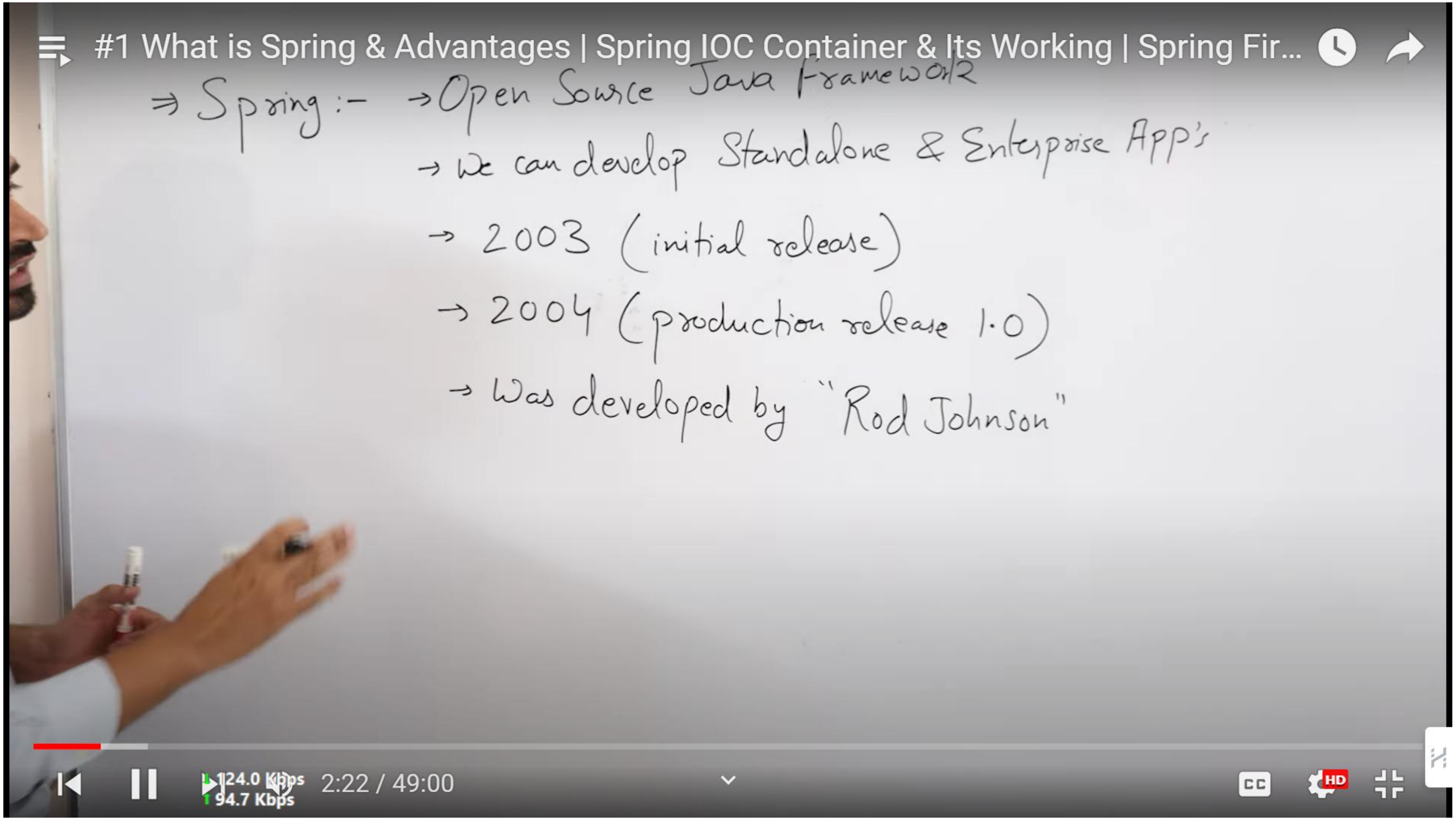


#1 What is Spring & Advantages | Spring IOC Container & Its Working | Spring Fir...

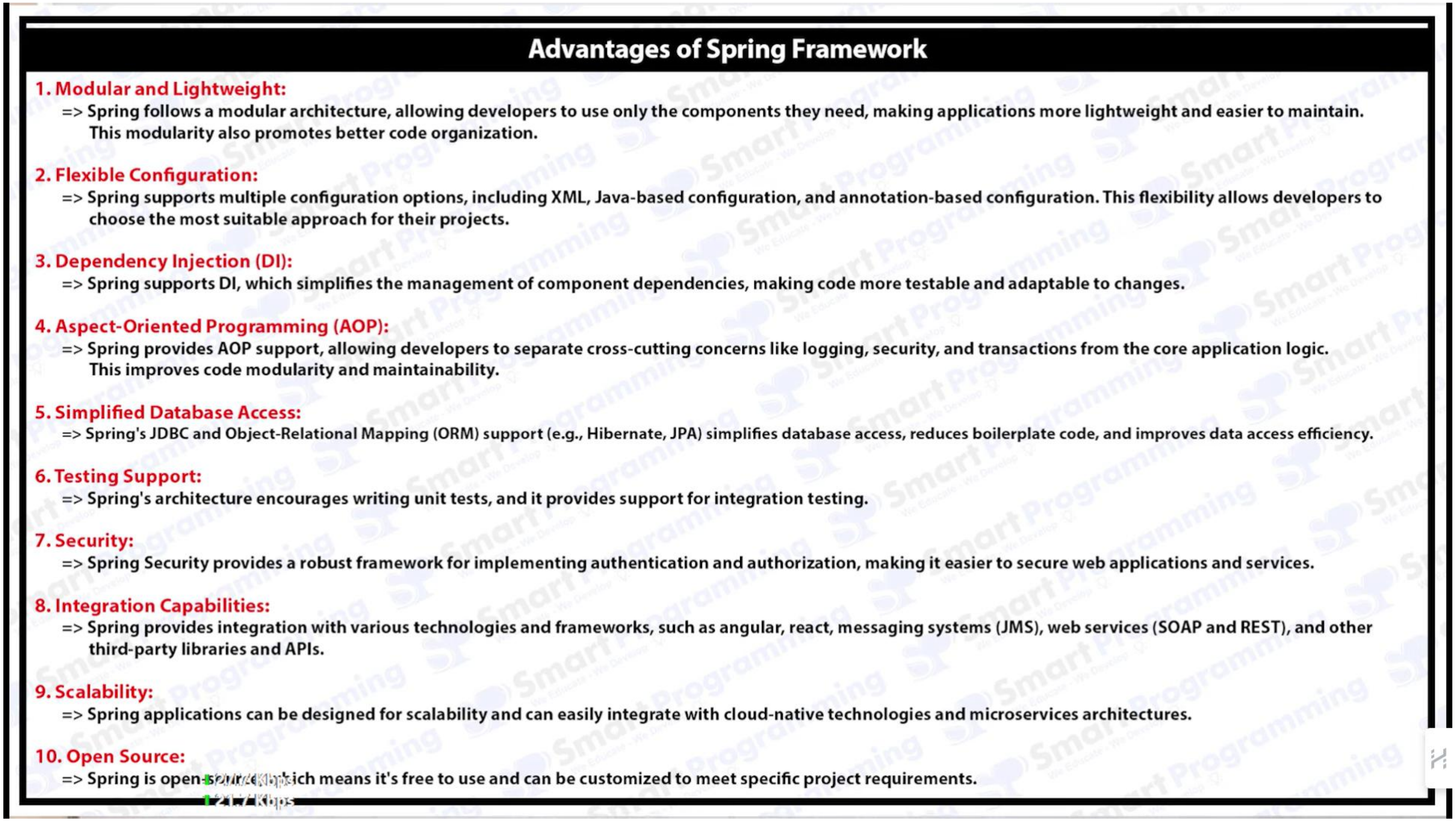
⇒ Spring :- → Open Source Java Framework

→ We can develop Standalone & Enterprise App's

→ 2003 (initial release)

→ 2004 (production release 1.0)

→ Was developed by "Rod Johnson"



Advantages of Spring Framework

1. Modular and Lightweight:

=> Spring follows a modular architecture, allowing developers to use only the components they need, making applications more lightweight and easier to maintain. This modularity also promotes better code organization.

2. Flexible Configuration:

=> Spring supports multiple configuration options, including XML, Java-based configuration, and annotation-based configuration. This flexibility allows developers to choose the most suitable approach for their projects.

3. Dependency Injection (DI):

=> Spring supports DI, which simplifies the management of component dependencies, making code more testable and adaptable to changes.

4. Aspect-Oriented Programming (AOP):

=> Spring provides AOP support, allowing developers to separate cross-cutting concerns like logging, security, and transactions from the core application logic. This improves code modularity and maintainability.

5. Simplified Database Access:

=> Spring's JDBC and Object-Relational Mapping (ORM) support (e.g., Hibernate, JPA) simplifies database access, reduces boilerplate code, and improves data access efficiency.

6. Testing Support:

=> Spring's architecture encourages writing unit tests, and it provides support for integration testing.

7. Security:

=> Spring Security provides a robust framework for implementing authentication and authorization, making it easier to secure web applications and services.

8. Integration Capabilities:

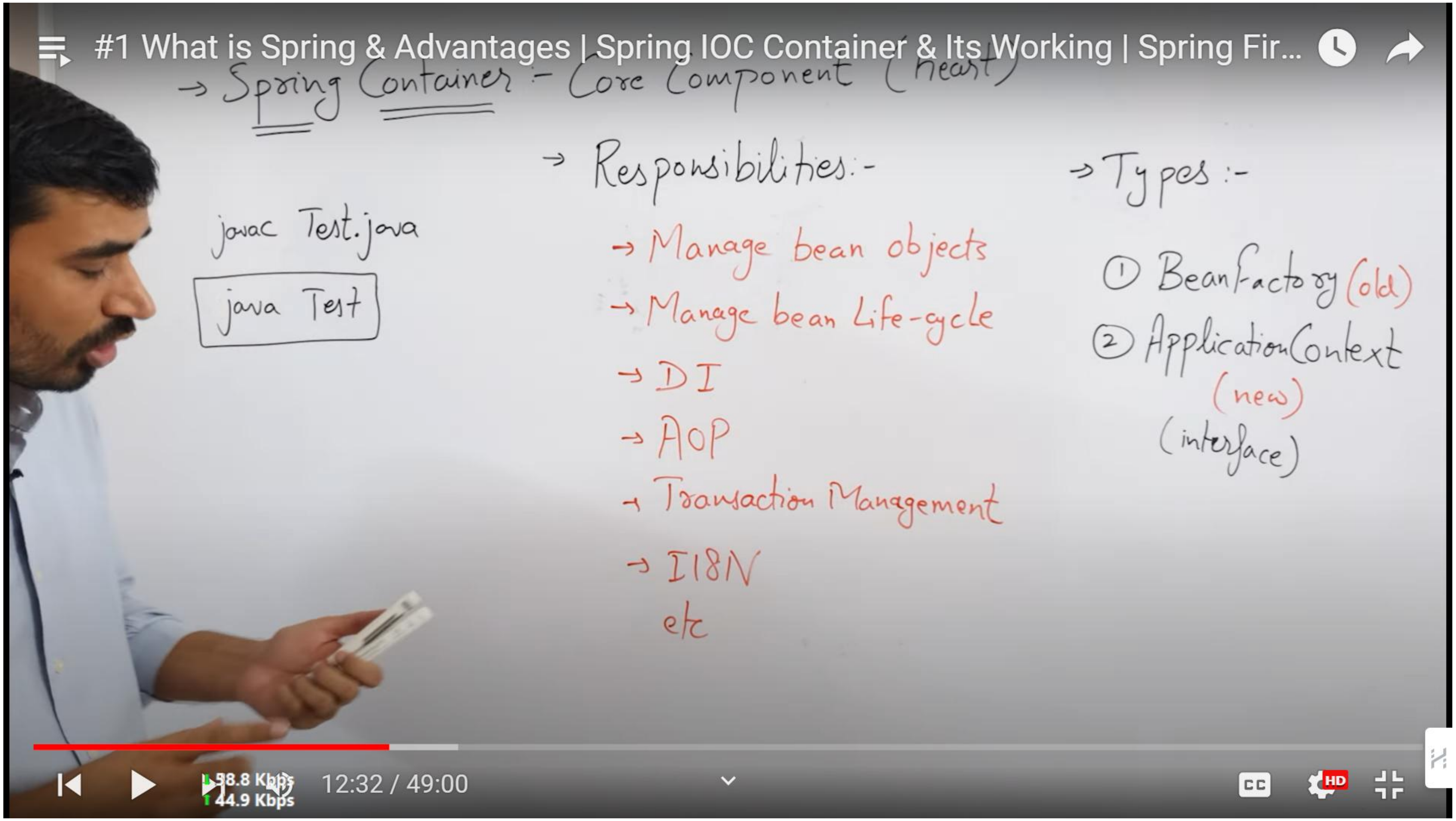
=> Spring provides integration with various technologies and frameworks, such as angular, react, messaging systems (JMS), web services (SOAP and REST), and other third-party libraries and APIs.

9. Scalability:

=> Spring applications can be designed for scalability and can easily integrate with cloud-native technologies and microservices architectures.

10. Open Source:

=> Spring is open-source, which means it's free to use and can be customized to meet specific project requirements.



#1 What is Spring & Advantages | Spring IOC Container & Its Working | Spring Fir...

→ Spring Container - Core Component (heart)

javac Test.java

java Test

→ Responsibilities:-

→ Manage bean objects

→ Manage bean Life-cycle

→ DI

→ AOP

→ Transaction Management

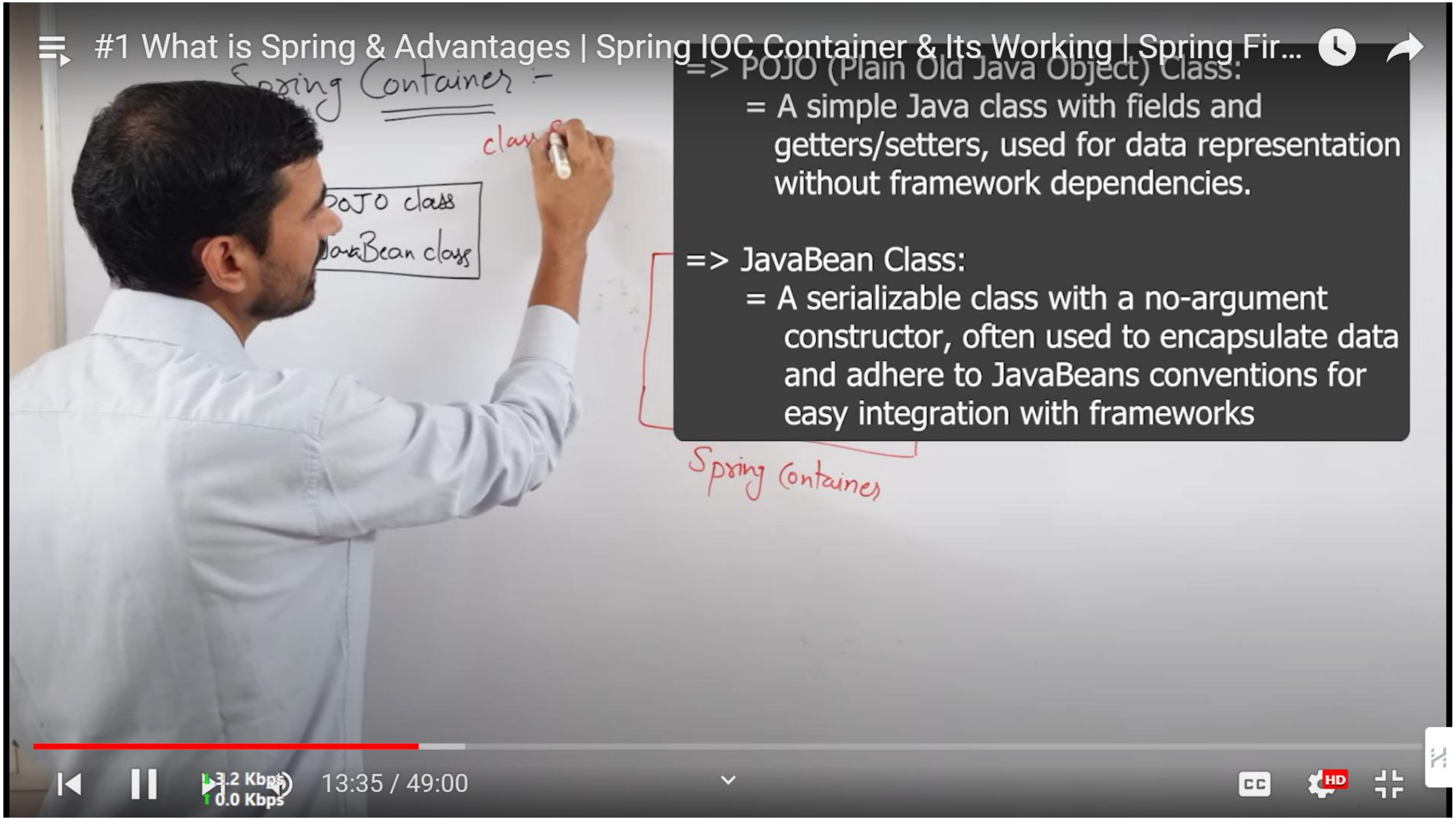
→ I18N

etc

→ Types :-

① BeanFactory (old)

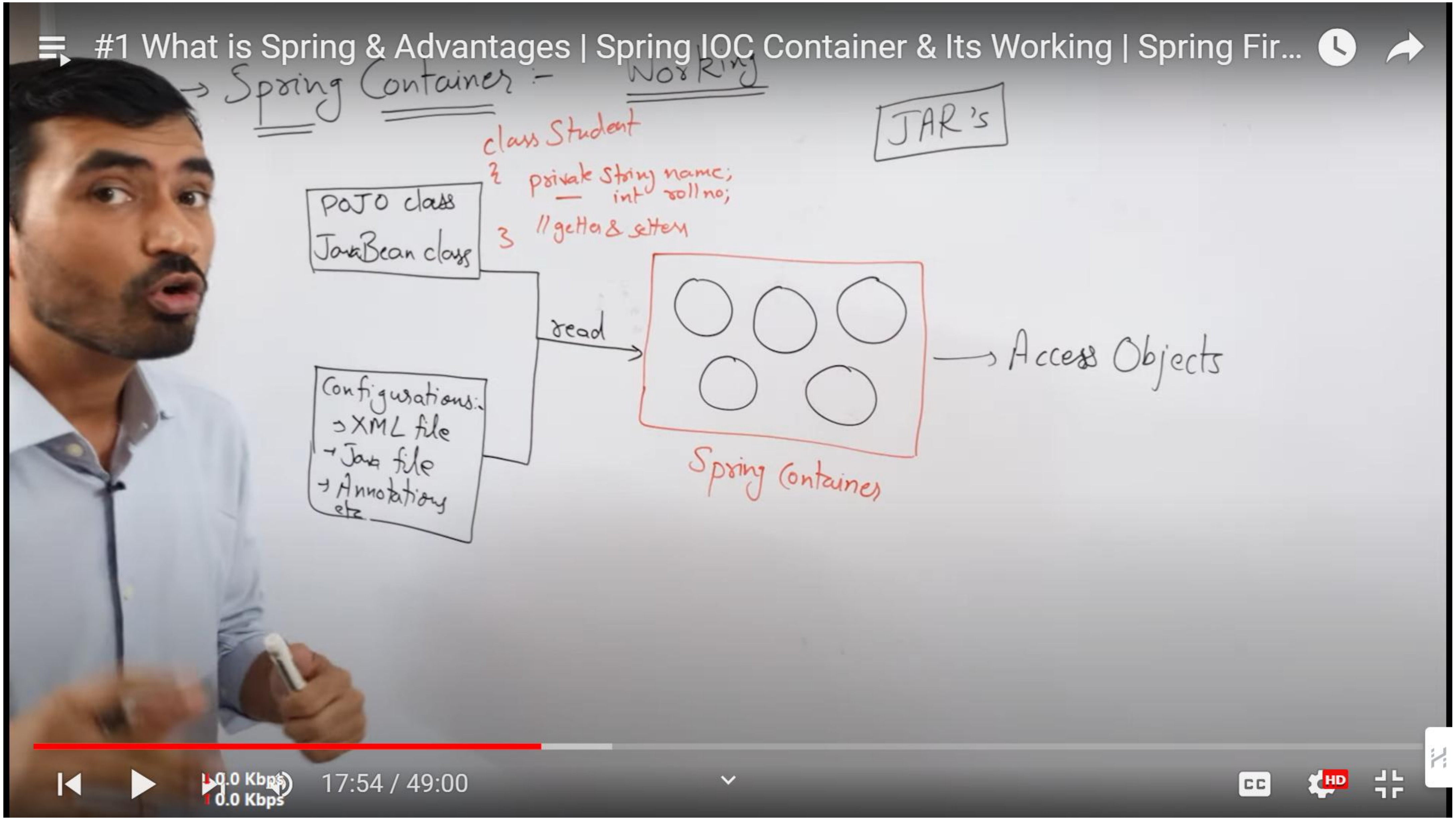
② ApplicationContext (new) (interface)

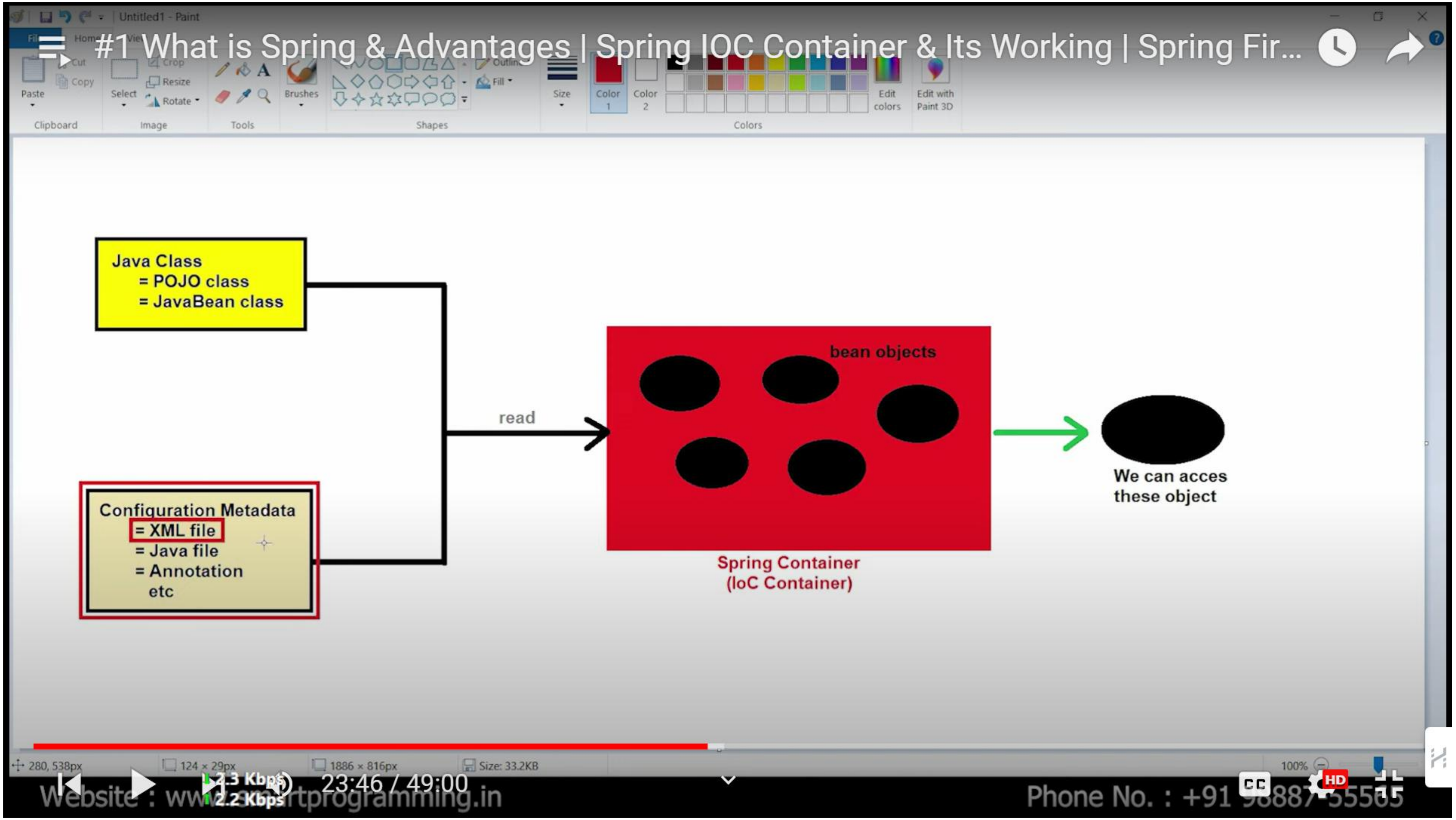


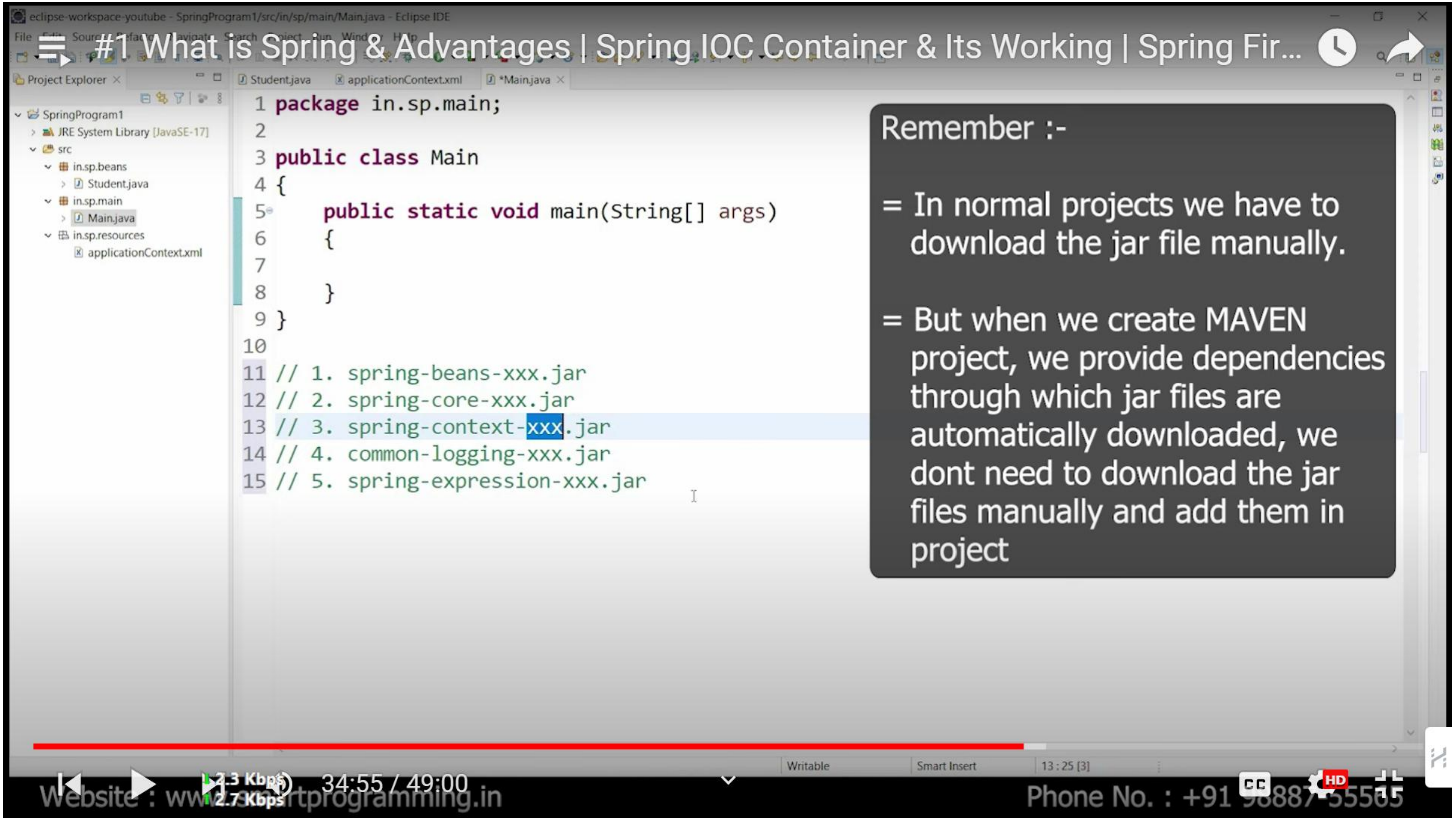
#1 What is Spring & Advantages | Spring IOC Container & Its Working | Spring Fir...

=> POJO (Plain Old Java Object) Class:
= A simple Java class with fields and getters/setters, used for data representation without framework dependencies.

=> JavaBean Class:
= A serializable class with a no-argument constructor, often used to encapsulate data and adhere to JavaBeans conventions for easy integration with frameworks


#1 What is Spring & Advantages | Spring IOC Container & Its Working | Spring Fir...
→ Spring Container :- Working
JAR's
class Student
{ private String name;
int rollno;
//getter & setter
}
POJO class
JavaBean class
Configurations:
→ XML file
→ Java file
→ Annotations etc
read
Spring Container
→ Access Objects
0.0 Kbps
0.0 Kbps
17:54 / 49:00
HD


#1 What is Spring & Advantages | Spring IOC Container & Its Working | Spring Fir...
Java Class
= POJO class
= JavaBean class
Configuration Metadata
= XML file
= Java file
= Annotation
etc
read
bean objects
Spring Container
(IoC Container)
We can acces
these object
23:46 / 49:00
Website : www.smartprogramming.in
Phone No. : +91 98887 55565

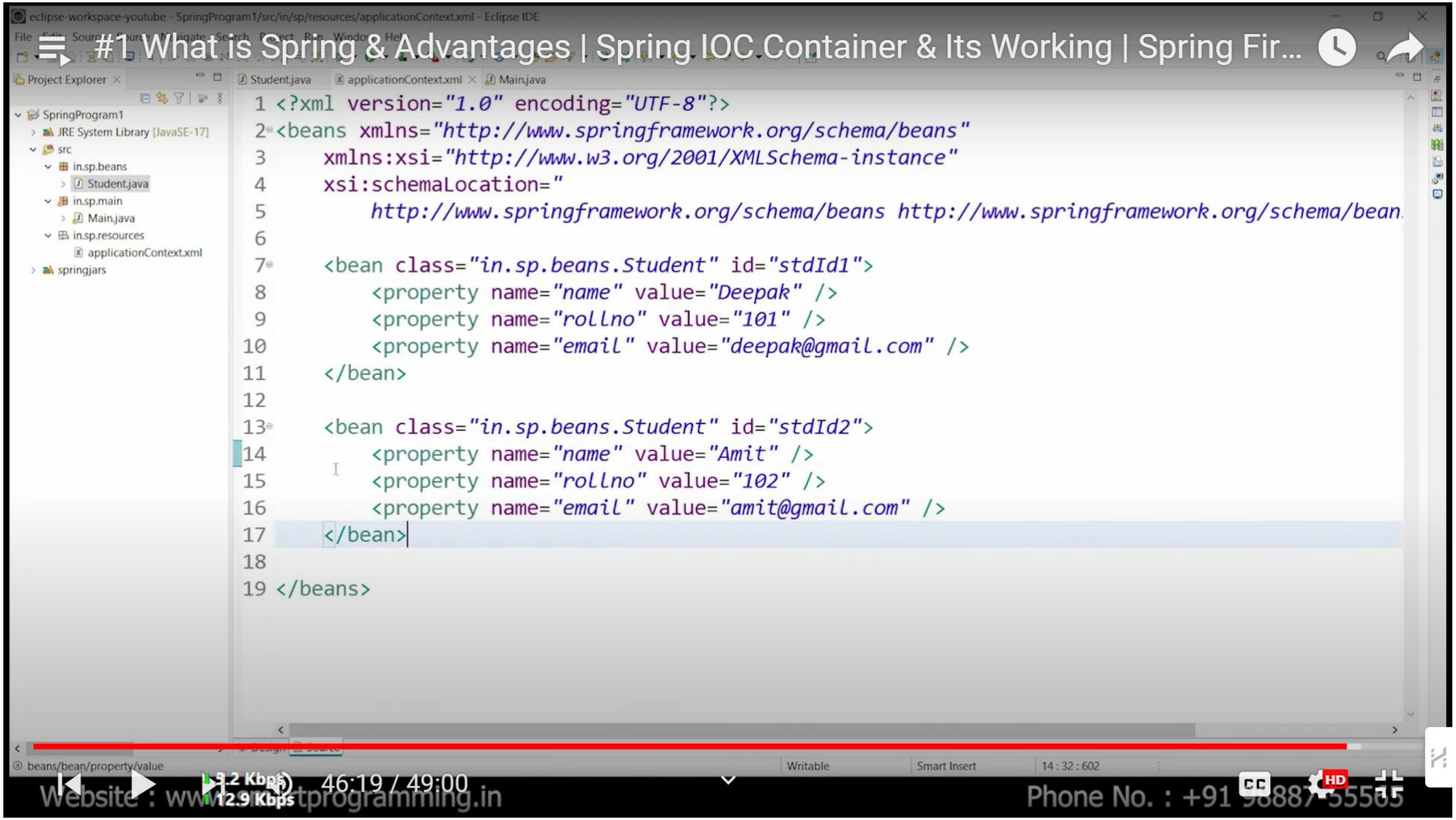

eclipse-workspace-youtube - SpringProgram1/src/in/sp/main/Main.java - Eclipse IDE
#1 What is Spring & Advantages | Spring IOC Container & Its Working | Spring Fir...
Project Explorer
SpringProgram1
JRE System Library [JavaSE-17]
src
in.sp.beans
Student.java
in.sp.main
Main.java
in.sp.resources
applicationContext.xml
Student.java
applicationContext.xml
*Main.java
Writable
Smart Insert
13 : 25 [3]
34:55 / 49:00
Website : www.smartprogramming.in
Phone No. : +91 98887 55565

```
package in.sp.main;

public class Main
{
    public static void main(String[] args)
    {

    }
}

// 1. spring-beans-xxx.jar
// 2. spring-core-xxx.jar
// 3. spring-context-xxx.jar
// 4. common-logging-xxx.jar
// 5. spring-expression-xxx.jar
```

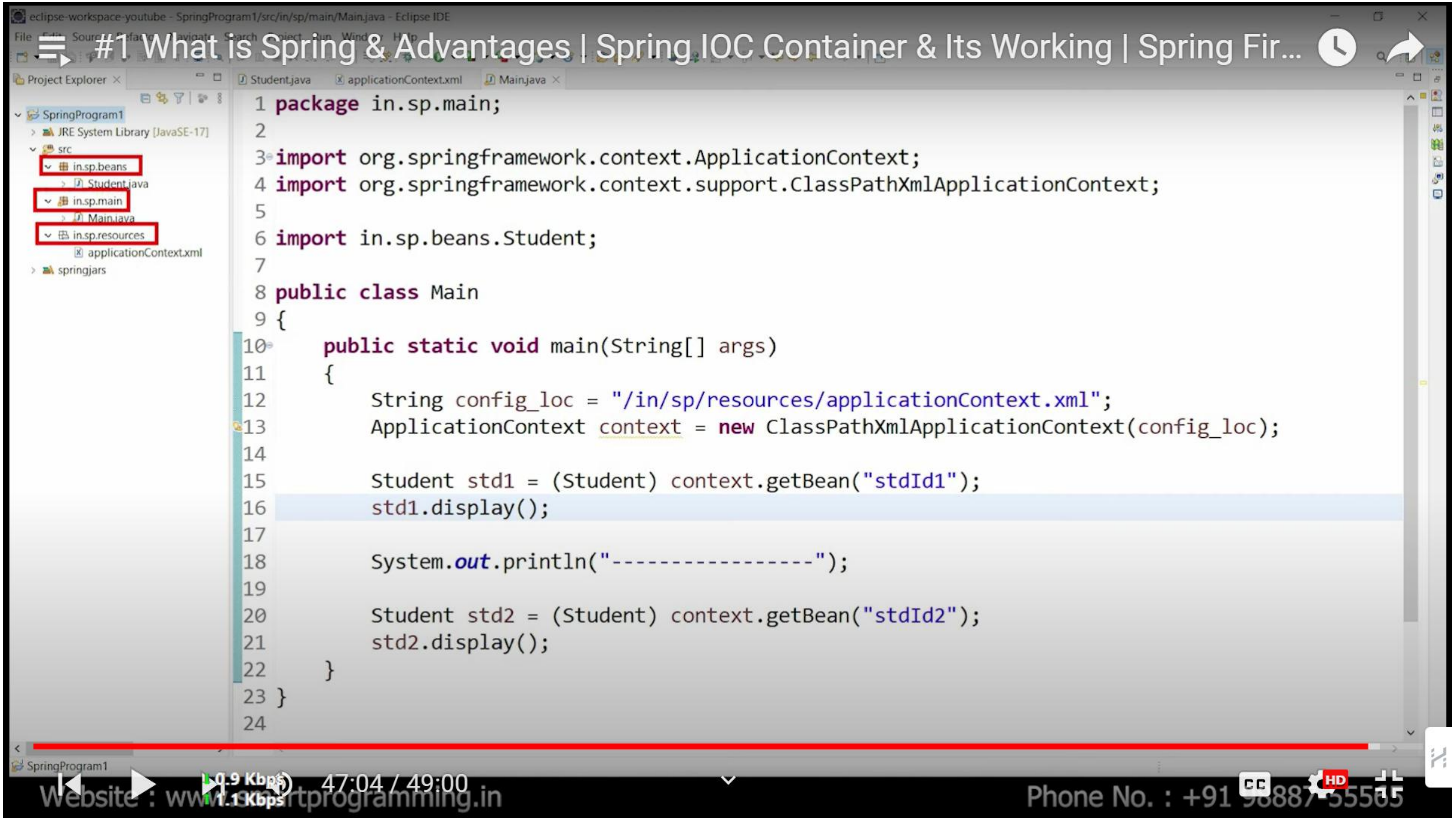
Remember :-

= In normal projects we have to download the jar file manually.

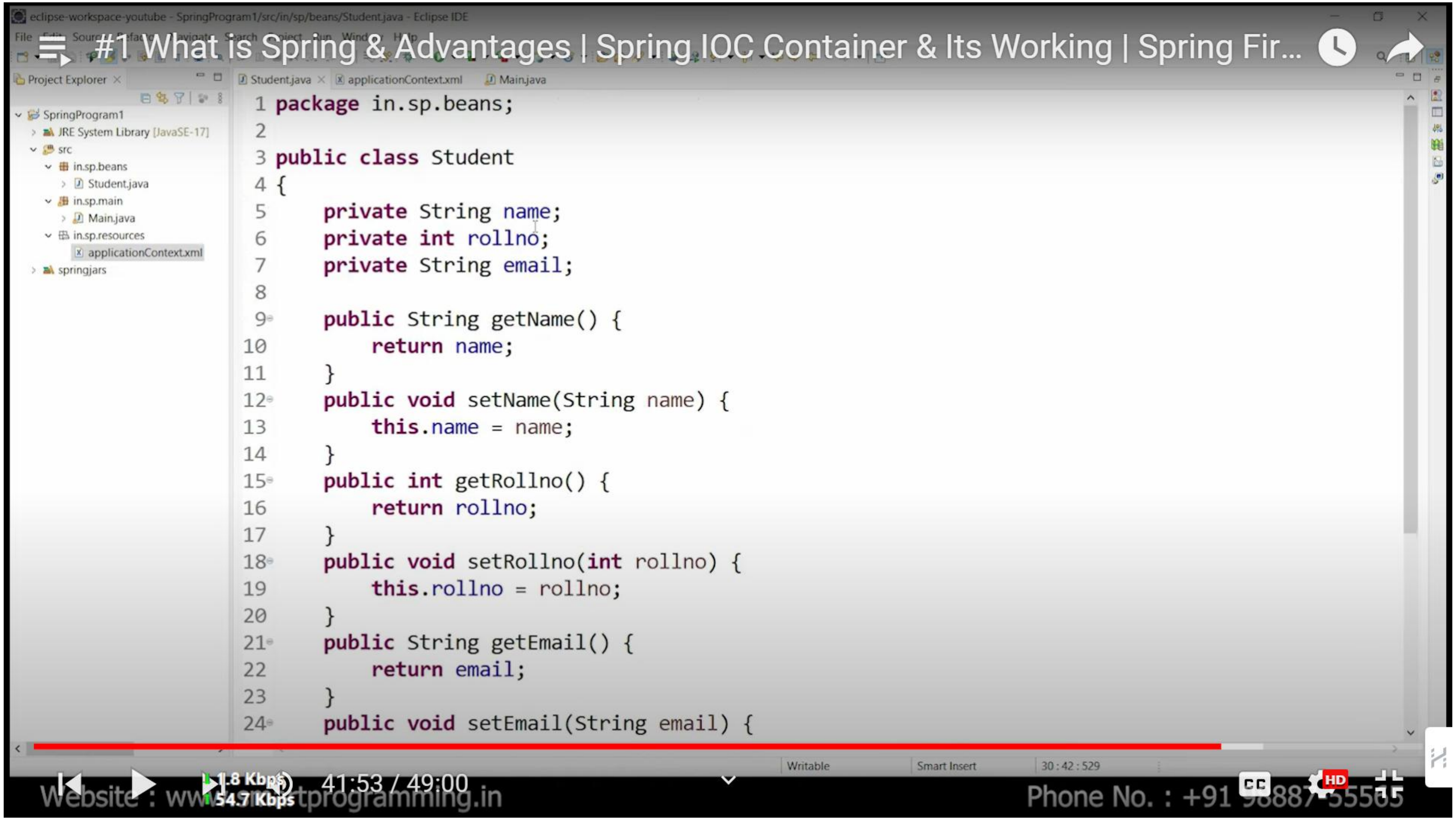
= But when we create MAVEN project, we provide dependencies through which jar files are automatically downloaded, we dont need to download the jar files manually and add them in project


eclipse-workspace-youtube - SpringProgram1/src/in/sp/resources/applicationContext.xml - Eclipse IDE
#1 What is Spring & Advantages | Spring IOC Container & Its Working | Spring Fir...
Project Explorer
SpringProgram1
JRE System Library [JavaSE-17]
src
in.sp.beans
Student.java
in.sp.main
Main.java
in.sp.resources
applicationContext.xml
springjars
Student.java
applicationContext.xml
Main.java
beans/bean/property/value
Writable
Smart Insert
14 : 32 : 602
46:19 / 49:00
Website : www.smartprogramming.in
Phone No. : +91 98887 55565

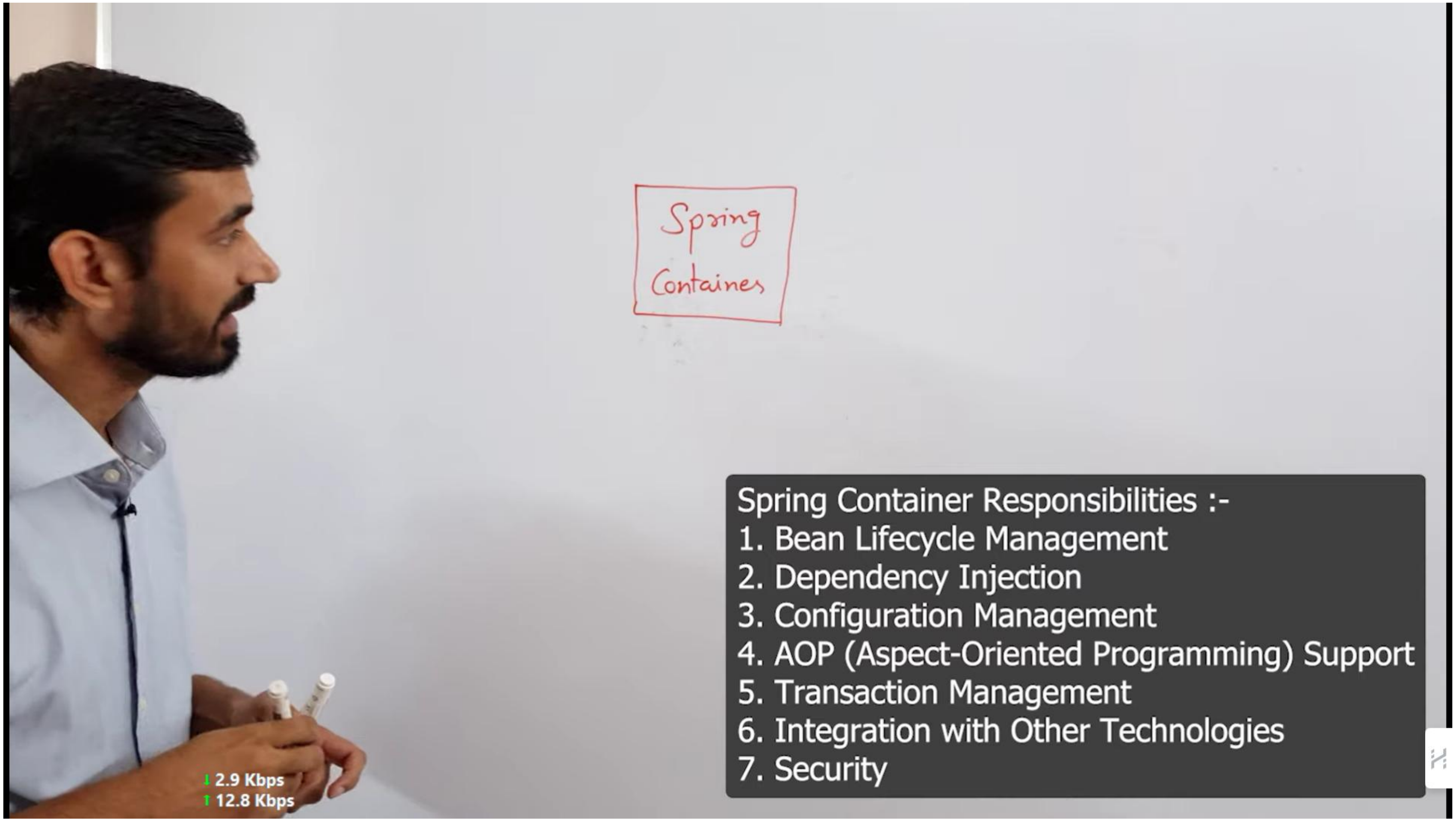
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/bean

    <bean class="in.sp.beans.Student" id="stdId1">
        <property name="name" value="Deepak" />
        <property name="rollno" value="101" />
        <property name="email" value="deepak@gmail.com" />
    </bean>

    <bean class="in.sp.beans.Student" id="stdId2">
        <property name="name" value="Amit" />
        <property name="rollno" value="102" />
        <property name="email" value="amit@gmail.com" />
    </bean>

</beans>
```


eclipse-workspace-youtube - SpringProgram1/src/in/sp/main/Main.java - Eclipse IDE
#1 What is Spring & Advantages | Spring IOC Container & Its Working | Spring Fir...
Project Explorer
SpringProgram1
JRE System Library [JavaSE-17]
src
in.sp.beans
Student.java
in.sp.main
Main.java
in.sp.resources
applicationContext.xml
springjars
Student.java
applicationContext.xml
Main.java
47:04 / 49:00
Website : www.smartprogramming.in
Phone No. : +91 98887 55565

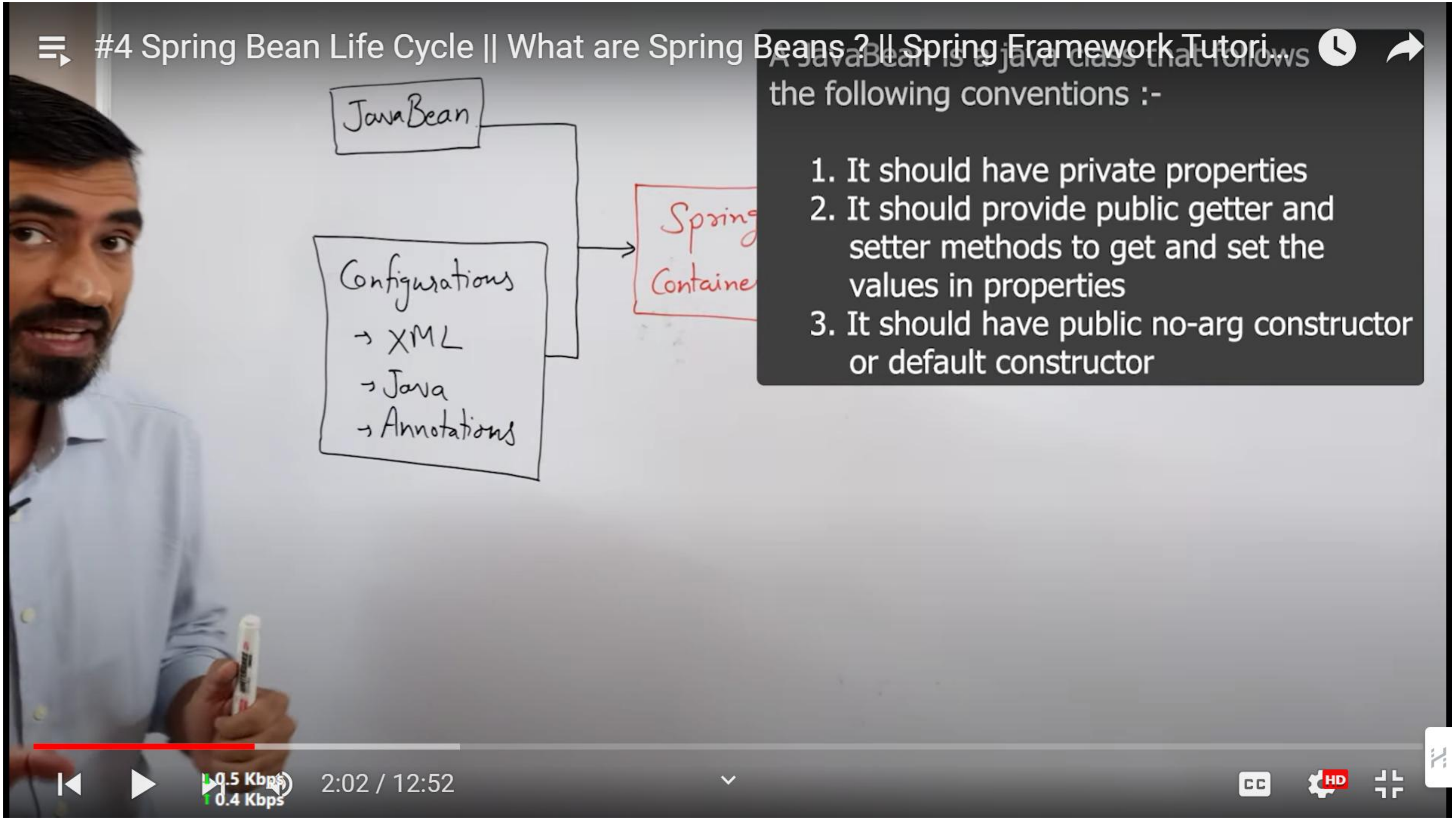
```java
package in.sp.main;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import in.sp.beans.Student;

public class Main
{
    public static void main(String[] args)
    {
        String config_loc = "/in/sp/resources/applicationContext.xml";
        ApplicationContext context = new ClassPathXmlApplicationContext(config_loc);

        Student std1 = (Student) context.getBean("stdId1");
        std1.display();

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

        Student std2 = (Student) context.getBean("stdId2");
        std2.display();
    }
}

```


eclipse-workspace-youtube - SpringProgram1/src/in/sp/beans/Student.java - Eclipse IDE
#1 What is Spring & Advantages | Spring IOC Container & Its Working | Spring Fir...
Project Explorer
SpringProgram1
JRE System Library [JavaSE-17]
src
in.sp.beans
Student.java
in.sp.main
Main.java
in.sp.resources
applicationContext.xml
springjars
Student.java
applicationContext.xml
Main.java
Writable
Smart Insert
30 : 42 : 529
41:53 / 49:00
Website : www.[illegible]tprogramming.in
Phone No. : +91 98887 55565

```java
package in.sp.beans;

public class Student
{
    private String name;
    private int rollno;
    private String email;

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getRollno() {
        return rollno;
    }
    public void setRollno(int rollno) {
        this.rollno = rollno;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
```

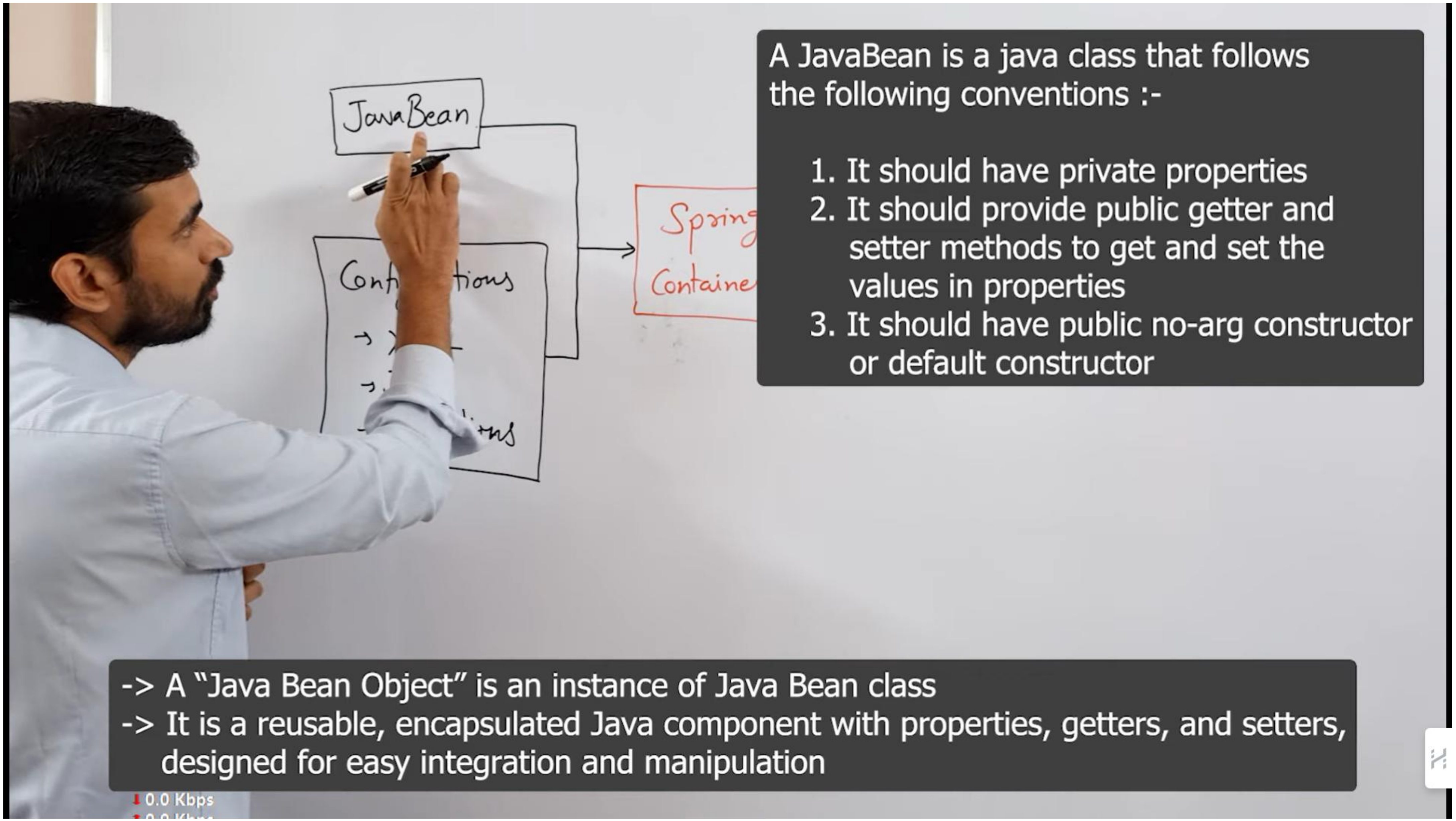


Spring Container Responsibilities :-
1. Bean Lifecycle Management
2. Dependency Injection
3. Configuration Management
4. AOP (Aspect-Oriented Programming) Support
5. Transaction Management
6. Integration with Other Technologies
7. Security



A JavaBean is a java class that follows the following conventions :-

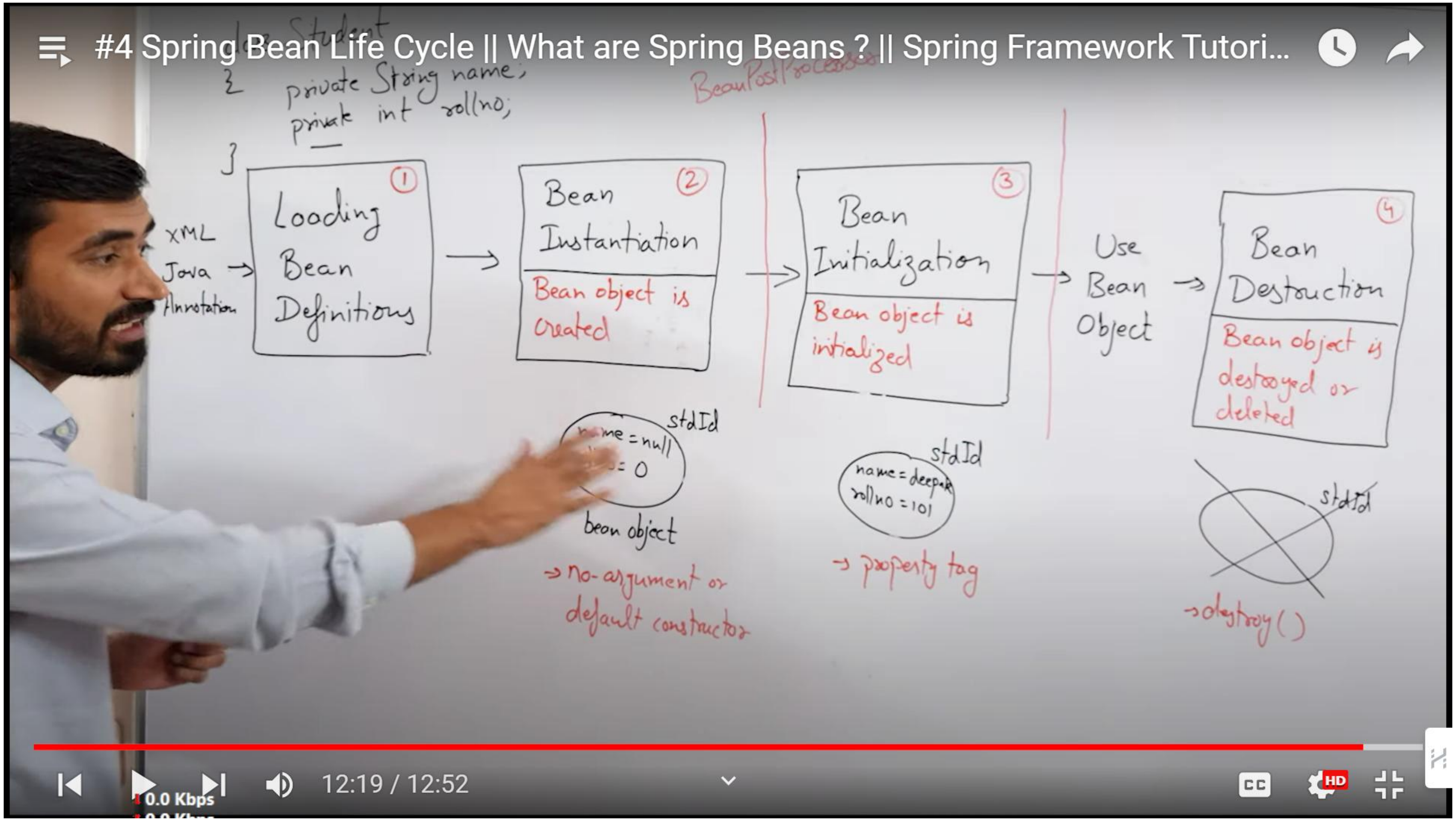
1. It should have private properties
2. It should provide public getter and setter methods to get and set the values in properties
3. It should have public no-arg constructor or default constructor

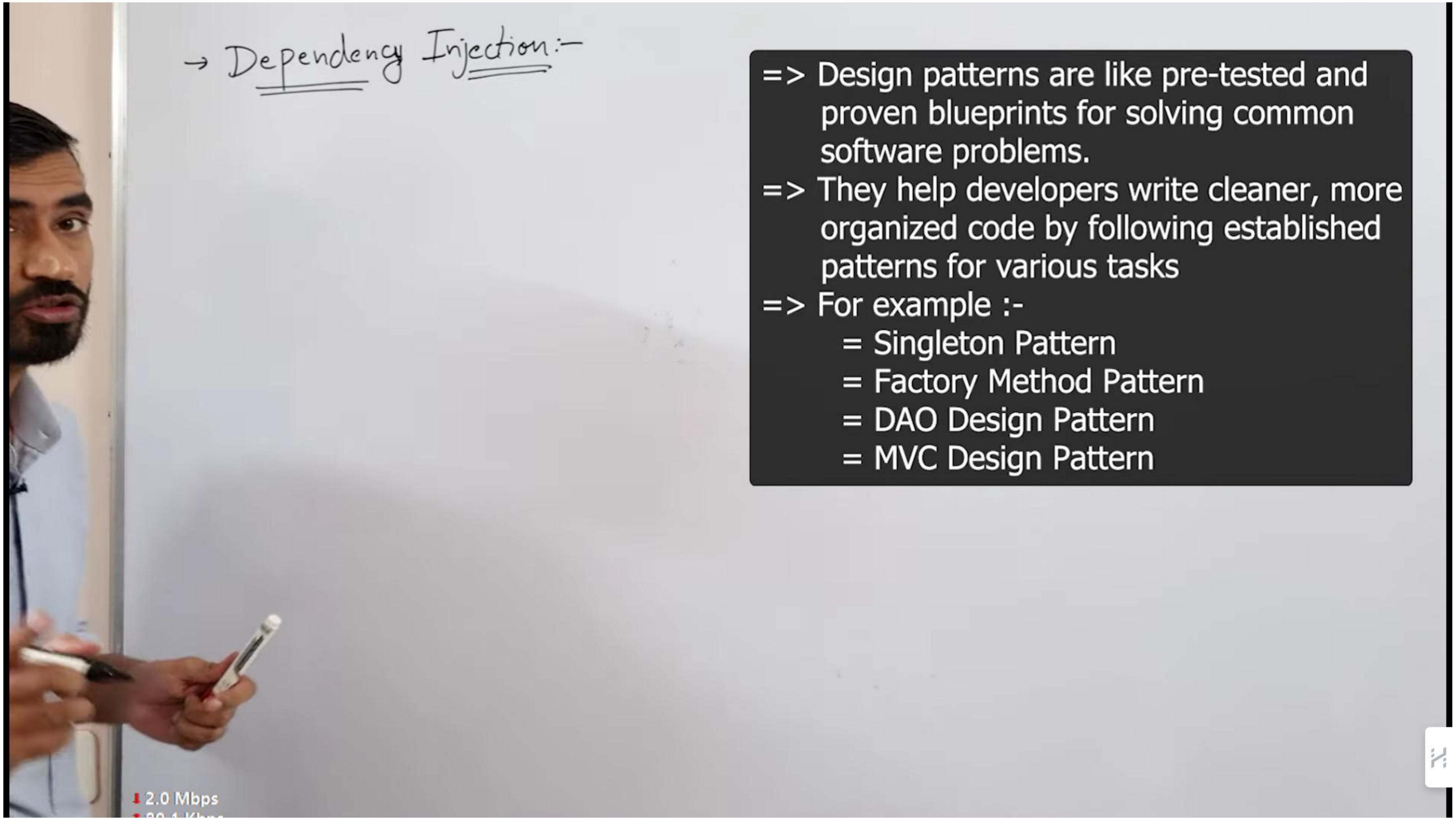


A JavaBean is a java class that follows the following conventions :-

1. It should have private properties
2. It should provide public getter and setter methods to get and set the values in properties
3. It should have public no-arg constructor or default constructor

-> A "Java Bean Object" is an instance of Java Bean class
-> It is a reusable, encapsulated Java component with properties, getters, and setters, designed for easy integration and manipulation


#4 Spring Bean Life Cycle || What are Spring Beans ? || Spring Framework Tutori...
class Student
{ private String name;
private int rollno;
}
BeanPostProcessor
XML
Java
Annotation
① Loading Bean Definitions
② Bean Instantiation
Bean object is created
③ Bean Initialization
Bean object is initialized
Use Bean Object
④ Bean Destruction
Bean object is destroyed or deleted
name = null
rollno = 0
stdId
bean object
→ no-argument or default constructor
name = deepak
rollno = 101
stdId
→ property tag
stdId
→ destroy()
12:19 / 12:52
0.0 Kbps



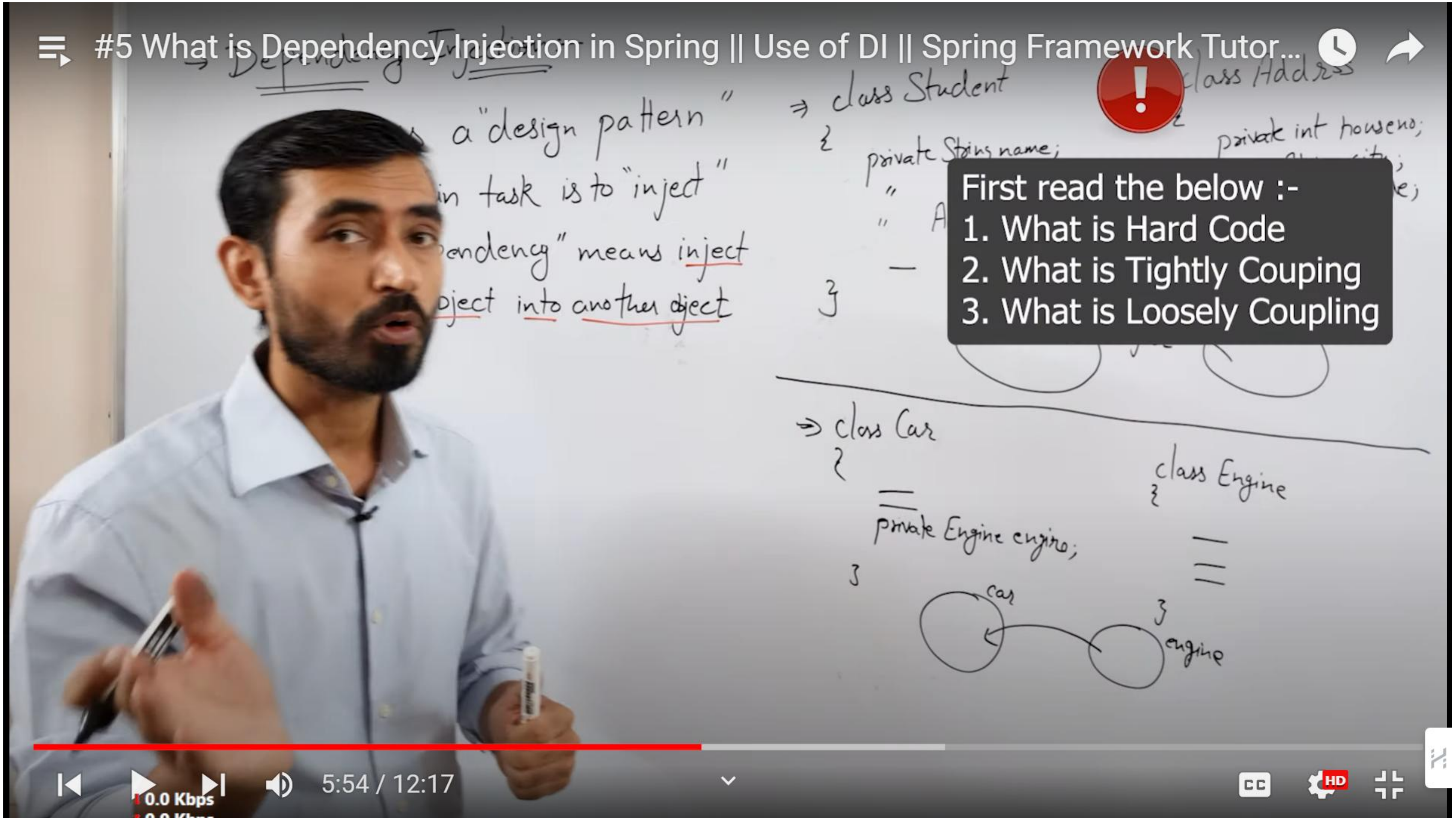
→ Dependency Injection :-

=> Design patterns are like pre-tested and proven blueprints for solving common software problems.

=> They help developers write cleaner, more organized code by following established patterns for various tasks

=> For example :-

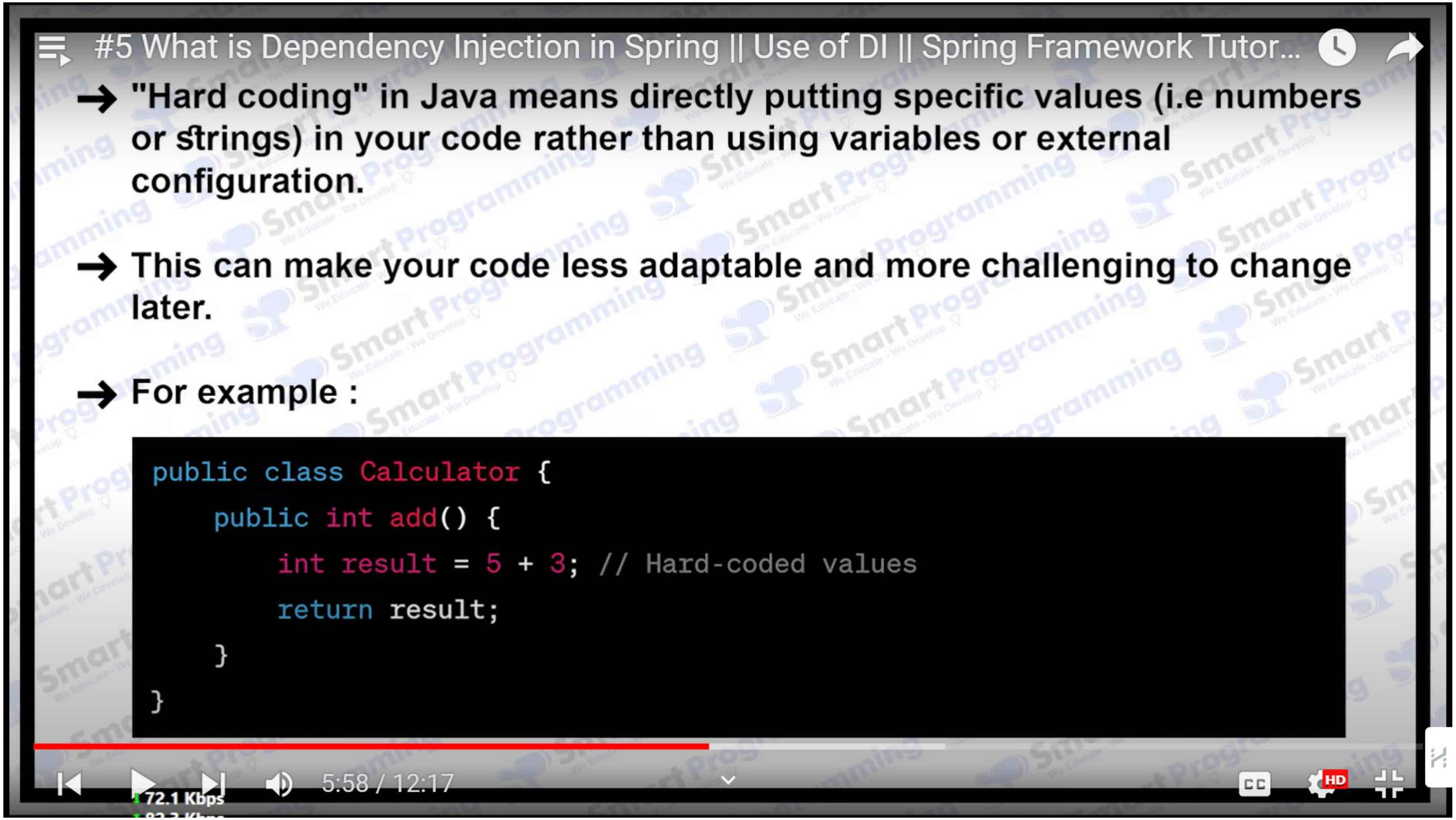
- = Singleton Pattern
- = Factory Method Pattern
- = DAO Design Pattern
- = MVC Design Pattern



#5 What is Dependency Injection in Spring || Use of DI || Spring Framework Tutor...

First read the below :-
1. What is Hard Code
2. What is Tightly Couping
3. What is Loosely Coupling

5:54 / 12:17



#5 What is Dependency Injection in Spring || Use of DI || Spring Framework Tutor...

→ "Hard coding" in Java means directly putting specific values (i.e numbers or strings) in your code rather than using variables or external configuration.

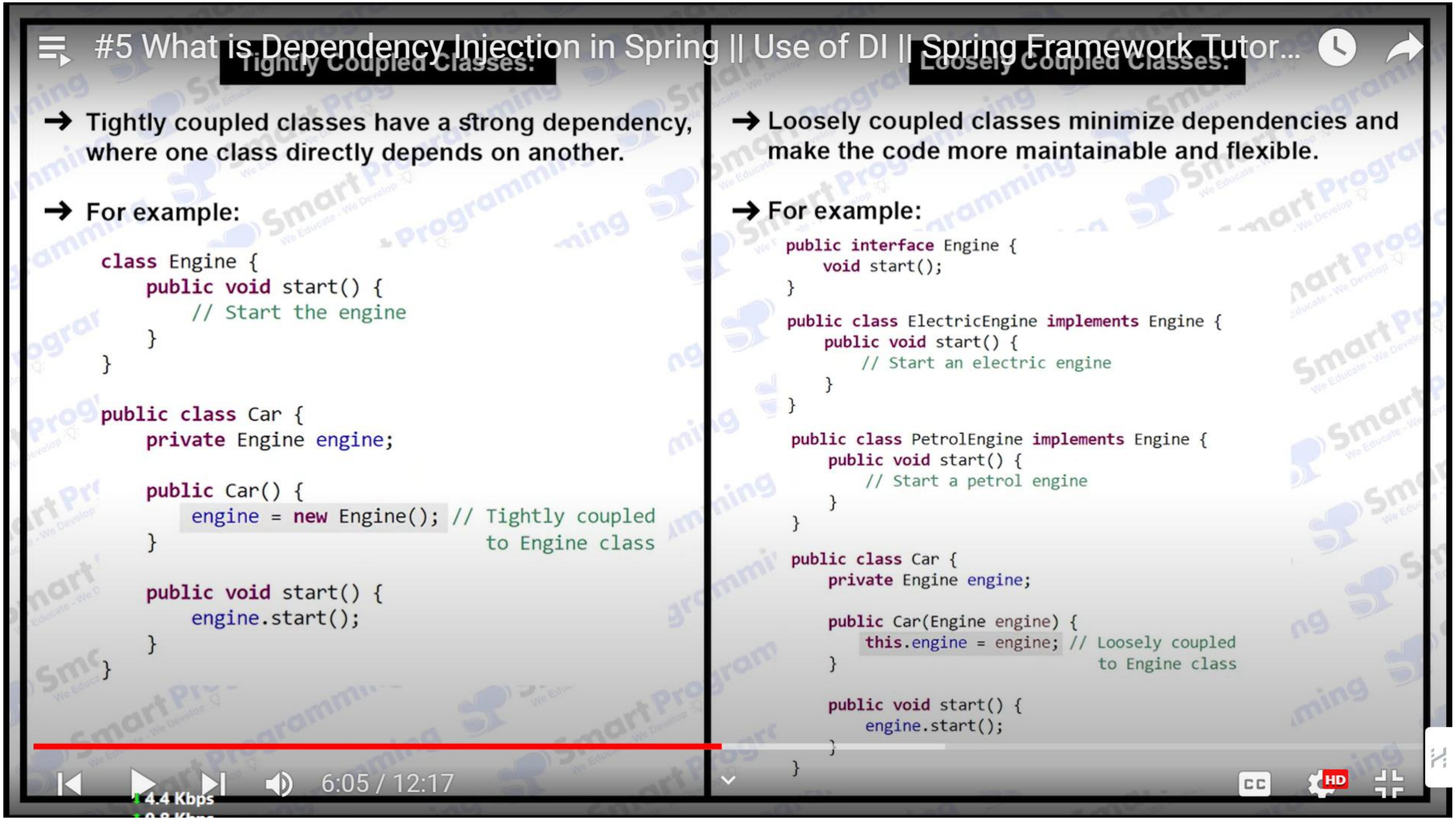
→ This can make your code less adaptable and more challenging to change later.

→ For example :

```java
public class Calculator {
    public int add() {
        int result = 5 + 3; // Hard-coded values
        return result;
    }
}
```

5:58 / 12:17

72.1 Kbps



#5 What is Dependency Injection in Spring || Use of DI || Spring Framework Tutor...
6:05 / 12:17
4.4 Kbps

Tightly Coupled Classes:

→ Tightly coupled classes have a strong dependency, where one class directly depends on another.

→ For example:

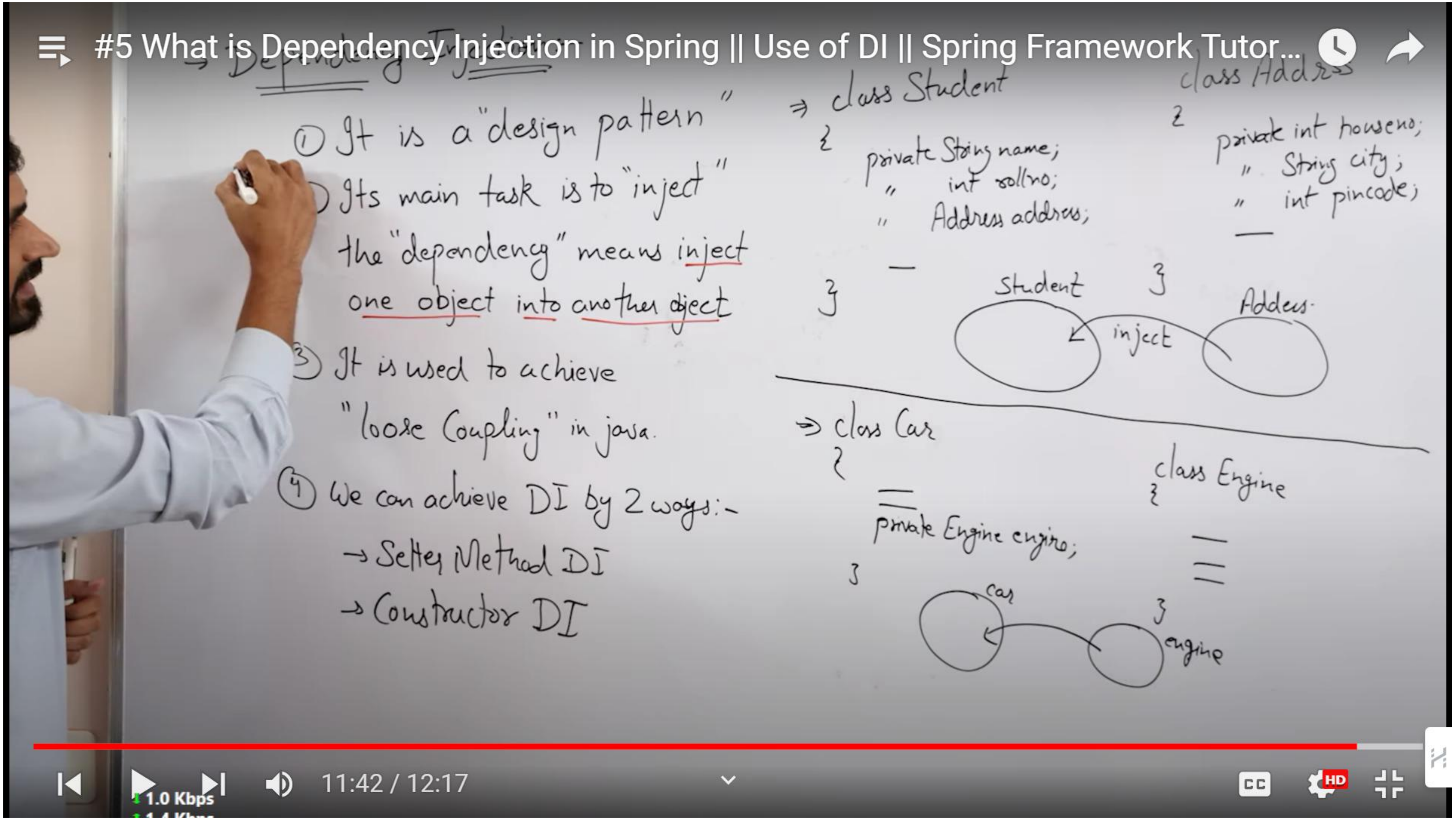
```java
class Engine {
    public void start() {
        // Start the engine
    }
}

public class Car {
    private Engine engine;

    public Car() {
        engine = new Engine(); // Tightly coupled
    }                            to Engine class

    public void start() {
        engine.start();
    }
}
```

Loosely Coupled Classes:

→ Loosely coupled classes minimize dependencies and make the code more maintainable and flexible.

→ For example:

```java
public interface Engine {
    void start();
}

public class ElectricEngine implements Engine {
    public void start() {
        // Start an electric engine
    }
}

public class PetrolEngine implements Engine {
    public void start() {
        // Start a petrol engine
    }
}

public class Car {
    private Engine engine;

    public Car(Engine engine) {
        this.engine = engine; // Loosely coupled
    }                            to Engine class

    public void start() {
        engine.start();
    }
}
```



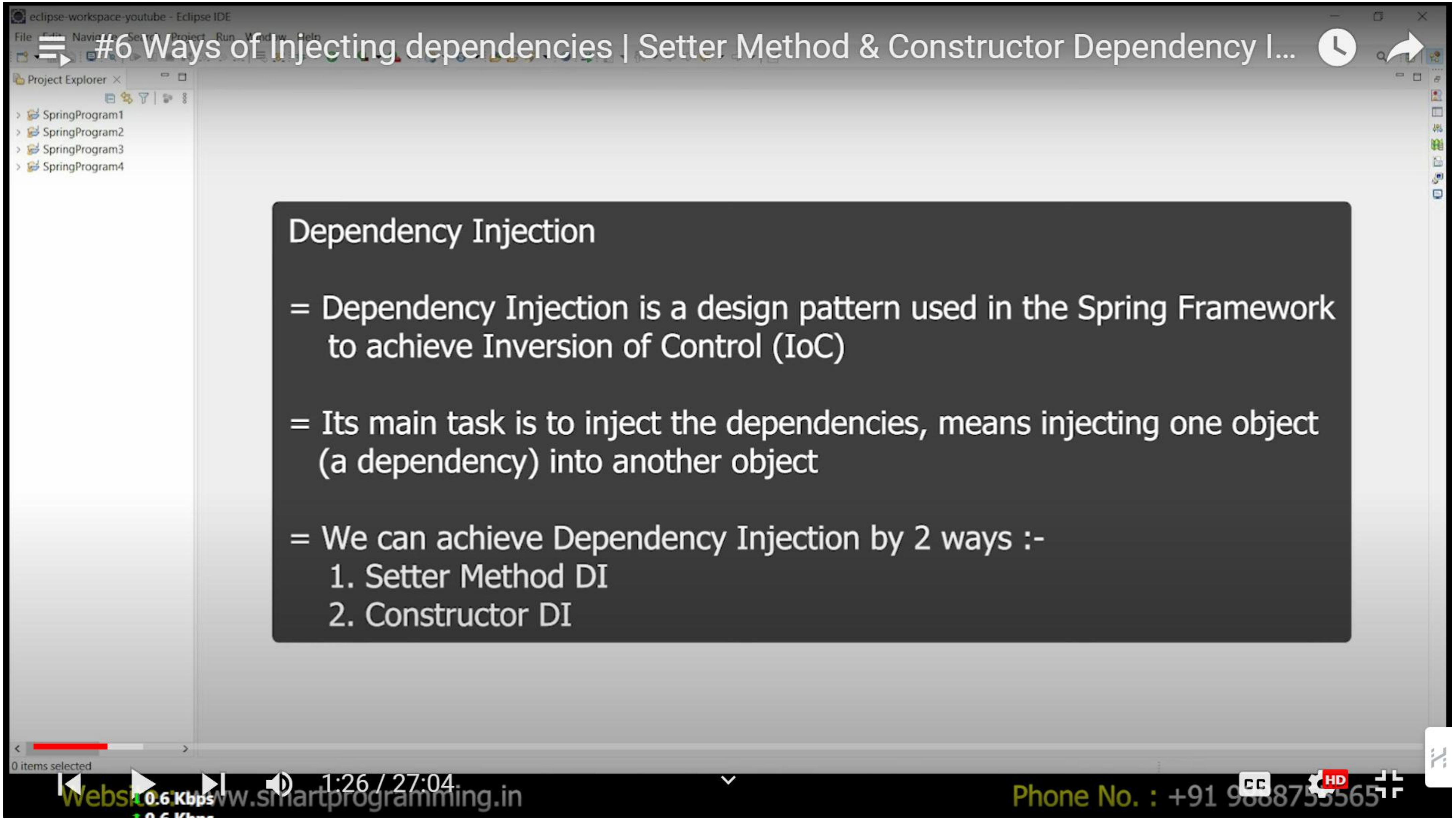
#5 What is Dependency Injection in Spring || Use of DI || Spring Framework Tutor...

→ Dependency Injection

1. It is a "design pattern"
2. Its main task is to "inject" the "dependency" means inject one object into another object
3. It is used to achieve "loose Coupling" in java.
4. We can achieve DI by 2 ways:-
   → Setter Method DI
   → Constructor DI

```
⇒ class Student
{
    private String name;
      "     int rollno;
      "     Address address;
      —
}

class Address
{
    private int houseno;
      "     String city;
      "     int pincode;
      —
}
```

```
⇒ class Car
{
    ═
    private Engine engine;
}

class Engine
{
    ═
    ═
}
```

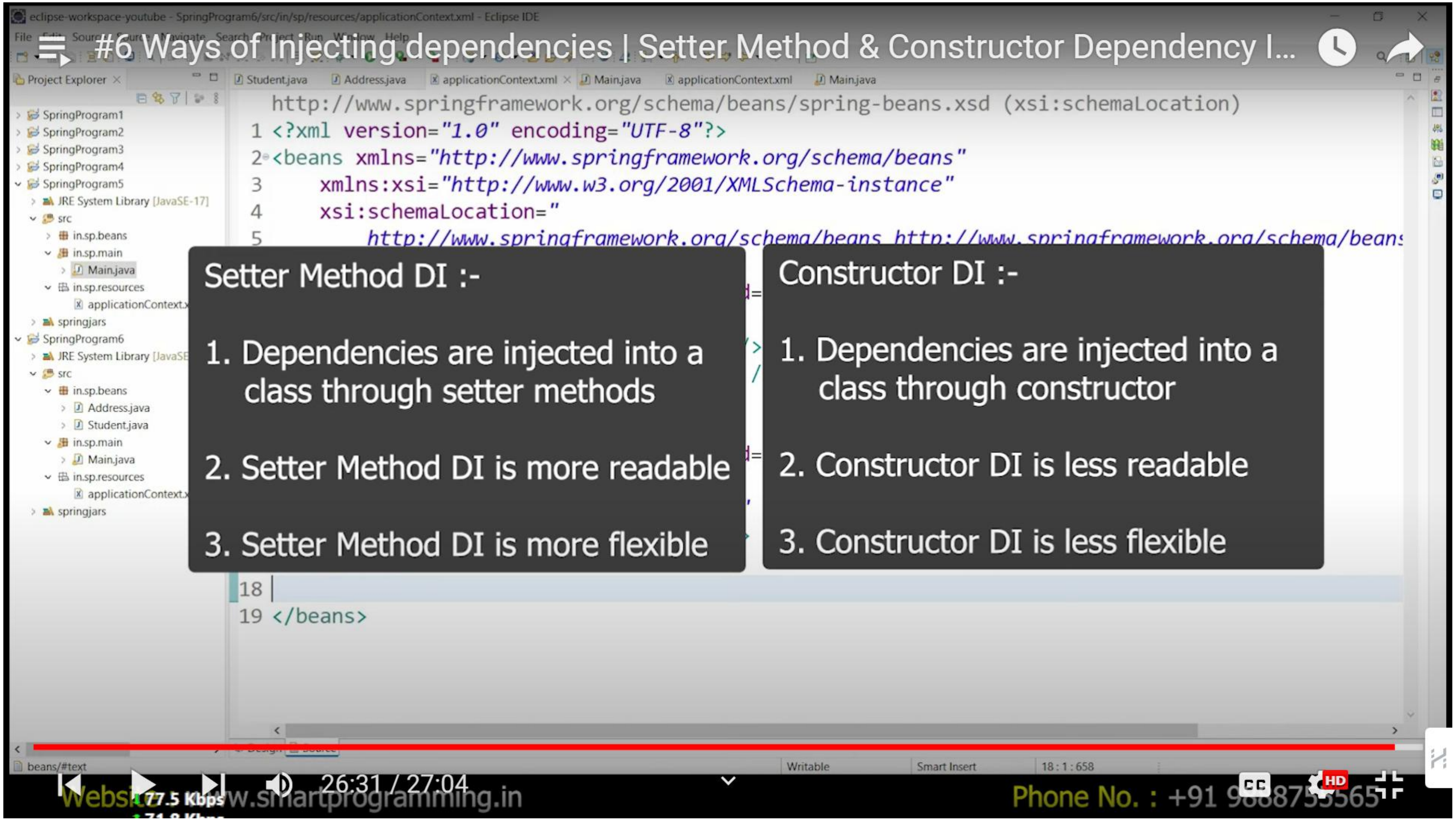

eclipse-workspace-youtube - Eclipse IDE
#6 Ways of Injecting dependencies | Setter Method & Constructor Dependency I...
Project Explorer
SpringProgram1
SpringProgram2
SpringProgram3
SpringProgram4
0 items selected
1:26 / 27:04
Website : www.smartprogramming.in
Phone No. : +91 9888755565

Dependency Injection

= Dependency Injection is a design pattern used in the Spring Framework to achieve Inversion of Control (IoC)

= Its main task is to inject the dependencies, means injecting one object (a dependency) into another object

= We can achieve Dependency Injection by 2 ways :-
1. Setter Method DI
2. Constructor DI



#6 Ways of Injecting dependencies | Setter Method & Constructor Dependency I...

```
http://www.springframework.org/schema/beans/spring-beans.xsd (xsi:schemaLocation)
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans

</beans>
```

Setter Method DI :-

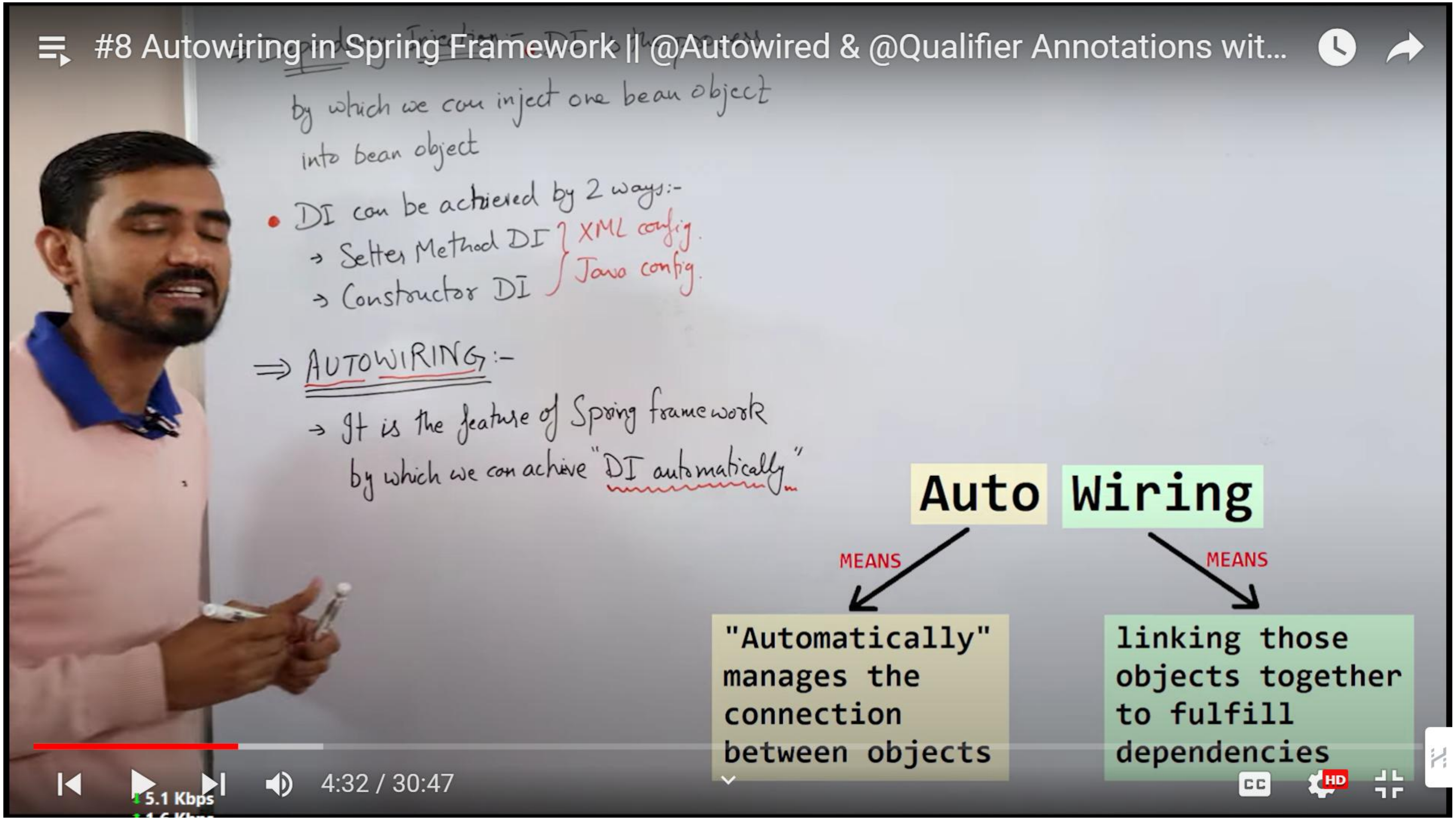
1. Dependencies are injected into a class through setter methods
2. Setter Method DI is more readable
3. Setter Method DI is more flexible

Constructor DI :-

1. Dependencies are injected into a class through constructor
2. Constructor DI is less readable
3. Constructor DI is less flexible

Website: www.smartprogramming.in

Phone No. : +91 9888755565



#8 Autowiring in Spring Framework || @Autowired & @Qualifier Annotations wit...

by which we can inject one bean object into bean object

- DI can be achieved by 2 ways:-
  - → Setter Method DI } XML config
  - → Constructor DI } Java config.

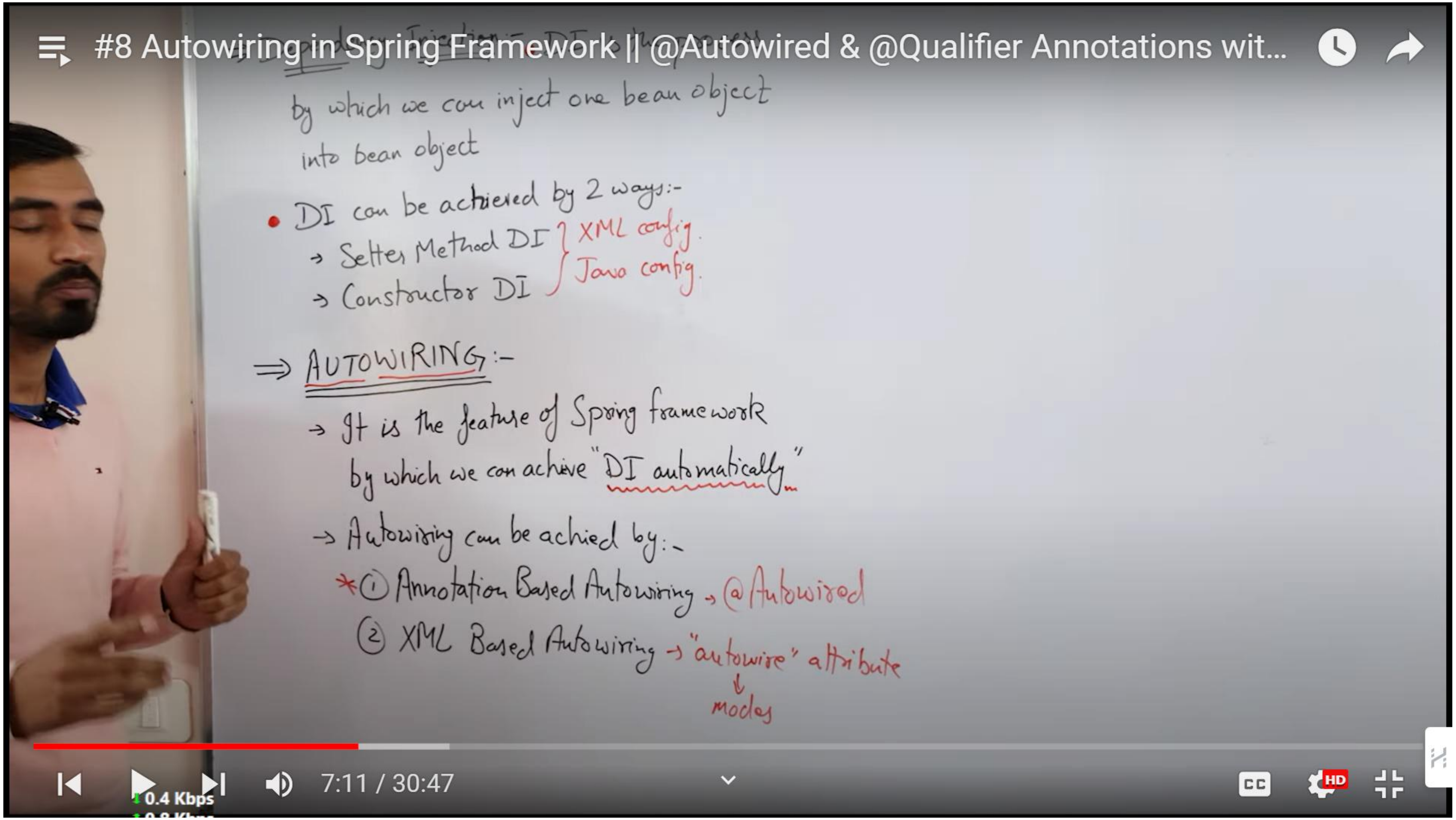
⇒ AUTOWIRING:-

→ It is the feature of Spring framework by which we can achive "DI automatically"

Auto Wiring

Auto MEANS "Automatically" manages the connection between objects

Wiring MEANS linking those objects together to fulfill dependencies



#8 Autowiring in Spring Framework || @Autowired & @Qualifier Annotations wit...

by which we can inject one bean object into bean object

- DI can be achieved by 2 ways:-
  - → Setter Method DI
  - → Constructor DI

  } XML config. / Java config.

⇒ AUTOWIRING:-

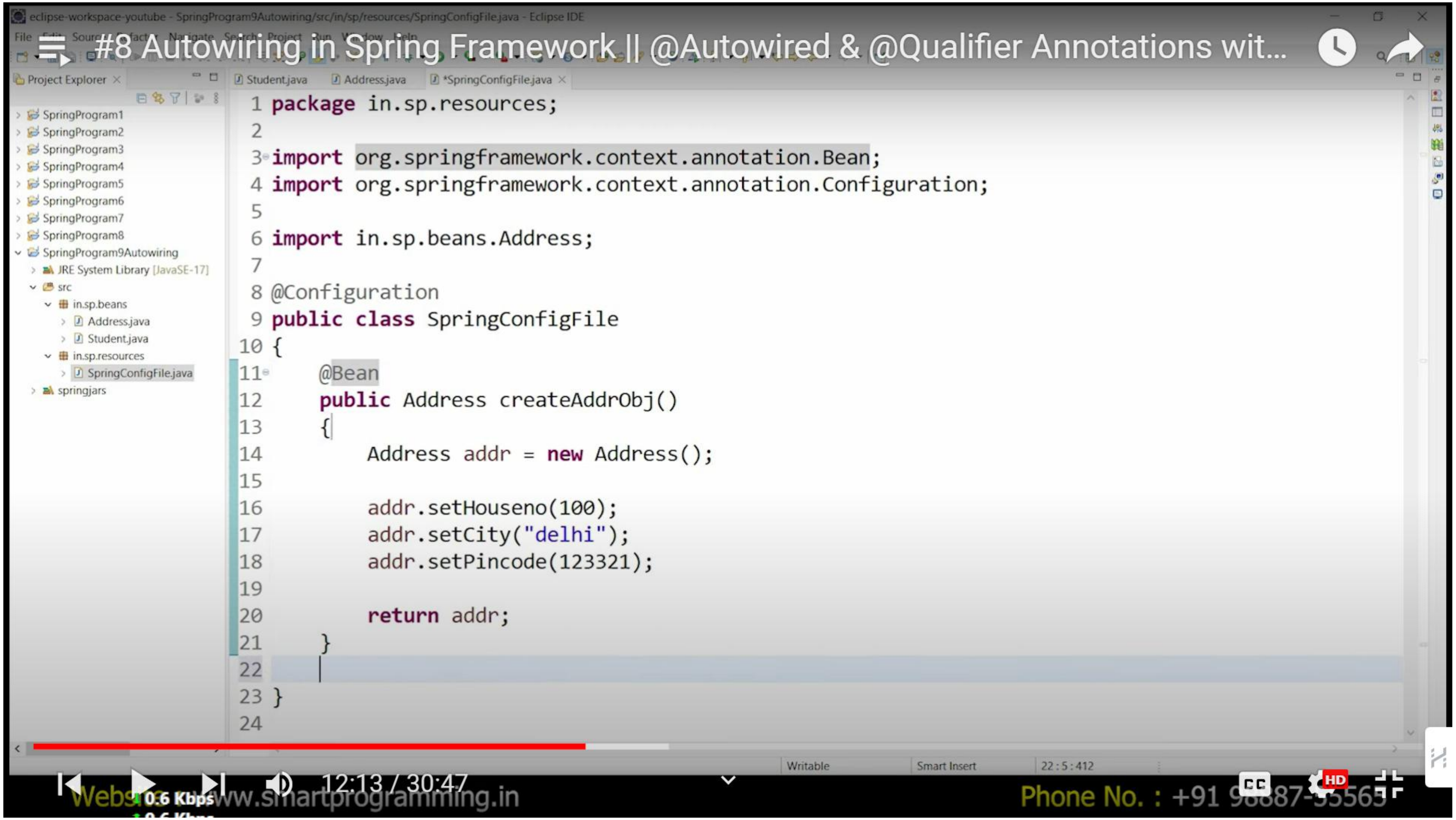
→ It is the feature of Spring framework by which we can achive "DI automatically"

→ Autowiring can be achied by:-

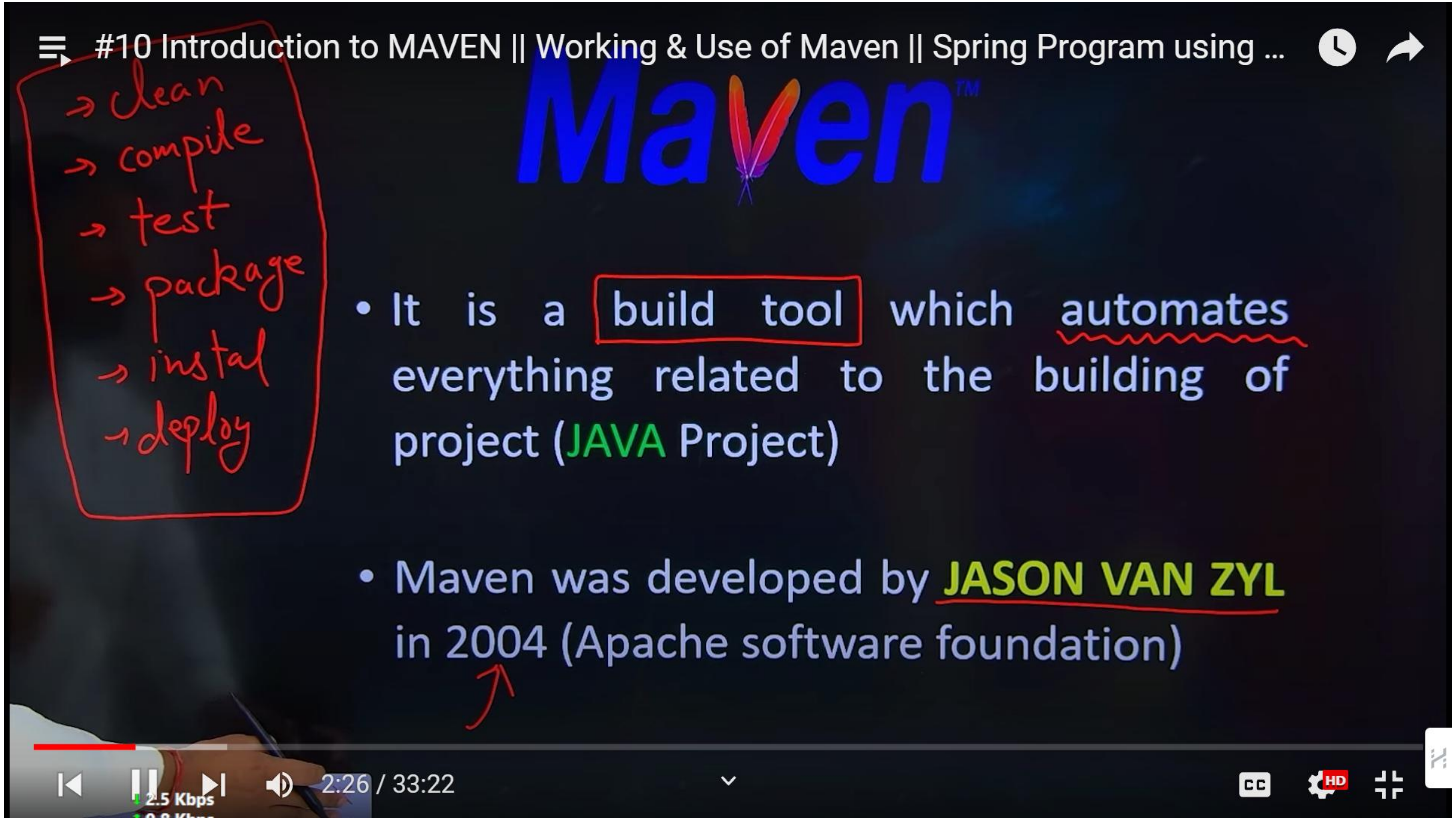
* ① Annotation Based Autowiring → @Autowired

② XML Based Autowiring → "autowire" attribute → modes

7:11 / 30:47



```java
package in.sp.resources;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import in.sp.beans.Address;

@Configuration
public class SpringConfigFile
{
    @Bean
    public Address createAddrObj()
    {
        Address addr = new Address();

        addr.setHouseno(100);
        addr.setCity("delhi");
        addr.setPincode(123321);

        return addr;
    }

}
```

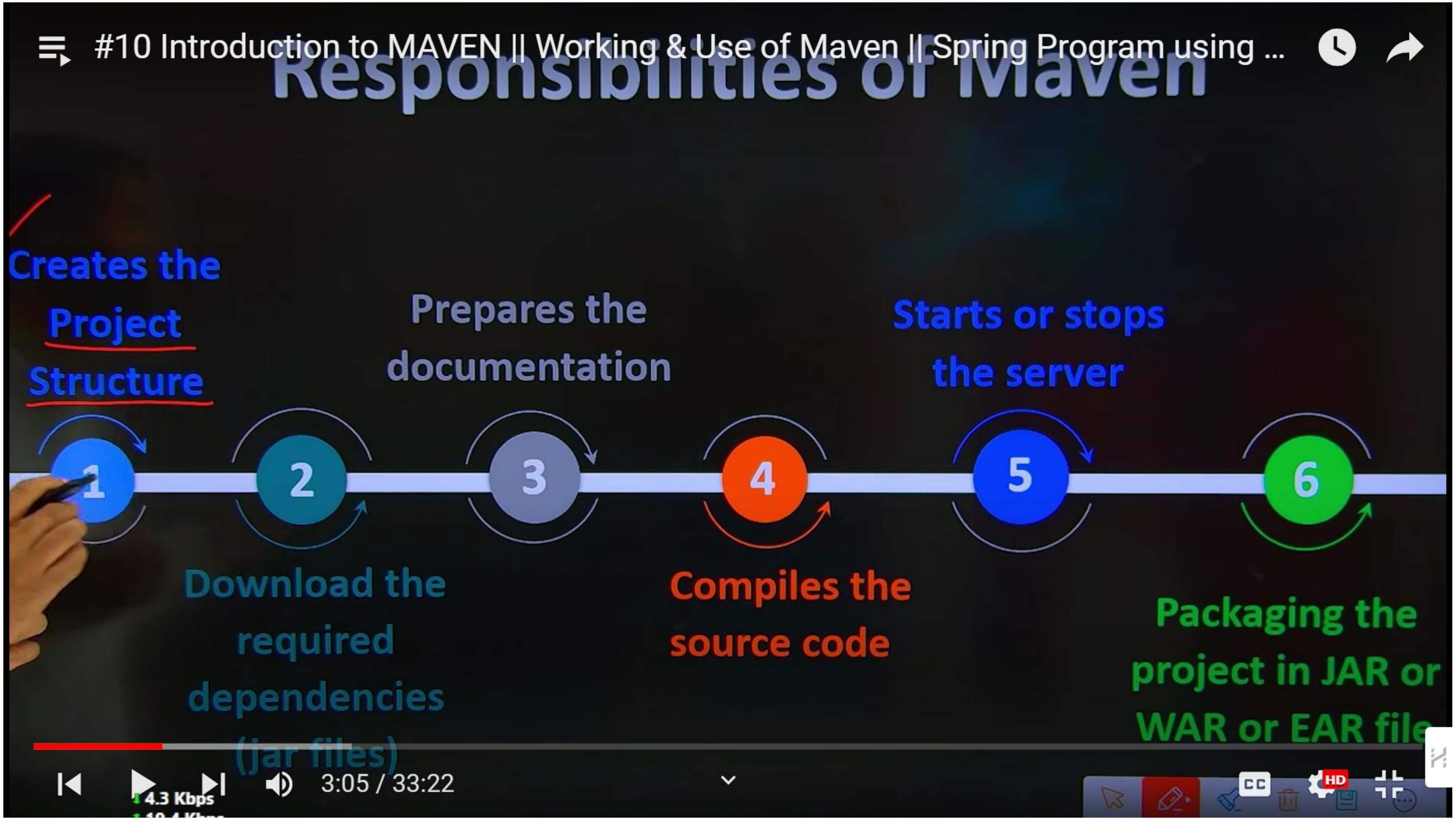


#10 Introduction to MAVEN || Working & Use of Maven || Spring Program using ...

Maven™

- It is a $\boxed{\text{build tool}}$ which automates everything related to the building of project (JAVA Project)
- Maven was developed by JASON VAN ZYL in 2004 (Apache software foundation)

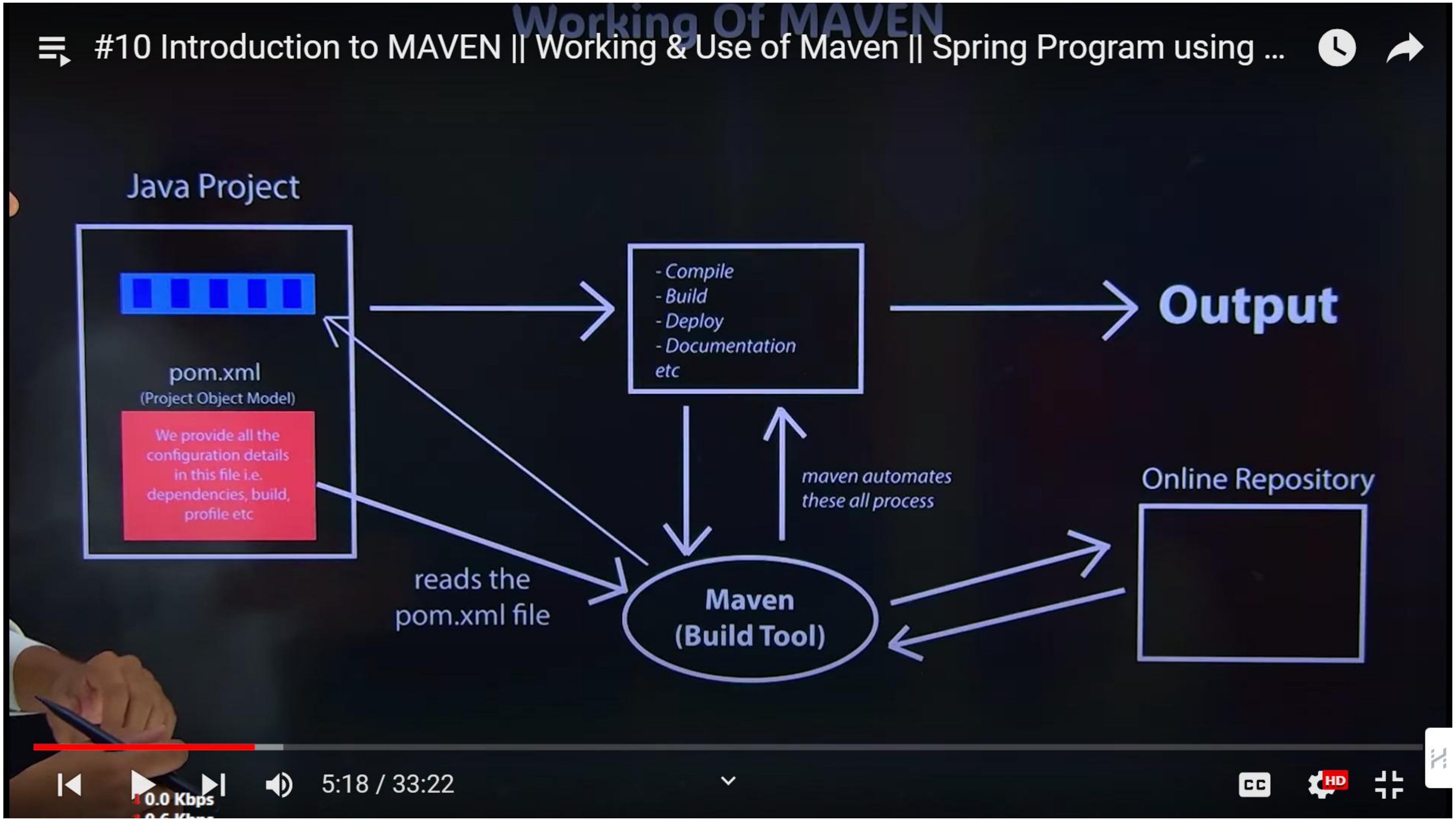
→ clean
→ compile
→ test
→ package
→ instal
→ deploy



#10 Introduction to MAVEN || Working & Use of Maven || Spring Program using ...

Responsibilities of Maven

1. Creates the Project Structure
2. Download the required dependencies (jar files)
3. Prepares the documentation
4. Compiles the source code
5. Starts or stops the server
6. Packaging the project in JAR or WAR or EAR file


#10 Introduction to MAVEN || Working & Use of Maven || Spring Program using ...

Working Of MAVEN