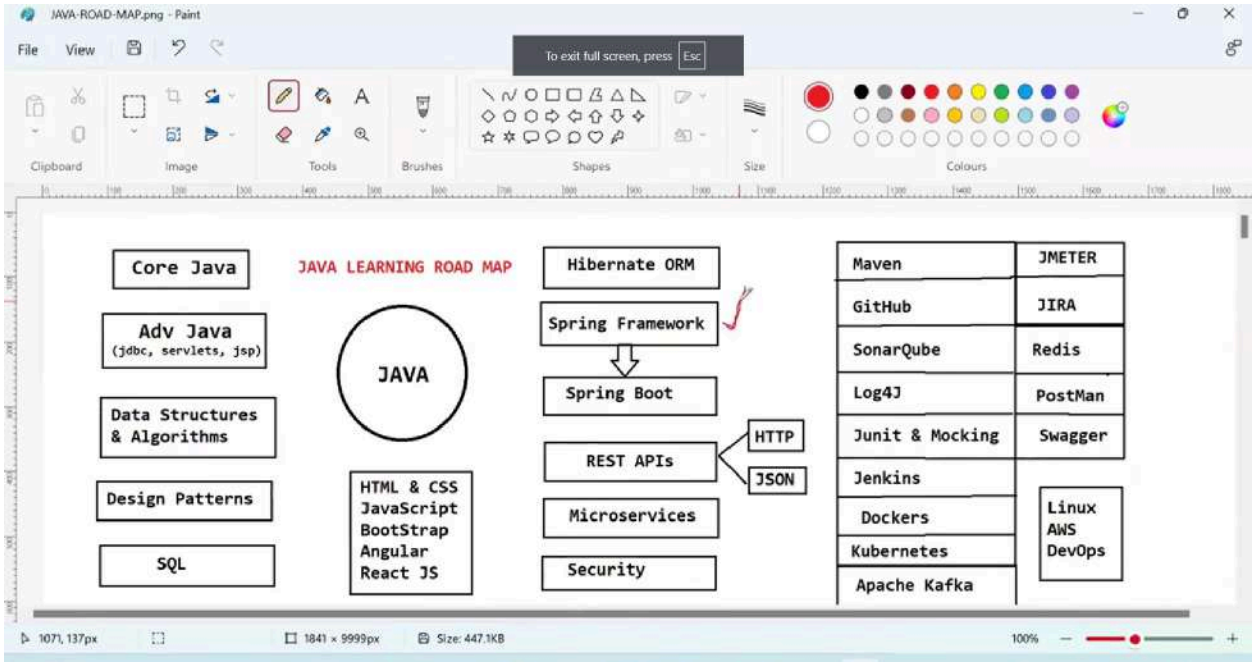
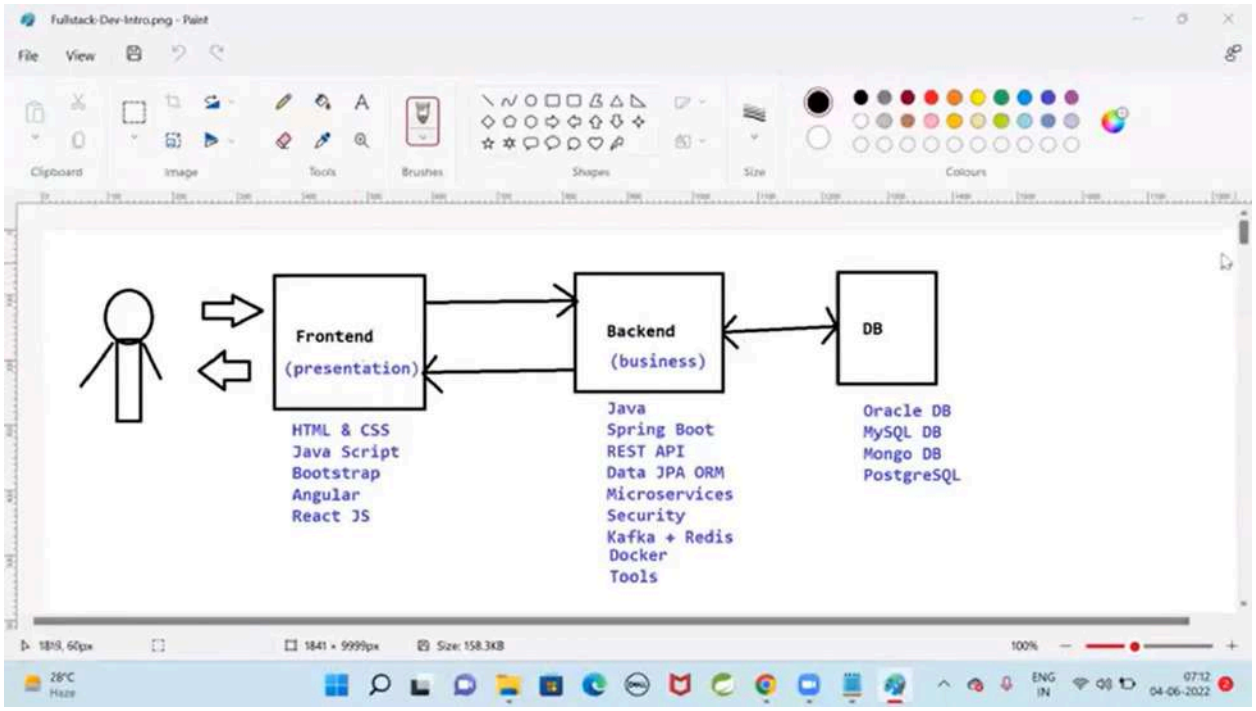
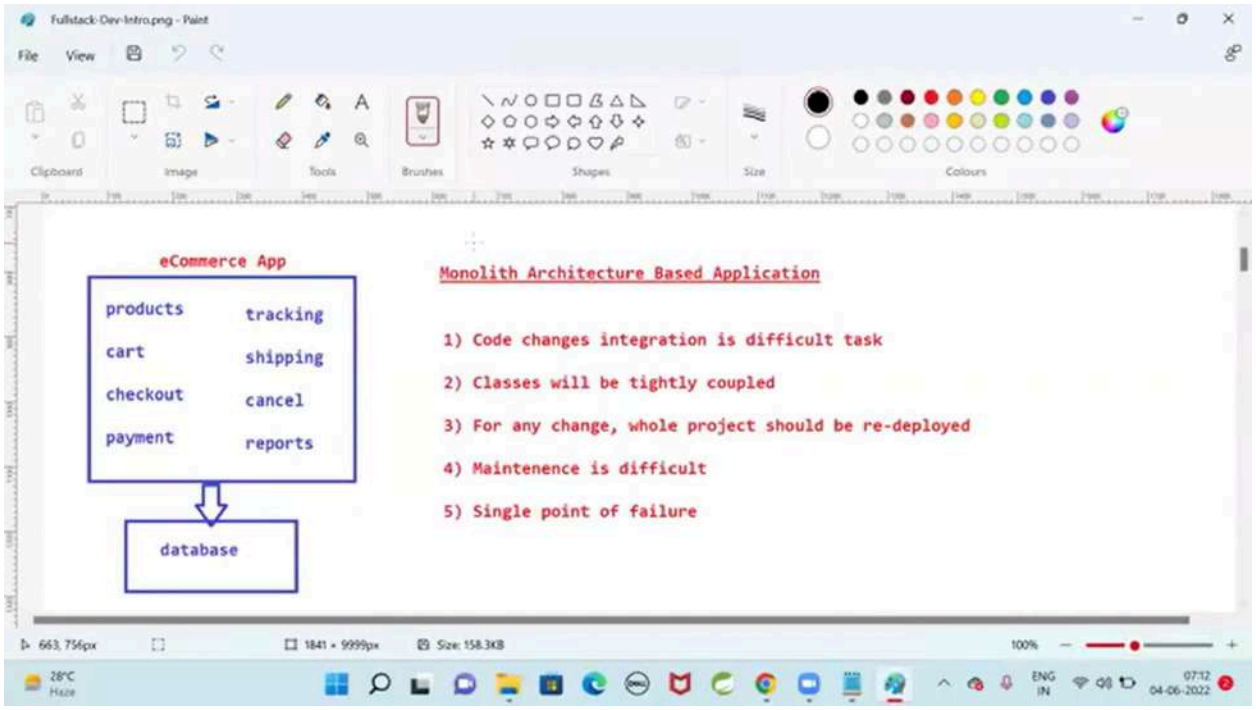
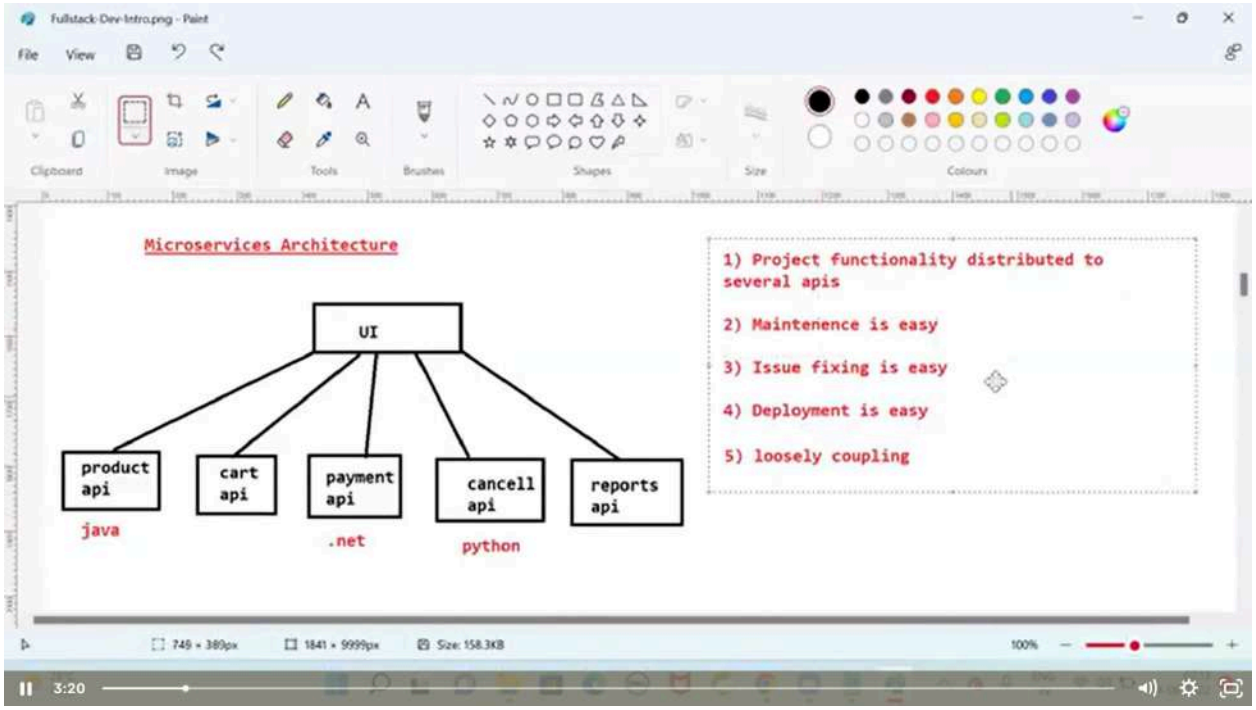
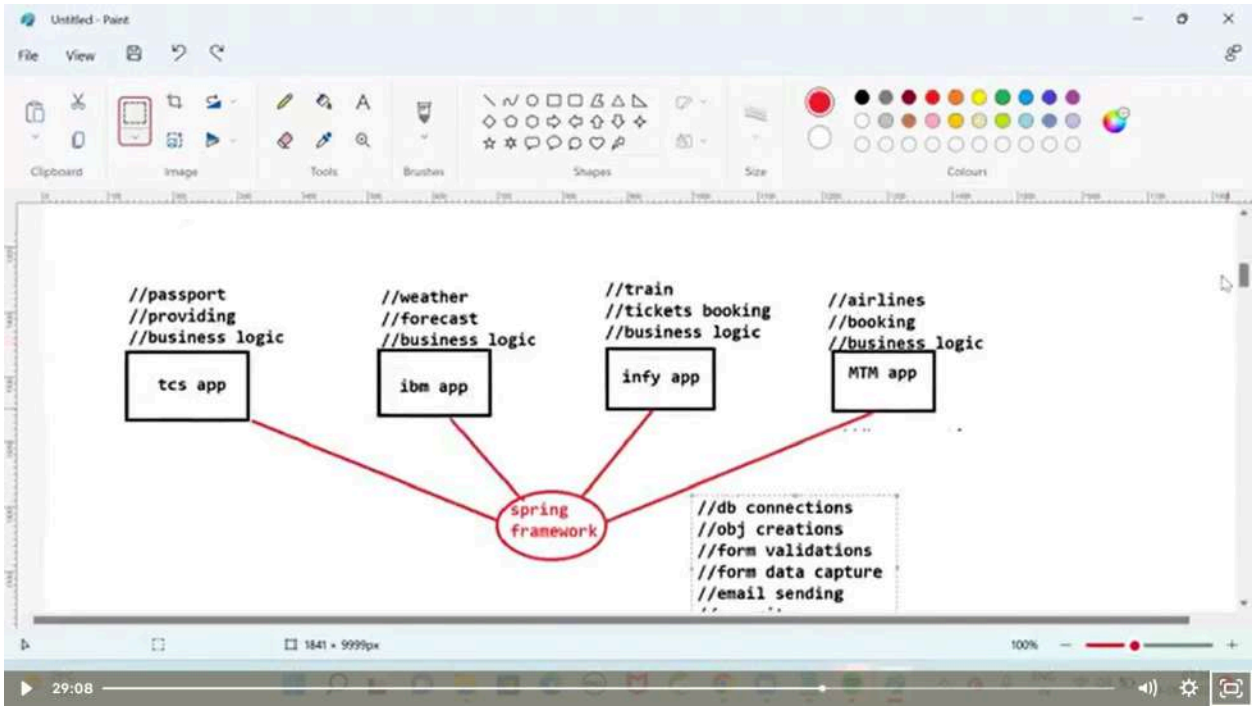
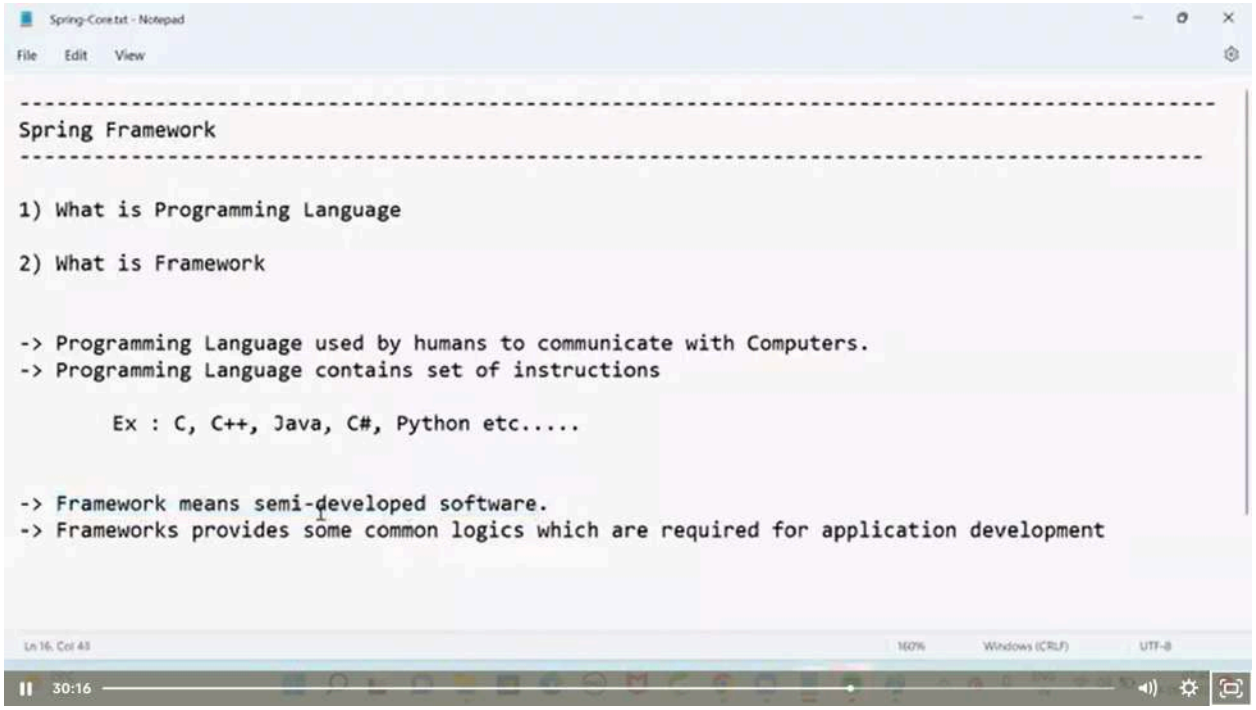

JAVA-ROAD-MAP.png - Paint
File
View
To exit full screen, press Esc
Clipboard
Image
Tools
Brushes
Shapes
Size
Colours
Core Java
JAVA LEARNING ROAD MAP
Hibernate ORM
Spring Framework
Spring Boot
REST APIs
HTTP
JSON
Microservices
Security
Adv Java
(jdbc, servlets, jsp)
Data Structures
& Algorithms
Design Patterns
SQL
JAVA
HTML & CSS
JavaScript
BootStrap
Angular
React JS
Maven
GitHub
SonarQube
Log4J
Junit & Mocking
Jenkins
Dockers
Kubernetes
Apache Kafka
JMETER
JIRA
Redis
PostMan
Swagger
Linux
AWS
DevOps
1071, 137px
1841 × 9999px
Size: 447.1KB
100%


Fullstack-Dev-Intro.png - Paint
File
View
Clipboard
Image
Tools
Brushes
Shapes
Size
Colours
Frontend
(presentation)
HTML & CSS
Java Script
Bootstrap
Angular
React JS
Backend
(business)
Java
Spring Boot
REST API
Data JPA ORM
Microservices
Security
Kafka + Redis
Docker
Tools
DB
Oracle DB
MySQL DB
Mongo DB
PostgreSQL
1819, 60px
1841 × 9999px
Size: 158.3KB
100%
28°C
Haze
ENG
IN
07:12
04-06-2022


Fullstack-Dev-Intro.png - Paint
File
View
eCommerce App
products
cart
checkout
payment
tracking
shipping
cancel
reports
database
Monolith Architecture Based Application
1) Code changes integration is difficult task
2) Classes will be tightly coupled
3) For any change, whole project should be re-deployed
4) Maintenence is difficult
5) Single point of failure


Fullstack-Dev-Intro.png - Paint
File
View
Microservices Architecture
UI
product api
java
cart api
payment api
.net
cancell api
python
reports api
1) Project functionality distributed to several apis
2) Maintenence is easy
3) Issue fixing is easy
4) Deployment is easy
5) loosely coupling
3:20


Untitled - Paint
File View
Clipboard
Image
Tools
Brushes
Shapes
Size
Colours
//passport
//providing
//business logic
tcs app
//weather
//forecast
//business logic
ibm app
//train
//tickets booking
//business logic
infy app
//airlines
//booking
//business logic
MTM app
spring framework
//db connections
//obj creations
//form validations
//form data capture
//email sending
1841 × 9999px
100%
29:08


Spring-Core.txt - Notepad
File Edit View
Spring Framework
1) What is Programming Language
2) What is Framework
-> Programming Language used by humans to communicate with Computers.
-> Programming Language contains set of instructions
Ex : C, C++, Java, C#, Python etc.....
-> Framework means semi-developed software.
-> Frameworks provides some common logics which are required for application development
Ln 16, Col 43
160%
Windows (CRLF)
UTF-8
30:16

Spring-Core.txt - Notepad

File Edit View

-> Frameworks provides some common logics which are required for application development

Ex: Hibernate, Struts, Spring etc...

-> If we use framework in our application development then we need to focus only on business logic (framework will provide common logics)

Hibernate : It is an ORM framework. Used to develop persistence layer of our application

Struts : It is a web framework. Used to develop web layer of our application.

Spring : It is an application development framework. Entire project can be developed by using this.

(we can do application end to end development)

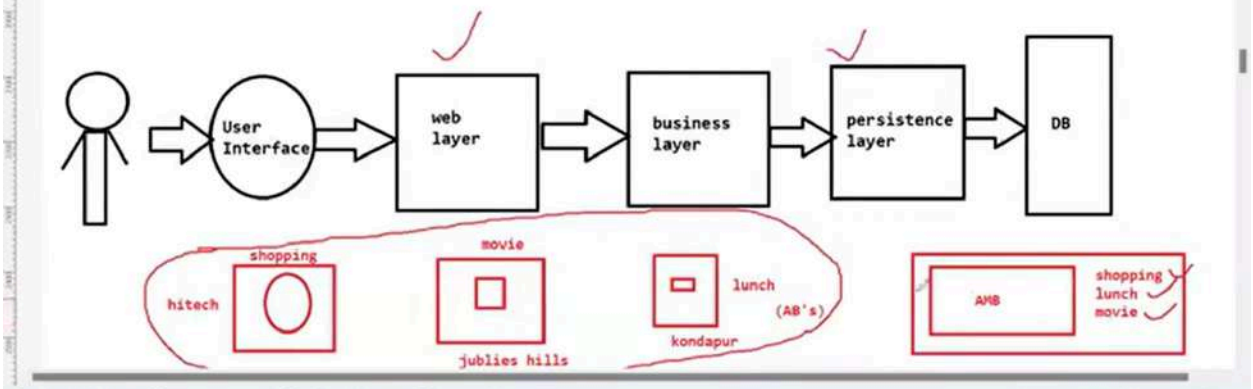
Ln 24, Col 37 150% Windows (CRLF) UTF-8

29°C Haze ENG IN 08:10 04-06-2022

Spring-Overview.png - Paint

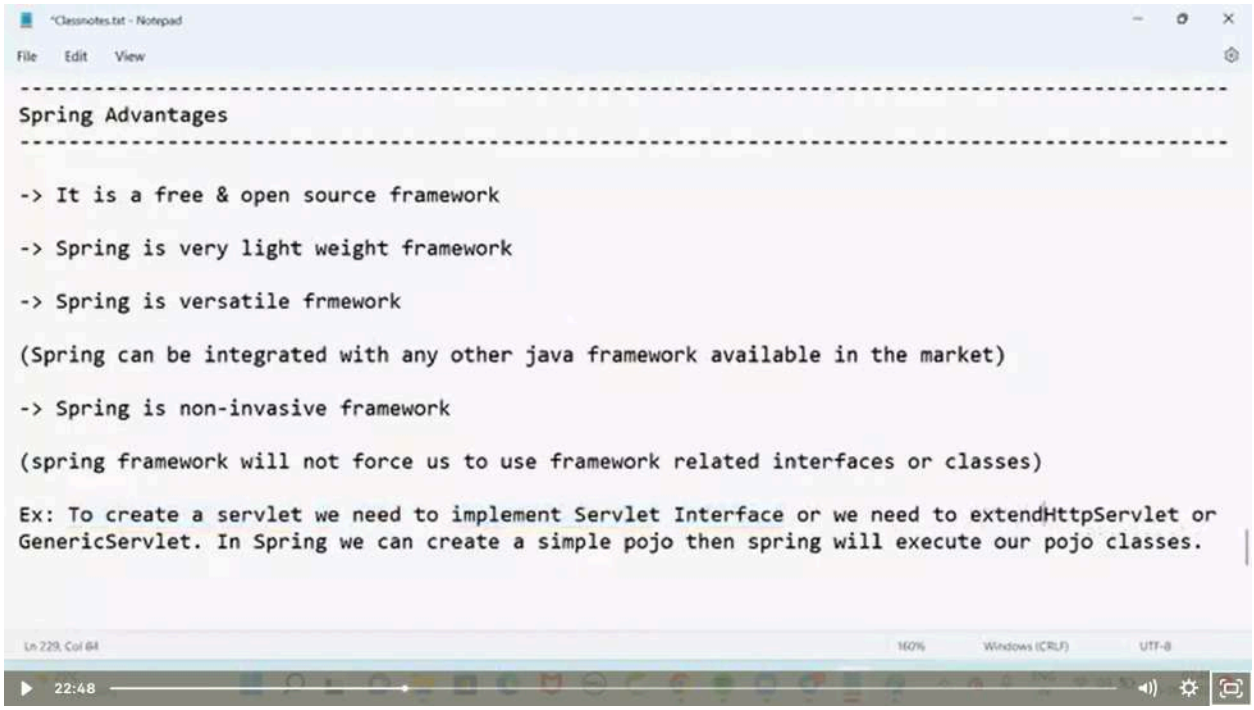
File View

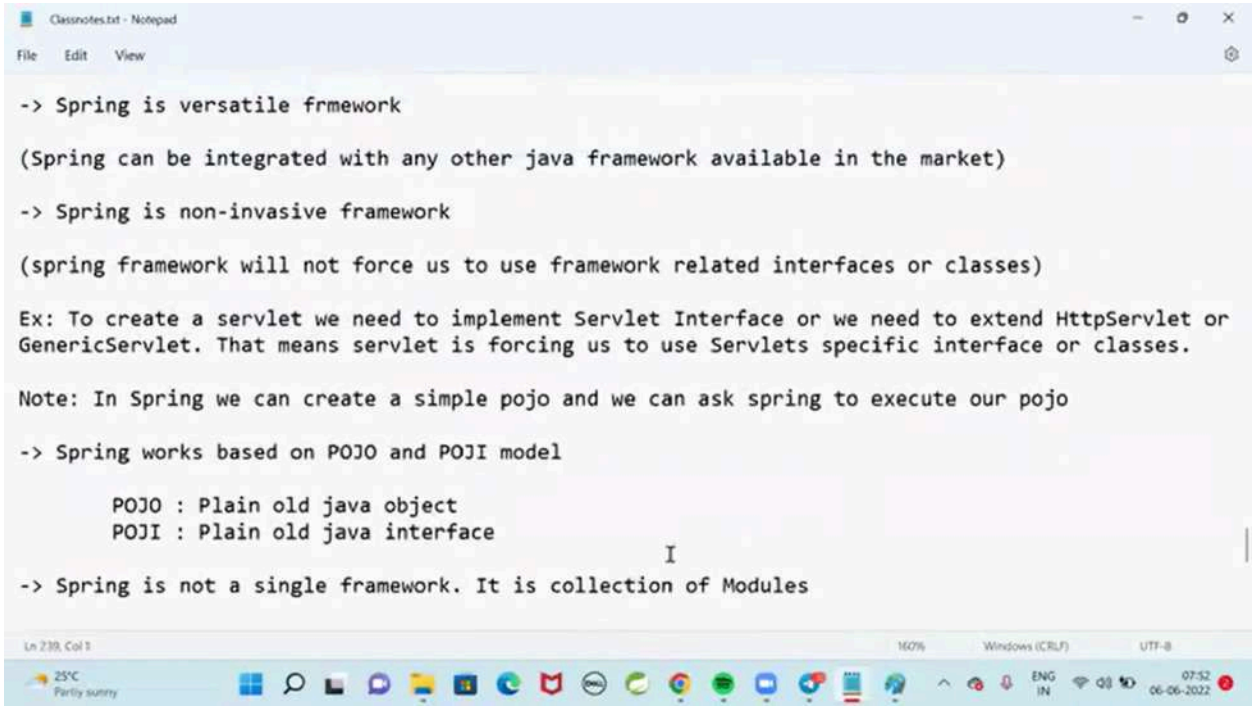
Clipboard Image Tools Brushes Shapes Size Colours




1459, 2444px 1841 × 9999px Size: 215.6KB 100%

27°C Haze ENG IN 07:37 06-06-2022


*Classnotes.txt - Notepad
File Edit View
Spring Advantages
-> It is a free & open source framework
-> Spring is very light weight framework
-> Spring is versatile frmework
(Spring can be integrated with any other java framework available in the market)
-> Spring is non-invasive framework
(spring framework will not force us to use framework related interfaces or classes)
Ex: To create a servlet we need to implement Servlet Interface or we need to extendHttpServlet or GenericServlet. In Spring we can create a simple pojo then spring will execute our pojo classes.
Ln 229, Col 84
160%
Windows (CRLF)
UTF-8
22:48


Classnotes.txt - Notepad
File Edit View
-> Spring is versatile frmework
(Spring can be integrated with any other java framework available in the market)
-> Spring is non-invasive framework
(spring framework will not force us to use framework related interfaces or classes)
Ex: To create a servlet we need to implement Servlet Interface or we need to extend HttpServlet or GenericServlet. That means servlet is forcing us to use Servlets specific interface or classes.
Note: In Spring we can create a simple pojo and we can ask spring to execute our pojo
-> Spring works based on POJO and POJI model
POJO : Plain old java object
POJI : Plain old java interface
-> Spring is not a single framework. It is collection of Modules
Ln 239, Col 1
160%
Windows (CRLF)
UTF-8
25°C
Partly sunny
ENG
IN
07:52
06-06-2022

Classnotes.txt - Notepad

File Edit View

To exit full screen, press Esc

```
-> Spring is not a single framework. It is collection of Modules

-----------------------------------------------------------------------------------------------
Spring Modules
-----------------------------------------------------------------------------------------------

1) Spring Core
2) Spring Context
3) Spring DAO / Spring JDBC
5) Spring AOP
6) Spring ORM
7) Spring Web MVC
8) Spring Security
9) Spring REST
10) Spring Data
11) Spring Cloud
12) Spring Batch etc...
```

Ln 254, Col 24 170% Windows (CRLF) UTF-8

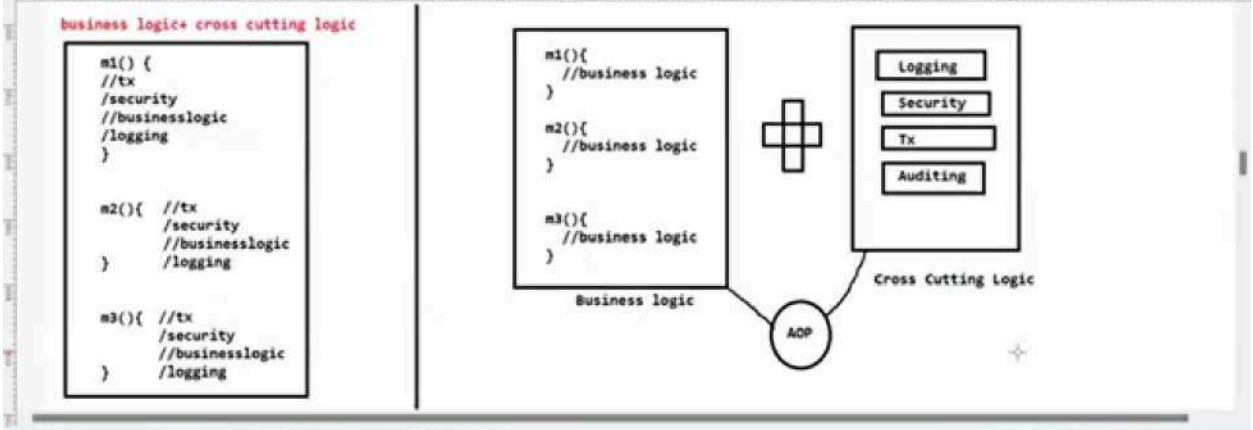
32:09

Spring-Overview.png - Paint

File View

To exit full screen, press Esc

Clipboard Image Tools Brushes Shapes Size Colours




1502, 4104px 1841 × 9999px Size: 215.6KB 100%

52:06

Classnotes.txt - Notepad

File Edit View

-> AOP stands for Aspect Oriented Programming. Spring AOP is used to deal with Cross Cutting logics in application.

Application = Business Logic + Cross Cutting Logic

Note: We can seperate business logic and cross cutting logic using AOP module.

-> Spring JDBC / Spring DAO module used to develop Persistence Layer

-> Spring ORM module is used to develop Persistence Layer with ORM features.

-> Spring Web MVC Module is used to develop Web Applications.

-> Spring Security module is used to implement Security Features in our application
(Authentication & Authorization)

-> Spring REST is used to develop RESTFul services (REST API)

-> Spring Data is used to persistence layer

Ln 277, Col 1 160% Windows (CRLF) UTF-8

58:08

Untitled - Paint

File View

Clipboard Image Tools Brushes Shapes Size Colours

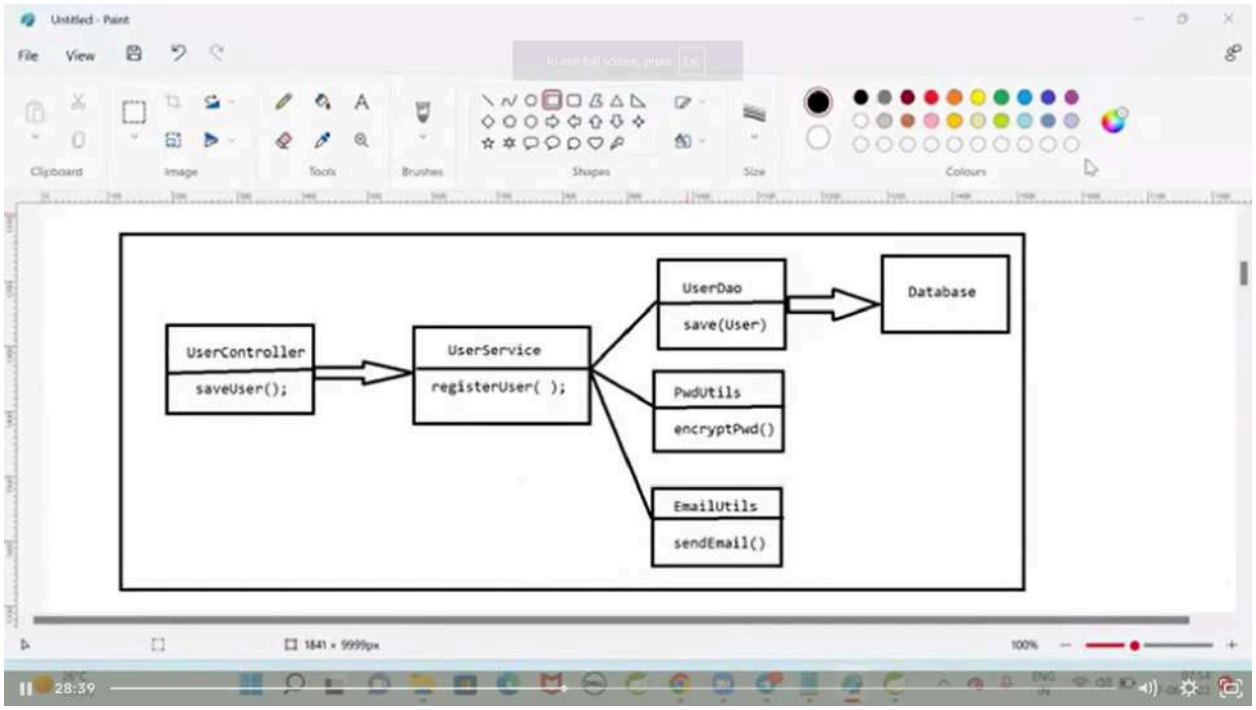
```java
package in.ashokit;

public class Engine {

	public int start() {
		// logic
		System.out.println("Engine started..!!");
		return 1;
	}
}
```

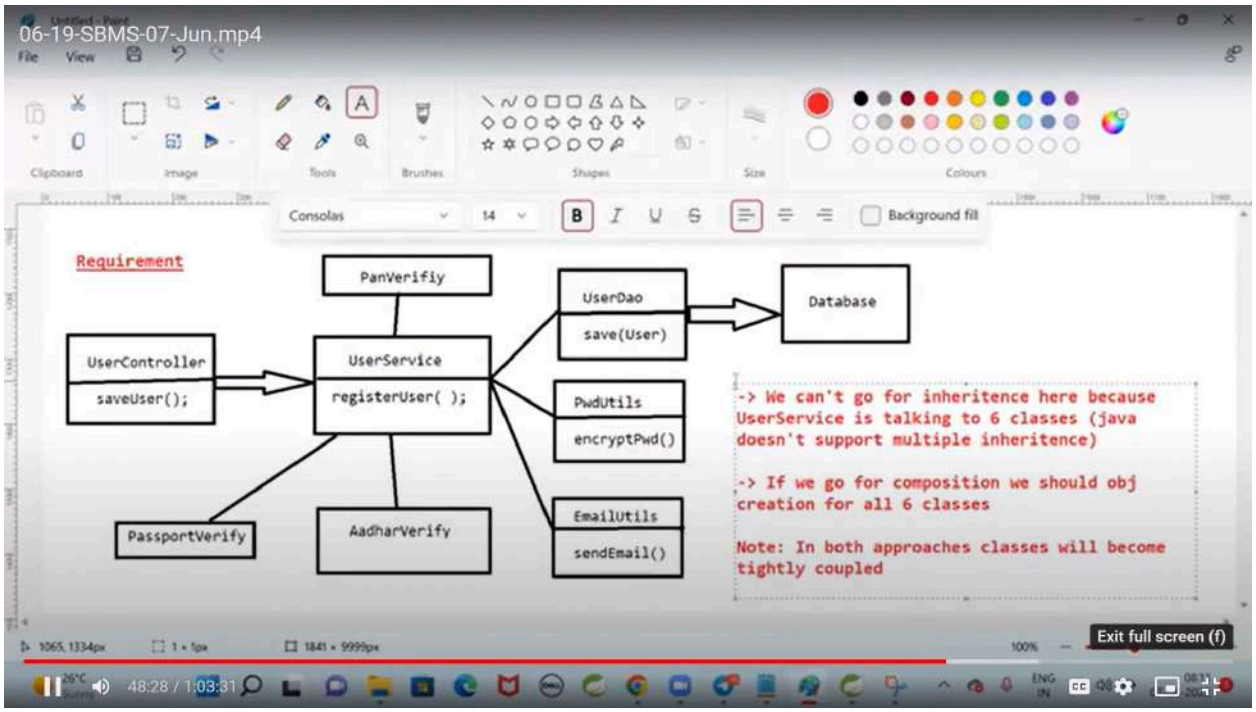
```java
package in.ashokit;

public class Car extends Engine {

	public void drive() {
		int status = super.start();

		if (status >= 1) {
			System.out.println("Journey Started...");
		} else {
			System.out.println("Engine Having Some Problem...");
		}
	}
}
```

1841 × 9999px 100%

Play Video

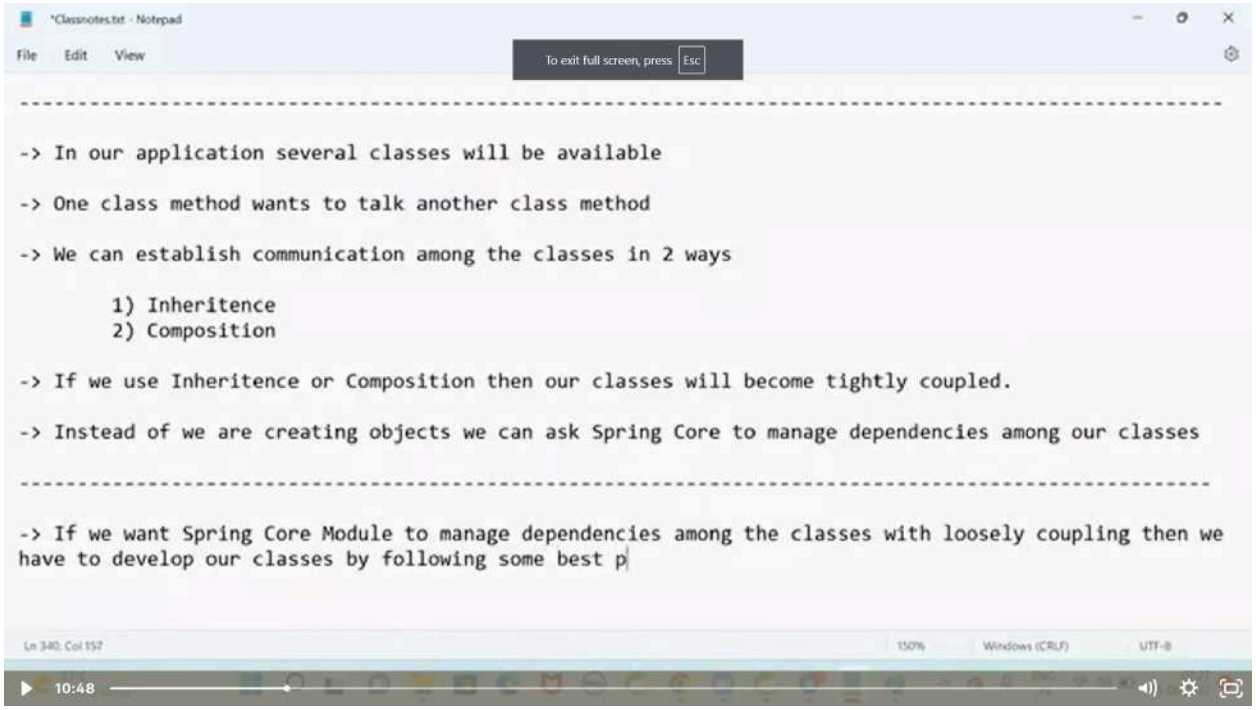
25:46



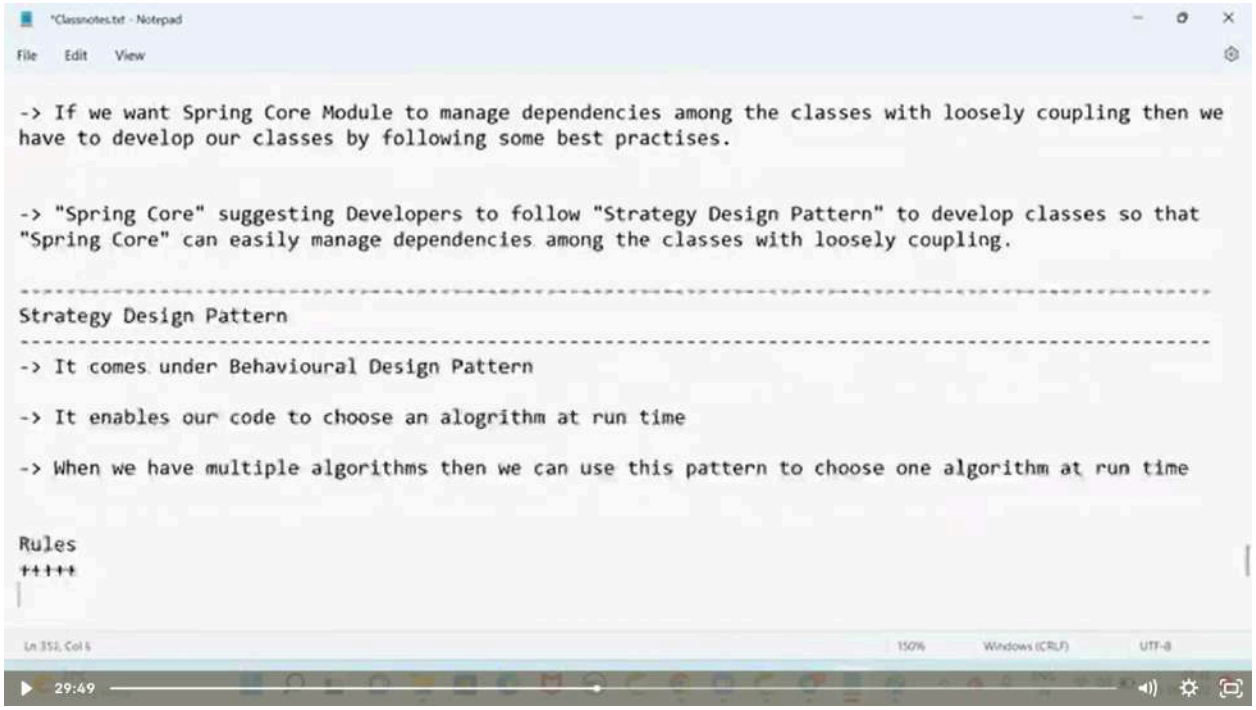



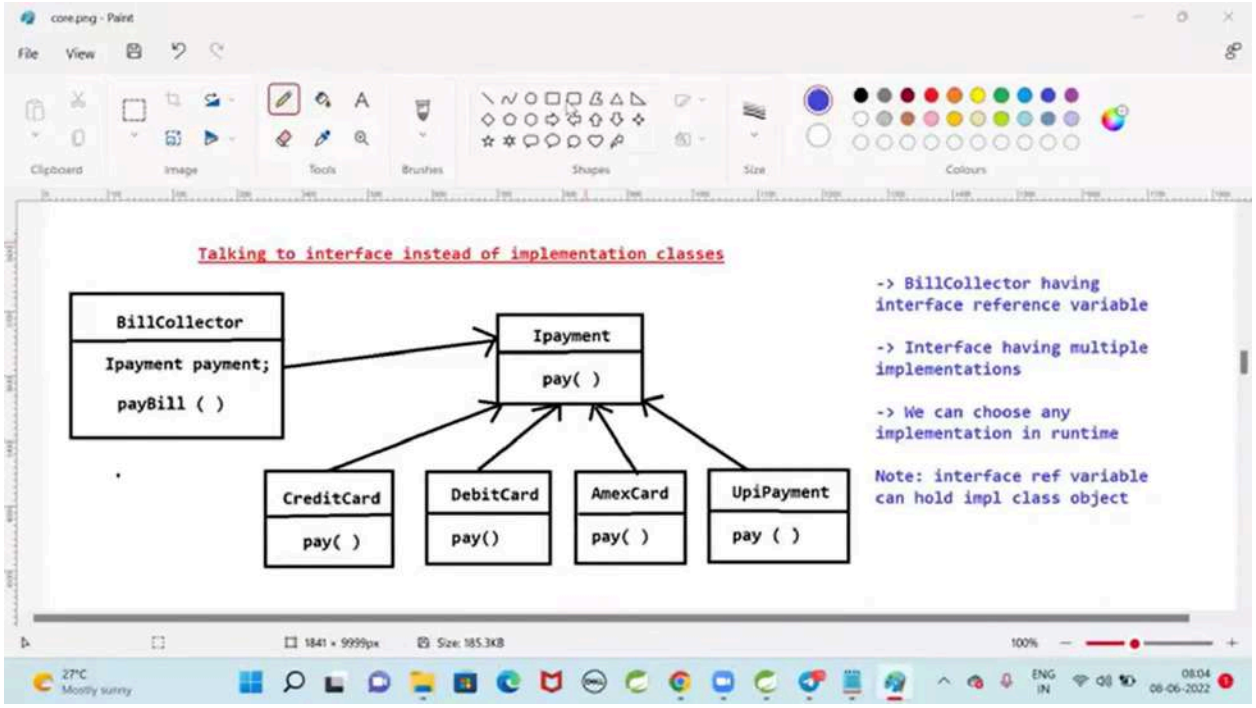

So above are how we classes communicate to each other
1. Through inheritance
2.through composition
But both are not recommended becoz when we use inheritance and one class want to extend multiple class but as we know java does not multiple inheritance.
And composition we need to create the objects for communicate but in future if other class change something then target class will affect so both not recommended.

– so overcome these problem spring core help for dependency injection so here all object creation are handle by IOC container



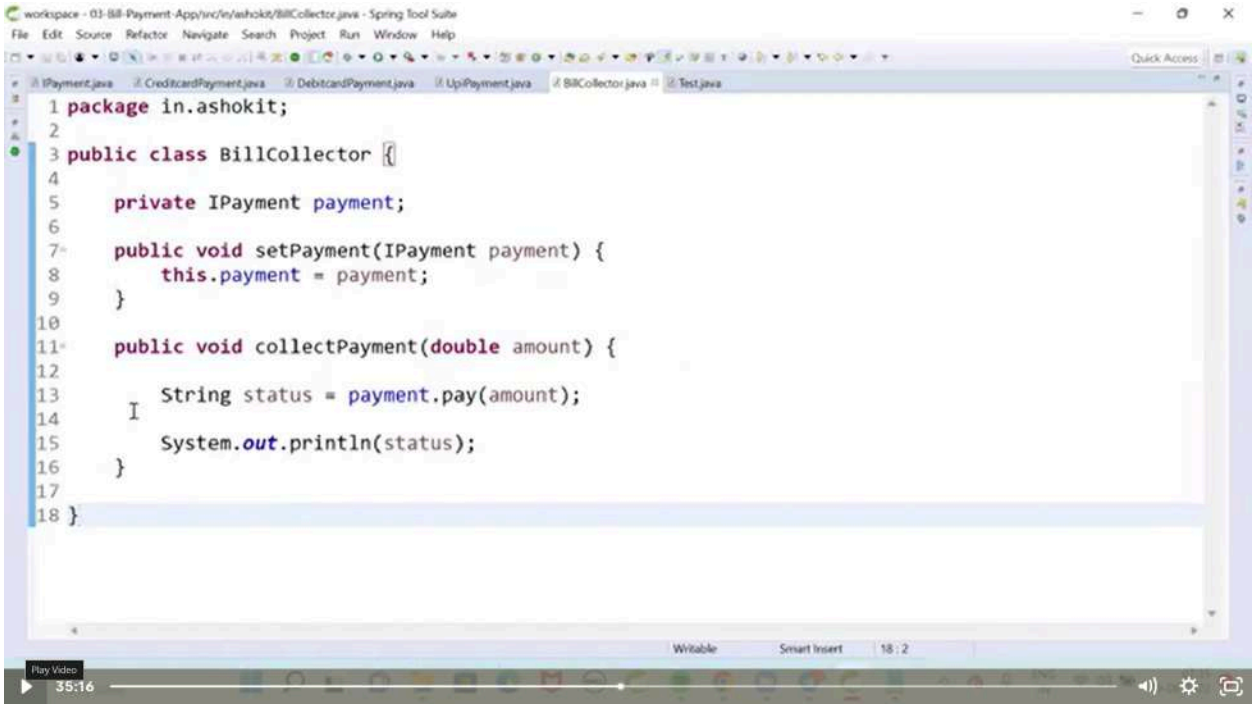

We have to make sure classes are loosely coupled

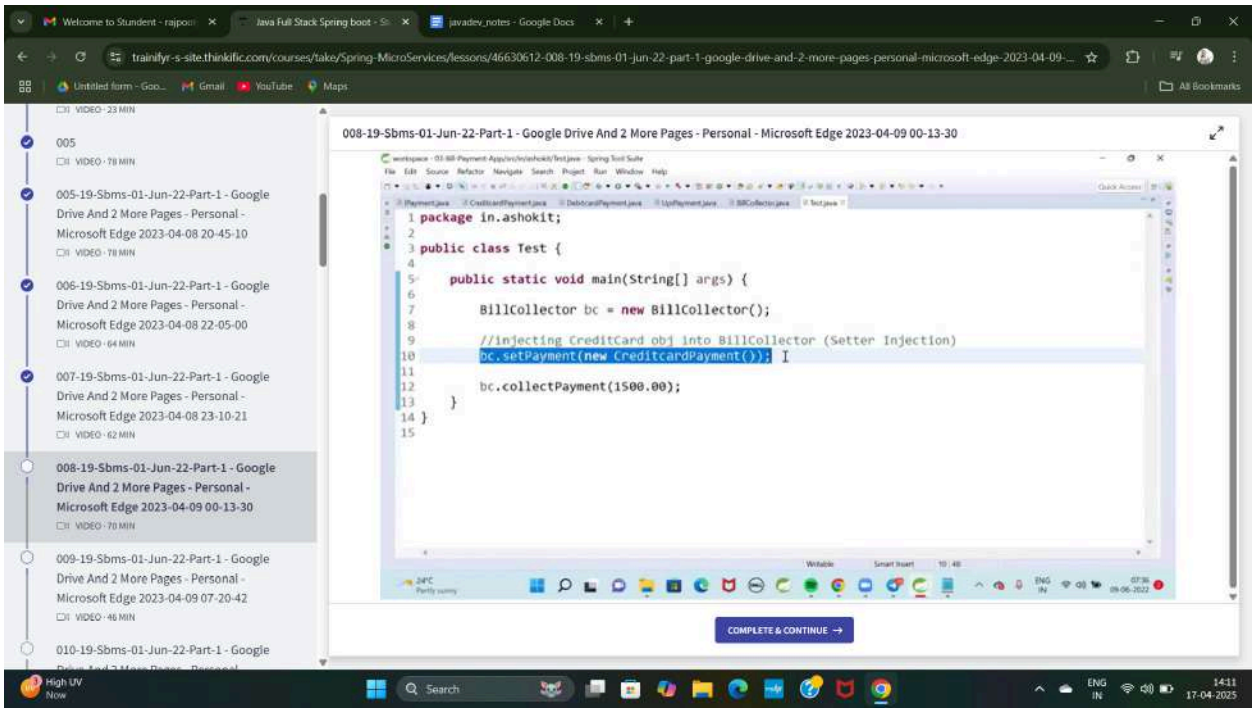




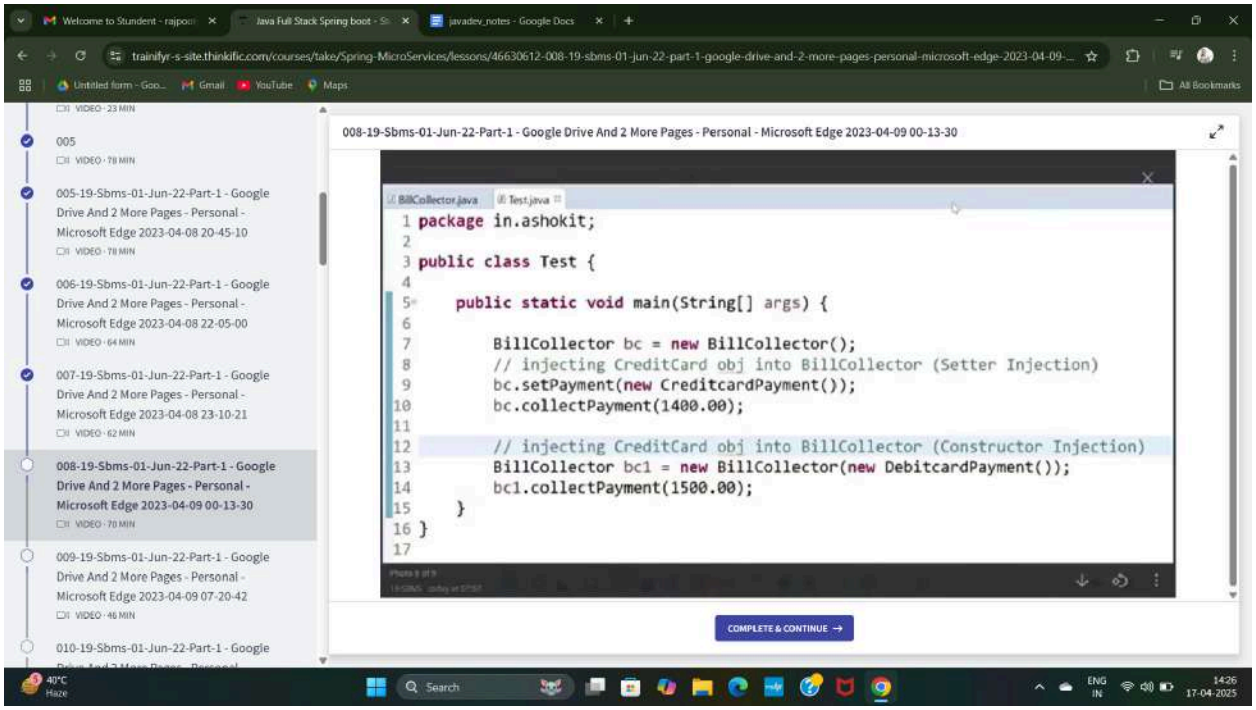

If we developed the classes through strategy design pattern then classees will be loosely coupled

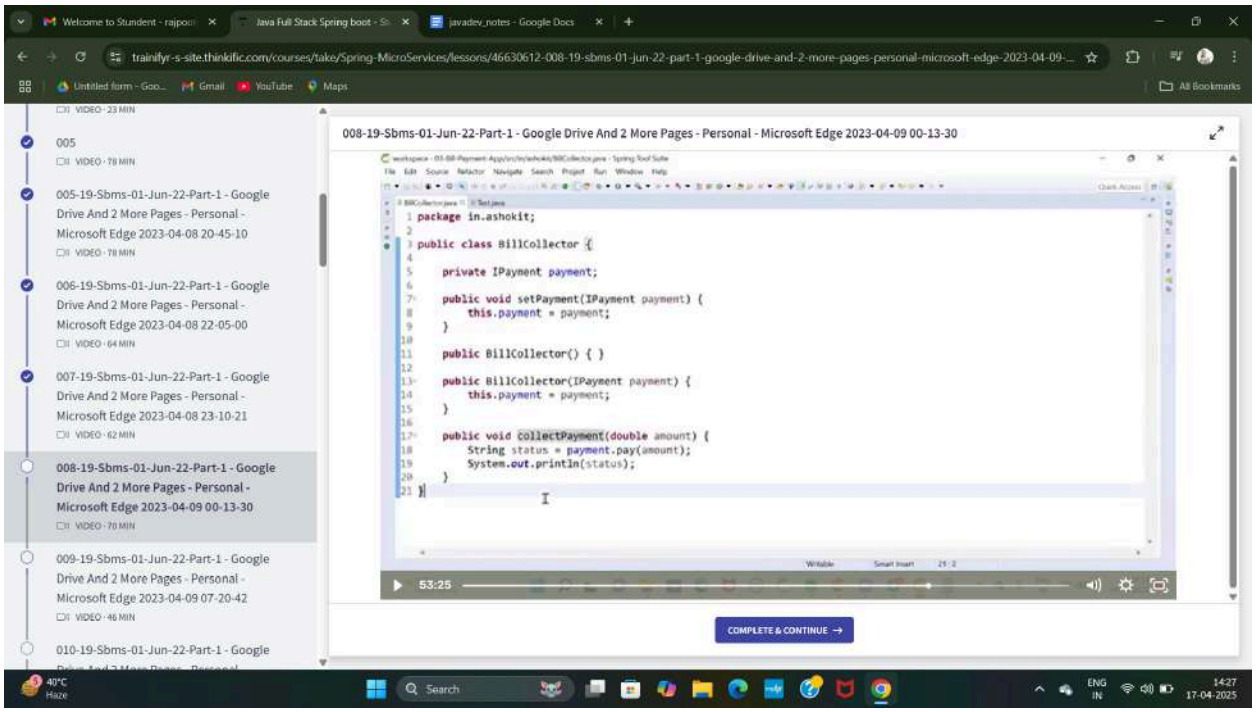
Note :- in short not a good way to  implement dependency injection through inheritance becoz of multiple inheritance java does not support and also through composition we need to make object of each class so not suitable this also becoz of tight coupling
Thsts why we use interface reference variable to make dependency injection loosly coupling

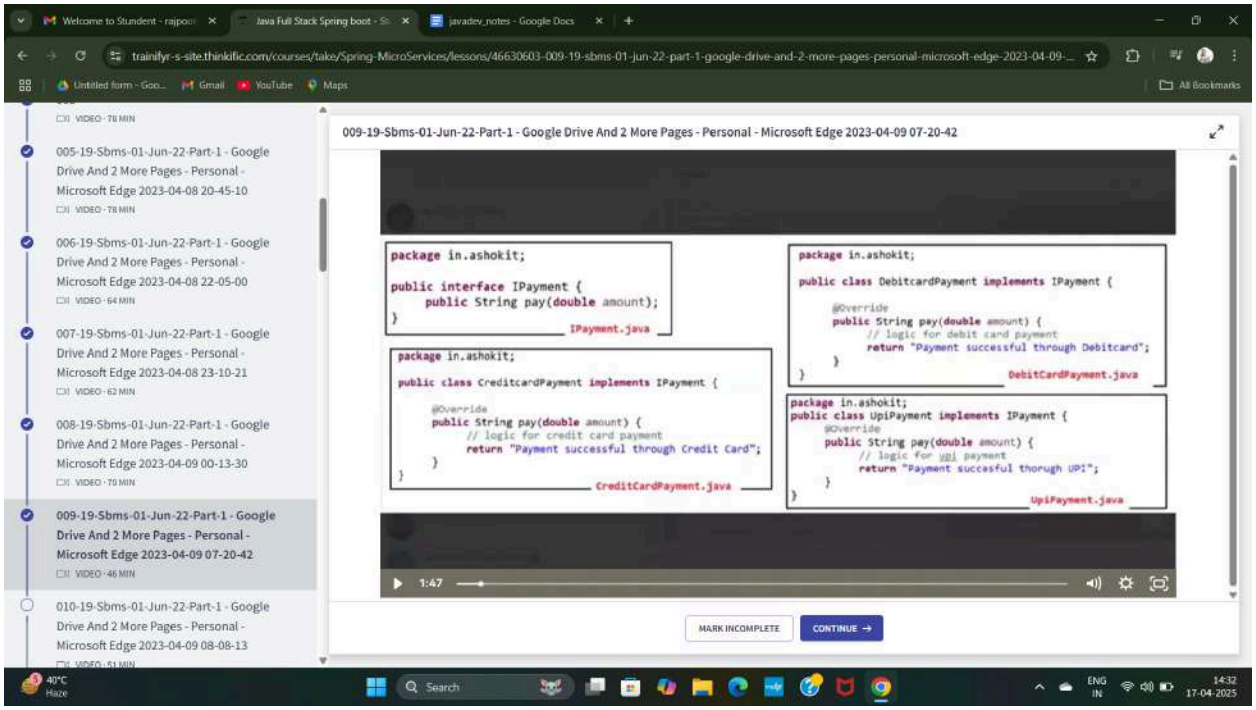


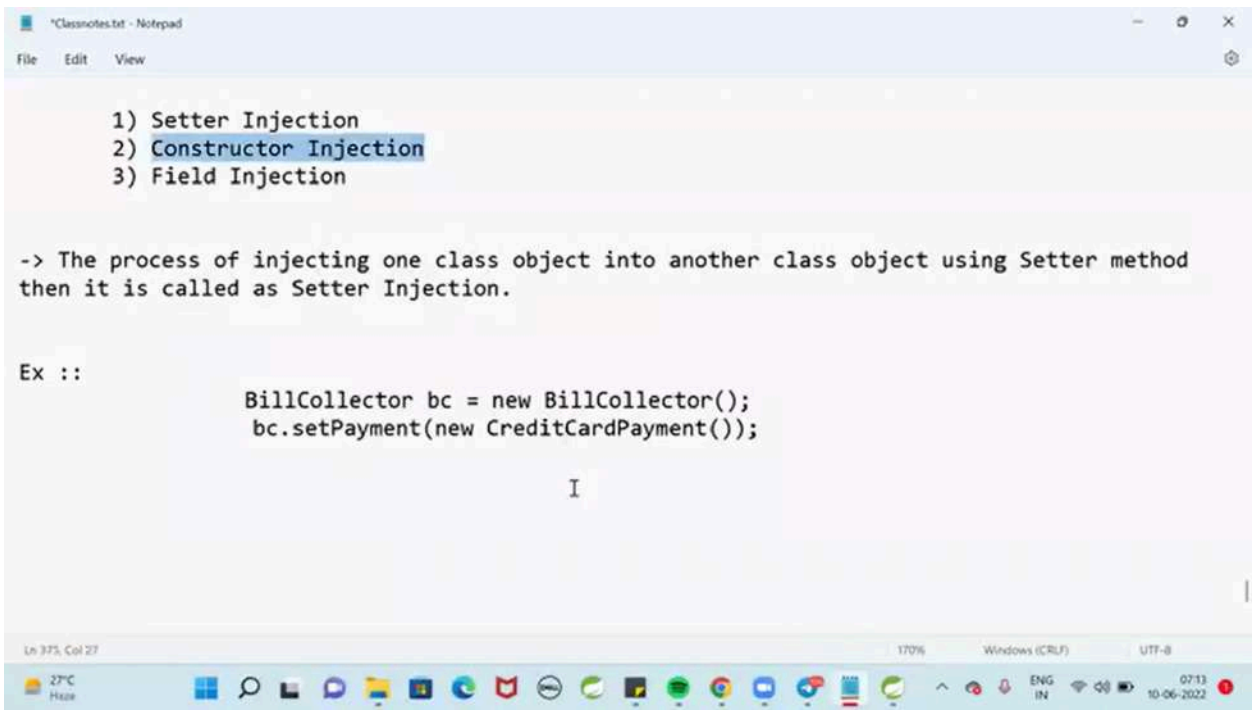



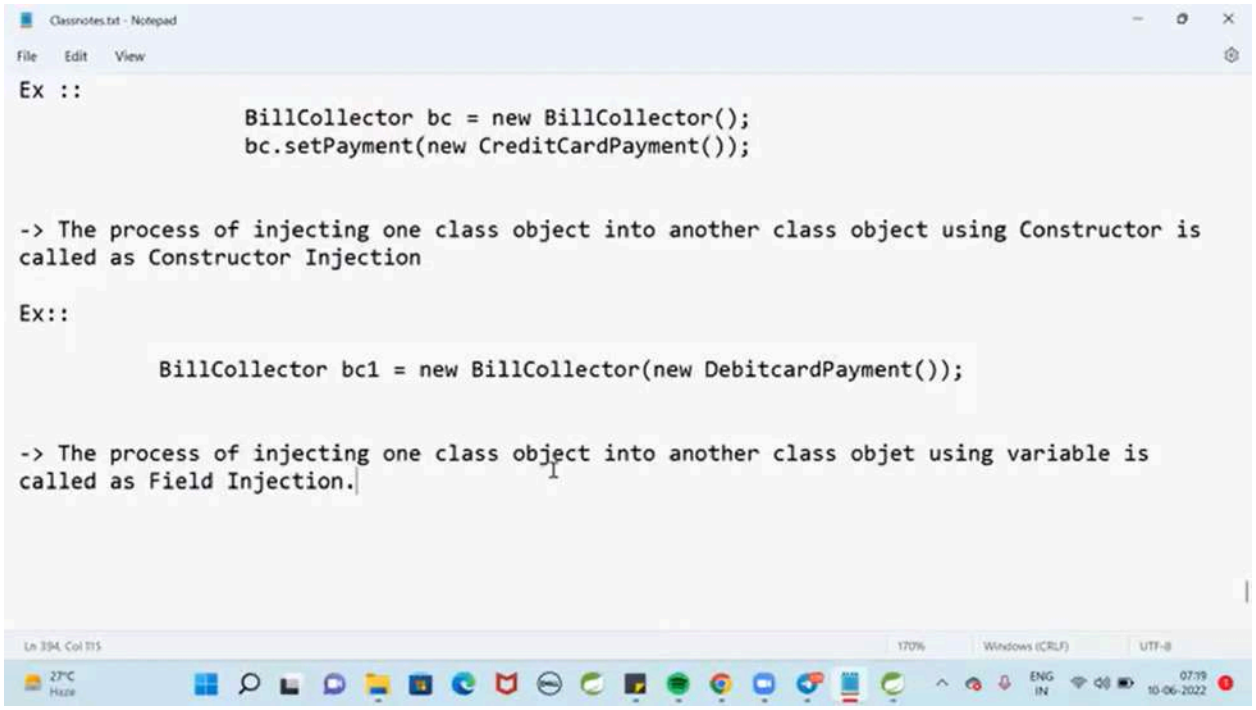

Here we have Ipayment interface and 4 their implemented classes have pay method each .

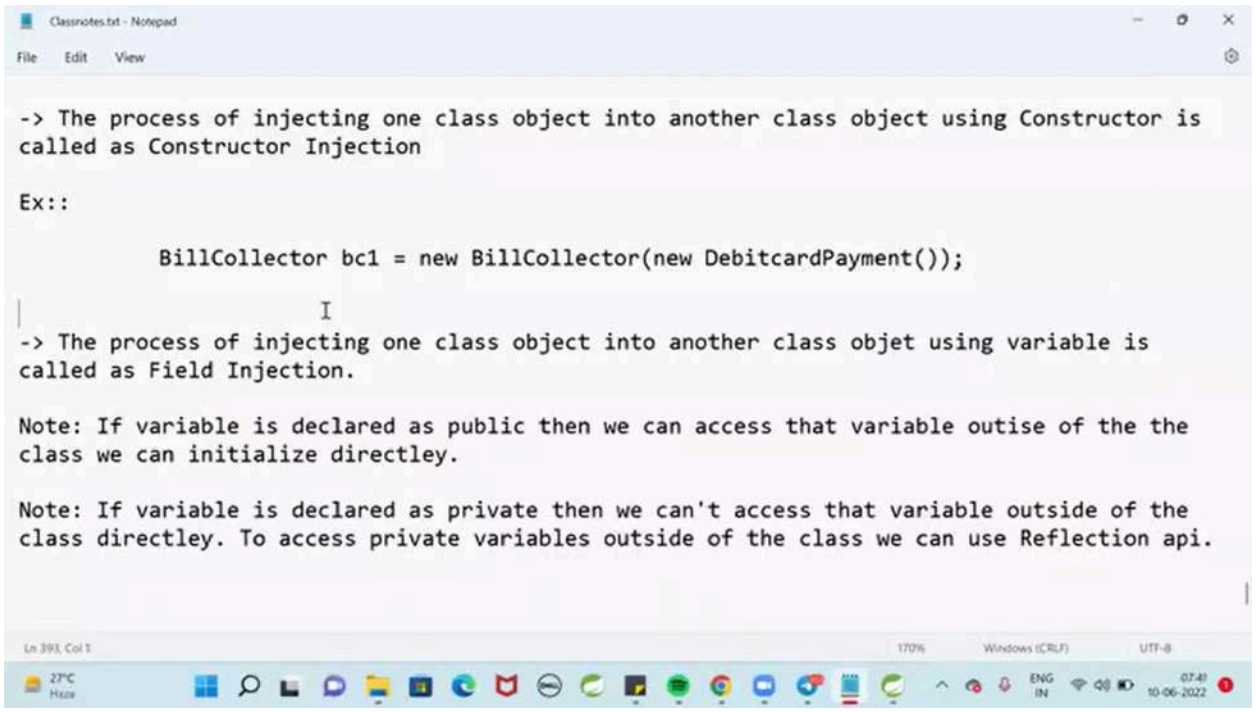

Welcome to Stundent - rajpoo
Java Full Stack Spring boot - Sb
javadev_notes - Google Docs
trainifyr-s-site.thinkific.com/courses/take/Spring-MicroServices/lessons/46630612-008-19-sbms-01-jun-22-part-1-google-drive-and-2-more-pages-personal-microsoft-edge-2023-04-09-...
Untitled form - Goo...
Gmail
YouTube
Maps
All Bookmarks
VIDEO · 23 MIN
005
VIDEO · 78 MIN
005-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-08 20-45-10
VIDEO · 78 MIN
006-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-08 22-05-00
VIDEO · 64 MIN
007-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-08 23-10-21
VIDEO · 62 MIN
008-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-09 00-13-30
VIDEO · 70 MIN
009-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-09 07-20-42
VIDEO · 46 MIN
010-19-Sbms-01-Jun-22-Part-1 - Google
008-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-09 00-13-30
BillCollector.java
Test.java
1 package in.ashokit;
2
3 public class Test {
4
5 public static void main(String[] args) {
6
7 BillCollector bc = new BillCollector();
8 // injecting CreditCard obj into BillCollector (Setter Injection)
9 bc.setPayment(new CreditcardPayment());
10 bc.collectPayment(1400.00);
11
12 // injecting CreditCard obj into BillCollector (Constructor Injection)
13 BillCollector bc1 = new BillCollector(new DebitcardPayment());
14 bc1.collectPayment(1500.00);
15 }
16 }
17
COMPLETE & CONTINUE →
40°C
Haze
Search
ENG
IN
14:26
17-04-2025

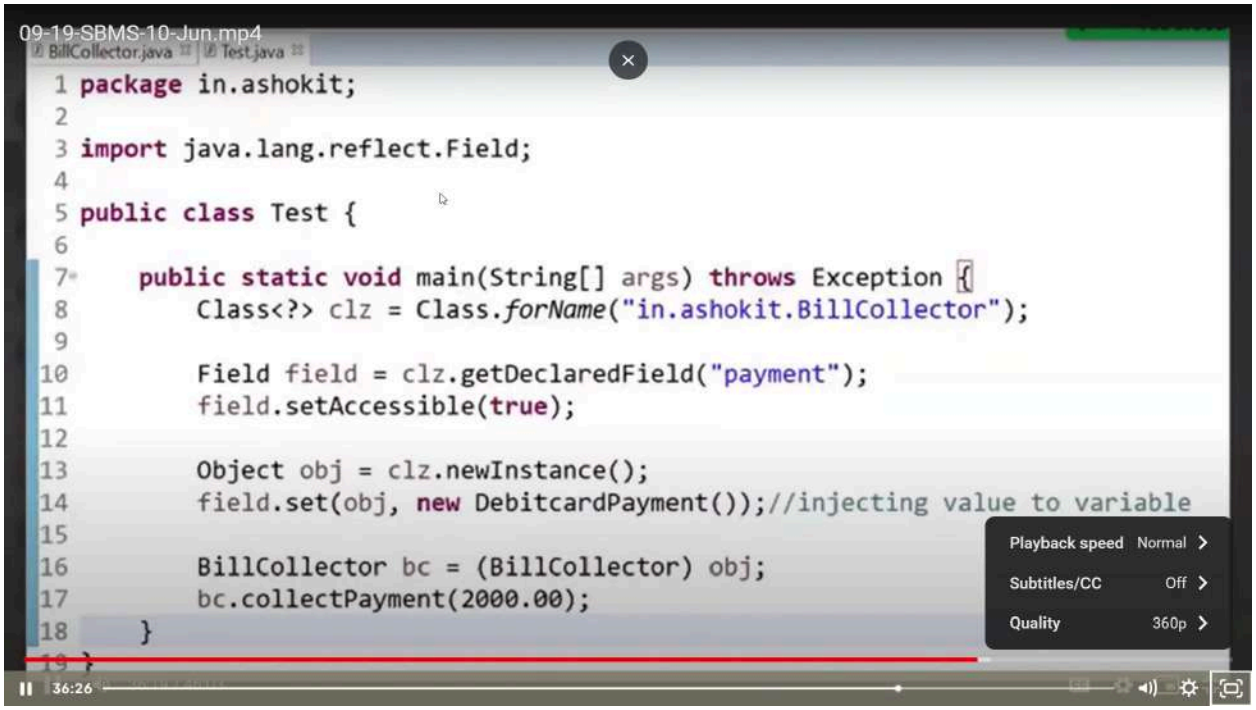

Welcome to Stundent - rajpoo
Java Full Stack Spring boot - Sb
javadev_notes - Google Docs
trainifyr-s-site.thinkific.com/courses/take/Spring-MicroServices/lessons/46630612-008-19-sbms-01-jun-22-part-1-google-drive-and-2-more-pages-personal-microsoft-edge-2023-04-09-...
Untitled form - Goo...
Gmail
YouTube
Maps
All Bookmarks
VIDEO · 23 MIN
005
VIDEO · 78 MIN
005-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-08 20-45-10
VIDEO · 78 MIN
006-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-08 22-05-00
VIDEO · 64 MIN
007-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-08 23-10-21
VIDEO · 62 MIN
008-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-09 00-13-30
VIDEO · 70 MIN
009-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-09 07-20-42
VIDEO · 46 MIN
010-19-Sbms-01-Jun-22-Part-1 - Google
008-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-09 00-13-30
1 package in.ashokit;
2
3 public class BillCollector {
4
5 private IPayment payment;
6
7 public void setPayment(IPayment payment) {
8 this.payment = payment;
9 }
10
11 public BillCollector() { }
12
13 public BillCollector(IPayment payment) {
14 this.payment = payment;
15 }
16
17 public void collectPayment(double amount) {
18 String status = payment.pay(amount);
19 System.out.println(status);
20 }
21 }
53:25
COMPLETE & CONTINUE →
40°C
Haze
Search
ENG
IN
14:27
17-04-2025


Java Full Stack Spring boot
javadev_notes - Google Docs
005-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-08 20-45-10
006-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-08 22-05-00
007-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-08 23-10-21
008-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-09 00-13-30
009-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-09 07-20-42
010-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-09 08-08-13
009-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-09 07-20-42
package in.ashokit;
public interface IPayment {
public String pay(double amount);
}
IPayment.java
package in.ashokit;
public class DebitcardPayment implements IPayment {
@Override
public String pay(double amount) {
// logic for debit card payment
return "Payment successful through Debitcard";
}
}
DebitCardPayment.java
package in.ashokit;
public class CreditcardPayment implements IPayment {
@Override
public String pay(double amount) {
// logic for credit card payment
return "Payment successful through Credit Card";
}
}
CreditCardPayment.java
package in.ashokit;
public class UpiPayment implements IPayment {
@Override
public String pay(double amount) {
// logic for upi payment
return "Payment succesful thorugh UPI";
}
}
UpiPayment.java
1:47
MARK INCOMPLETE
CONTINUE →

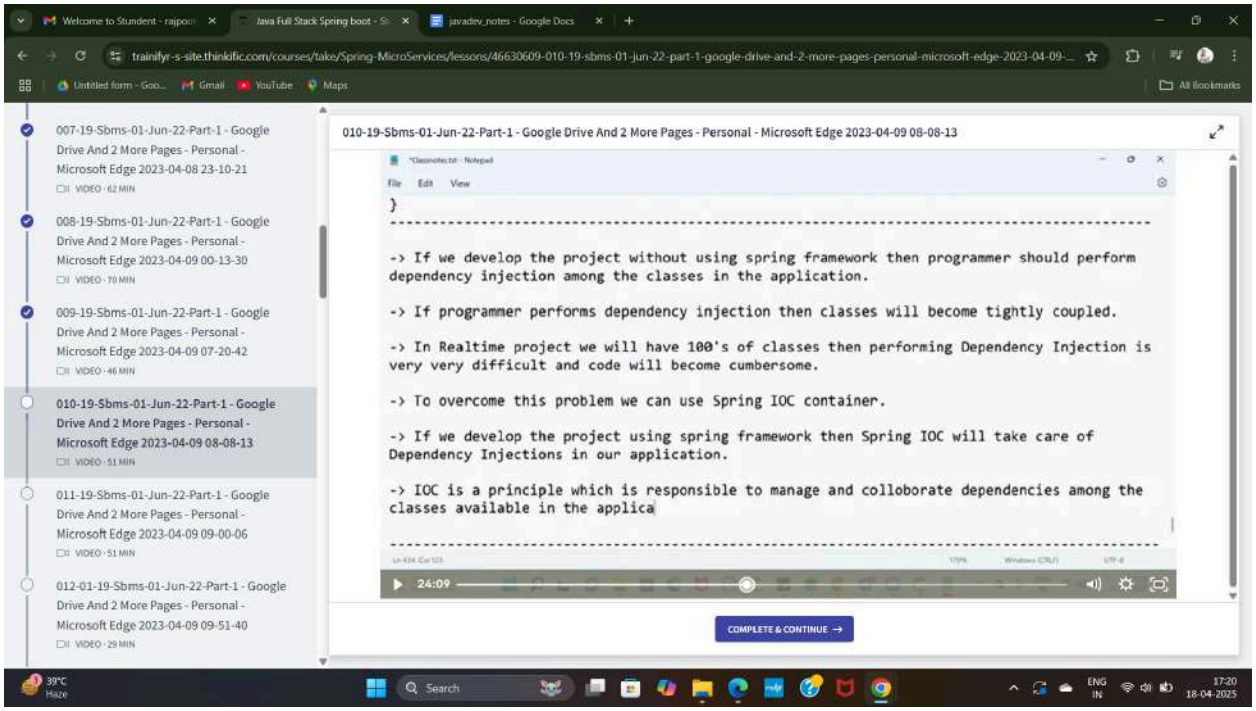

*Classnotes.txt - Notepad
File Edit View
1) Setter Injection
2) Constructor Injection
3) Field Injection
-> The process of injecting one class object into another class object using Setter method then it is called as Setter Injection.
Ex ::
BillCollector bc = new BillCollector();
bc.setPayment(new CreditCardPayment());
Ln 375, Col 27
170%
Windows (CRLF)
UTF-8


Classnotes.txt - Notepad
File Edit View
Ex ::
BillCollector bc = new BillCollector();
bc.setPayment(new CreditCardPayment());
-> The process of injecting one class object into another class object using Constructor is called as Constructor Injection
Ex::
BillCollector bc1 = new BillCollector(new DebitcardPayment());
-> The process of injecting one class object into another class objet using variable is called as Field Injection.


Classnotes.txt - Notepad
File Edit View
-> The process of injecting one class object into another class object using Constructor is called as Constructor Injection
Ex::
BillCollector bc1 = new BillCollector(new DebitcardPayment());
-> The process of injecting one class object into another class objet using variable is called as Field Injection.
Note: If variable is declared as public then we can access that variable outise of the the class we can initialize directley.
Note: If variable is declared as private then we can't access that variable outside of the class directley. To access private variables outside of the class we can use Reflection api.




I did’t understand feild injection properly yet (reminder need to work on this)



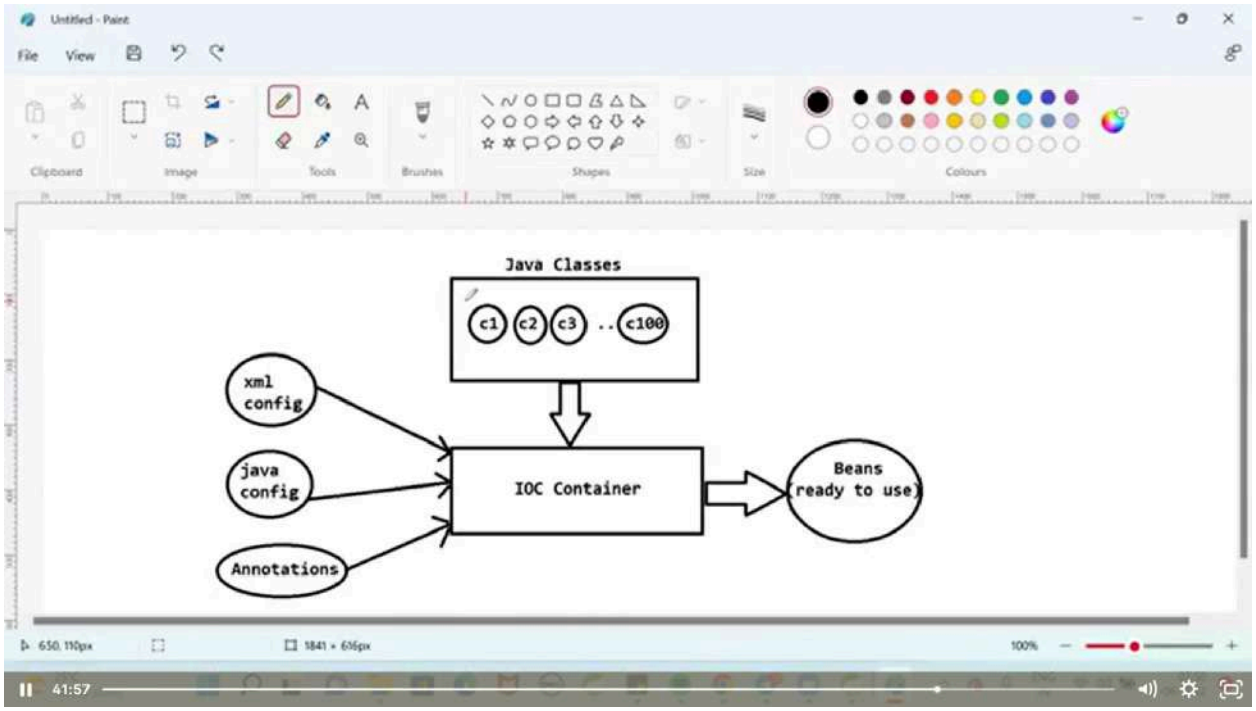

-> IOC will maintain which class object inject in which class

-> or whether through setter injection or construction injection although constructor injection will recommended.

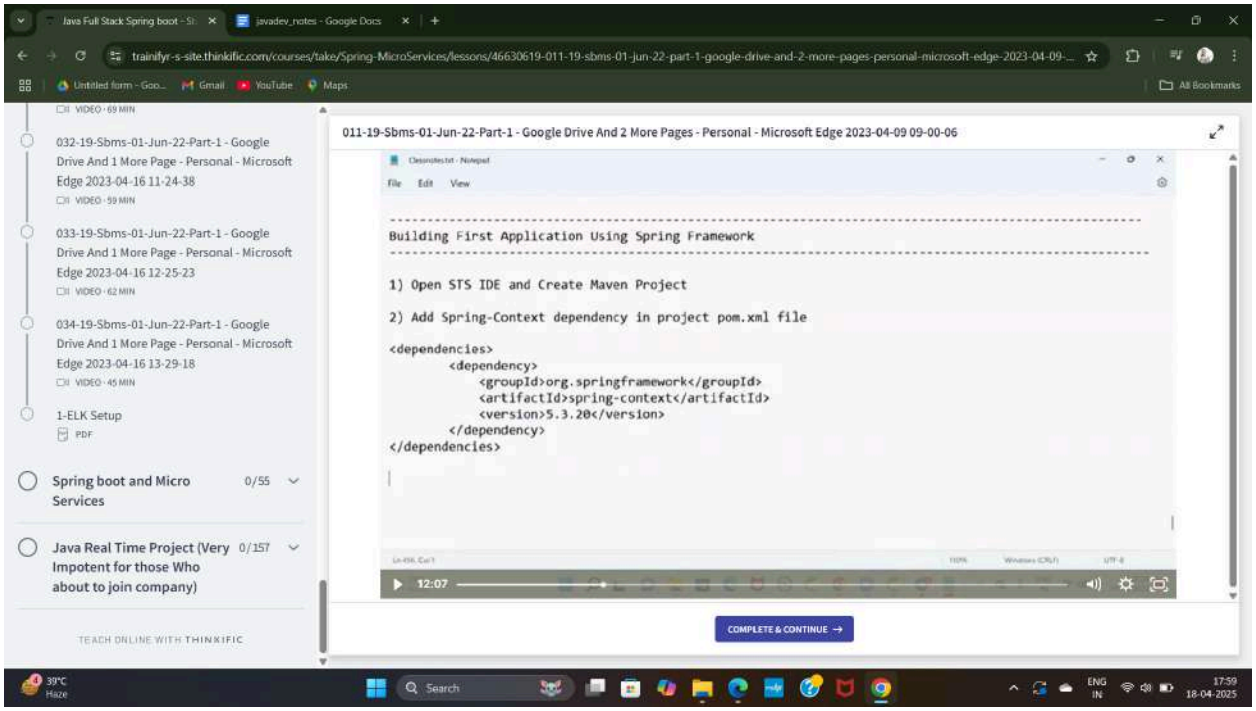
-> as IOC container does not know about which one is dependent class and which one is target class so we need to provide instruction to the IOC container in three ways

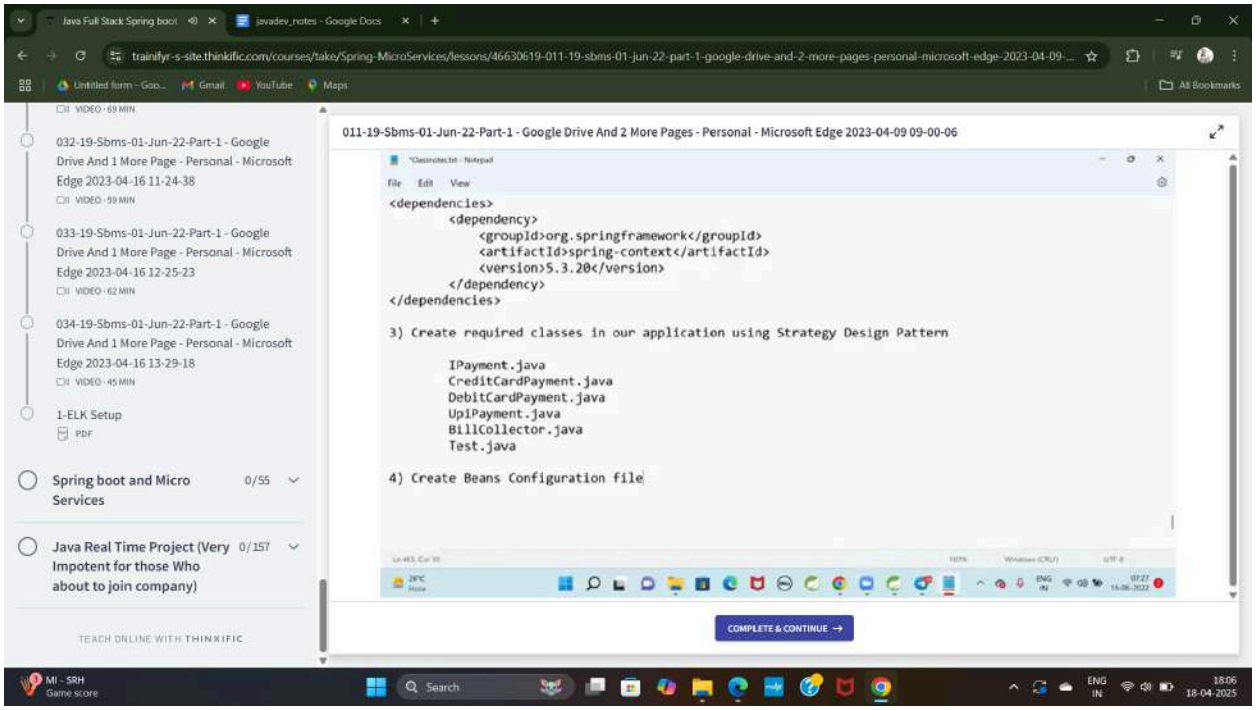
1. Xml config

2.java config
3.annotations.
-> beans we get as an output from IOC container

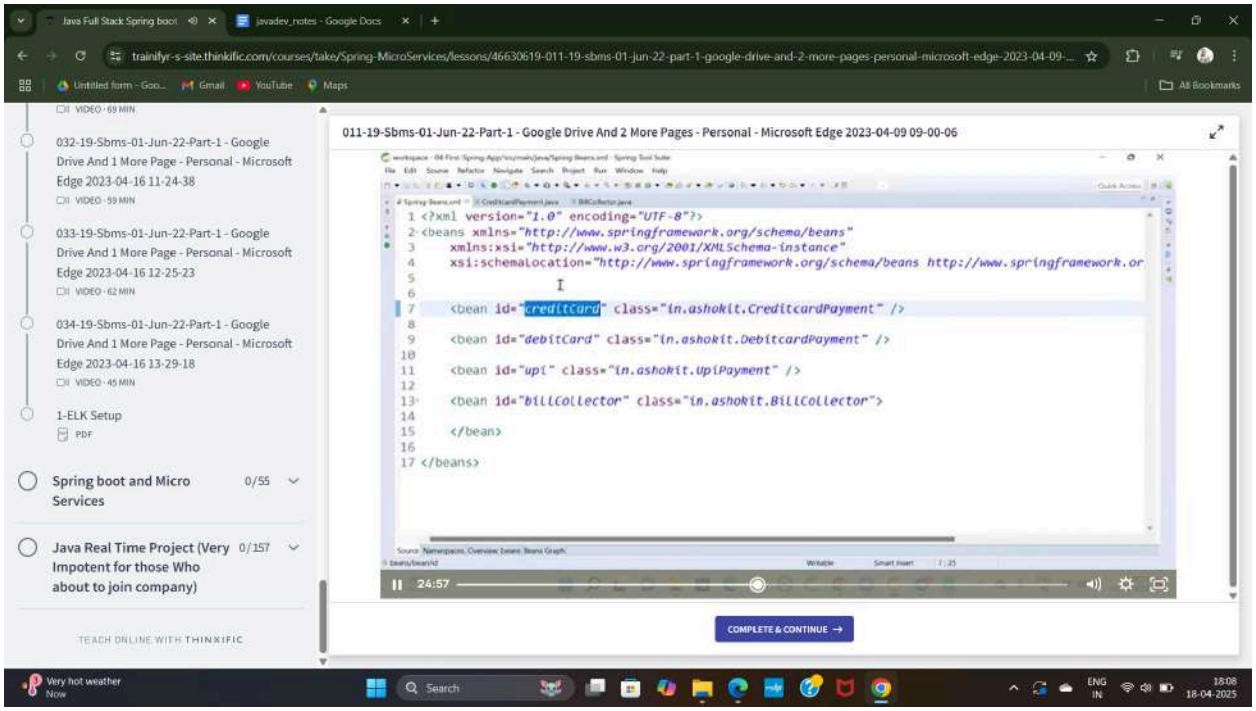



So all of the above we implement it was normal java base application there is no spring here so now i will use spring framework for my next phase .



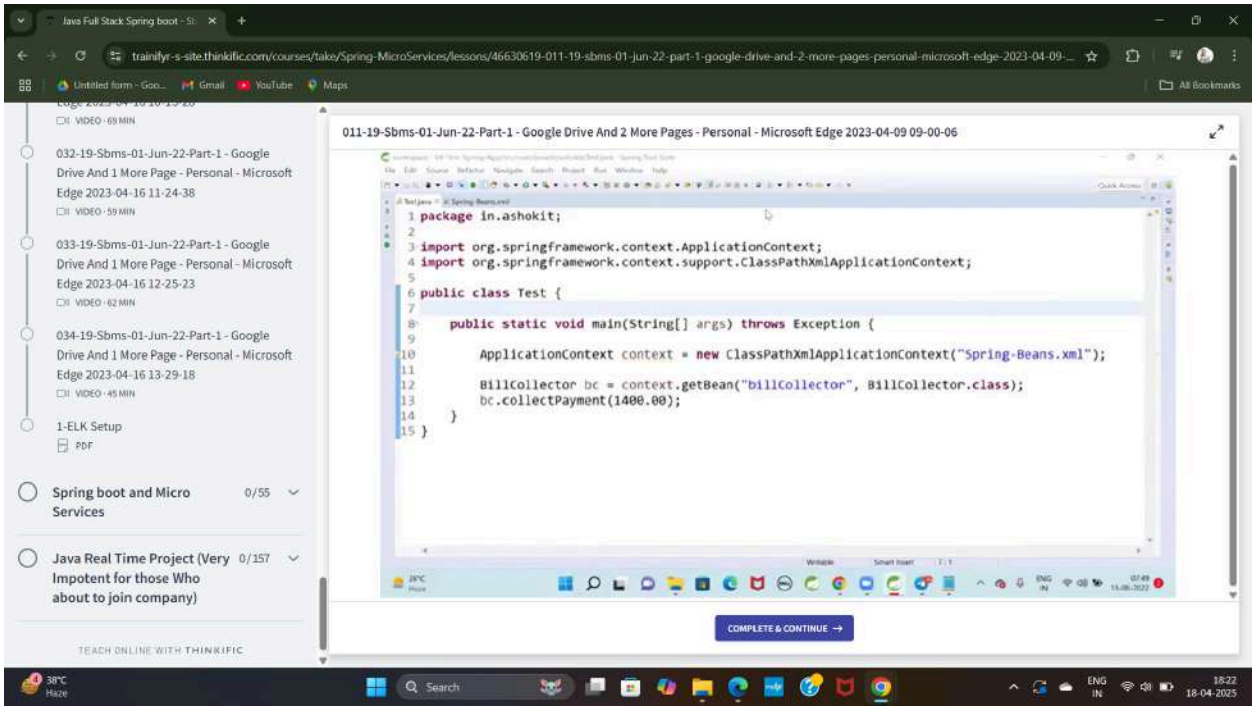


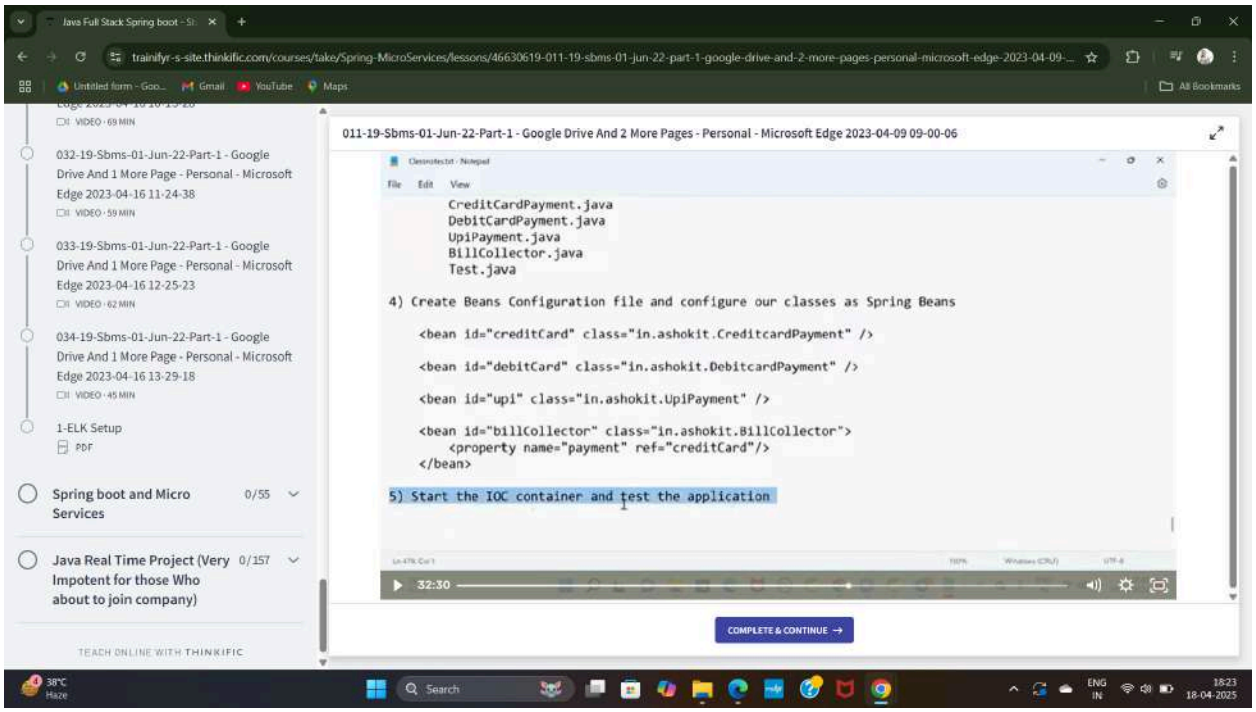

We will configure the clases as spring bean in XML file



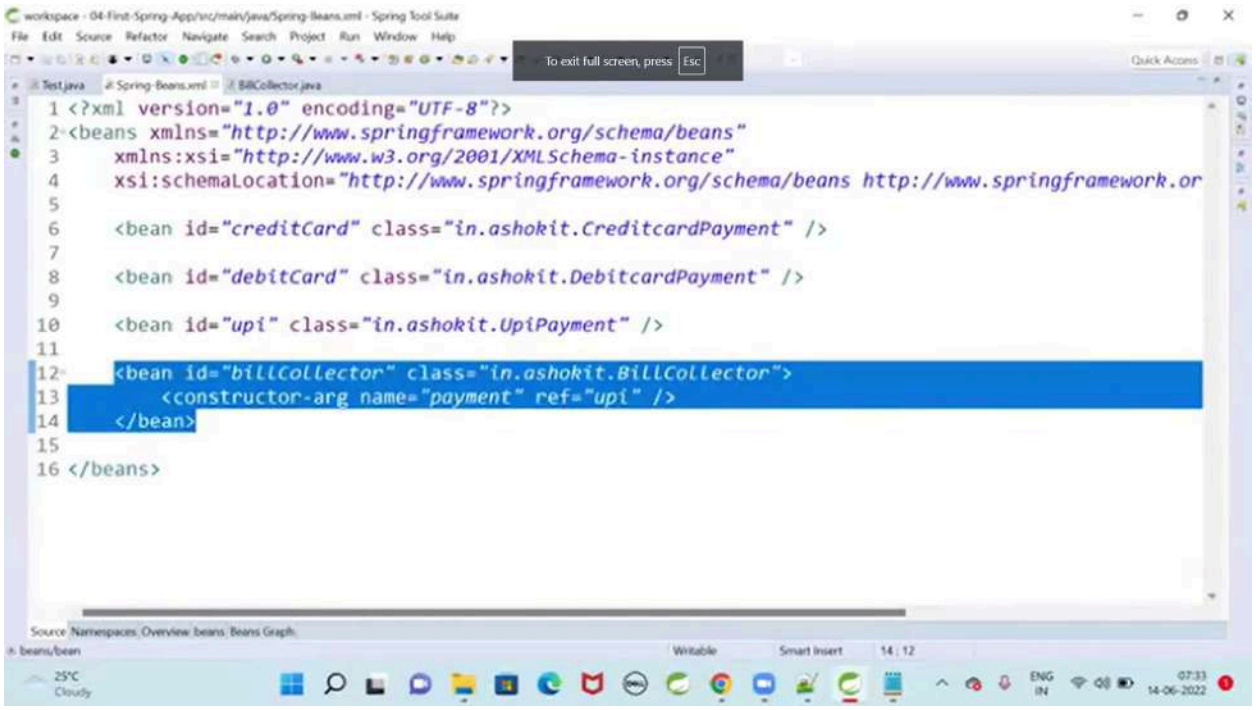

Property tag represent setter injection , ref indicate which class object will be inject for given variable

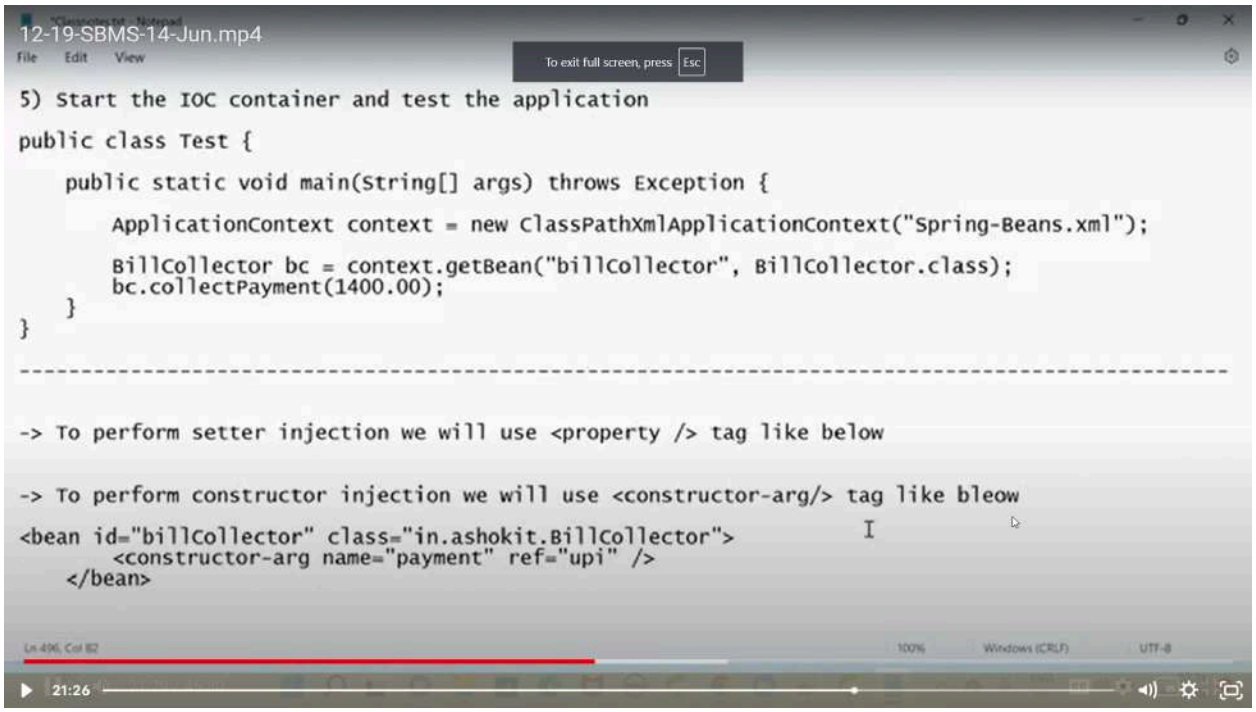
-> classpathXmlApplicationContect read the xml file and start our IOC container for DI



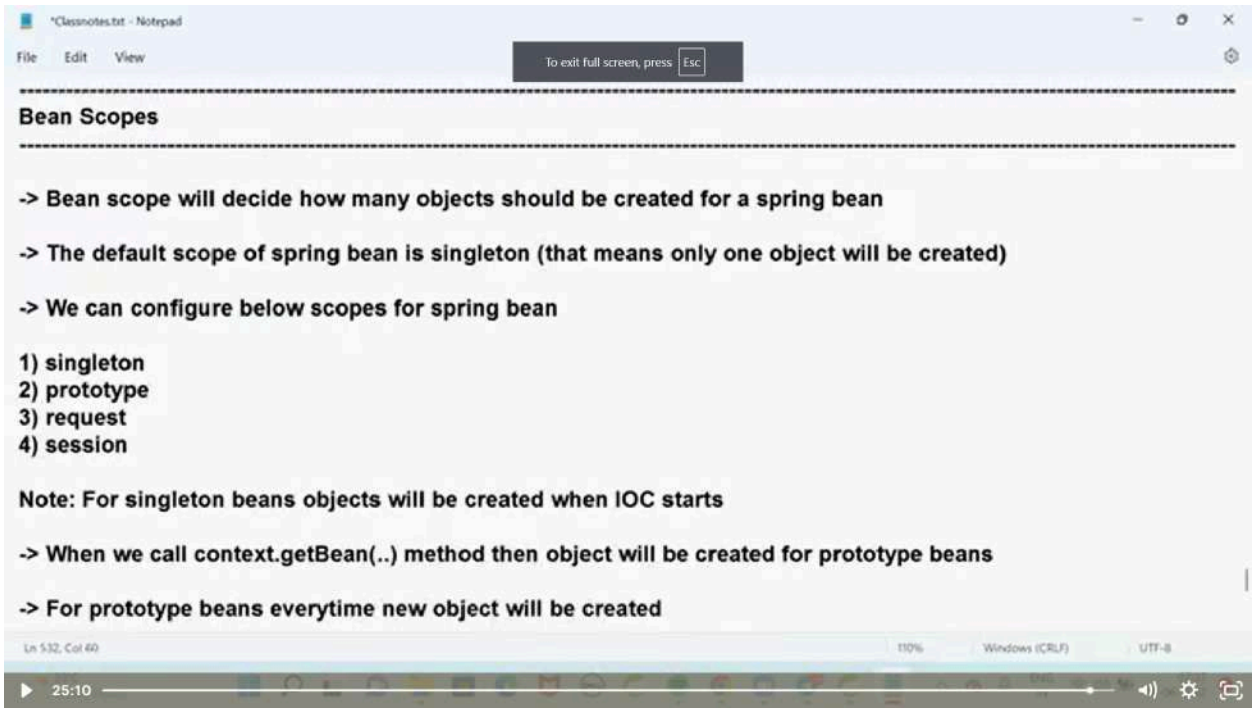



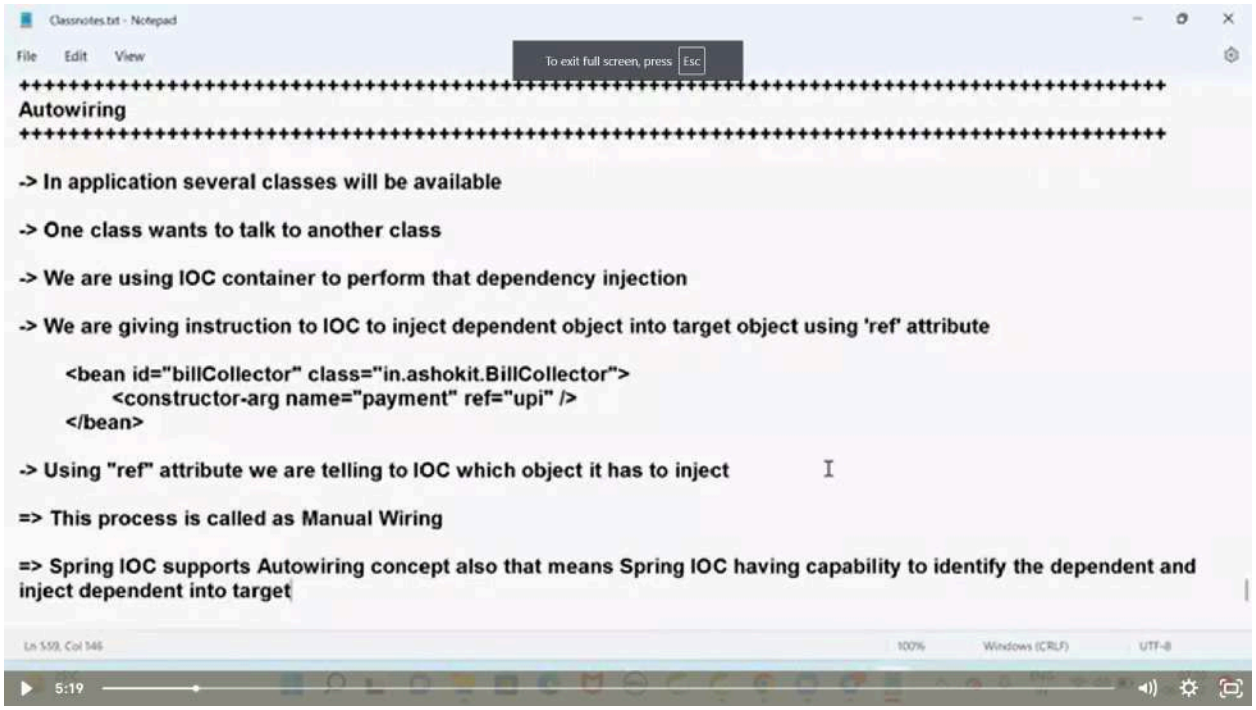

So we applicationbContect here to start the IOC container and one can use bean factory also


workspace - 04-First-Spring-App/src/main/java/Spring-Beans.xml - Spring Tool Suite
File Edit Source Refactor Navigate Search Project Run Window Help
To exit full screen, press Esc
Test.java
Spring-Beans.xml
BillCollector.java
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.or
5
6 <bean id="creditCard" class="in.ashokit.CreditcardPayment" />
7
8 <bean id="debitCard" class="in.ashokit.DebitcardPayment" />
9
10 <bean id="upi" class="in.ashokit.UpiPayment" />
11
12 <bean id="billCollector" class="in.ashokit.BillCollector">
13 <constructor-arg name="payment" ref="upi" />
14 </bean>
15
16 </beans>
Source Namespaces Overview beans Beans Graph
beans/bean
Writable
Smart Insert
14 : 12
25°C
Cloudy
ENG
IN
07:33
14-06-2022

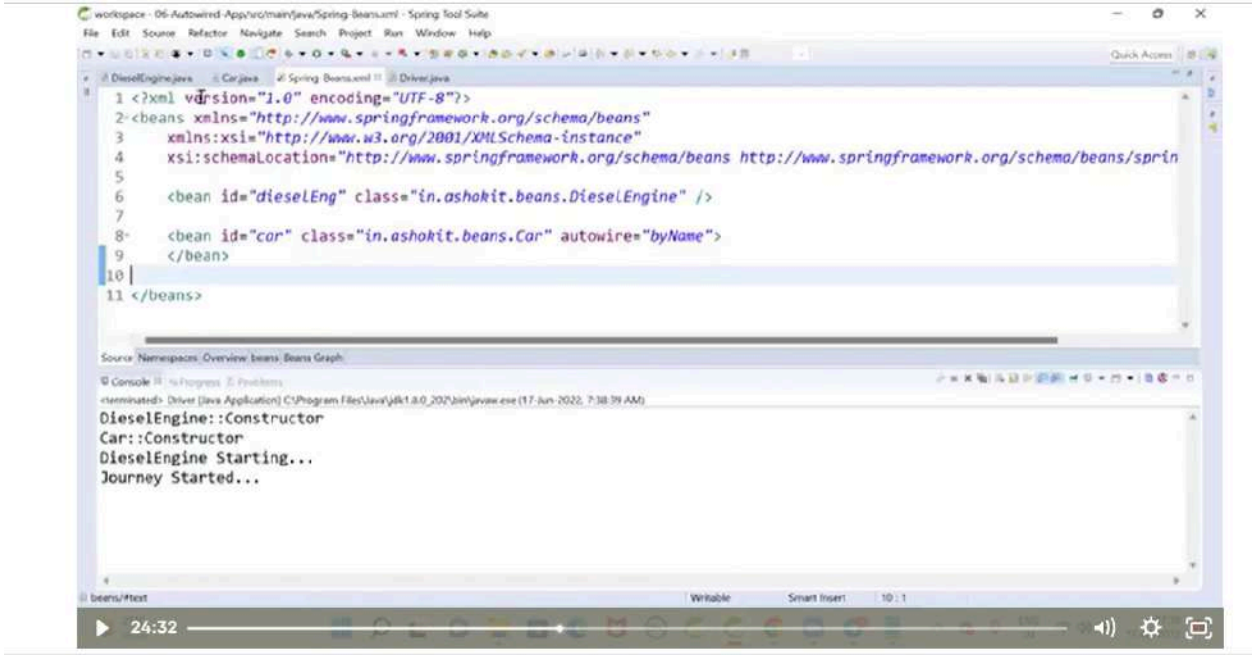

12-19-SBMS-14-Jun.mp4
File Edit View
To exit full screen, press Esc
5) Start the IOC container and test the application
public class Test {
public static void main(String[] args) throws Exception {
ApplicationContext context = new ClassPathXmlApplicationContext("Spring-Beans.xml");
BillCollector bc = context.getBean("billCollector", BillCollector.class);
bc.collectPayment(1400.00);
}
}
-> To perform setter injection we will use <property /> tag like below
-> To perform constructor injection we will use <constructor-arg/> tag like bleow
<bean id="billCollector" class="in.ashokit.BillCollector">
<constructor-arg name="payment" ref="upi" />
</bean>
Ln 496, Col 82
100%
Windows (CRLF)
UTF-8
21:26

```
-> To perform constructor injection we will use <constructor-arg/> tag like bleow

<bean id="billCollector" class="in.ashokit.BillCollector">
        <constructor-arg name="payment" ref="upi" />
</bean>

-> When we perform both setter and constructor injection for same variable then setter injection
will override constructor injection because construcor will execute first to initialize the
variable then setter will execute and it will re-initialize the variable.

<bean id="billCollector" class="in.ashokit.BillCollector">
        <property name="payment" ref="creditCard" />
        <constructor-arg name="payment" ref="upi" />
</bean>
```

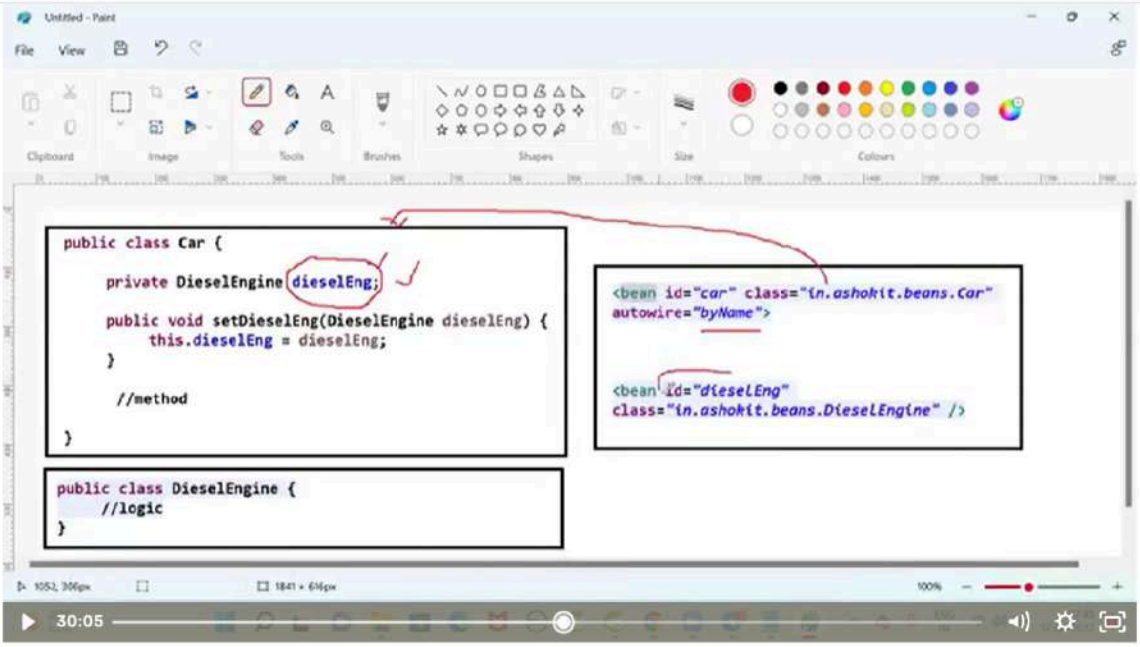





014-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-09 18-39-23

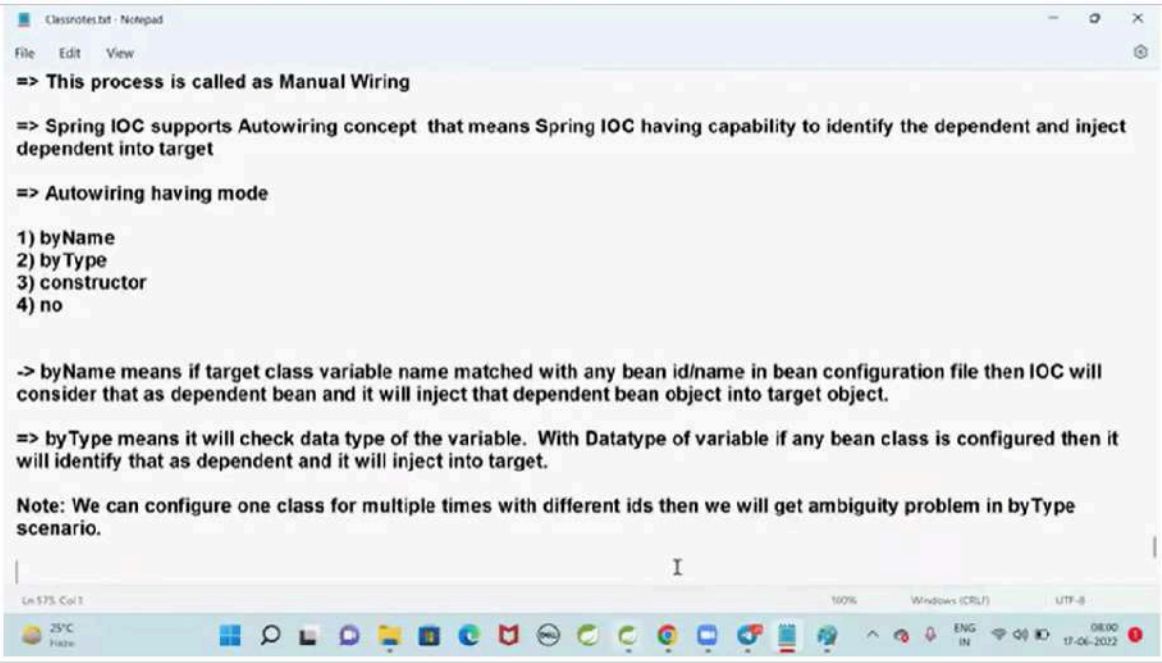


014-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-09 18-39-23

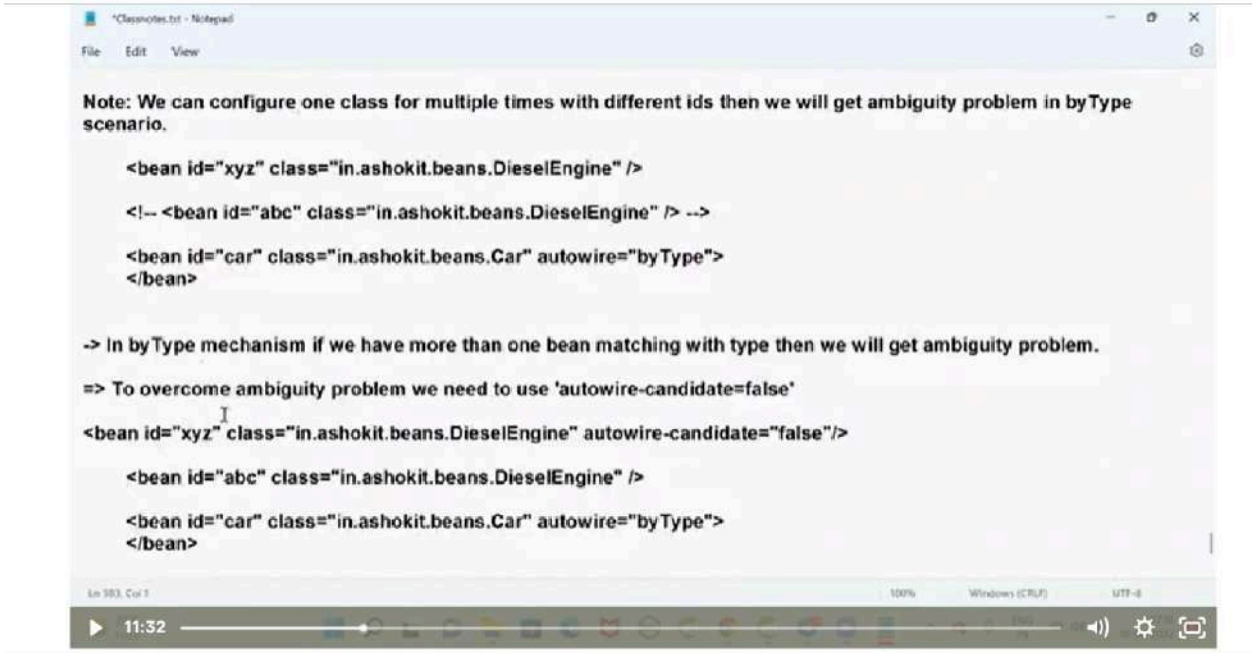



So here there is no need of manual DI it will be done using autowire so where we use autowire it will check in that class ..is that any reference variable if yes then check in xml file of same reference id name if find then it will inject automatically so no need of property tag or constructor arg .

014-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-09 18-39-23



015-19-Sbms-01-Jun-22-Part-1 - Google Drive And 2 More Pages - Personal - Microsoft Edge 2023-04-09 19-42-00




```
<bean id="xyz" class="in.ashokit.beans.DieselEngine" autowire-candidate="false"/>

<bean id="abc" class="in.ashokit.beans.DieselEngine" />

<bean id="car" class="in.ashokit.beans.Car" autowire="byType">
</bean>
```

Note: When we configure autowiring with "byName" or "byType" it is performing setter injection by default and setter method is mandatory in target bean.

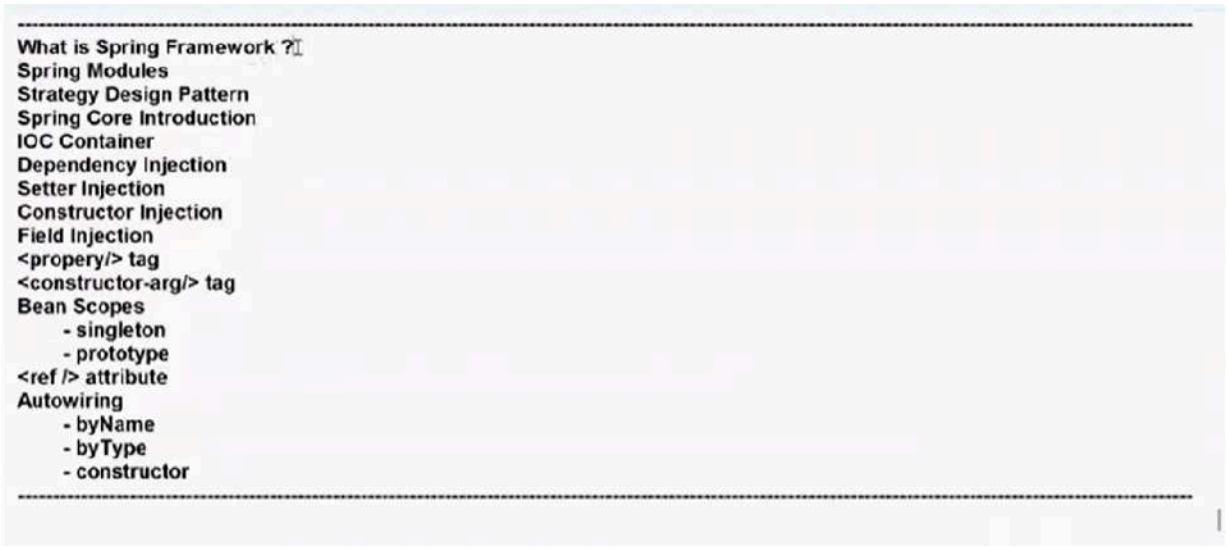
=> If we want to perfom Autowiring through constructor then we can use 'constructor' mode

```
<bean id="xyz" class="in.ashokit.beans.DieselEngine" autowire-candidate="false"/>

<bean id="abc" class="in.ashokit.beans.DieselEngine" />

<bean id="car" class="in.ashokit.beans.Car" autowire="constructor"/>
```

=> In 'constructor' byType will be used to identify dependent bean object.

---

By default autowiring happen to through setter methode




So these are the things we learned so far in spring core now will start spring boot .