e.g:

(i) Implementing lifecycle methods of spring bean using XML file. / How to call init / destroy method using XML file.

→ create new package.

→ create new bean(class) in that package. i.e. Samosa

Samosa.java

```
package ....

p. class Samosa {
    Private double price;
    public double getPrice(){
        return price;
    }

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

    public Samosa(){
        super();
    }
}
```

getter/setter method.

Constructor.

```
public void init(){
    S.o.pln("Inside init method");
}
p. v. destroy(){
    S.o.pln("Inside destroy method");
}
```

Init method

destroy method

classmate Date Page

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

toString method

config.xml

To identify the name of bean

```
<bean class = "com.springcore.lifecycle.Samosa" name = "s1"
    init-method = "init" destroy-method = "destroy">
    <property name = "price" value = "10" />
</bean>
```

Package Name

Class name

Test.java → Main method (Creating object of bean property)

```
package ....
p. c. Test {
    p. s. v. m. (String[] args){
        AbstractApplicationContext context = new ClassPathXmlApplicationContext
            ("com/springcore/lifecycle/config.xml");
        Samosa s1 = (Samosa) context.getBean("s1");
        S.o.pln(s1);
        context.registerShutdownHook();
    }
}
```

Object creating

Storing bean in variable

Type casting

getting bean

filename

This method is only with AbstractApplicationContext

Shutdown hook will enable & the destroy method will be called.

InitializingBean → This method is only for Initializing purpose.

DisposableBean → This method is only for destroy() purpose.

(ii) 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
```

(Init & Destroy)

To run destroy() method we want to run

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

init()

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

destroy()

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

Test.java

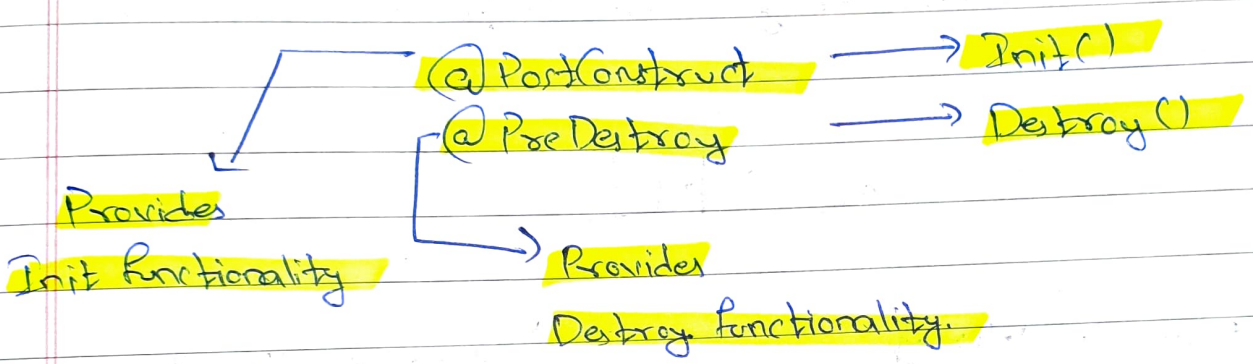
```
package ....;

import ApplicationContext;
import AbstractApplicationContext;
//      ClassPathXmlApplicationContext;

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

AbstractApplicationContext context = new ClassPa...
context.registerShutdownHook();
Pepsi pl = (Pepsi) context.getBean("pi");
}
```

to run destroy() method.

Config.xml :-

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

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




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