Inversion of Control (IOC)

Major Project by:-
1) Kashif Manzoor
2) Mohd Athar
3) Md. Umair Abdullah

Inversion of Control is a framework

Inversion of Control

A class should not configure its dependencies statically but should be configured by some other class from outside.

A class should concentrate on fulfilling its responsibilities like the flow of an application, and not on creating objects

Concept Behind Dependency Injection

Inversion of Control :: IOC

- IOC is a software behave as container.
- IOC is a concept through which it create object / instance of classes like (POJO / JAVA beans) and store all instances itself and give the reference of classes to the other classes where it calling (main method)
- It cannot create / make a instance more than 1 in IOC container (using factory method ~~using~~ with Singleton pattern) of one class
- It gives the reference of one classes to the one or more than one other classes

IoC Container Features

Creating the objects

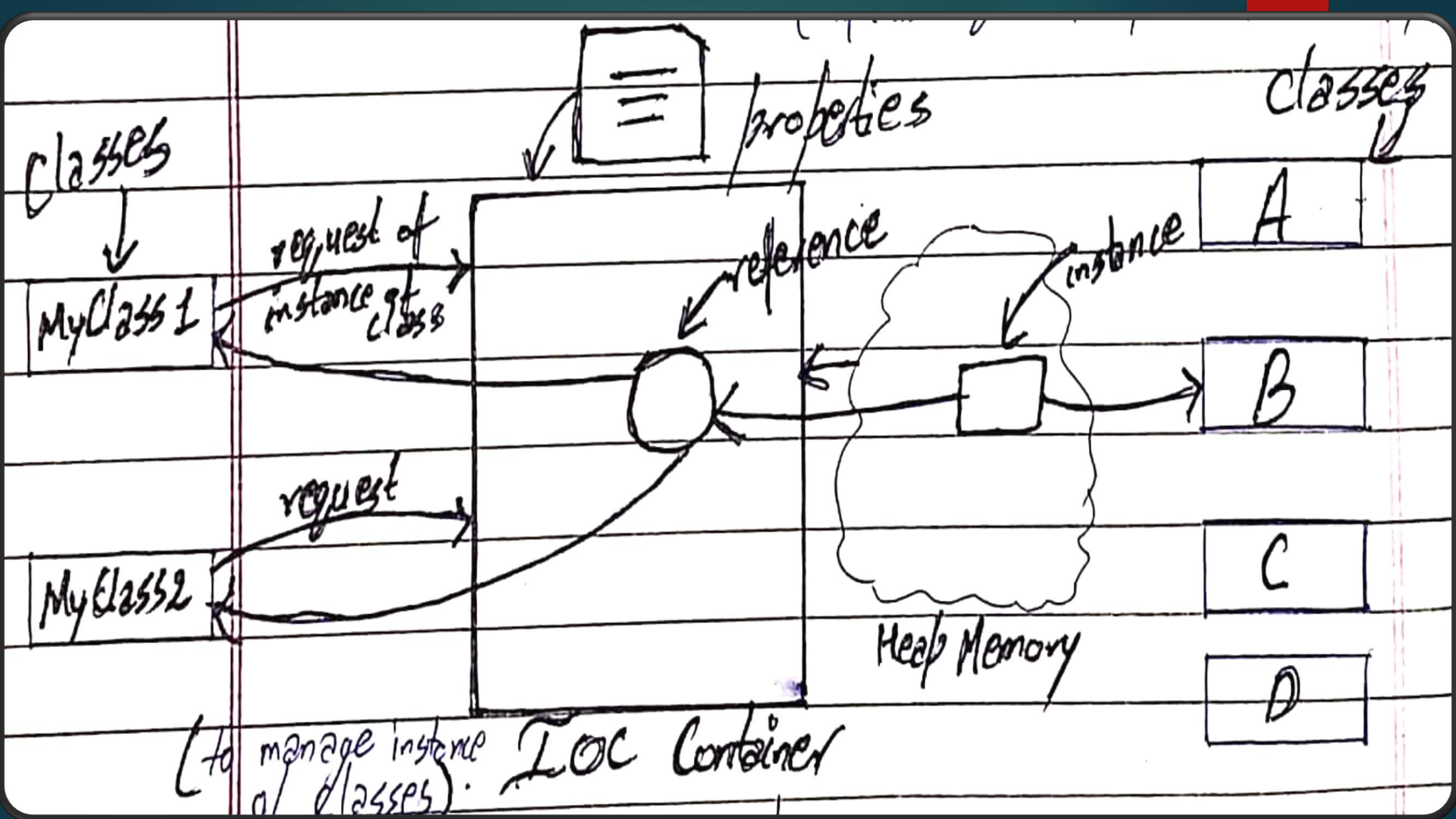
Wiring them together

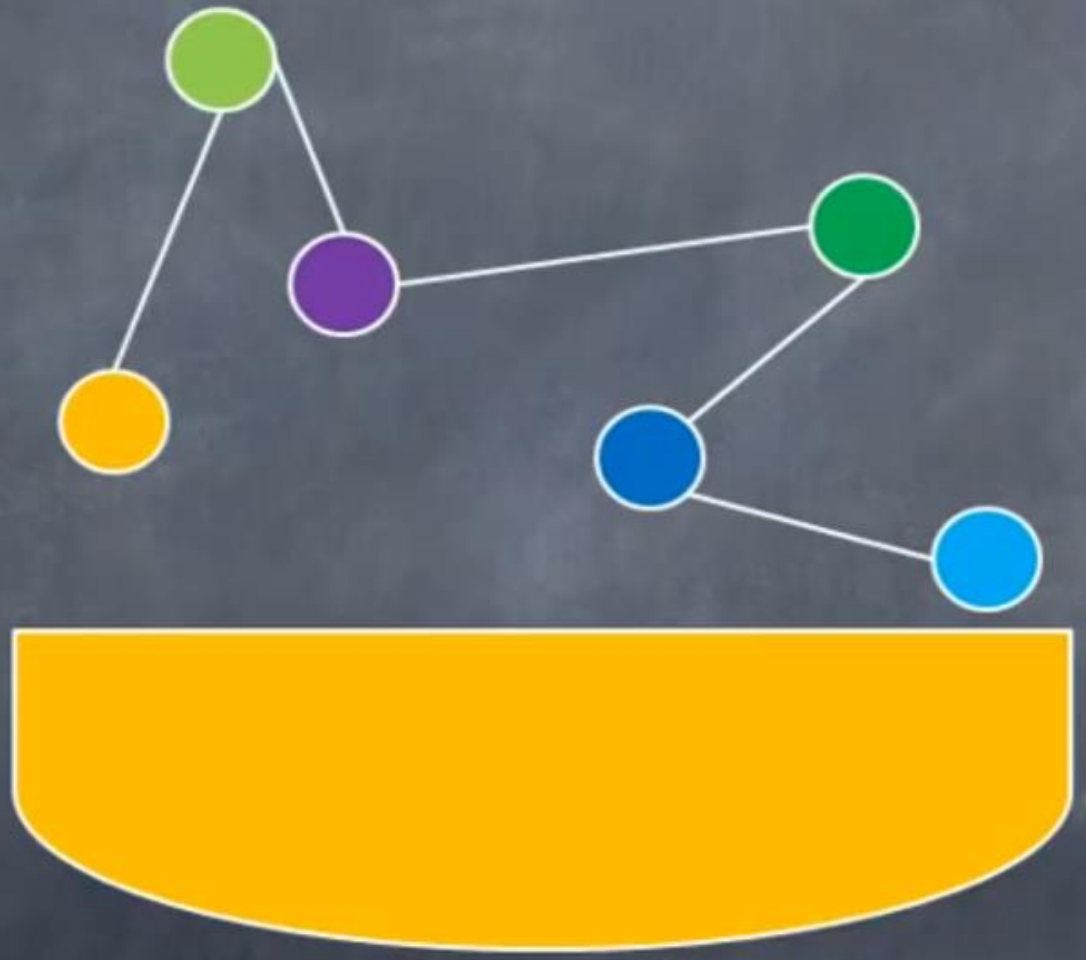
Managing their complete life cycle

Configuring

IOC

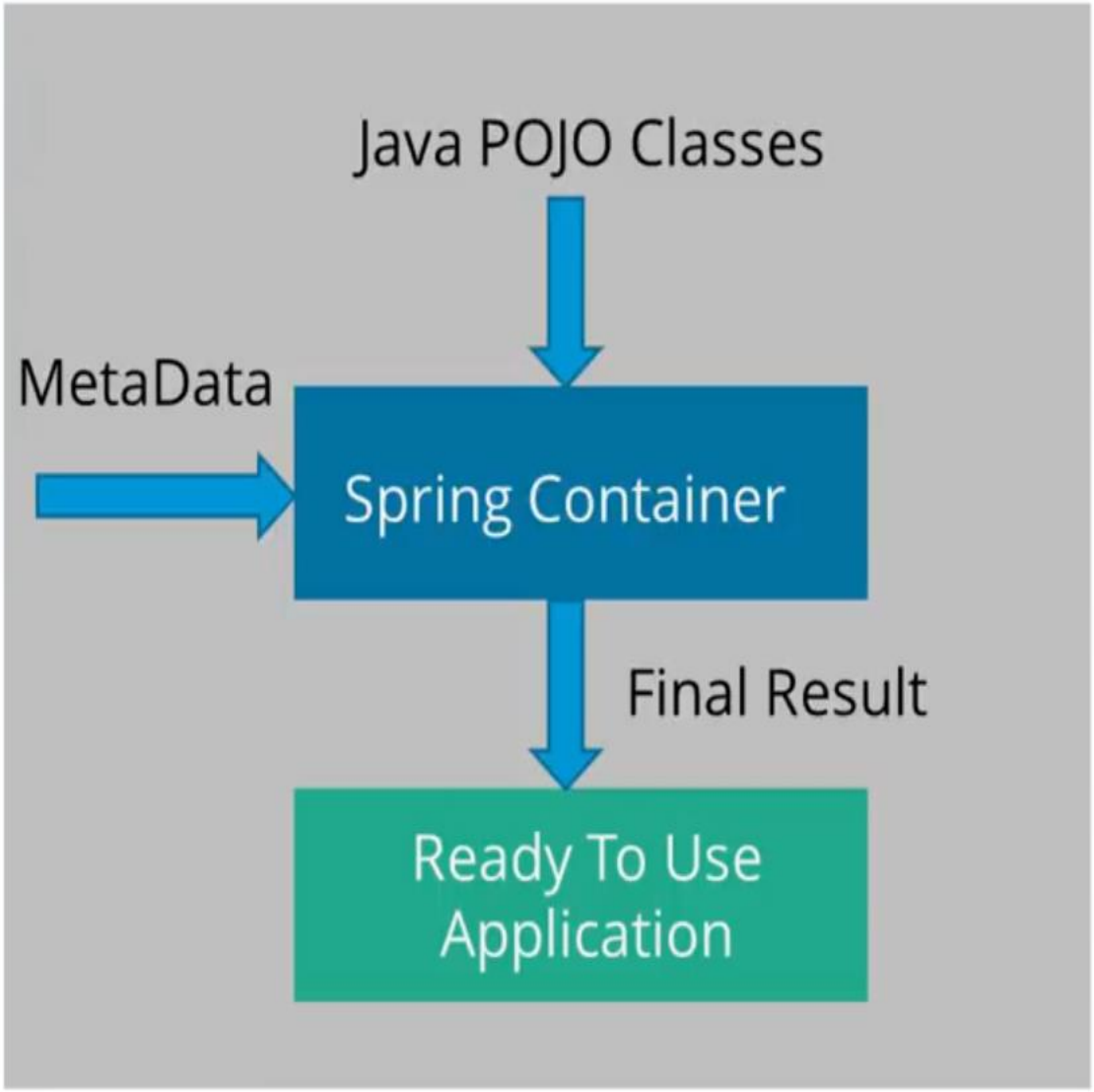
The Spring container is at the core of the Spring Framework. The container will create the objects, wire them together, configure them, and manage their complete lifecycle from creation till destruction


properties
Classes
MyClass1
MyClass2
request of instance of class
request
reference
instance
Heap Memory
Classes
A
B
C
D
(to manage instance of classes) IOC Container

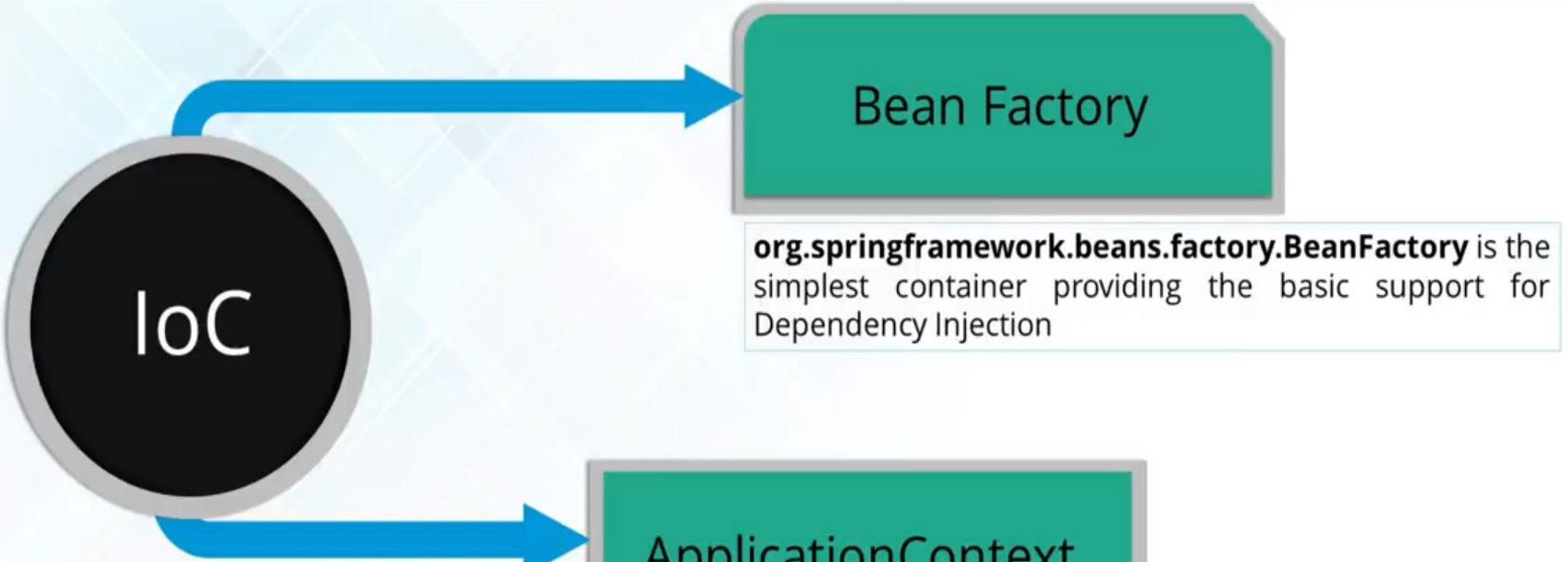


IoC Container Features

The Spring IoC container by using Java POJO classes and configuration metadata produces a fully configured and executable system or application.



Types Of IoC Container



org.springframework.beans.factory.BeanFactory is the simplest container providing the basic support for Dependency Injection

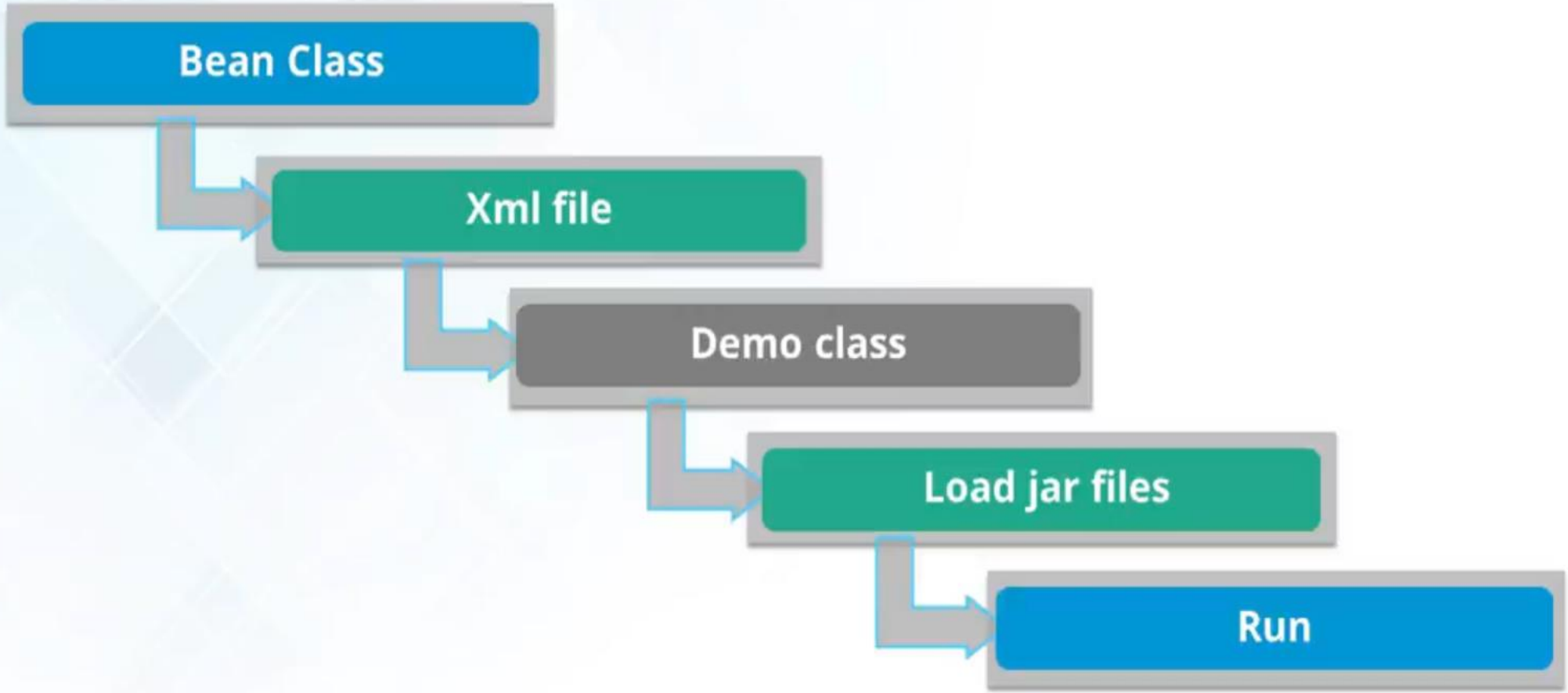
org.springframework.context.ApplicationContext built on top of the BeanFactory interface. This container adds more enterprise-specific functionality

Uses :→

Inversion of control is used to increase modularity of the program and make it extensible and has applications in object-oriented programming and other programming paradigms.

with decoupling dependency between high-level and low-level layers through shared abstractions.

Start Coding in 5 Simple Steps

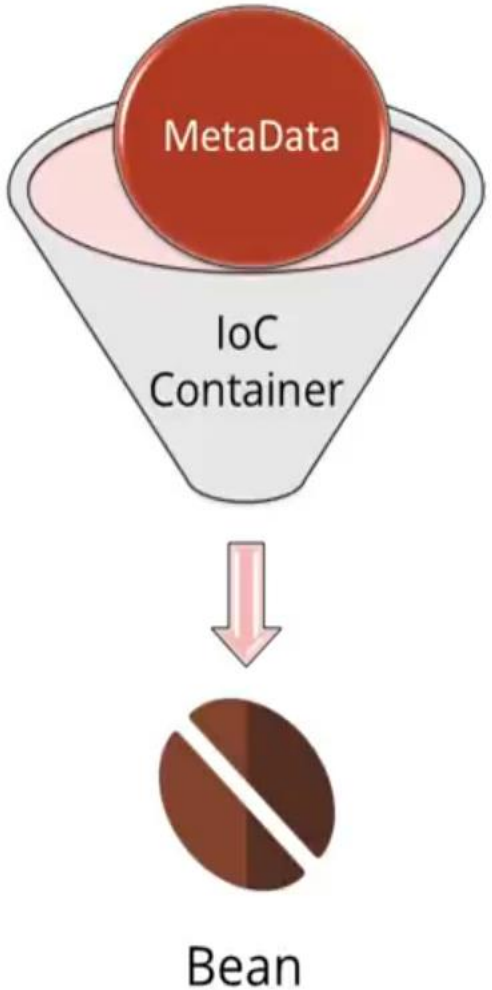


Bean Object

1. Beans are the objects that form the backbone of our application and are managed by the Spring IoC container.

2. Spring IoC container instantiates, assembles, and manages the bean object

3. The configuration metadata that are supplied to the container are used create **Beans** object



Step1: Programmer create a package with classes and a properties file.

Step2: This properties file contains information about independent classes and their dependent classes.

Step3: When programmer use IOC container by specifing related properties file for the package.

Step4: IOC container methods will be invoked and create instances of independent and their dependent class and ~~tete~~ then inject references of dependent classes to independent classes by calling their constructors.

Creating the Bean class

- **Bean Class**
- Xml file
- Demo Class
- Load jar files
- Run

package-info.java | StudentDemo.java | StudentConfig.xml | *StudentBean.java

```java
package org.edureka.firstSpring;

public class StudentBean
{
    String name;
    public String getName()
    {
      return name;
    }
    public void setName(String name)
    {
      this.name = name;
    }
    public void displayInfo()
    {
      System.out.println("Hello: "+ name);
    }
}
```

getter() method

setter() method

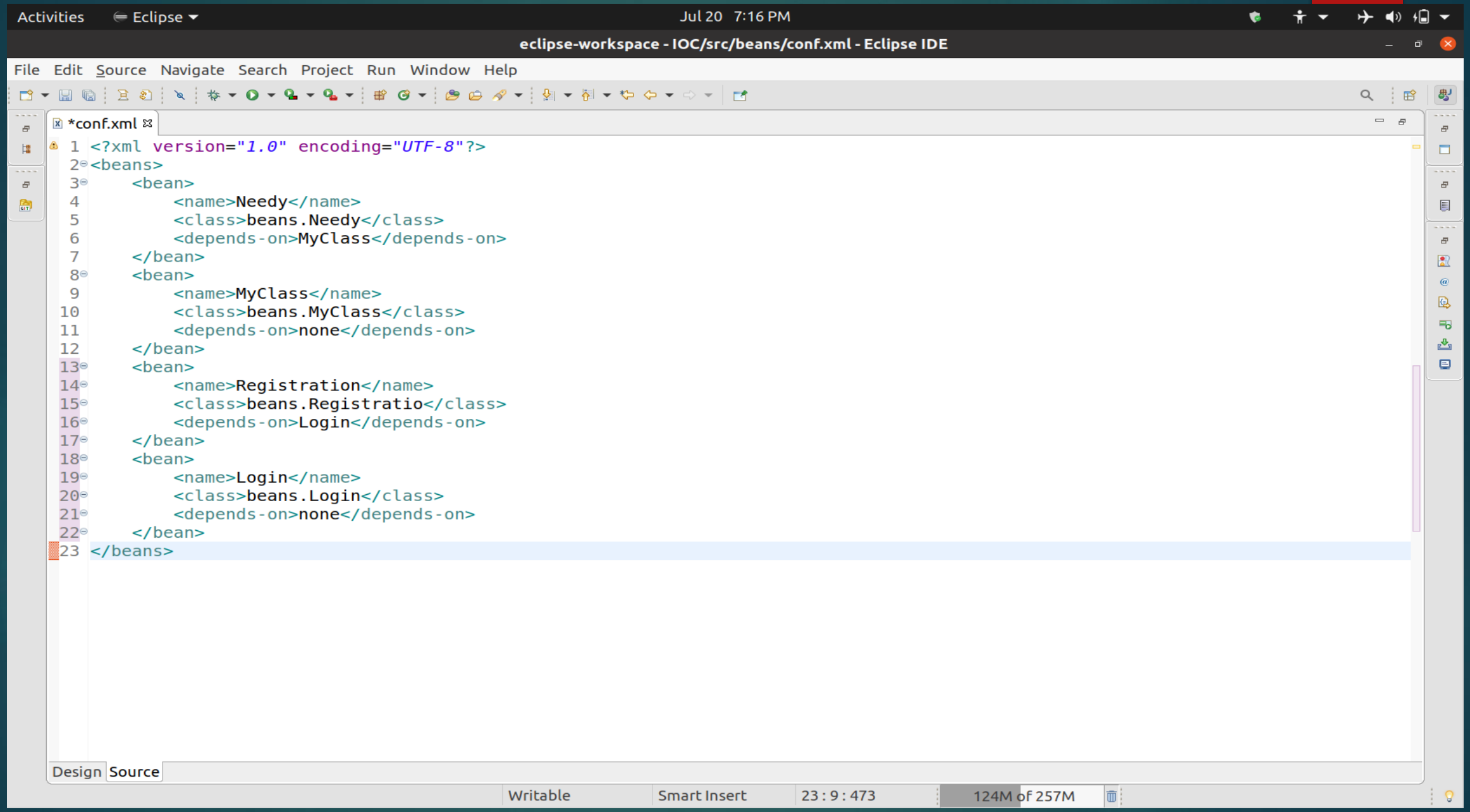
Creating the xml File

- Bean Class
- **Xml file**
- Demo Class
- Load jar files
- Run

package-info.java | *StudentDemo.java | *StudentConfig.xml | *StudentBean.java

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans
3     xmlns="http://www.springframework.org/schema/beans"
4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5     xmlns:p="http://www.springframework.org/schema/p"
6     xsi:schemaLocation="http://www.springframework.org/schema/beans
7           http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
8
9
10 <bean id="studentbean" class="org/edureka/firstSpring/StudentBean">
11 <property name="name" value="Edureka"></property>
12 </bean>
13 </beans>
```

Providing a bean
To IoC container


Activities
Eclipse
Jul 20 7:16 PM
eclipse-workspace - IOC/src/beans/conf.xml - Eclipse IDE
File Edit Source Navigate Search Project Run Window Help
*conf.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans>
3 <bean>
4 <name>Needy</name>
5 <class>beans.Needy</class>
6 <depends-on>MyClass</depends-on>
7 </bean>
8 <bean>
9 <name>MyClass</name>
10 <class>beans.MyClass</class>
11 <depends-on>none</depends-on>
12 </bean>
13 <bean>
14 <name>Registration</name>
15 <class>beans.Registratio</class>
16 <depends-on>Login</depends-on>
17 </bean>
18 <bean>
19 <name>Login</name>
20 <class>beans.Login</class>
21 <depends-on>none</depends-on>
22 </bean>
23 </beans>
Design Source
Writable Smart Insert 23 : 9 : 473 124M of 257M

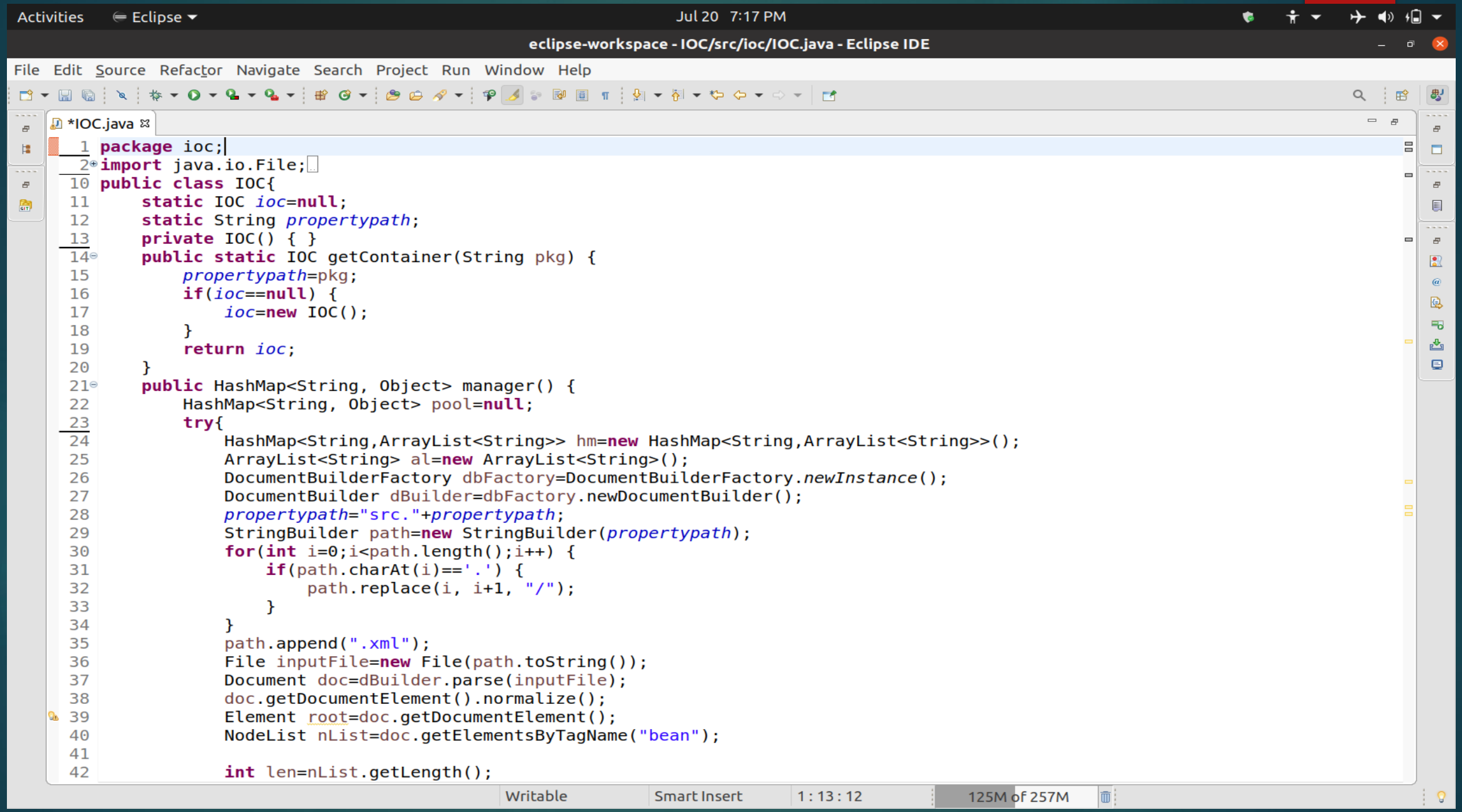
Creating the Demo class

- Bean Class
- Xml file
- **Demo Class**
- Load jar files
- Run

package-info.java | *StudentDemo.java | StudentConfig.xml | *StudentBean.java

```
1  package org.edureka.firstSpring;
2  import org.springframework.context.ApplicationContext;
3  import org.springframework.context.support.ClassPathXmlApplicationContext;
4  public class StudentDemo
5  {
6    public static void main(String[] args)
7    {
8      ApplicationContext appCon=new ClassPathXmlApplicationContext("StudentConfig.xml");
9      StudentBean factory=(StudentBean)appCon.getBean("studentbean");
10     factory.displayInfo();
11   }
12 }
13
```

Injecting a bean file


Activities
Eclipse
Jul 20 7:17 PM
eclipse-workspace - IOC/src/ioc/IOC.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*IOC.java
1 package ioc;
2 import java.io.File;
10 public class IOC{
11 static IOC ioc=null;
12 static String propertypath;
13 private IOC() { }
14 public static IOC getContainer(String pkg) {
15 propertypath=pkg;
16 if(ioc==null) {
17 ioc=new IOC();
18 }
19 return ioc;
20 }
21 public HashMap<String, Object> manager() {
22 HashMap<String, Object> pool=null;
23 try{
24 HashMap<String,ArrayList<String>> hm=new HashMap<String,ArrayList<String>>();
25 ArrayList<String> al=new ArrayList<String>();
26 DocumentBuilderFactory dbFactory=DocumentBuilderFactory.newInstance();
27 DocumentBuilder dBuilder=dbFactory.newDocumentBuilder();
28 propertypath="src."+propertypath;
29 StringBuilder path=new StringBuilder(propertypath);
30 for(int i=0;i<path.length();i++) {
31 if(path.charAt(i)=='.') {
32 path.replace(i, i+1, "/");
33 }
34 }
35 path.append(".xml");
36 File inputFile=new File(path.toString());
37 Document doc=dBuilder.parse(inputFile);
38 doc.getDocumentElement().normalize();
39 Element root=doc.getDocumentElement();
40 NodeList nList=doc.getElementsByTagName("bean");
41
42 int len=nList.getLength();
Writable
Smart Insert
1 : 13 : 12
125M of 257M

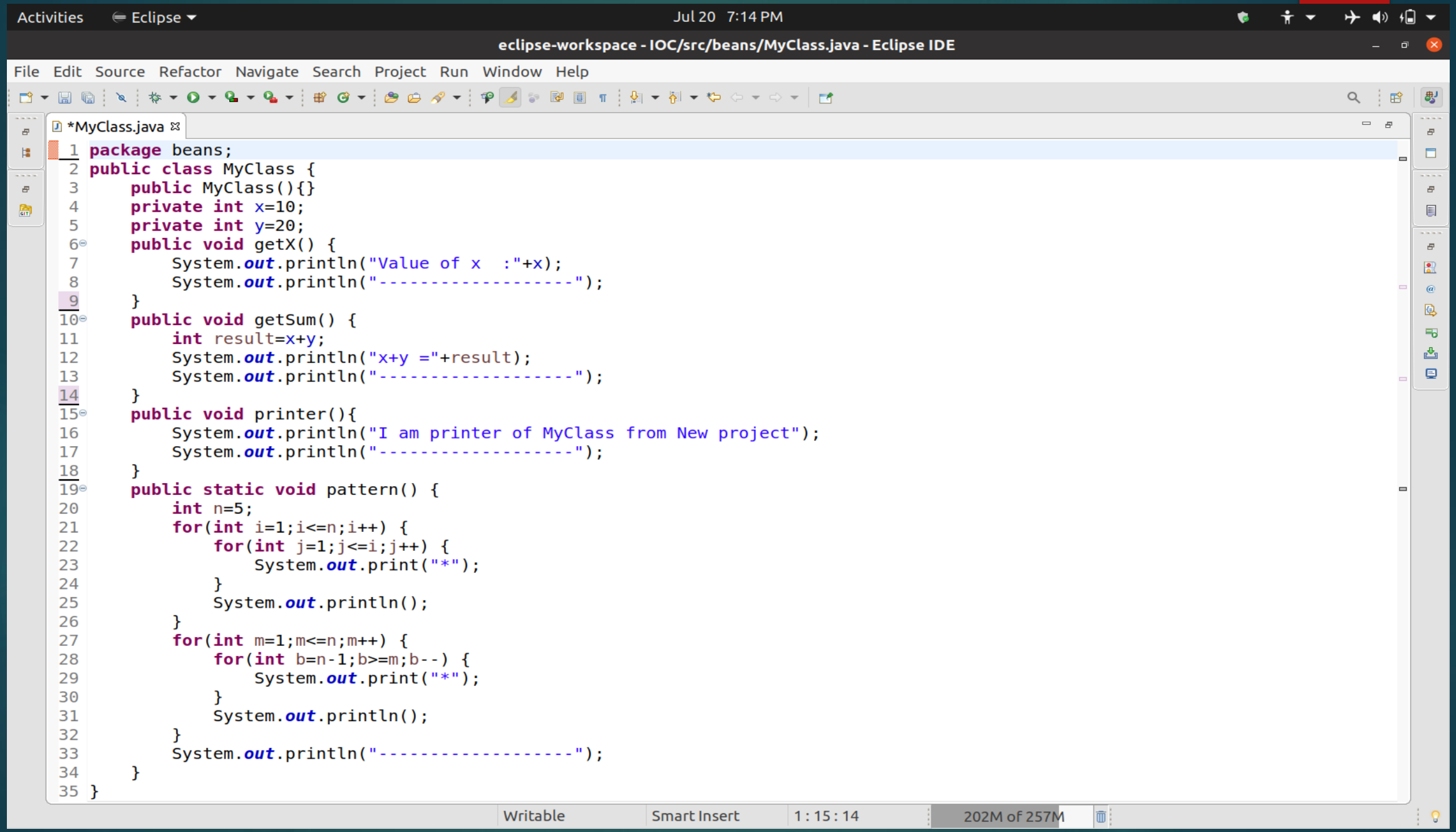
Now we understand IOC in terms of code.

1) Firstly object created by developer.

2) Secondly object created by IOC itself and preserve in container and give reference to required class.

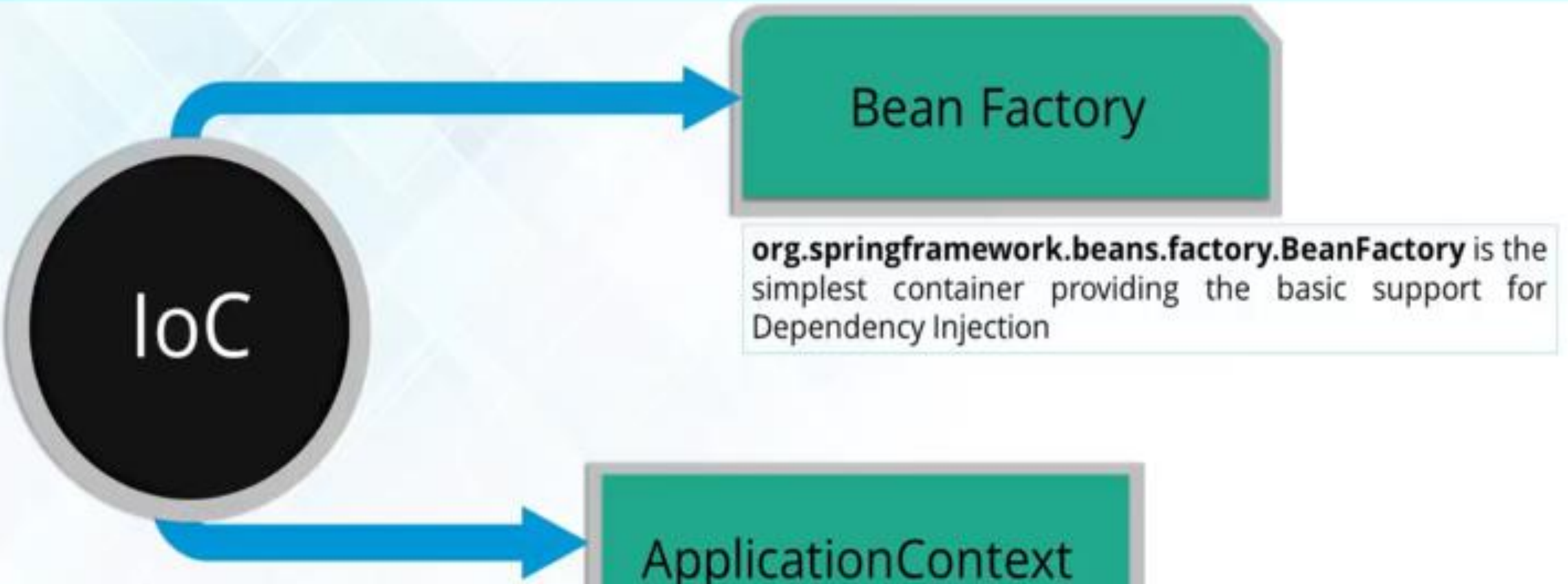
Object created by developer

```
class Employee{

    // Attributes
    int eid;
    String ename;
    String address;
    char gender;

    // Methods
    //...
}

Object Construction

Employee eRef = new Employee();
eRef.eid = 101;
eRef.ename = "John Watson";
eRef.address = "Redwood Shores";
eRef.gender = 'M';

Spring
Core | IOC (Inversion of Control)

You dont create objects.
objects shall be configured in an XML file by the develop
```


Activities
Eclipse
Jul 20 7:14 PM
eclipse-workspace - IOC/src/beans/MyClass.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*MyClass.java
package beans;
public class MyClass {
public MyClass(){}
private int x=10;
private int y=20;
public void getX() {
System.out.println("Value of x :"+x);
System.out.println("--------------------");
}
public void getSum() {
int result=x+y;
System.out.println("x+y ="+result);
System.out.println("--------------------");
}
public void printer(){
System.out.println("I am printer of MyClass from New project");
System.out.println("--------------------");
}
public static void pattern() {
int n=5;
for(int i=1;i<=n;i++) {
for(int j=1;j<=i;j++) {
System.out.print("*");
}
System.out.println();
}
for(int m=1;m<=n;m++) {
for(int b=n-1;b>=m;b--) {
System.out.print("*");
}
System.out.println();
}
System.out.println("--------------------");
}
}
Writable
Smart Insert
1 : 15 : 14
202M of 257M

Object created by IOC itself

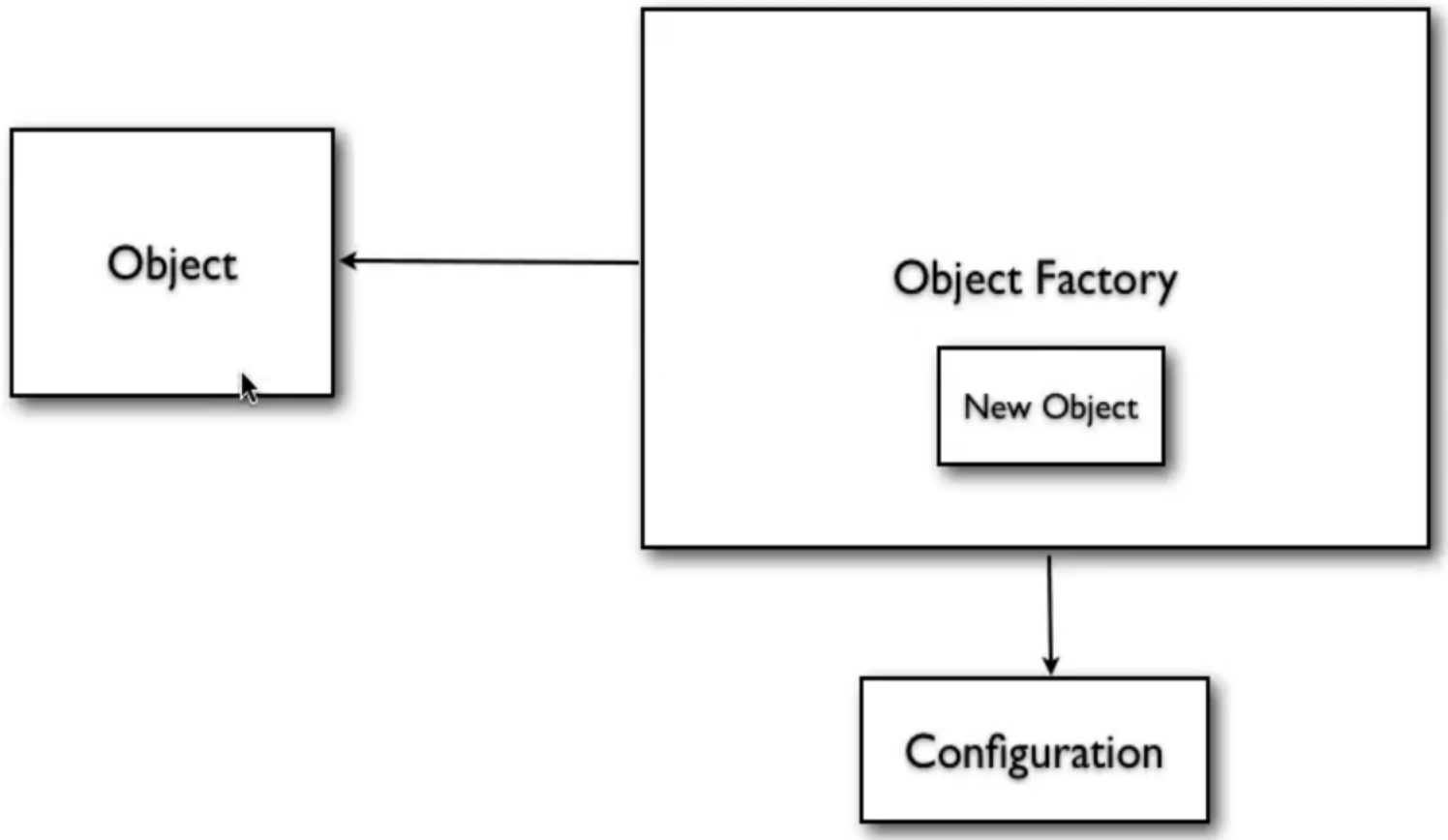
Types Of IoC Container



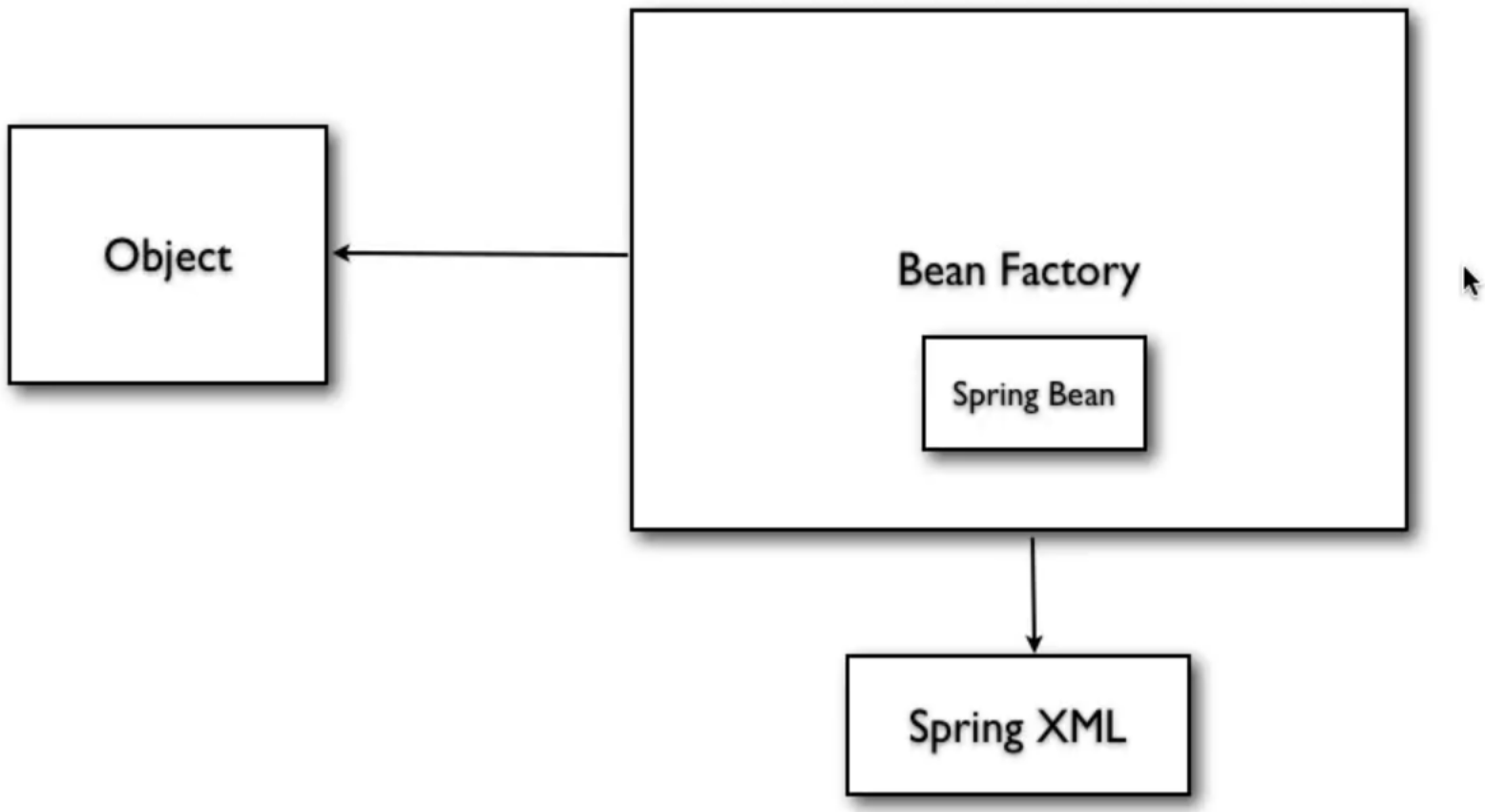
org.springframework.beans.factory.BeanFactory is the simplest container providing the basic support for Dependency Injection

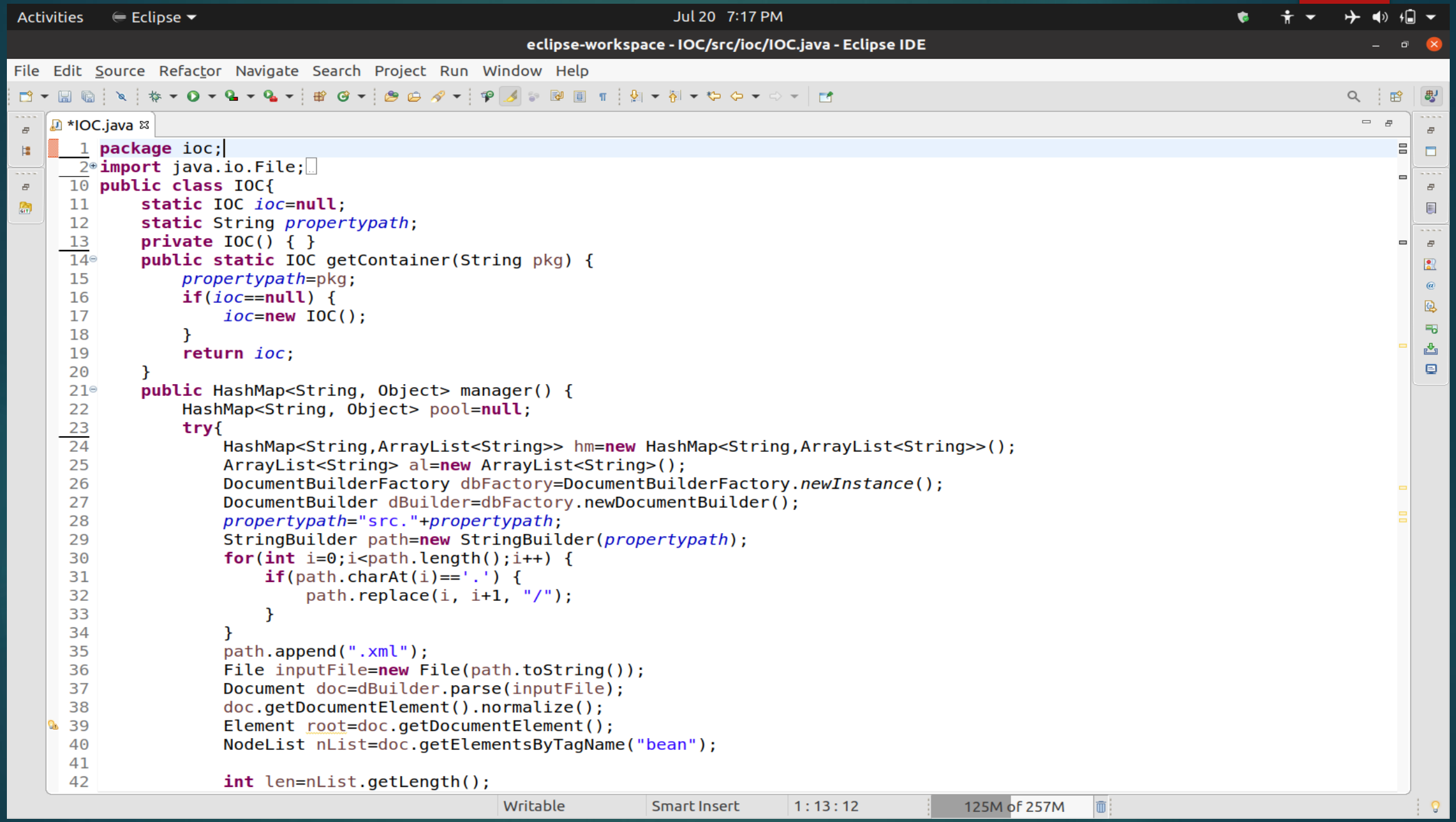
org.springframework.context.ApplicationContext built on top of the BeanFactory interface. This container adds more enterprise-specific functionality

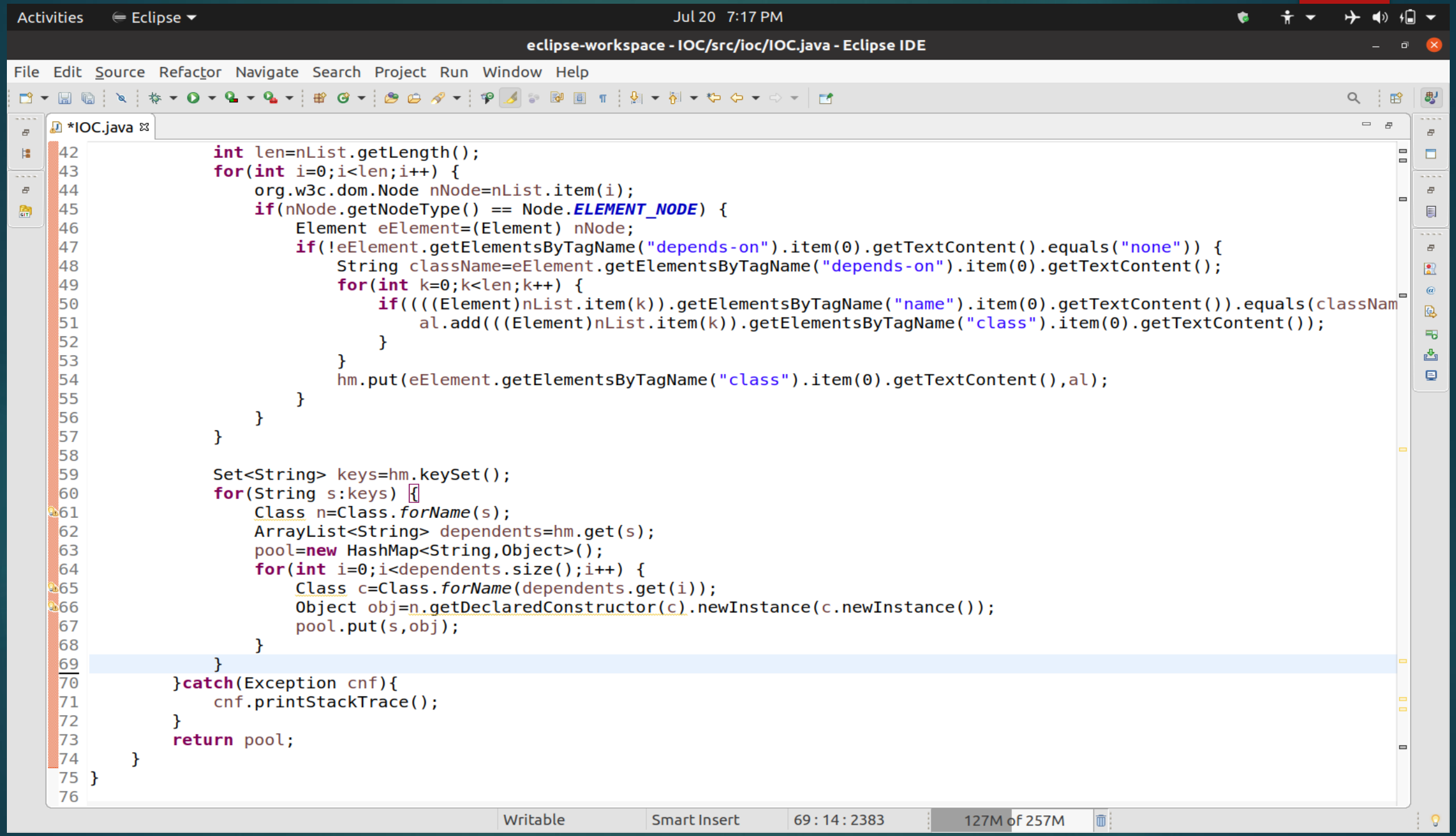
Factory Pattern



Spring Bean Factory




Activities
Eclipse
Jul 20 7:17 PM
eclipse-workspace - IOC/src/ioc/IOC.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*IOC.java
package ioc;
import java.io.File;
public class IOC{
static IOC ioc=null;
static String propertypath;
private IOC() { }
public static IOC getContainer(String pkg) {
propertypath=pkg;
if(ioc==null) {
ioc=new IOC();
}
return ioc;
}
public HashMap<String, Object> manager() {
HashMap<String, Object> pool=null;
try{
HashMap<String,ArrayList<String>> hm=new HashMap<String,ArrayList<String>>();
ArrayList<String> al=new ArrayList<String>();
DocumentBuilderFactory dbFactory=DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder=dbFactory.newDocumentBuilder();
propertypath="src."+propertypath;
StringBuilder path=new StringBuilder(propertypath);
for(int i=0;i<path.length();i++) {
if(path.charAt(i)=='.') {
path.replace(i, i+1, "/");
}
}
path.append(".xml");
File inputFile=new File(path.toString());
Document doc=dBuilder.parse(inputFile);
doc.getDocumentElement().normalize();
Element root=doc.getDocumentElement();
NodeList nList=doc.getElementsByTagName("bean");
int len=nList.getLength();
Writable
Smart Insert
1 : 13 : 12
125M of 257M


Activities
Eclipse
Jul 20 7:17 PM
eclipse-workspace - IOC/src/ioc/IOC.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*IOC.java
int len=nList.getLength();
for(int i=0;i<len;i++) {
org.w3c.dom.Node nNode=nList.item(i);
if(nNode.getNodeType() == Node.ELEMENT_NODE) {
Element eElement=(Element) nNode;
if(!eElement.getElementsByTagName("depends-on").item(0).getTextContent().equals("none")) {
String className=eElement.getElementsByTagName("depends-on").item(0).getTextContent();
for(int k=0;k<len;k++) {
if((((Element)nList.item(k)).getElementsByTagName("name").item(0).getTextContent()).equals(classNam
al.add(((Element)nList.item(k)).getElementsByTagName("class").item(0).getTextContent());
}
}
hm.put(eElement.getElementsByTagName("class").item(0).getTextContent(),al);
}
}
}
Set<String> keys=hm.keySet();
for(String s:keys) {
Class n=Class.forName(s);
ArrayList<String> dependents=hm.get(s);
pool=new HashMap<String,Object>();
for(int i=0;i<dependents.size();i++) {
Class c=Class.forName(dependents.get(i));
Object obj=n.getDeclaredConstructor(c).newInstance(c.newInstance());
pool.put(s,obj);
}
}
}catch(Exception cnf){
cnf.printStackTrace();
}
return pool;
}
}
Writable
Smart Insert
69 : 14 : 2383
127M of 257M

What is Dependency Injection ?

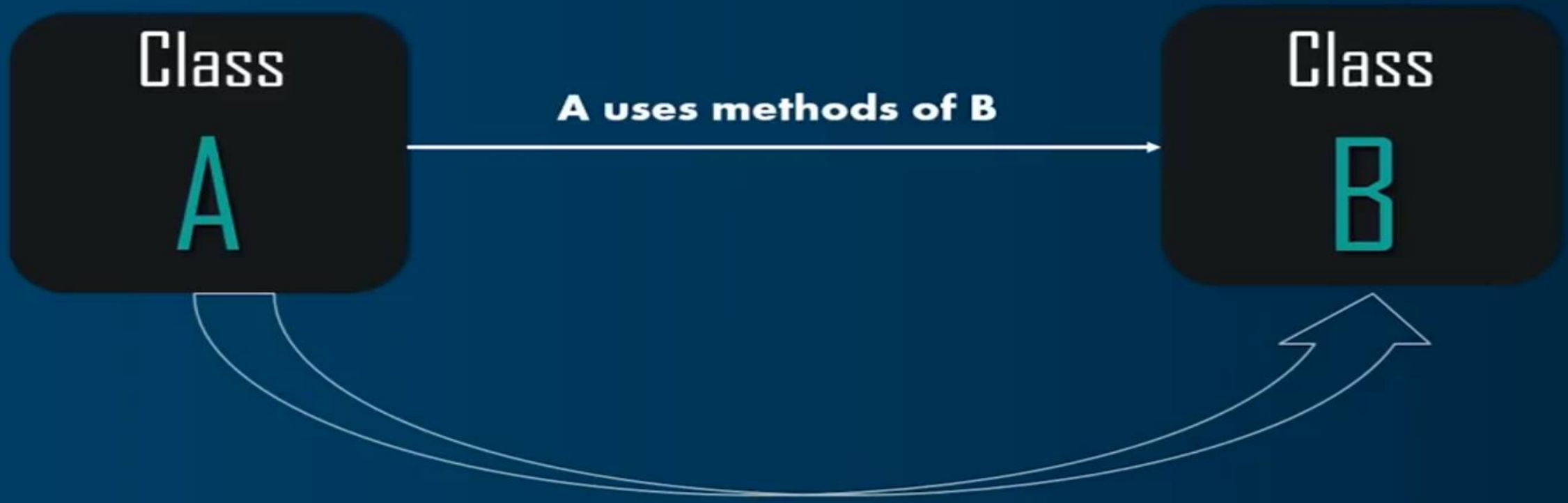
Dependency Injection

1. It is a design pattern which removes the dependency from the programming code, that makes the Application easy to manage and test.

2. Dependency Injection makes our programming code loosely coupled, which means change in implementation doesn't affects the user.

Dependency Injection (DI) is a design pattern that removes the dependency from the programming code so that it can be easy to manage and test the application. Dependency Injection makes our programming code loosely coupled.

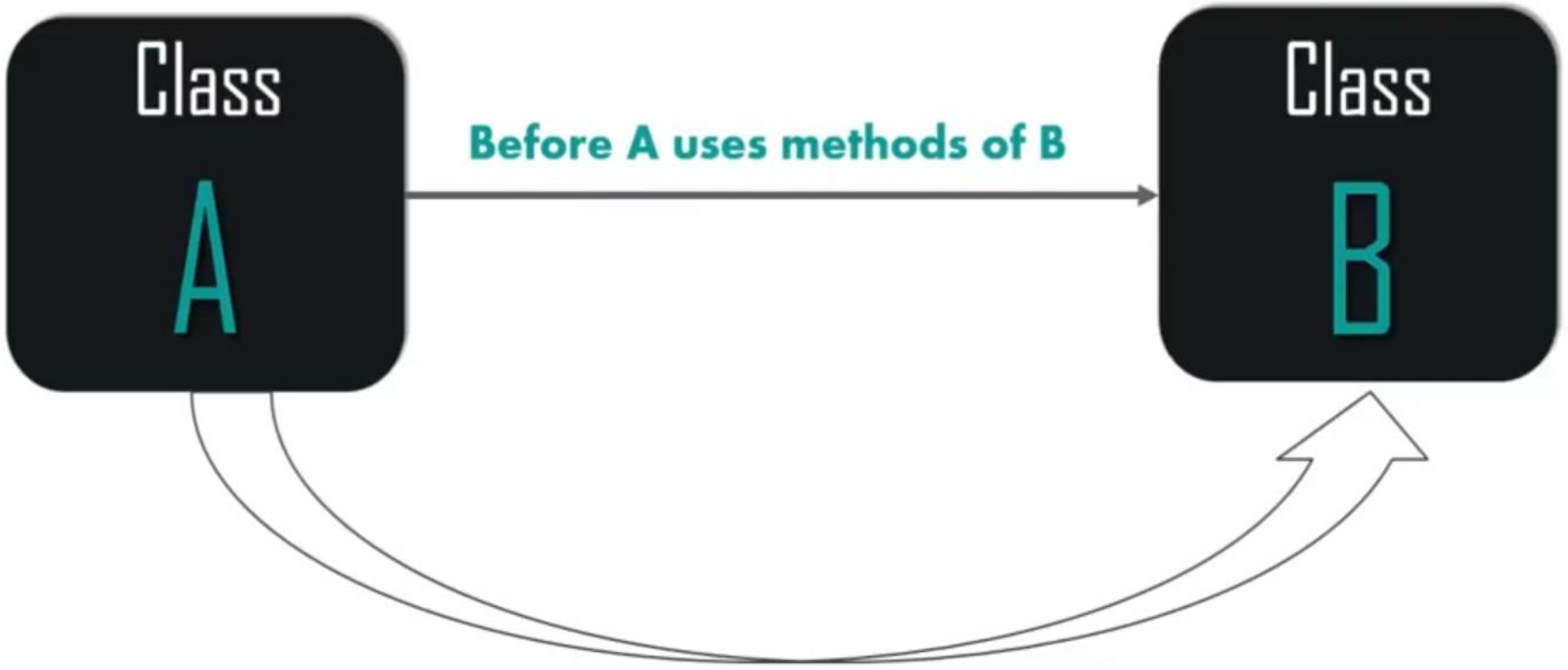
The dependencies are handled to the object when it's created. This also illustrates the Hollywood Principle at work: Don't call around for your dependencies, we'll give them to you when we need you.



Class A is Dependent on B

Dependency

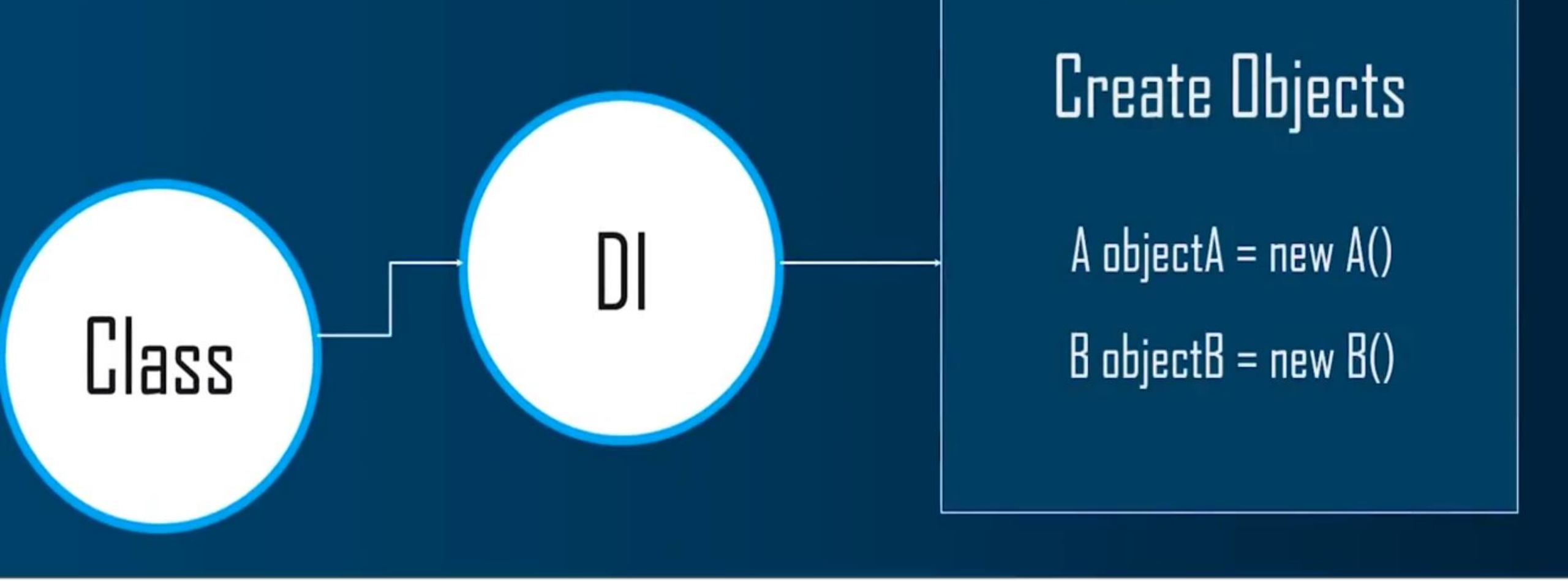
Dependency In Programming



Class A needs to create an instance of class B

What is Dependency Injection(DI)?

The process of creating an object for some other class and let the class directly using the dependency is called dependency injection.

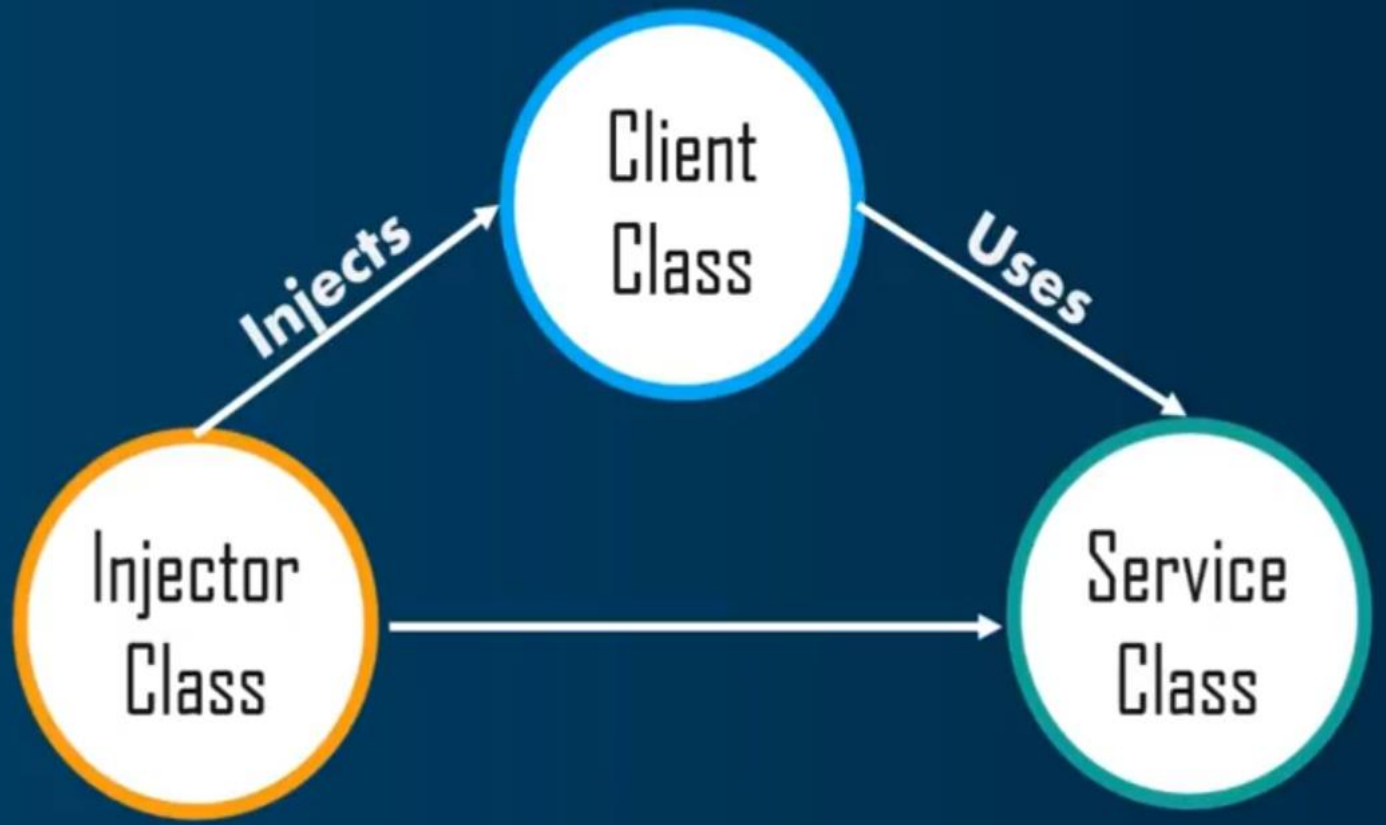


What is Dependency Injection(DI)?

The process of creating an object for some other class and let the class directly using the dependency is called dependency injection.

What is Dependency Injection(DI)?

The process of creating an object for some other class and let the class directly using the dependency is called dependency injection.



What is Dependency Injection(DI)?

Dependency Injection uses three types of classes. The injector class creates an object of the service class. Then, the injector class injects the object to a client object.

Types of Dependency Injection

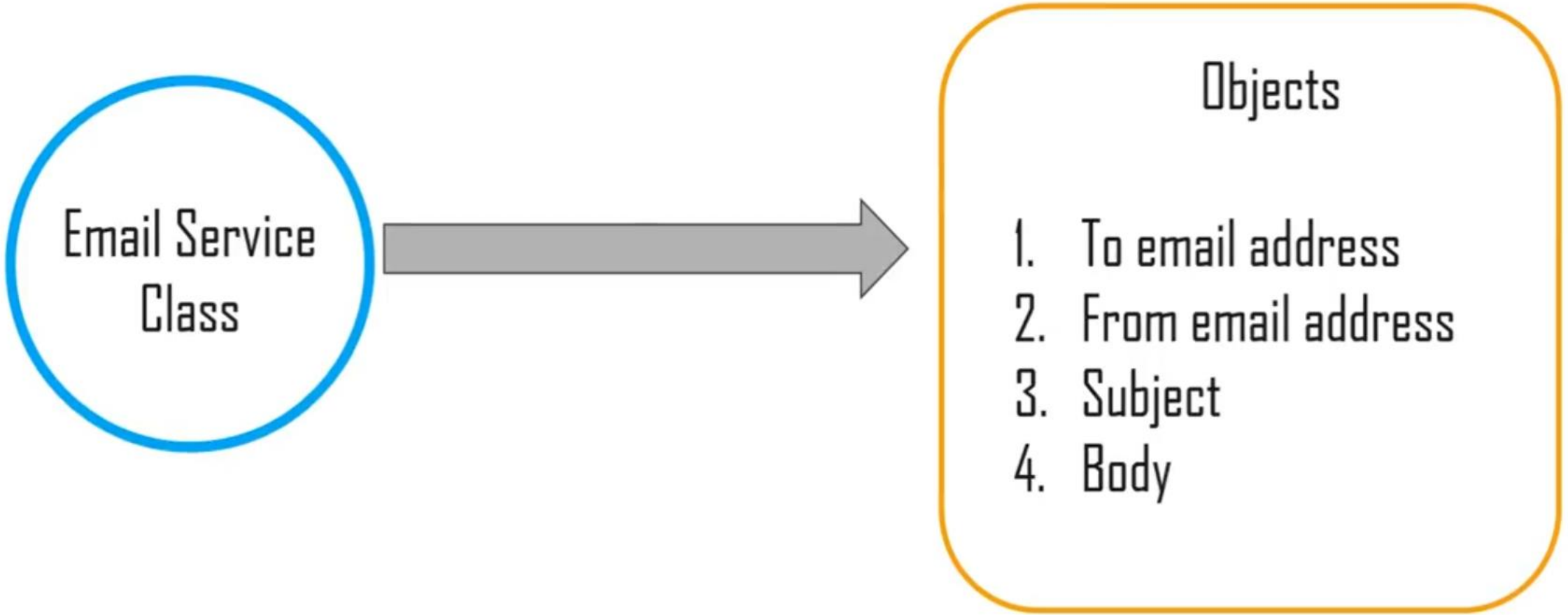
Constructor	Setter	Interface
Dependencies are provided through a class constructor	Injector method injects the dependency to the setter method exposed by the client.	Injector uses Interface to provide the dependency to the client class.

Types Of Dependency Injection

Spring framework avails two ways to inject dependency :

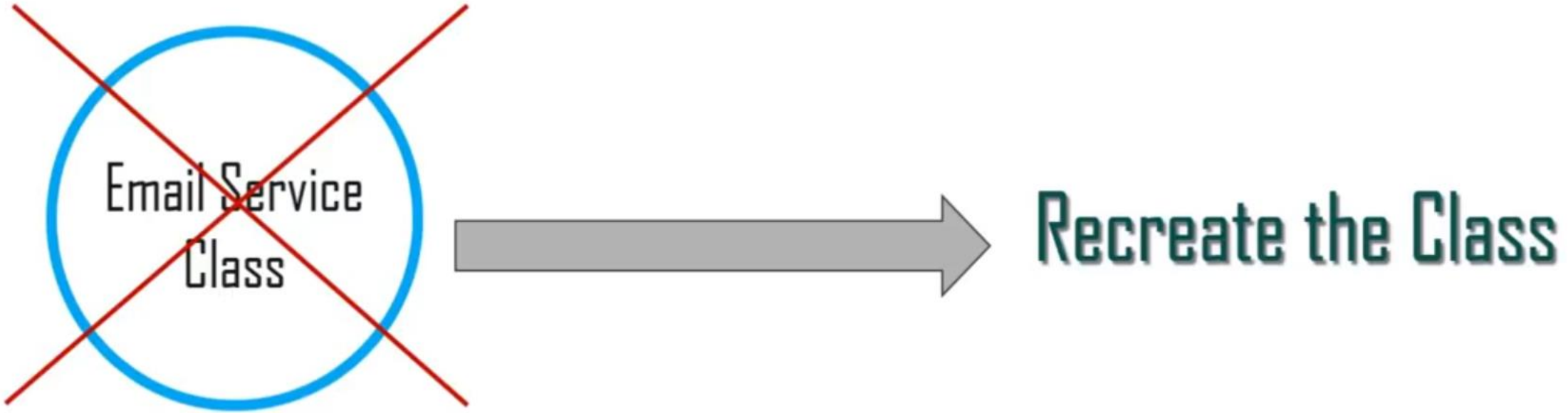
By Constructor	1	The **<constructor-arg>** subelement of **<bean>** is used for constructor injection
By Setter method	2	The **<property>** subelement of **<bean>** is used for setter injection

Benefits of DI



Benefits of DI

Text or Voice Messages?



Benefits of DI

Using Dependency Injection

Change Objects at run time

Benefits of DI

Enables an easy way to interconnect the components

Application can be easily extended

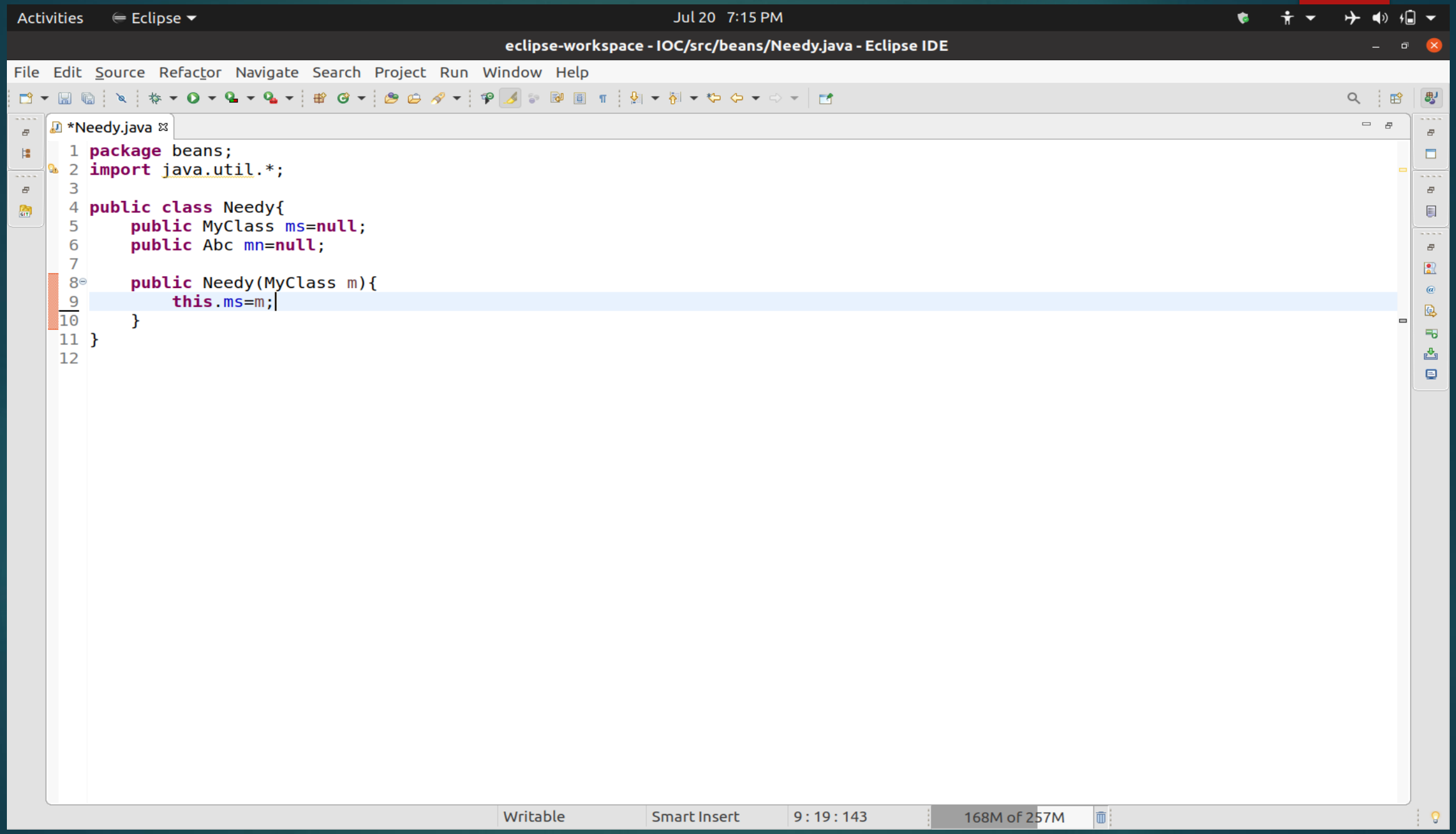
Unit Testing is made much easier

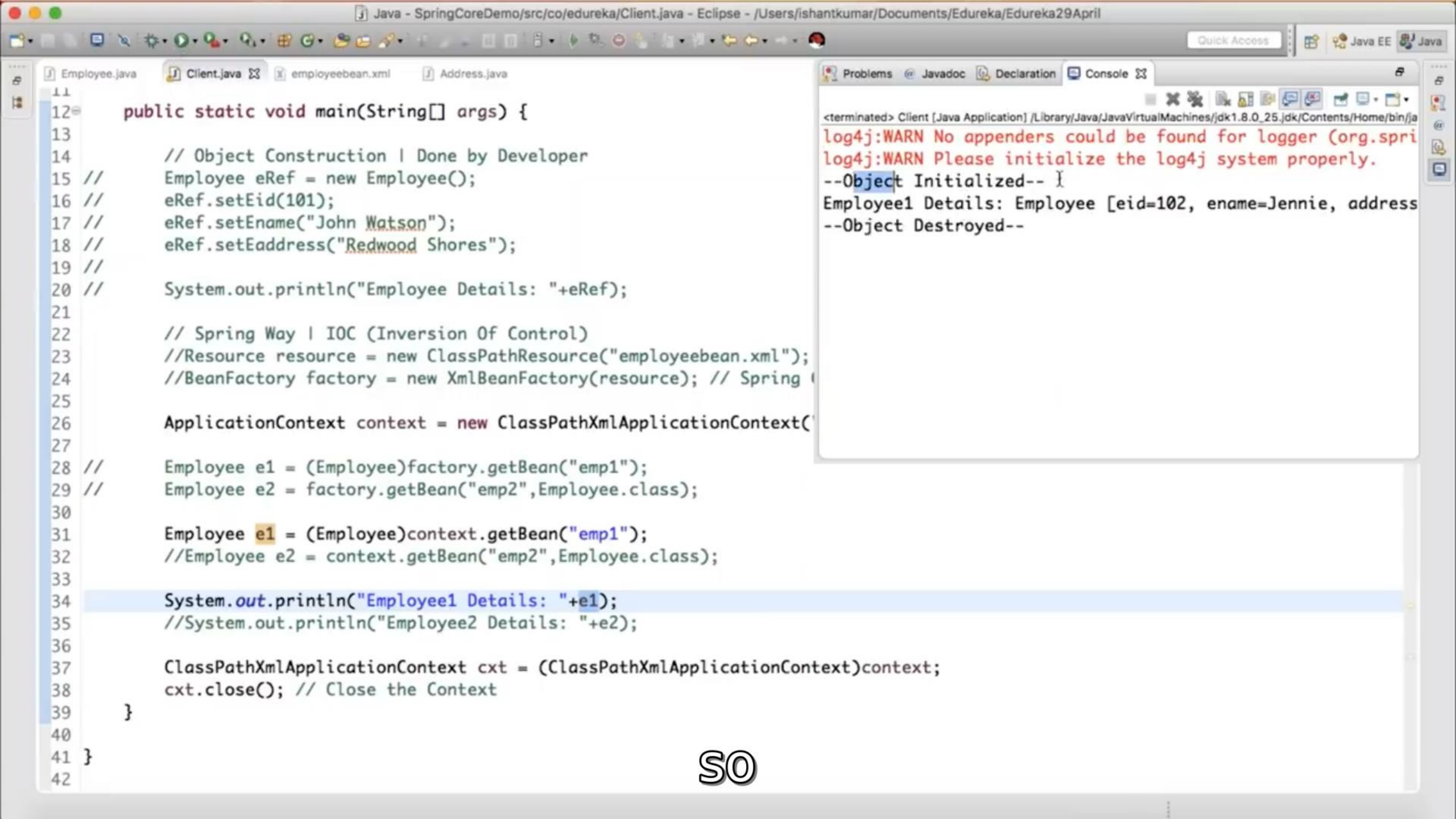
Reduction of Boiler plate code

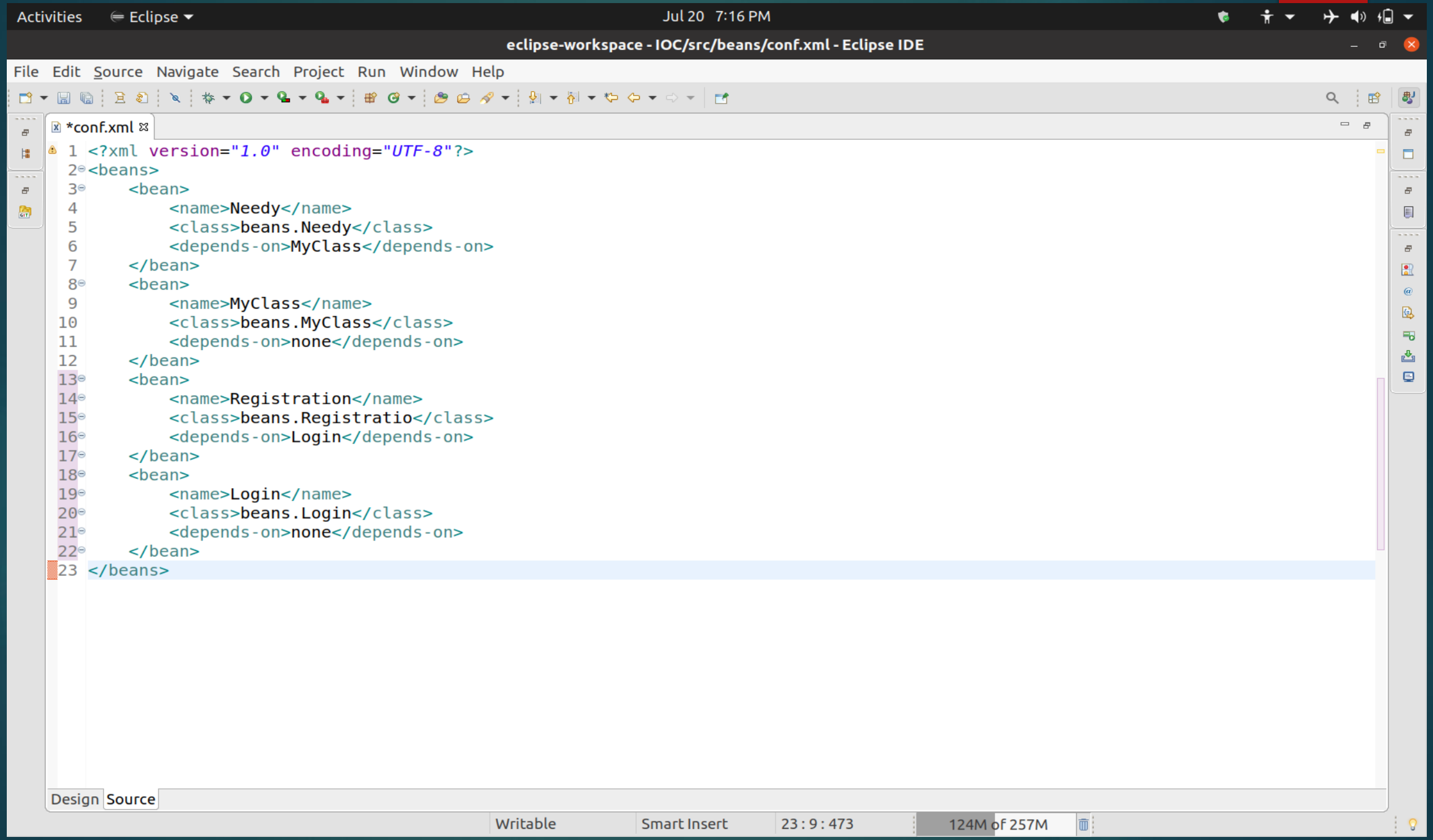
```
Spring
Core | IOC (Inversion of Control)

You dont create objects.
objects shall be configured in an XML file by the developer.
Spring Container -> Responsible to construct the Java Objects by parsing XML File




class Employee{
    int eid;
    String ename;
    Address address; // HAS-A

    Employee(){
        eid = 0;
        ename = "NA";
        address = new Address(); // Object Construction
    }
}

class Address{
    String city;
    String state;
```

```
45
46      void setAddress(Address adrs){
47          address = adrs;
48      }
49  }
50
51  class Address{
52      String city;
53      String state;
54      int zipCode;
55  }
56
57
58  Employee e = new Employee();
59  Construction of Address Object relies on construction of Employee Object
60
61  Address a = new Address();
62
63  e.setAddress(a);
64  Employee e1 = new Employee(a);
65
66  Spring Way: XML | IOC
67  Dependency Injection
68      Constructor
69      Setter
```

Using Constructor Injection


Activities
Eclipse
Jul 20 7:15 PM
eclipse-workspace - IOC/src/beans/Needy.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*Needy.java
1 package beans;
2 import java.util.*;
3
4 public class Needy{
5 public MyClass ms=null;
6 public Abc mn=null;
7
8 public Needy(MyClass m){
9 this.ms=m;
10 }
11 }
12
Writable
Smart Insert
9 : 19 : 143
168M of 257M


Java - SpringCoreDemo/src/co/edureka/Client.java - Eclipse - /Users/ishantkumar/Documents/Edureka/Edureka29April
Employee.java
Client.java
employeebean.xml
Address.java
Problems
Javadoc
Declaration
Console
<terminated> Client [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home/bin/ja
log4j:WARN No appenders could be found for logger (org.spri
log4j:WARN Please initialize the log4j system properly.
--Object Initialized--
Employee1 Details: Employee [eid=102, ename=Jennie, address
--Object Destroyed--
public static void main(String[] args) {
// Object Construction | Done by Developer
// Employee eRef = new Employee();
// eRef.setEid(101);
// eRef.setEname("John Watson");
// eRef.setEaddress("Redwood Shores");
// System.out.println("Employee Details: "+eRef);
// Spring Way | IOC (Inversion Of Control)
//Resource resource = new ClassPathResource("employeebean.xml");
//BeanFactory factory = new XmlBeanFactory(resource); // Spring
ApplicationContext context = new ClassPathXmlApplicationContext(
// Employee e1 = (Employee)factory.getBean("emp1");
// Employee e2 = factory.getBean("emp2",Employee.class);
Employee e1 = (Employee)context.getBean("emp1");
//Employee e2 = context.getBean("emp2",Employee.class);
System.out.println("Employee1 Details: "+e1);
//System.out.println("Employee2 Details: "+e2);
ClassPathXmlApplicationContext cxt = (ClassPathXmlApplicationContext)context;
cxt.close(); // Close the Context
}
}
SO


Activities
Eclipse
Jul 20 7:16 PM
eclipse-workspace - IOC/src/beans/conf.xml - Eclipse IDE
File Edit Source Navigate Search Project Run Window Help
*conf.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans>
3 <bean>
4 <name>Needy</name>
5 <class>beans.Needy</class>
6 <depends-on>MyClass</depends-on>
7 </bean>
8 <bean>
9 <name>MyClass</name>
10 <class>beans.MyClass</class>
11 <depends-on>none</depends-on>
12 </bean>
13 <bean>
14 <name>Registration</name>
15 <class>beans.Registratio</class>
16 <depends-on>Login</depends-on>
17 </bean>
18 <bean>
19 <name>Login</name>
20 <class>beans.Login</class>
21 <depends-on>none</depends-on>
22 </bean>
23 </beans>
Design
Source
Writable
Smart Insert
23 : 9 : 473
124M of 257M

Using Setter Injection

```
31          return ename;
32      }
33
34      public void setEname(String ename) {
35          this.ename = ename;
36      }
37
38      public Address getAddress() {
39          return address;
40      }
41
42      // Setter Injection
43      public void setAddress(Address address) {
44          this.address = address;
45      }
46
47      public void myInit(){
48          System.out.println("--Object Initialized--");
49      }
50
51      public void myDestroy(){
52          System.out.println("--Object Destroyed--");
53      }
54
55      @Override
56      public String toString() {
57          return "Employee [eid=" + eid + ", ename=" + ename + ", address=" + address + "]";
58      }
59
60  }
61
```

Java - SpringCoreDemo/src/employeebean.xml - Eclipse - /Users/ishantkumar/Documents/Edureka/Edureka29April

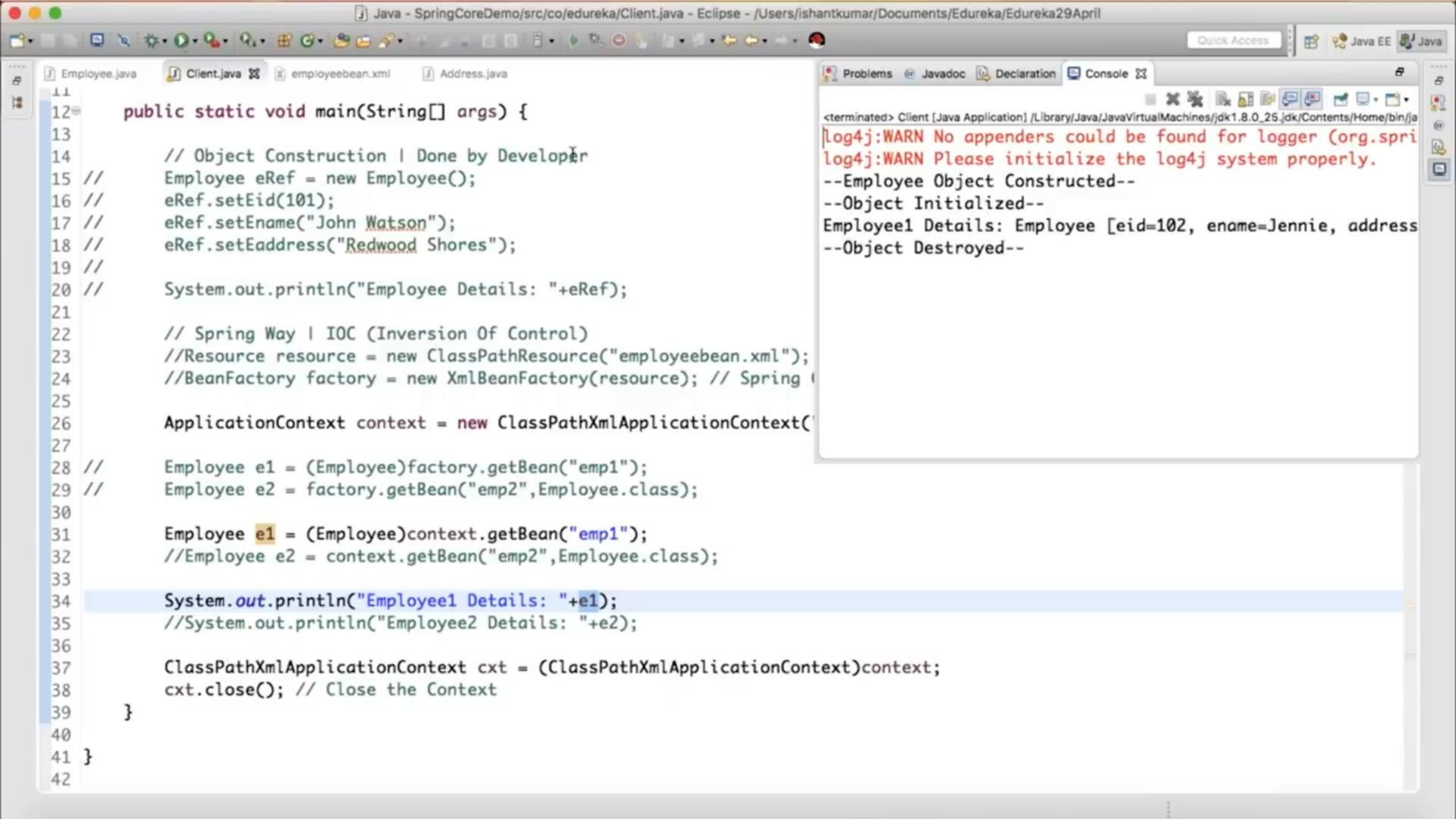
Employee.java | Client.java | employeebean.xml | Address.java

```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"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.0.xsd">

    <bean id="adrs" class="co.edureka.Address">
        <property name="city" value="Bengaluru"/>
        <property name="state" value="Karnataka"/>
        <property name="zipCode" value="520001"/>
    </bean>

    <bean id="emp1" class="co.edureka.Employee" init-method="myInit" destroy-method="myDestroy">
        <property name="eid" value="102"/>
        <property name="ename" value="Jennie"/>
        <!-- <constructor-arg ref="adrs"/> -->
        <property name="address" ref="adrs"/>
    </bean>

    <!-- <bean id="emp2" class="co.edureka.Employee">
        <property name="eid" value="103"/>
        <property name="ename" value="Jack"/>
        <property name="eaddress" value="Eastern Shores"/>
    </bean> -->

</beans>
```

Design | Source

this

Writable | Smart Insert | 21 : 42


Java - SpringCoreDemo/src/co/edureka/Client.java - Eclipse - /Users/ishantkumar/Documents/Edureka/Edureka29April
Employee.java
Client.java
employeebean.xml
Address.java
Problems
Javadoc
Declaration
Console
<terminated> Client [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home/bin/ja
log4j:WARN No appenders could be found for logger (org.spri
log4j:WARN Please initialize the log4j system properly.
--Employee Object Constructed--
--Object Initialized--
Employee1 Details: Employee [eid=102, ename=Jennie, address
--Object Destroyed--
public static void main(String[] args) {
// Object Construction | Done by Developer
// Employee eRef = new Employee();
// eRef.setEid(101);
// eRef.setEname("John Watson");
// eRef.setEaddress("Redwood Shores");
// System.out.println("Employee Details: "+eRef);
// Spring Way | IOC (Inversion Of Control)
//Resource resource = new ClassPathResource("employeebean.xml");
//BeanFactory factory = new XmlBeanFactory(resource); // Spring
ApplicationContext context = new ClassPathXmlApplicationContext(
// Employee e1 = (Employee)factory.getBean("emp1");
// Employee e2 = factory.getBean("emp2",Employee.class);
Employee e1 = (Employee)context.getBean("emp1");
//Employee e2 = context.getBean("emp2",Employee.class);
System.out.println("Employee1 Details: "+e1);
//System.out.println("Employee2 Details: "+e2);
ClassPathXmlApplicationContext cxt = (ClassPathXmlApplicationContext)context;
cxt.close(); // Close the Context
}
}

THANK YOU