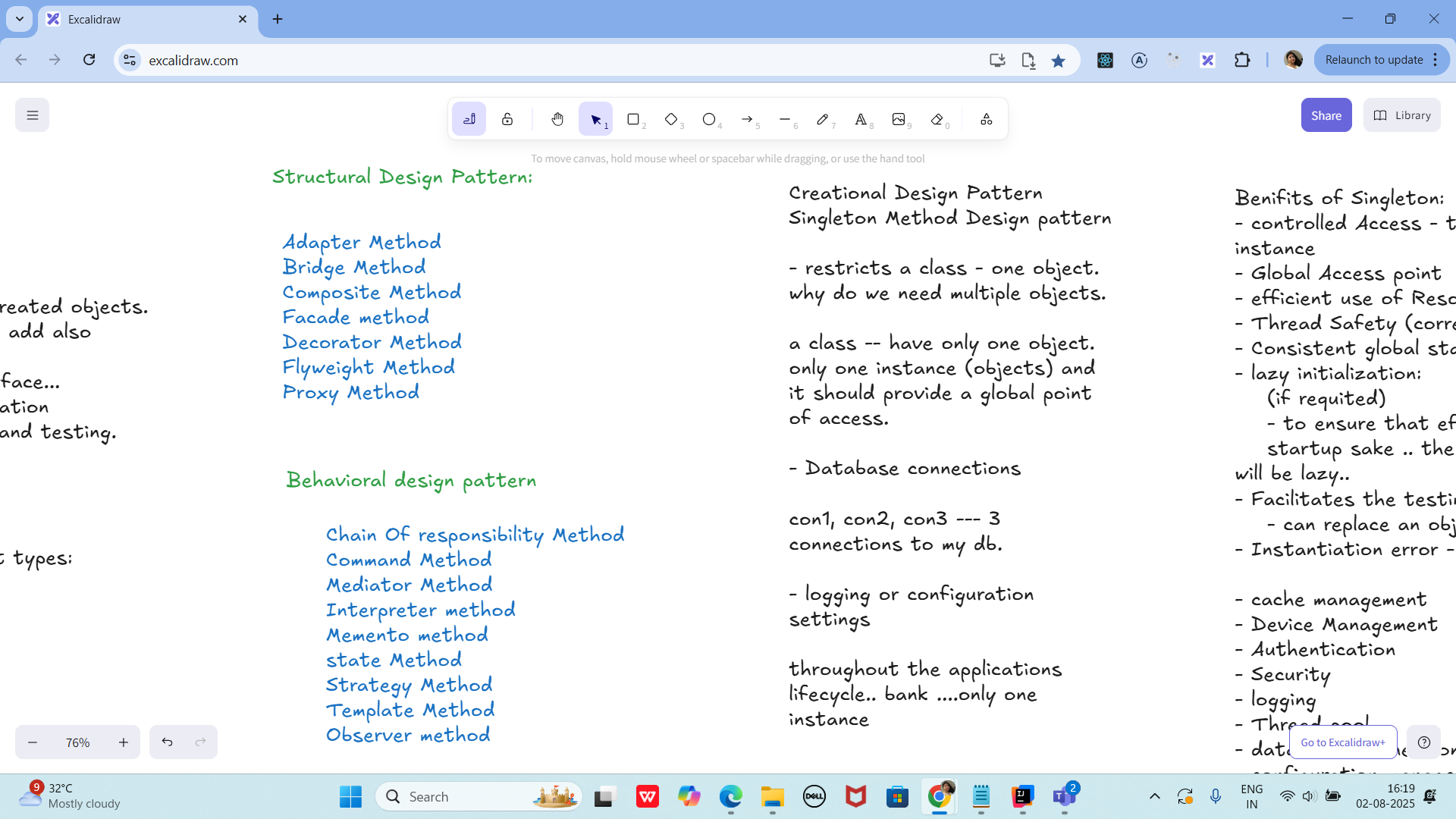
p.preparation(); p.baking(); p.cutting(); p.boxing();

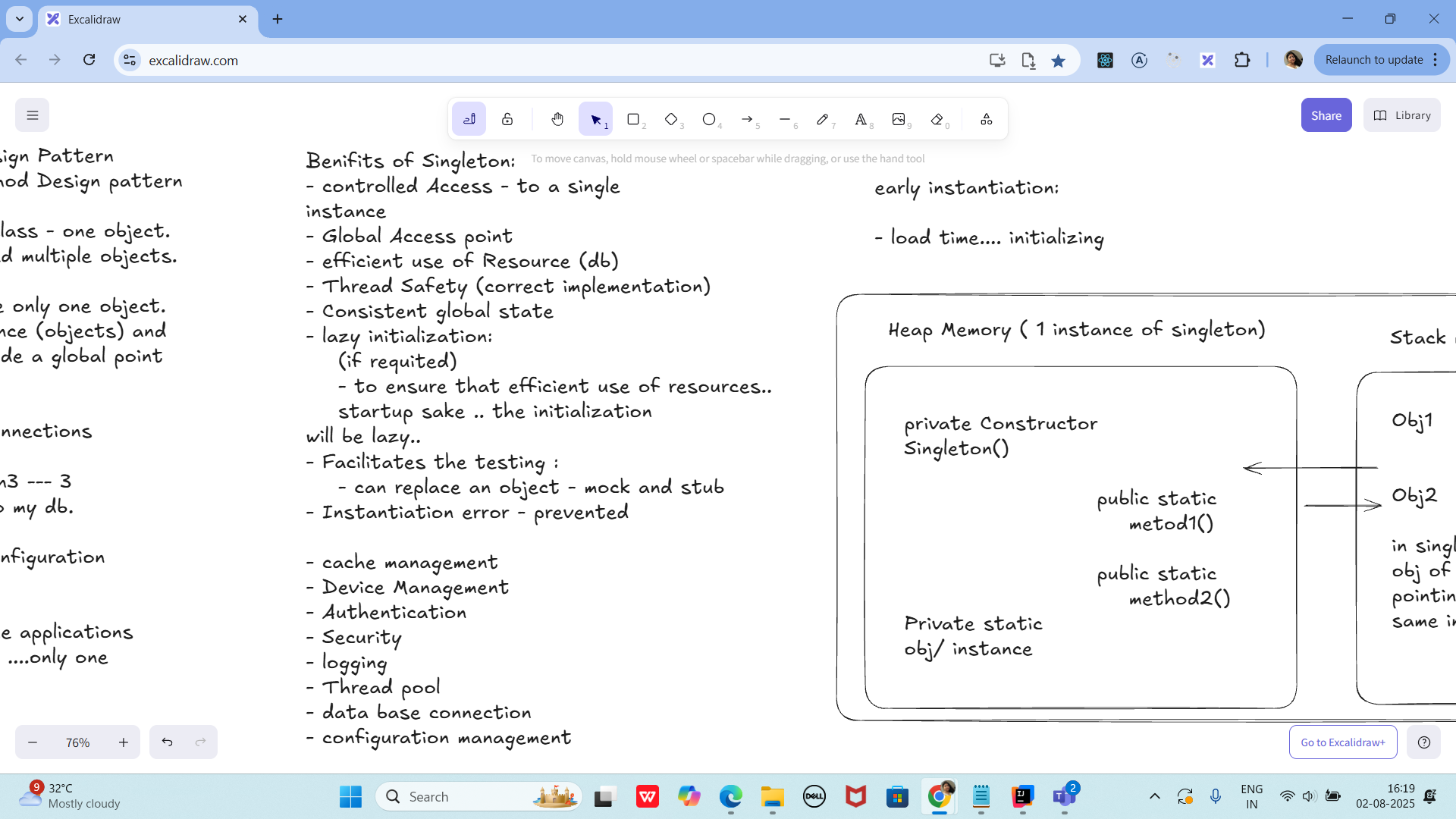
}

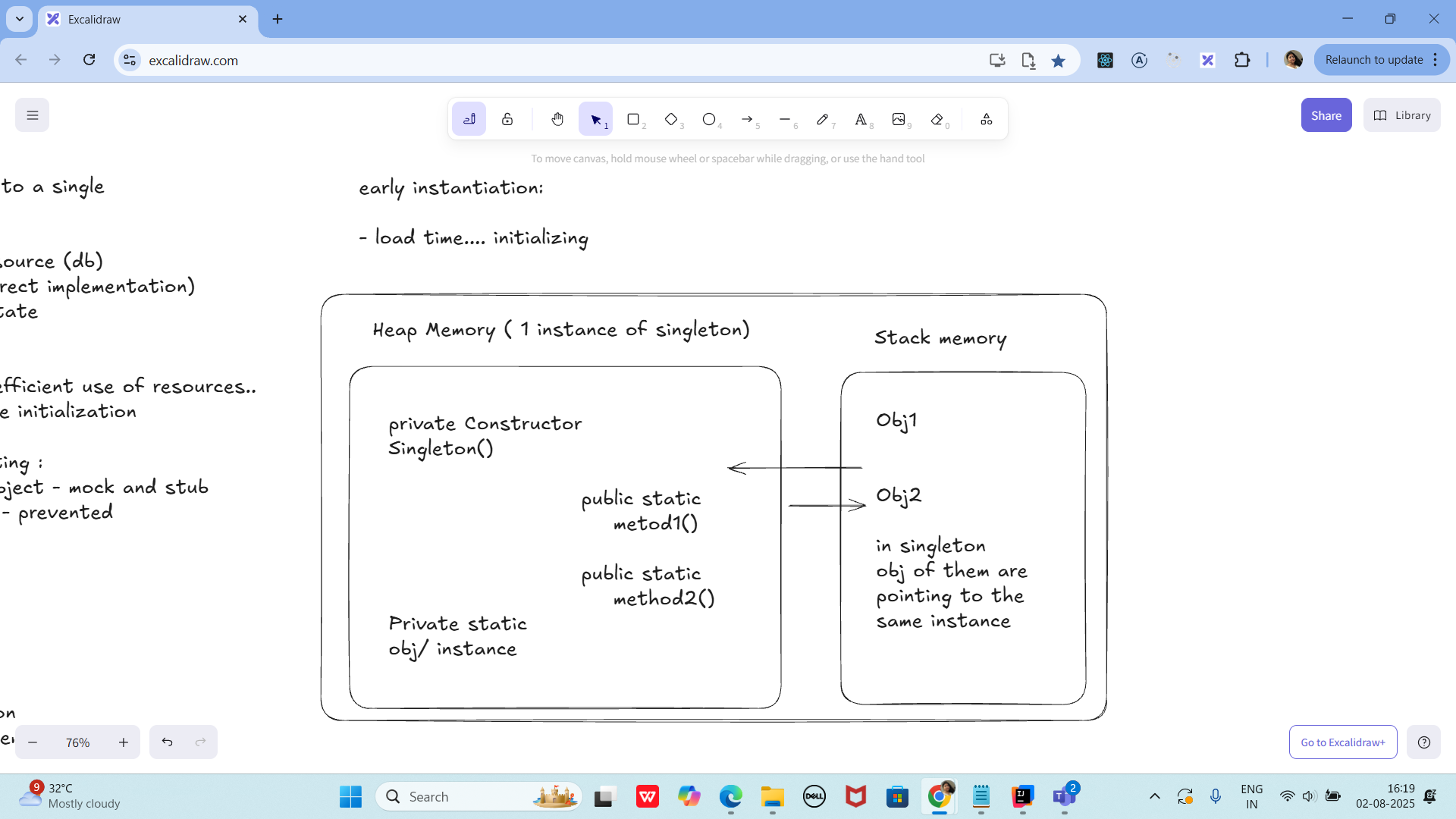
}

—------------------------------------------------------------------------------------------------------

Singleton Method Design principle







Task 03:

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");

}

}

public class SingletonDP {

public static void main(String[] args) {

SingletonDemo.*getInstance*().*doHere*();

// SingletonDemo obj = new SingletonDemo();

// obj.doHere();

// SingletonDemo obj2 = new SingletonDemo();

// obj2.doHere();

}

}

**Answer**:

class SingletonDemo {

private static volatile SingletonDemo instance;

private SingletonDemo(){ System.out.println(&quot;initiating the singleton&quot;); }

public static SingletonDemo getInstance(){

if(instance==null){

synchronized(SingletonDemo.class){

if(instance==null){ instance = new SingletonDemo(); }

}

}

return instance;

}

public void doHere(){ System.out.println(&quot;doing here some thing&quot;); }

}

public class SingletonDP{

public static void main(String[] args){

SingletonDemo.getInstance().doHere();

}

}

—----------------------------------------------------------------------------------------------------------------------------------

—----------------------------------------------------------------------------------------------------------------------------------

Home Tasks:

Home Task 1:

Write a test case for the below java file

public class Junit4Test {

public int compare(int n1, int n2) {

if (n1 &gt; n2) return 1;

return -1;

}

}

Answer:

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class Junit4TestTest {

@Test

public void compare\_returns1\_whenFirstGreater(){

assertEquals(1, new Junit4Test().compare(5,3));

}

@Test

public void compare\_returnsMinus1\_whenFirstNotGreater(){

assertEquals(-1, new Junit4Test().compare(2,7));

}

}

Home task 2:

public int compare(int n1, int n2) {

if (n1 &gt; n2) return 1;

else if (n1 &lt; n2) return -1;

return 0;

}

Answer:

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class CompareV2Test {

@Test public void gt(){ assertEquals(1, compare(9,3)); }

@Test public void lt(){ assertEquals(-1, compare(3,9)); }

@Test public void eq(){ assertEquals(0, compare(5,5)); }

private int compare(int n1,int n2){

if(n1&gt;n2) return 1; else if(n1&lt;n2) return -1; return 0;

}

}

Hoe Task 3:

Include a test case and run the test suite.

Test Suite:

 its a bundle of unit test cases and run them

all together.

@Suite

@RunWith

TestSuite Demo

import org.junit.runner.RunWith;

import org.junit.runners.Suite;

@RunWith(Suite.class)

@Suite.SuiteClasses( {

JunitTest01.class,

JunitTest02.class,

JunitTest03.class

})

public class TestSuiteDemo {

}

JunitTest01.java

import org,junit.Test;

import static org.junit.Assert.assetEquals;

import org.junit.Igore;

public class junitTest01 {

String msg = &quot;running test01 &quot;;

MessageUtil msgUtilobj = new MessageUtil(msg);

//MessageUtil is a utility class

//store a message

@Test

public void msgTest() {

sout(&quot;we are inside the msgtest()&quot;);

assertEquals(msg, msgUtilobj.printMessage());

}

}

junitTest02.java

test Runner

Home Task 4:

Implement the above code also add one more test case and test runner.. And execute the test

suite.

Answer:

// TestSuiteDemo.java (JUnit4)

import org.junit.runner.RunWith;

import org.junit.runners.Suite;

@RunWith(Suite.class)

@Suite.SuiteClasses({ JunitTest01.class, JunitTest02.class, Junit4TestTest.class })

public class TestSuiteDemo { }

// JunitTest01.java

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class JunitTest01 {

@Test

public void msgTest(){

MessageUtil mu = new MessageUtil(&quot;running test01 &quot;);

assertEquals(&quot;running test01 &quot;, mu.printMessage());

}

}

// JunitTest02.java

import org.junit.Test;

import static org.junit.Assert.assertTrue;

public class JunitTest02 {

@Test

public void simpleTruth(){ assertTrue(2+2==4); }

}

// TestRunner.java (optional)

import org.junit.runner.JUnitCore;

import org.junit.runner.Result;

import org.junit.runner.notification.Failure;

public class TestRunner{

public static void main(String[] args){

Result r = JUnitCore.runClasses(TestSuiteDemo.class);

for(Failure f: r.getFailures()) System.out.println(f.toString());

System.out.println(&quot;SUCCESS: &quot;+r.wasSuccessful());

}

}

Home Task 5:

Plz try this code in uml

@startuml

participant Pax as p

actor Actor as a #Green

boundary Boundary as b

control C as c

database Dbase as db

entity table as t

collections col as c

queue que as q

autonumber

p -[#red]> a : pax to actor

a ->x p: ohh is it!

autonumber 100

b ->> p: boundary to pax

note left: plz make a note on left

c -\ p: control to pax

db \\- p: db to pax

autonumber 50 10

t //-- db: table to db

c ->o db: collection to db

note right: plz make a note

q <->o p: q to pax

Prasunamba -> Batch1 : learning uml

Batch1 -> Prasunamba: ok will learn

@enduml

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

**Info Box**

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

Info Box (Carry forward from Day 22 02nd Aug juz fyi..

Plz refer Best Programming Practices - Doc 08 in Docs to study

Updated link at 12.17 ..

<https://excalidraw.com/#json=lKmZDbhZ5V6ZRlQ3zJOXv,K6r0FziLUTf3htjdUrnN4w>

Updated link at 15.57

<https://excalidraw.com/#json=Mp0F2MJiO57HKm74LVHXK,XmhQybYlqAswiHdRIeokDQ>

Junit dependency

<https://junit.org/junit4/dependency-info.html>

<dependencies >

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

</dependency>

</dependencies >

The above dependency you need to add in the dependencies tag of pom.xml..

Junit Official link

Download

Hamcrest and Junit.jar file

<https://junit.org/>

Or add the below dependency

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-api</artifactId>

<version>6.0.0-M2</version>

</dependency>

Hamcrest dependency

<!-- https://mvnrepository.com/artifact/org.hamcrest/hamcrest-junit -->

<dependency>

<groupId>org.hamcrest</groupId>

<artifactId>hamcrest-junit</artifactId>

<version>2.0.0.0</version>

<scope>test</scope>

</dependency>

JUNIT Jupiter api - junit.jar file link

<https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api>

Junit Hancrest

<https://mvnrepository.com/artifact/org.hamcrest/hamcrest>

Exclidraw - updated link at 12.33..

[https://excalidraw.com/#jso=cY4X2JOrLIvswxiToAxLB,01GKPhOFi8r3tEbo8hLCbw](https://excalidraw.com/#json=cY4X2JOrLIvswxiToAxLB,01GKPhOFi8r3tEbo8hLCbw)

Excalidraw - updated link at 16.20

<https://excalidraw.com/#json=udpoNQTqmw4D3clX_tr_9,w7du3hqQcDaDyG7PGsZtzw>

Codes for reference link → for uml codes

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

Plz go through uml diagrams in page nos:

Docs 04 OOAD UML in doc to study

8 to 16

12

18 to 39

57 to 58

97

118

123

136

154 to 160

Plantuml site juz found .. plz check for reference

<https://plantuml.com/use-case-diagram>

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