* Ambiguity Problem & its Solution with Constructor Injection.

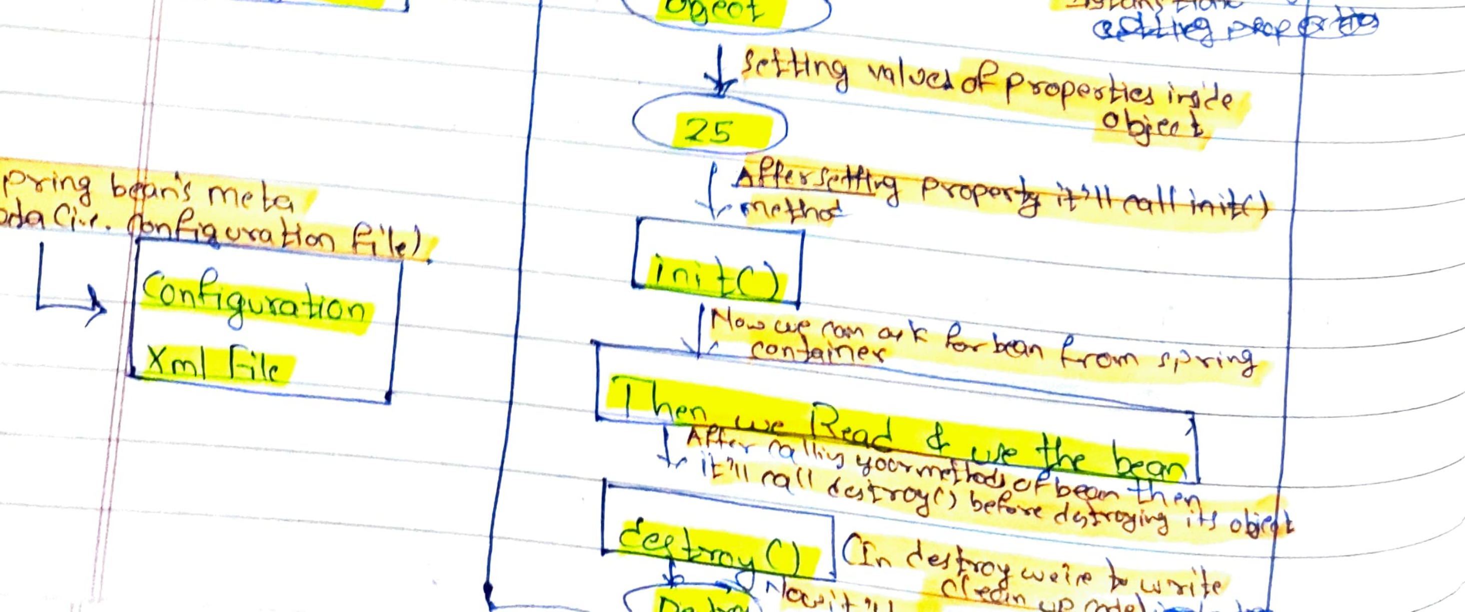
* Life Cycle Methods of Spring Bean:

→ Spring provide two important methods to every bean
(i) public void init() → For initialization
(ii) public void destroy() → For destroy

We can change the name of these method (i.e. init/destroy) but signature must be same (i.e. public void

(i) public void init(): Initialization code. loading config. Connecting db, webservices etc code into init()

(ii) public void destroy(): Clean up code (writing only close-up/clean-up code into destroy()).


A java class which has its default constructor, some properties, getter, setter, etc. (A simple Java class)
Life Cycle
Spring bean
create object
Object
Object Instantiate (Object creation)
object
Setting values of properties inside object
25
After setting property it'll call init() method
init()
Spring bean's meta data (i.e. Configuration file)
Configuration Xml File
Now we can ask for bean from spring container
Then we Read & use the bean
After calling your methods of bean then it'll call destroy() before destroying its object
destroy()
(In destroy we're to write clean up code)
Now it'll

Implementing Bean Lifecycle using Interfaces InitializingBean DisposableBean. / (Init / Destroy method called / configure using Spring Interfaces).

Create bean & implement two interfaces.

InitializingBean → It will provide one method & in that we've to write our init() code.

DisposableBean → It'll provide destroy functionality.

Pepsi.java:

```
package .....;
import org.springframework.beans.factory.DisposableBean;
import org.springframework.beans.factory.InitializingBean;

public class Pepsi implements InitializingBean, DisposableBean {
        private double price;

        public double getPrice(){
            return price;
        }

        public void setPrice(double price){
            this.price = price;
        }

        public Pepsi(){
            super();
        }
        @Override
```

Destroy()

To run destroy() after
run

```
public String toString(){
        return "Pepsi [price = " +price+"]";
}

        public void afterPropertiesSet() throws Exception{
Init()      ~~// TODO Auto generated~~
            //init
            S.o.pln("taking pepsi : init");
        }

        public void destroy() throws Exception{
Destroy()       //destroy
            S.o.pln("Going to put bottle: destroy");
        }
}
```

Test.java

```
package .....;

import  ApplicationContext;
import  AbstractApplicationContext;
  "     ClassPathXml ApplicationContext;

public ~~Class~~ class Test{
        p. s. v.m (String ~~ar~~[]args){

        AbstractApplication Context context=new Class pa
Imp to  context.registerShutdownHook();
```

Config.xml.

```
<bean class="com.cp. ---- "name="p1">
    <property name="price" value="5"/>
</bean>
```

(iii) Implementing Bean LifeCycle using Annotations @PostConstruct @PreDestroy.

$$\begin{tikzcd}
 & \text{@PostConstruct} \arrow[dl] \arrow[r] & \text{Init()} \\
\text{Provides Init functionality} & \text{@PreDestroy} \arrow[d] \arrow[r] & \text{Destroy()} \\
 & \text{Provides Destroy functionality} &
\end{tikzcd}$$

For java 8 it's ~~inp~~ inbuild, but for higher version we need to add the dependencies in ~~[illegible]~~ pom xml file, so we just need to copy it from web search & paste it into pomXML file.