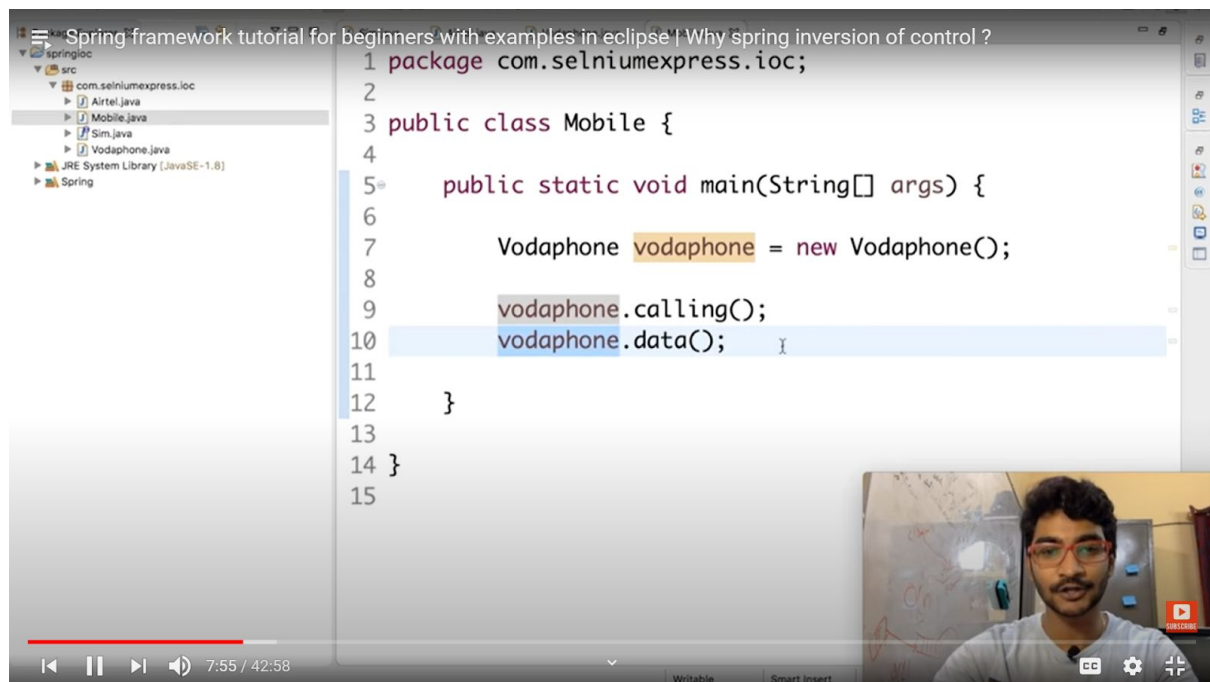
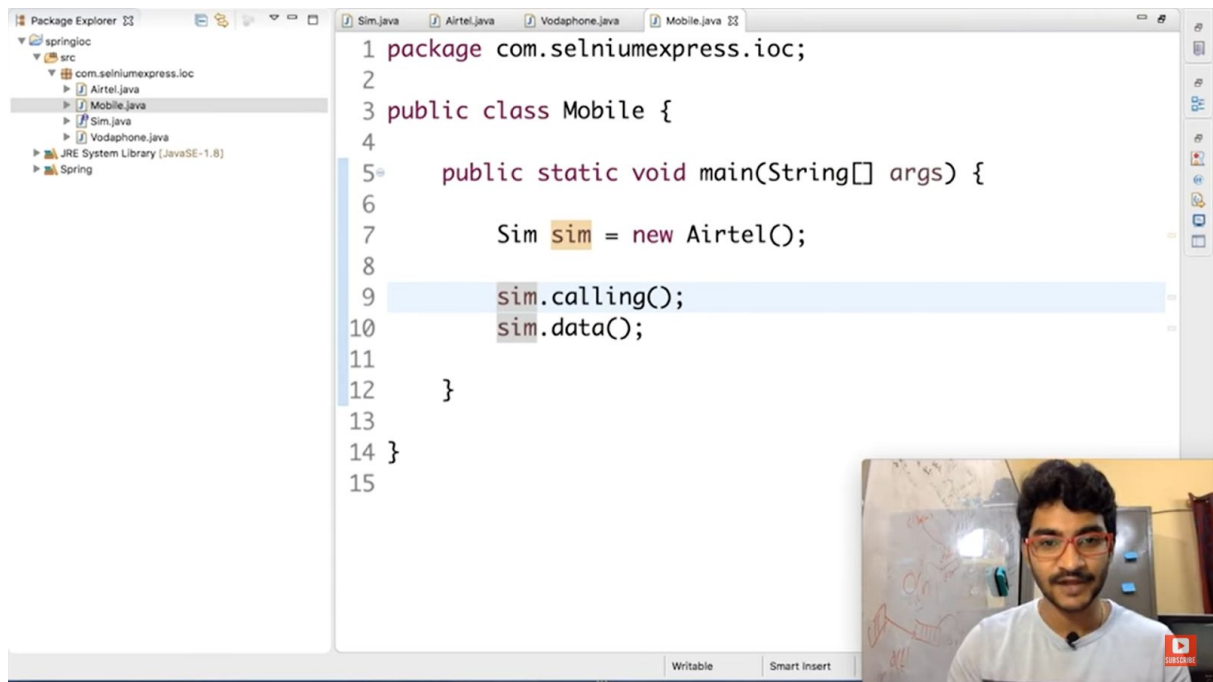


Now to Use Vodafone sim we have to make changes to code again



To overcome this problem using concept of interface



DON'T TOUCH THE SOURCE CODE

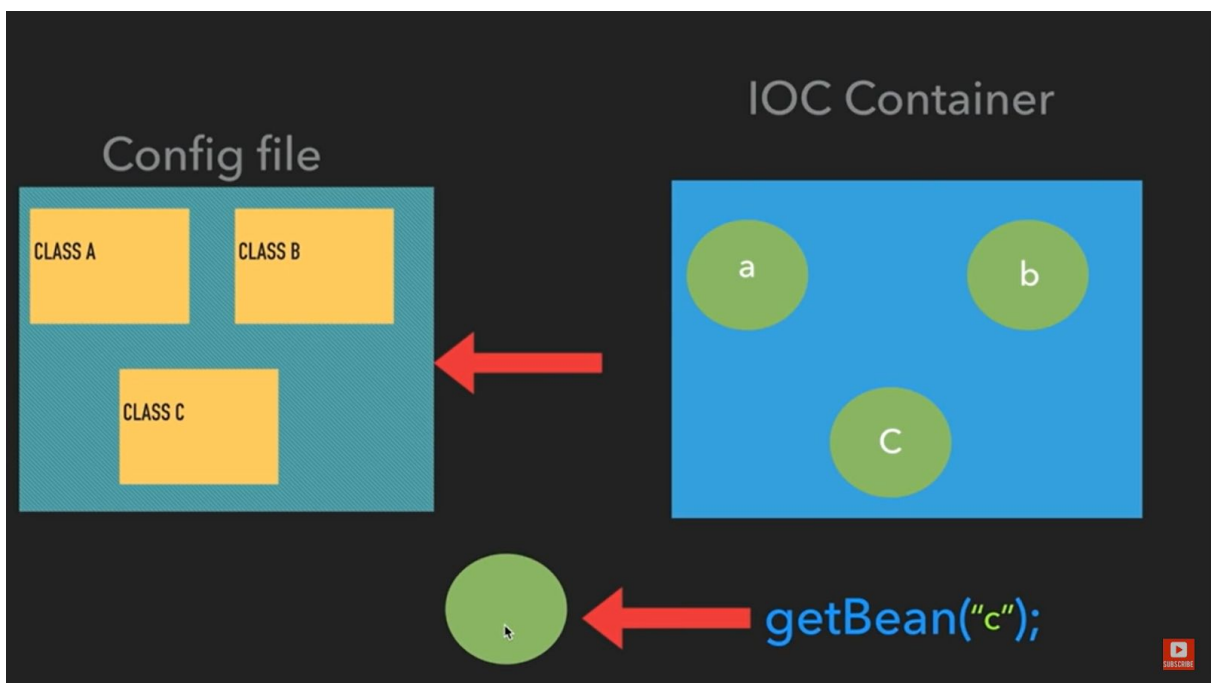
THIS APP SHOULD BE
CONFIGURABLE.

#spring





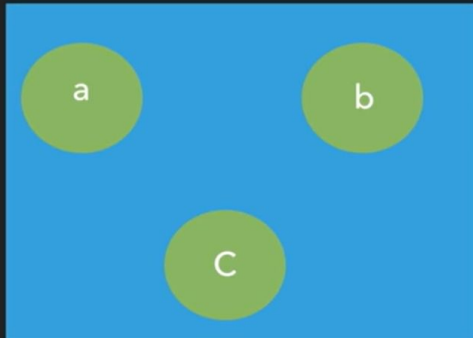
Mention all classes in config files jiske objects bnane he. IOC container reads config file and contains Objects of these classes and these objects in container are called BEAN.



To use bean in IOC Container we have two types

1. BeanFactory
2. ApplicationContext: It is provides much more functionality and used nowadays

IOC Container



2 types of IOC container :

- BeanFactory — Interface
- • ApplicationContext — Interface

implements

ClassPathXmlApplicationContext

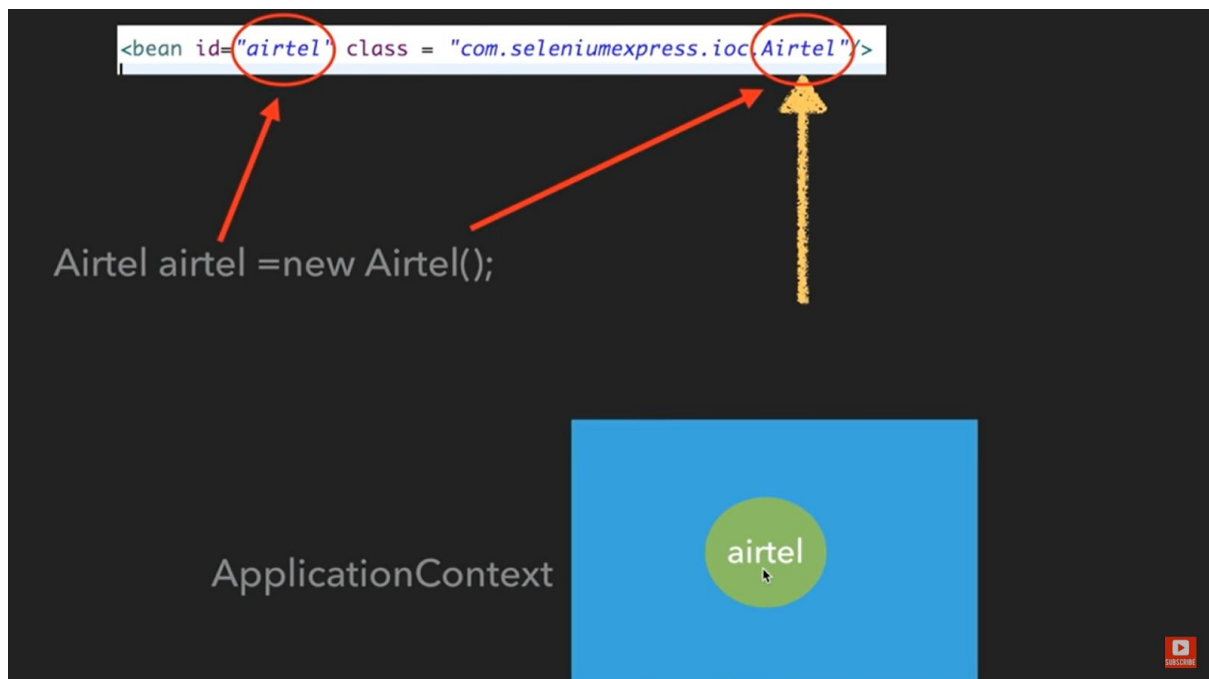
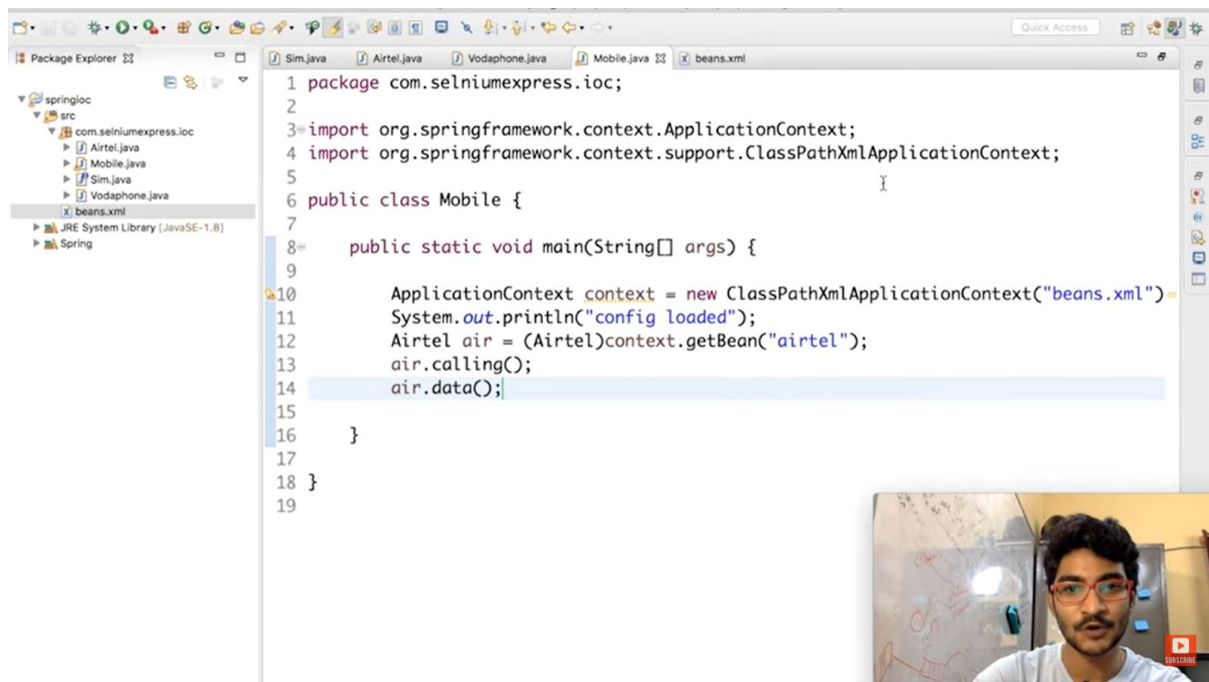


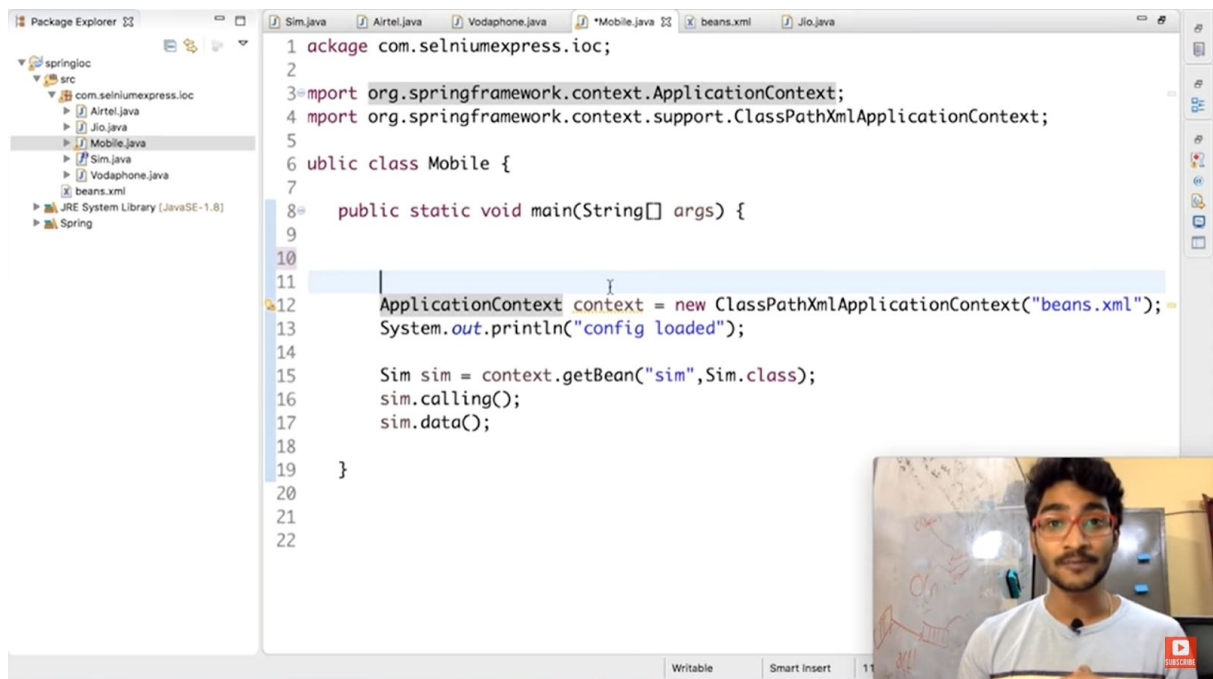
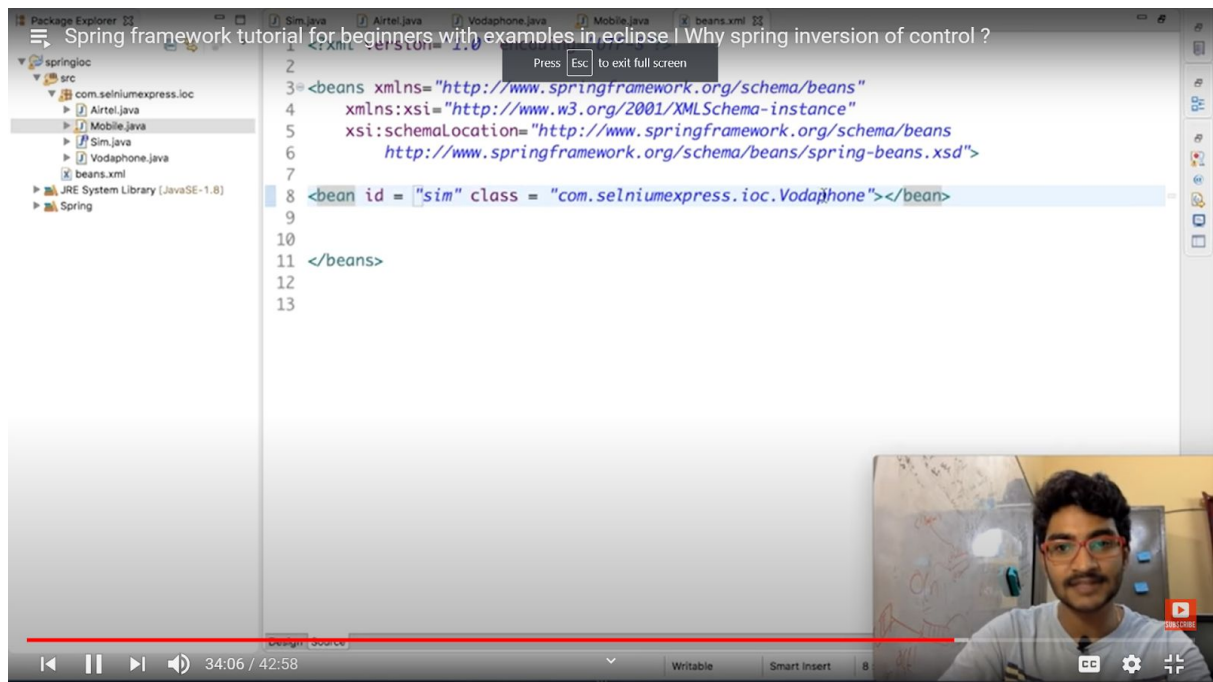
beans.xml is the config file

The screenshot shows an IDE with the Package Explorer on the left and the Source editor on the right. The Package Explorer shows a project named 'springloc' with a package 'com.seleniumexpress.ioc' containing files 'Airtel.java', 'Mobile.java', 'Sim.java', 'Vodafone.java', and 'beans.xml'. The Source editor displays the content of 'beans.xml' with the following XML code:

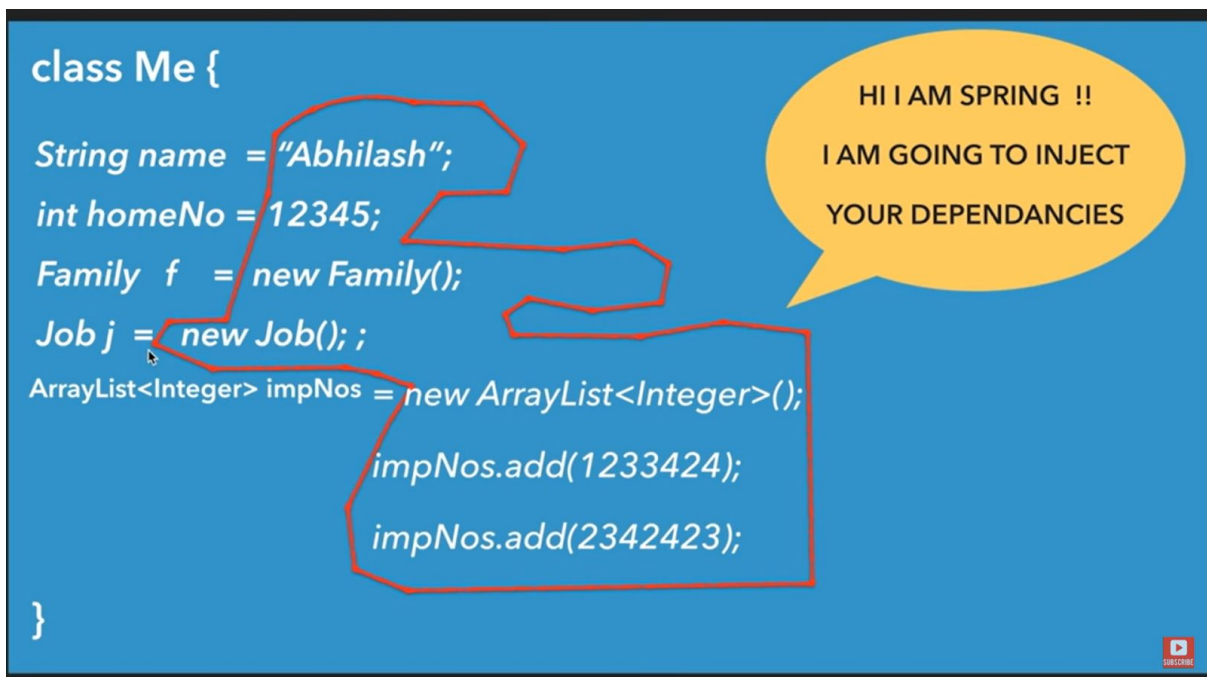
```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <beans xmlns="http://www.springframework.org/schema/beans"
4       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5       xsi:schemaLocation="http://www.springframework.org/schema/beans
6                           http://www.springframework.org/schema/beans/spring-beans.xsd">
7
8   <bean id = "airtel" class = "com.seleniumexpress.ioc.Airtel"></bean>
9
10
11 </beans>
12
13
```

In the bottom right corner, there is a small video inset showing a person with glasses and a microphone, likely the presenter.





Inversion of control means control of creating objects is taken by framework



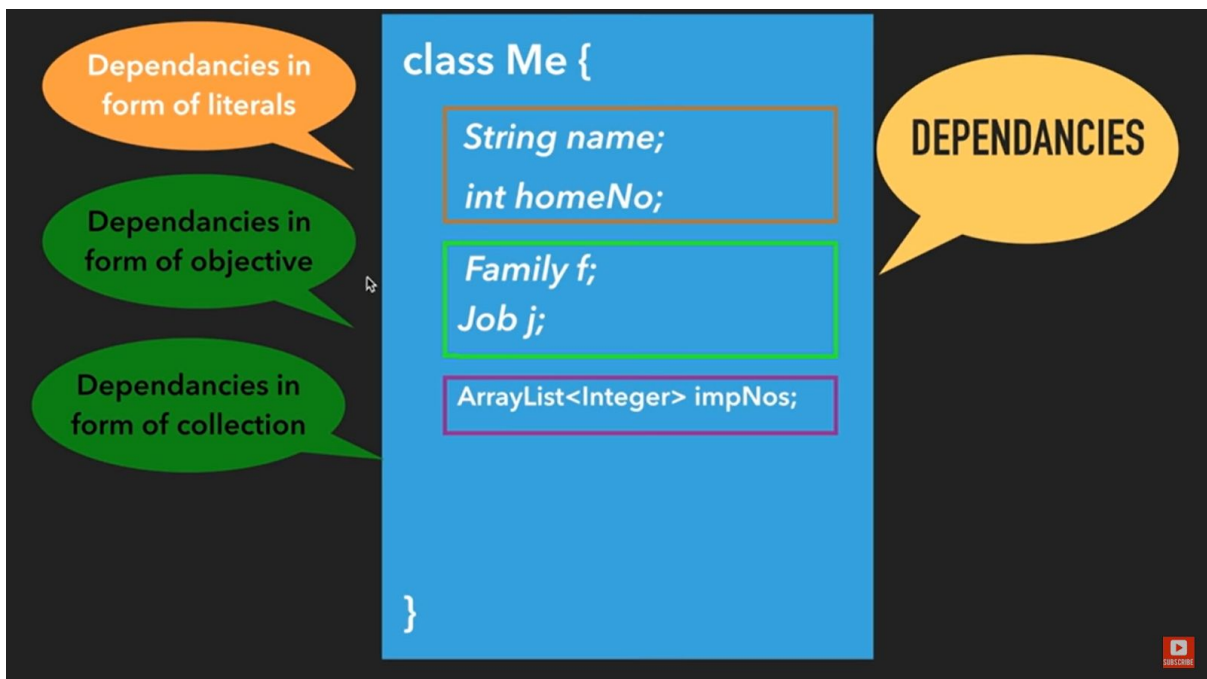
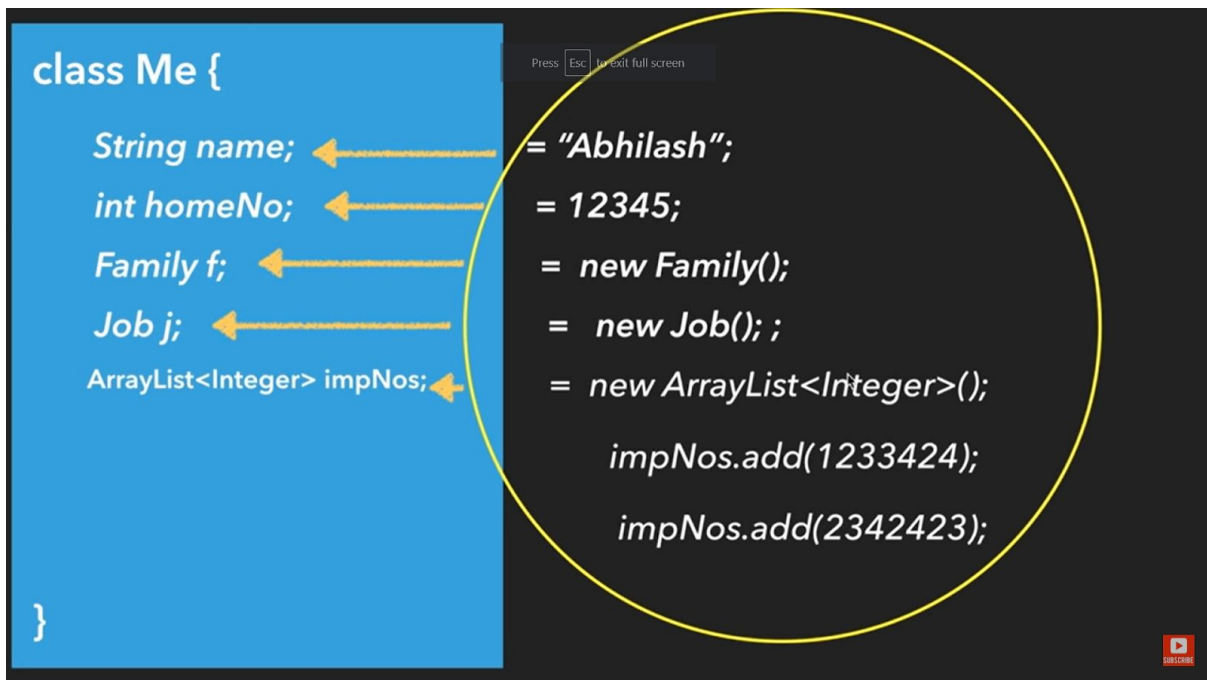
Job ;

Family; are dependancies

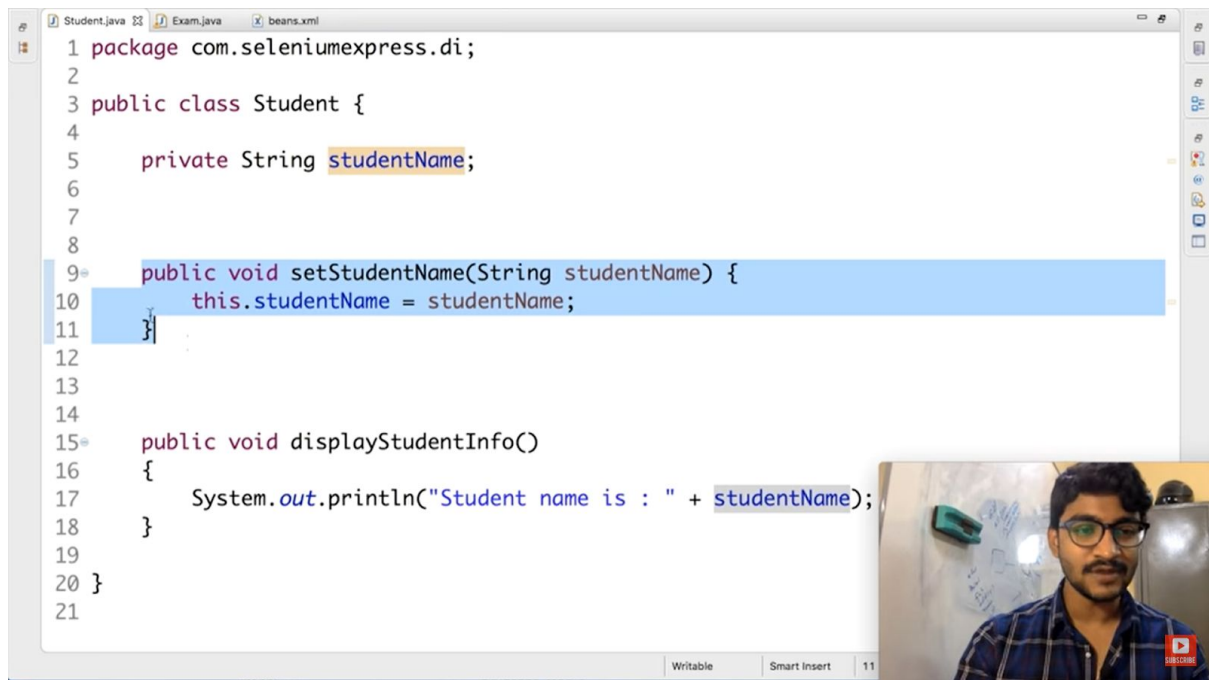
Rather than Hardcoding their values

Spring will help us to inject values to the dependency

Injection is done using SetterInjection and ConstructorInjection



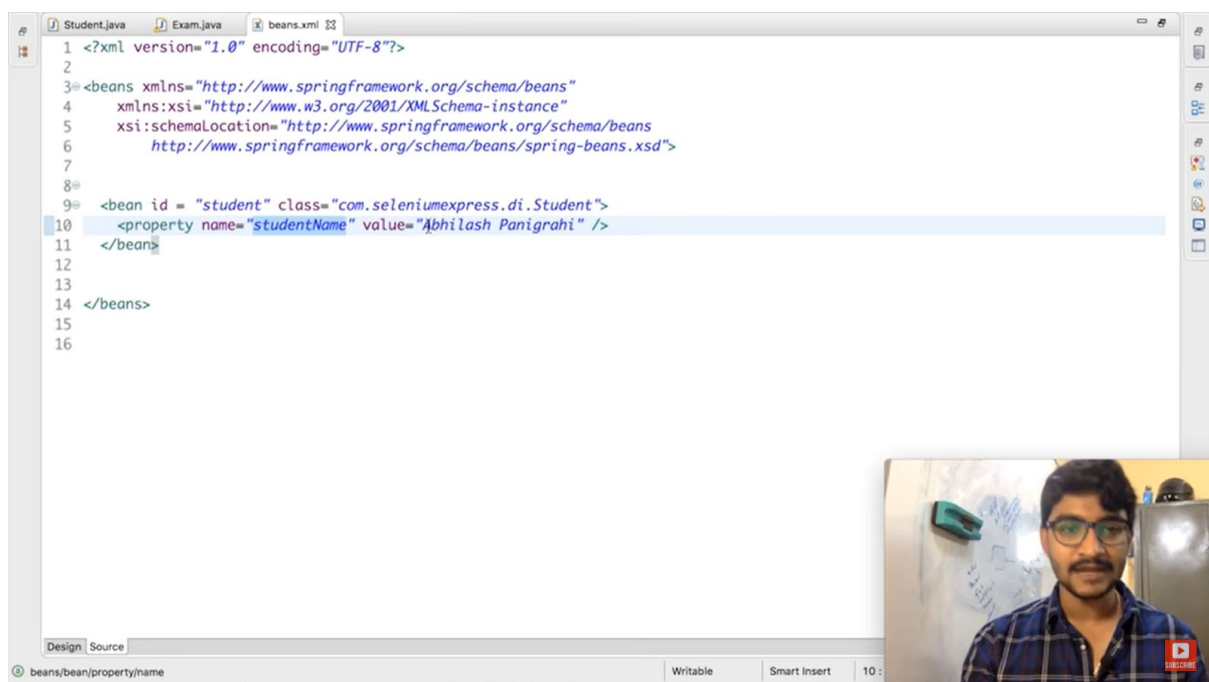
Dependancies in form of literals is known as PROPERTY



The screenshot shows an IDE with three tabs: Student.java, Exam.java, and beans.xml. The Student.java file contains the following code:

```
1 package com.seleniumexpress.di;  
2  
3 public class Student {  
4     private String studentName;  
5  
6  
7  
8  
9     public void setStudentName(String studentName) {  
10         this.studentName = studentName;  
11     }  
12  
13  
14  
15     public void displayStudentInfo()  
16     {  
17         System.out.println("Student name is : " + studentName);  
18     }  
19  
20 }  
21
```

A video inset in the bottom right corner shows a man with glasses and a beard, wearing a blue plaid shirt, speaking. A red 'SUBSCRIBE' button is visible in the bottom right corner of the video inset.



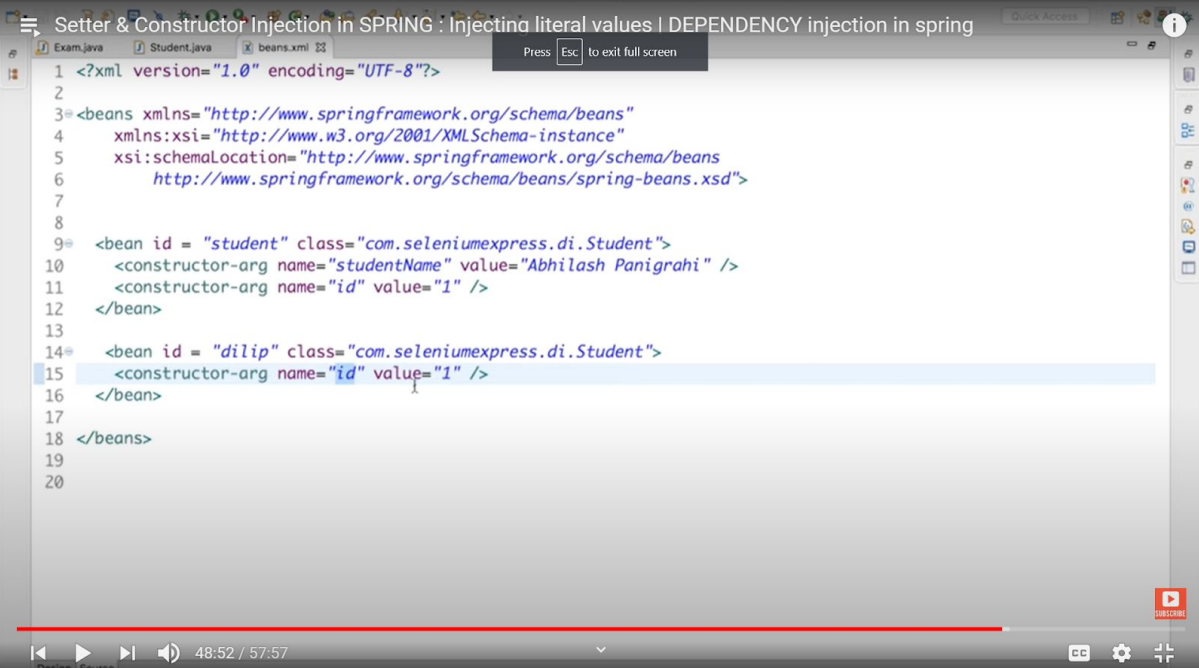
The screenshot shows an IDE with three tabs: Student.java, Exam.java, and beans.xml. The beans.xml file contains the following code:

```
1 <?xml version="1.0" encoding="UTF-8"?>  
2  
3 <beans xmlns="http://www.springframework.org/schema/beans"  
4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
5     xsi:schemaLocation="http://www.springframework.org/schema/beans  
6         http://www.springframework.org/schema/beans/spring-beans.xsd">  
7  
8  
9     <bean id="student" class="com.seleniumexpress.di.Student">  
10         <property name="studentName" value="Abhilash Panigrahi" />  
11     </bean>  
12  
13  
14 </beans>  
15  
16
```

A video inset in the bottom right corner shows the same man as in the first screenshot, speaking. A red 'SUBSCRIBE' button is visible in the bottom right corner of the video inset.

Here spring is using SETTER METHODS internally to inject values

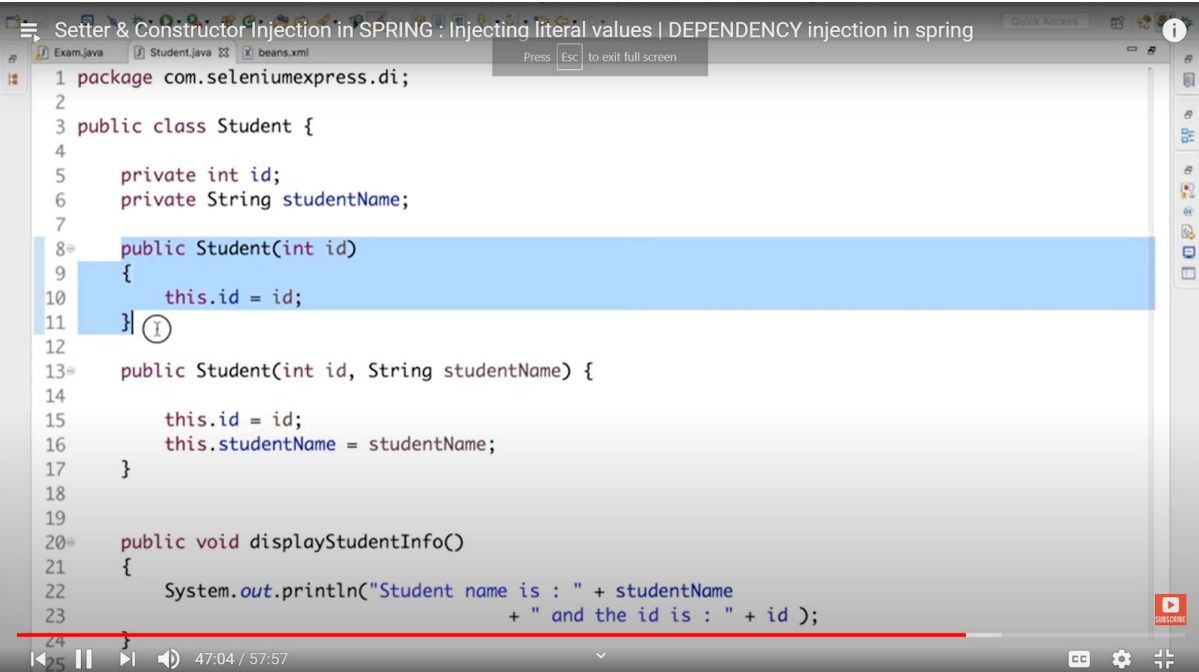
Constructor Injection



The screenshot shows a video player with a title bar that reads "Setter & Constructor Injection in SPRING : Injecting literal values | DEPENDENCY injection in spring". The video content displays an XML file named "beans.xml" with the following code:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <beans xmlns="http://www.springframework.org/schema/beans"
4       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5       xsi:schemaLocation="http://www.springframework.org/schema/beans
6       http://www.springframework.org/schema/beans/spring-beans.xsd">
7
8
9   <bean id="student" class="com.seleniumexpress.di.Student">
10     <constructor-arg name="studentName" value="Abhilash Panigrahi" />
11     <constructor-arg name="id" value="1" />
12   </bean>
13
14   <bean id="dilip" class="com.seleniumexpress.di.Student">
15     <constructor-arg name="id" value="1" />
16   </bean>
17
18 </beans>
19
20
```

The video player interface includes a progress bar at the bottom showing 48:52 / 57:57, a volume icon, and a "SUBSCRIBE" button in the bottom right corner.

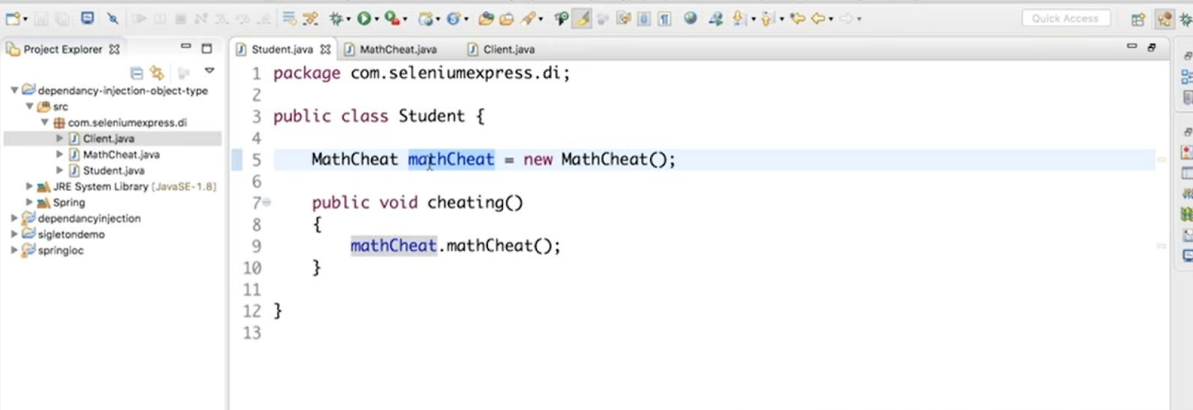


The screenshot shows a video player with the same title bar as the first image. The video content displays a Java file named "Student.java" with the following code:

```
1 package com.seleniumexpress.di;
2
3 public class Student {
4
5     private int id;
6     private String studentName;
7
8     public Student(int id)
9     {
10         this.id = id;
11     }
12
13     public Student(int id, String studentName) {
14
15         this.id = id;
16         this.studentName = studentName;
17     }
18
19
20     public void displayStudentInfo()
21     {
22         System.out.println("Student name is : " + studentName
23                             + " and the id is : " + id );
24     }
25
```

The video player interface includes a progress bar at the bottom showing 47:04 / 57:57, a volume icon, and a "SUBSCRIBE" button in the bottom right corner.

INJECTING Dependencies in form of objects

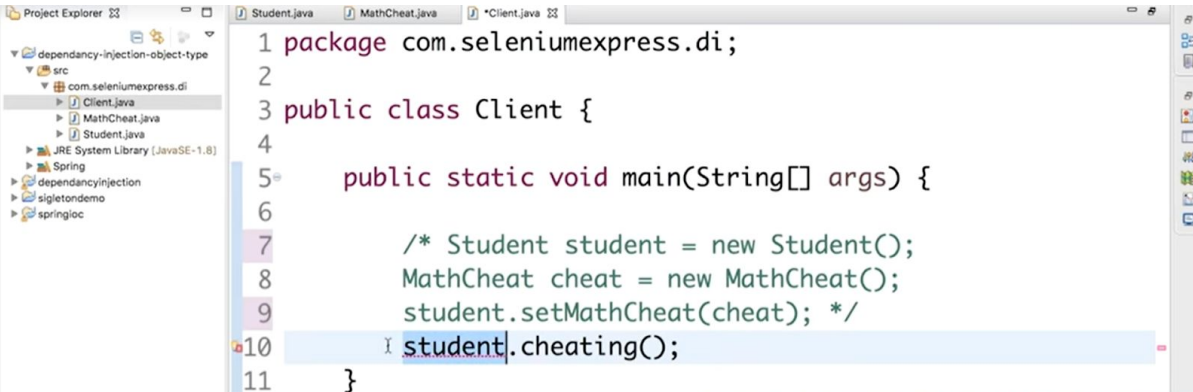


```
1 package com.seleniumexpress.di;
2
3 public class Student {
4
5     MathCheat mathCheat = new MathCheat();
6
7     public void cheating()
8     {
9         mathCheat.mathCheat();
10    }
11
12 }
13
```

how can I inject the value of "mathCheat" reference from a different class?

The commented code below

We expect Spring to do for us

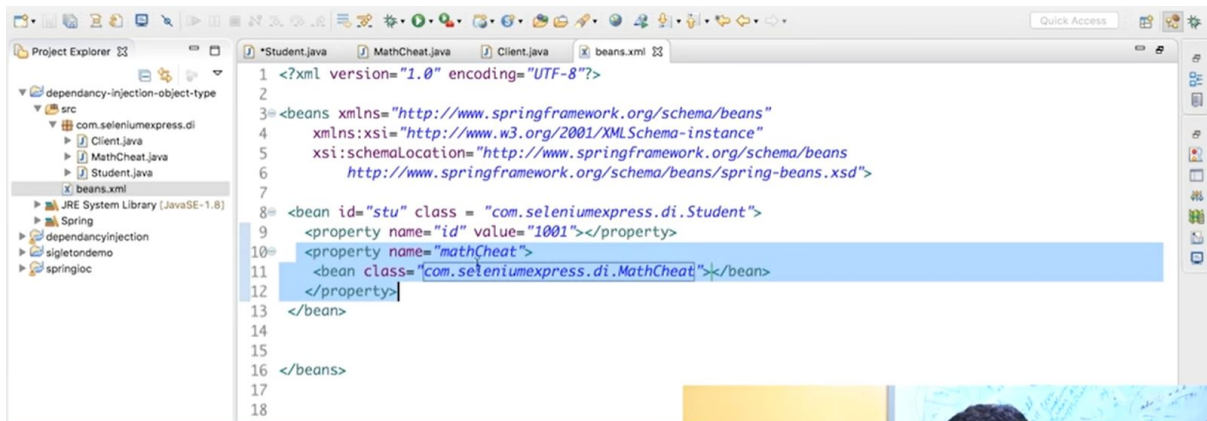


```
1 package com.seleniumexpress.di;
2
3 public class Client {
4
5     public static void main(String[] args) {
6
7         /* Student student = new Student();
8         MathCheat cheat = new MathCheat();
9         student.setMathCheat(cheat); */
10        student.cheating();
11    }
12
13 }
14
```

To tell Spring to do it there are ways

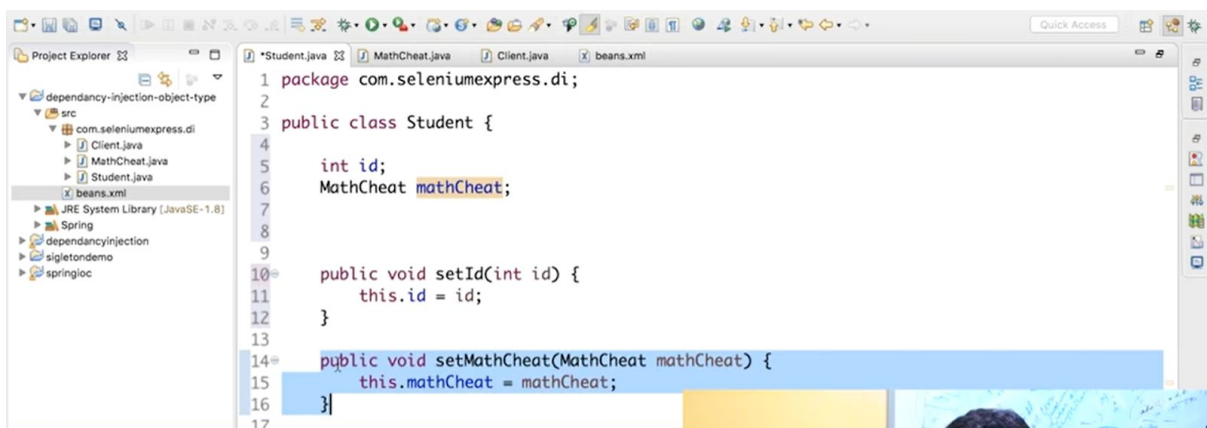
1. XML based

2. Annotations based



Internally

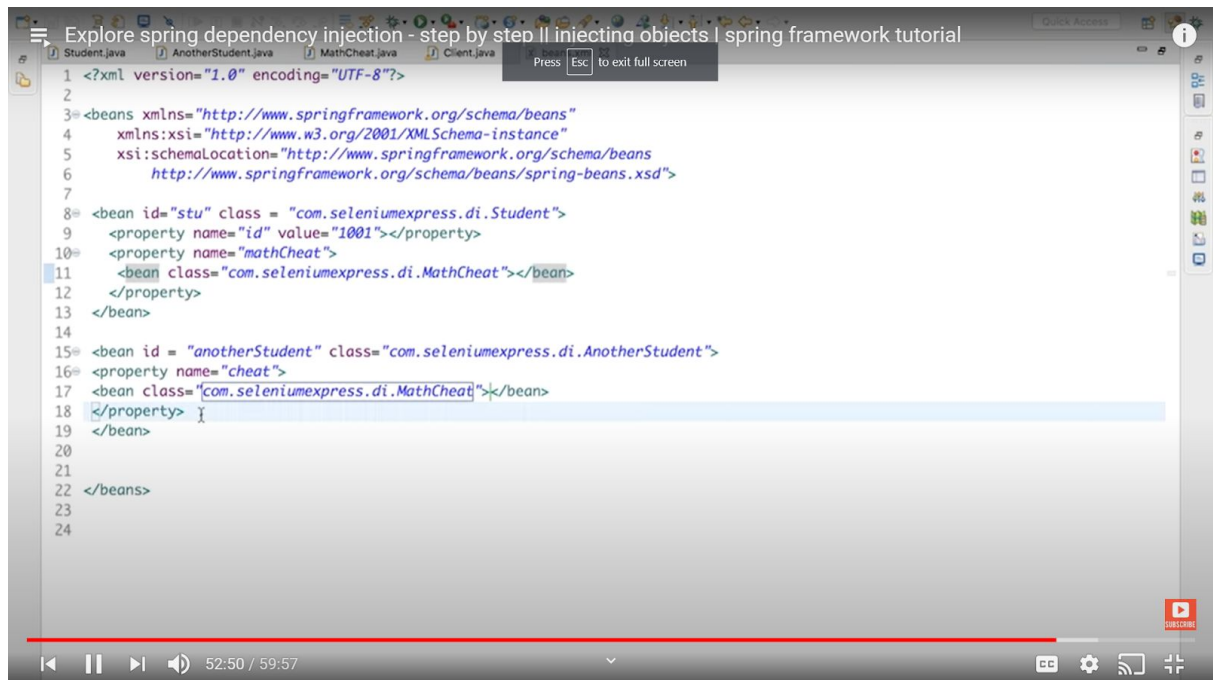
```
Student stu = new Student();
MathCheat mathCheat = new MathCheat();
stu.setId(1001);
stu.setMathCheat(mathCheat);
```



Internally

```
Student stu = new Student();
MathCheat mathCheat = new MathCheat();
stu.setId(1001);
stu.setMathCheat(mathCheat);
```


Similarly for 100 students 100 mathCheat objects

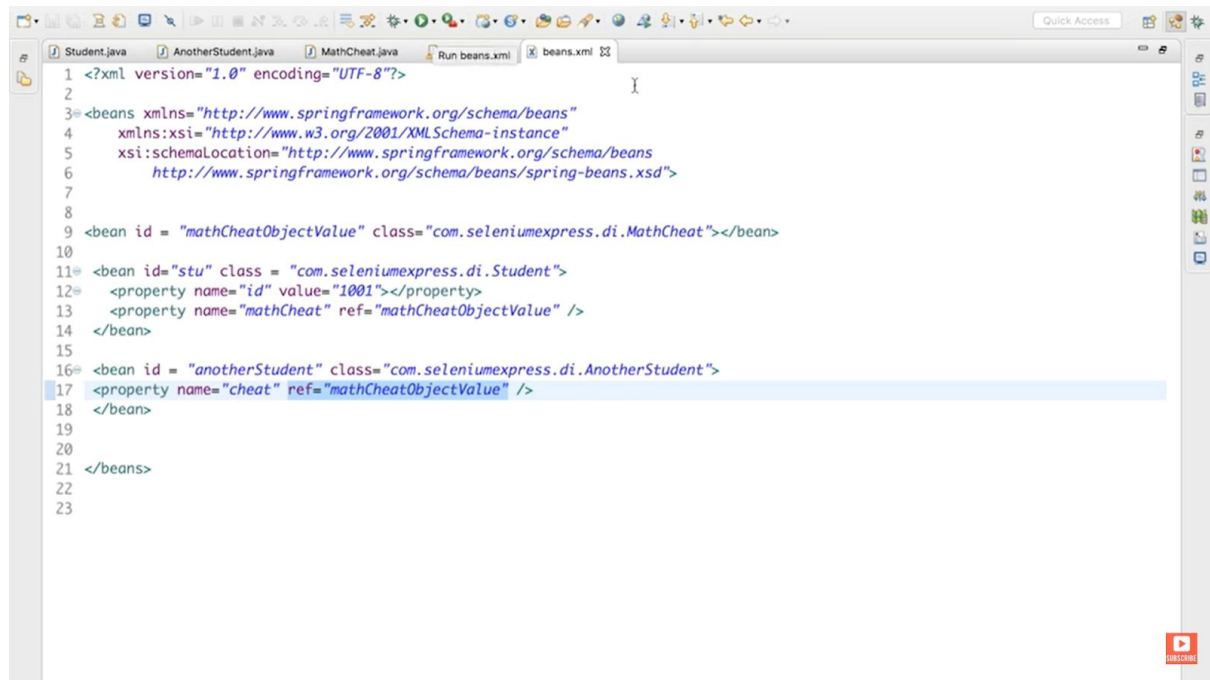


```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <beans xmlns="http://www.springframework.org/schema/beans"
4       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5       xsi:schemaLocation="http://www.springframework.org/schema/beans
6         http://www.springframework.org/schema/beans/spring-beans.xsd">
7
8   <bean id="stu" class="com.seleniumexpress.di.Student">
9     <property name="id" value="1001"></property>
10    <property name="mathCheat">
11      <bean class="com.seleniumexpress.di.MathCheat"></bean>
12    </property>
13  </bean>
14
15  <bean id="anotherStudent" class="com.seleniumexpress.di.AnotherStudent">
16    <property name="cheat">
17      <bean class="com.seleniumexpress.di.MathCheat"></bean>
18    </property>
19  </bean>
20
21 </beans>
22
23
24
```

Spring helps us to create light weight applications by creating objects only when needed

We create a one common object mathCheat

Overcoming above problem by creating one mathCheat object which will be used by any student class as and when needed



```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <beans xmlns="http://www.springframework.org/schema/beans"
4       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5       xsi:schemaLocation="http://www.springframework.org/schema/beans
6       http://www.springframework.org/schema/beans/spring-beans.xsd">
7
8
9 <bean id="mathCheatObjectValue" class="com.seleniumexpress.di.MathCheat"></bean>
10
11 <bean id="stu" class="com.seleniumexpress.di.Student">
12   <property name="id" value="1001"></property>
13   <property name="mathCheat" ref="mathCheatObjectValue" />
14 </bean>
15
16 <bean id="anotherStudent" class="com.seleniumexpress.di.AnotherStudent">
17   <property name="cheat" ref="mathCheatObjectValue" />
18 </bean>
19
20
21 </beans>
22
23
```

Spring helps in achieving

1. Dependency Injection
2. Loose Coupling

In Java we achieve Loose Coupling by Interface

Usage of new keyword denotes tight coupling