Course Outline

Spring Boot & Microservices

- Spring Core **Part-1**
- Spring Boot **Part-2**
- Spring Data JPA **Part-3**
- Spring Boot WebMVC **Part-** 4
- Spring Rest **Part- 5**
- Microservices **Part- 6**
- Miscellaneous **Part- 7**

❖ **Spring Security**
❖ **Messaging Queues**
❖ **Redis Cache**
❖ **Tools**
❖ **Deployment**
❖ **Backend Mini Project**

Course Details

Parameters	Information
Prerequisites	Core Java, Advanced Java, Spring(Advantage)
Training Mode	Online
Class Timings	7AM – 8AM IST (Monday - Saturday)
Batch Code	**47_SBMS**
Course Duration	3.5 – 4 Months
Course Fee	8K (Online) , 10K (Online + Backup Videos)
Videos Access	1 Year
Class Notes	Daily You Can Download Notes Through Ashok IT Portal
Course Content	https://ashokitech.com/uploads/course/847783647_1661854272.pdf

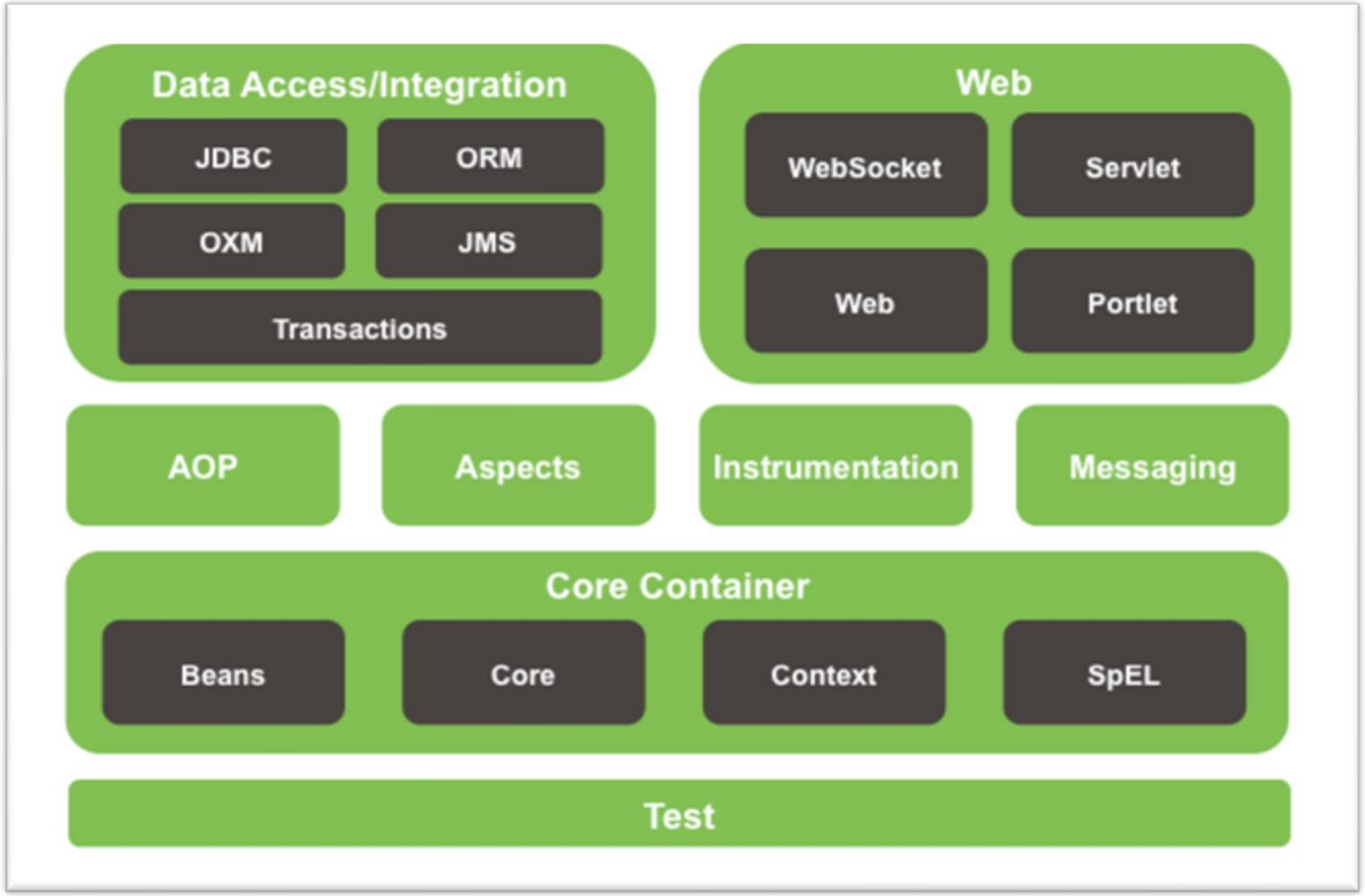
Introduction

ASHOK IT
Learn Here.. Lead Anywhere..!!

spring
Framework

- ❖ Java Based Open Source Framework
- ❖ Enterprise Application Development
- ❖ Modularized Framework
- ❖ Simplicity of POJOS
- ❖ Dependency Injection & Inversion of Control (IOC)
- ❖ Lightweight Framework

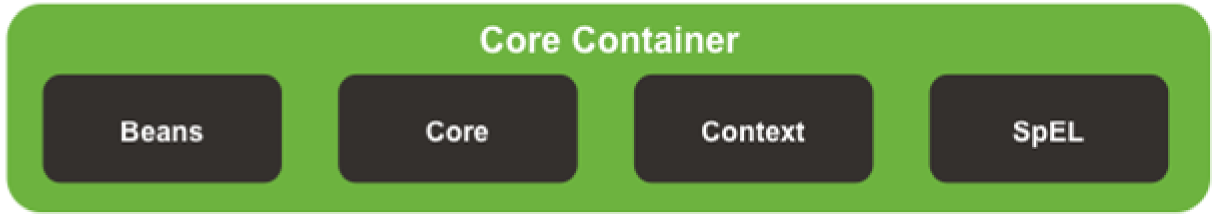
Spring Architecture



- ❖ Spring Core
- ❖ Spring Web
- ❖ Spring DAO
- ❖ Spring AOP

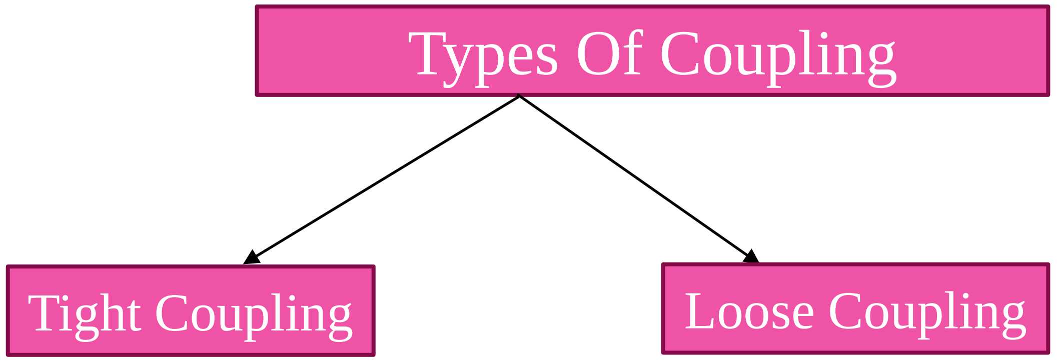
Spring Core

- ❖ Spring Core Module is base Module for remaining Modules of Spring Framework.
- ❖ Spring Core is the central part of spring framework to work with spring container. It manages how the beans are created, configured in a spring application.
- ❖ This module makes spring application as ”**Light -Weight**” by providing loose coupling between the objects.
- ❖ This module provides services like **Dependency Injection(DI),** Email, I18N, AOP programming etc.



Coupling

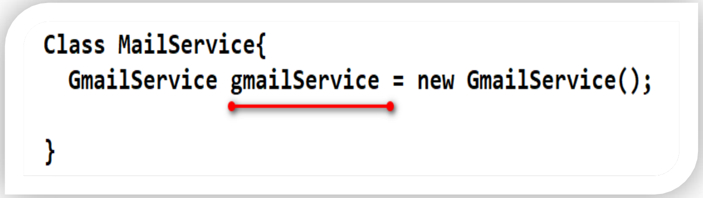
- Java is an Object-oriented programming language. Coupling in Java plays an important role when you work with Java Classes and Objects.
- It basically refers to the extent of knowledge one class knows about the other class.
- If one class calls another class / one class communicating with another Class called as "**Collaboration**"



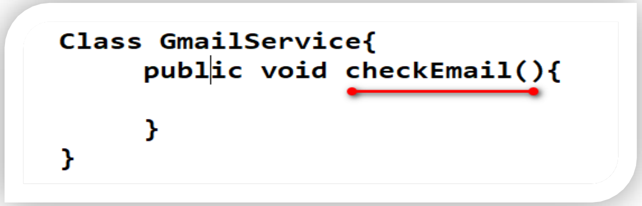
Tight Coupling

Tight Coupling between objects is going to occur in the following two cases

1) When a caller class is directly creating objects of its dependencies.

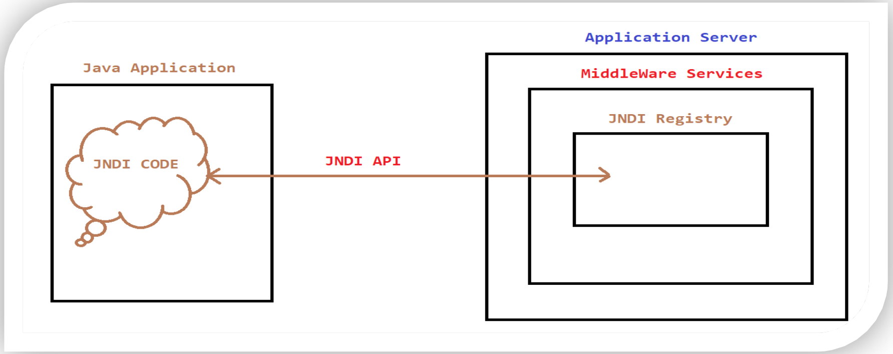


```
Class MailService{
  GmailService gmailService = new GmailService();

}
```



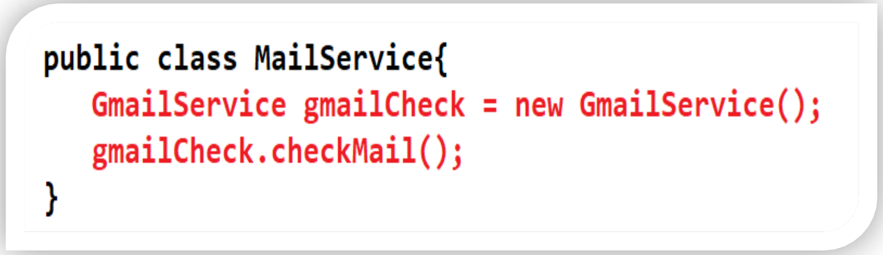
```
Class GmailService{
      public void checkEmail(){

      }
}
```

2) When a servlet class is collecting Data Source Object from JNDI(Java Naming and Directory Interface) registry then there is a tight coupling between a servlet object and Data Source Object.

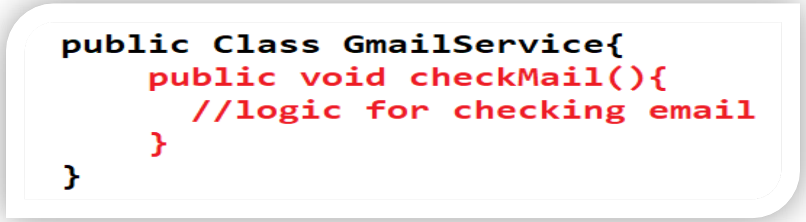


Drawbacks Of Tight Coupling

1) If any modification done on dependent Object then code impact will be on Main Class.

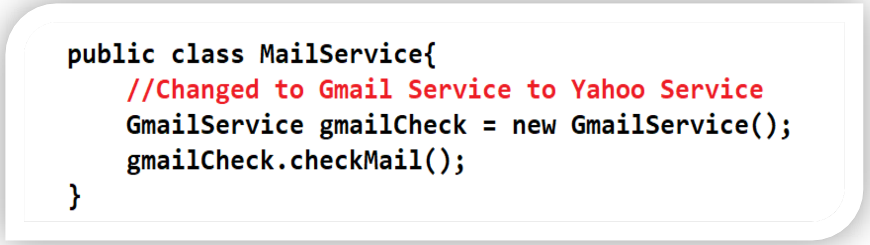


```
public class MailService{
   GmailService gmailCheck = new GmailService();
   gmailCheck.checkMail();
}
```



```
public Class GmailService{
      public void checkMail(){
        //logic for checking email
      }
}
```

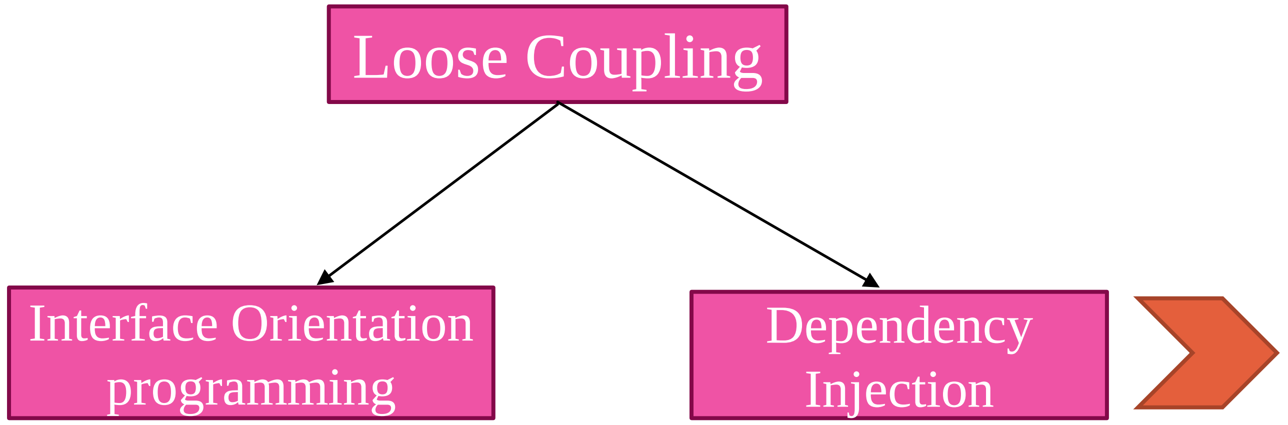
2) If dependency object got changed to another class then Main Object need to be modified.



```
public class MailService{
    //Changed to Gmail Service to Yahoo Service
    GmailService gmailCheck = new GmailService();
    gmailCheck.checkMail();
}
```

Loose Coupling

Loose Coupling means mostly independent components / Loose Coupling between Java Objects are existed below two techniques

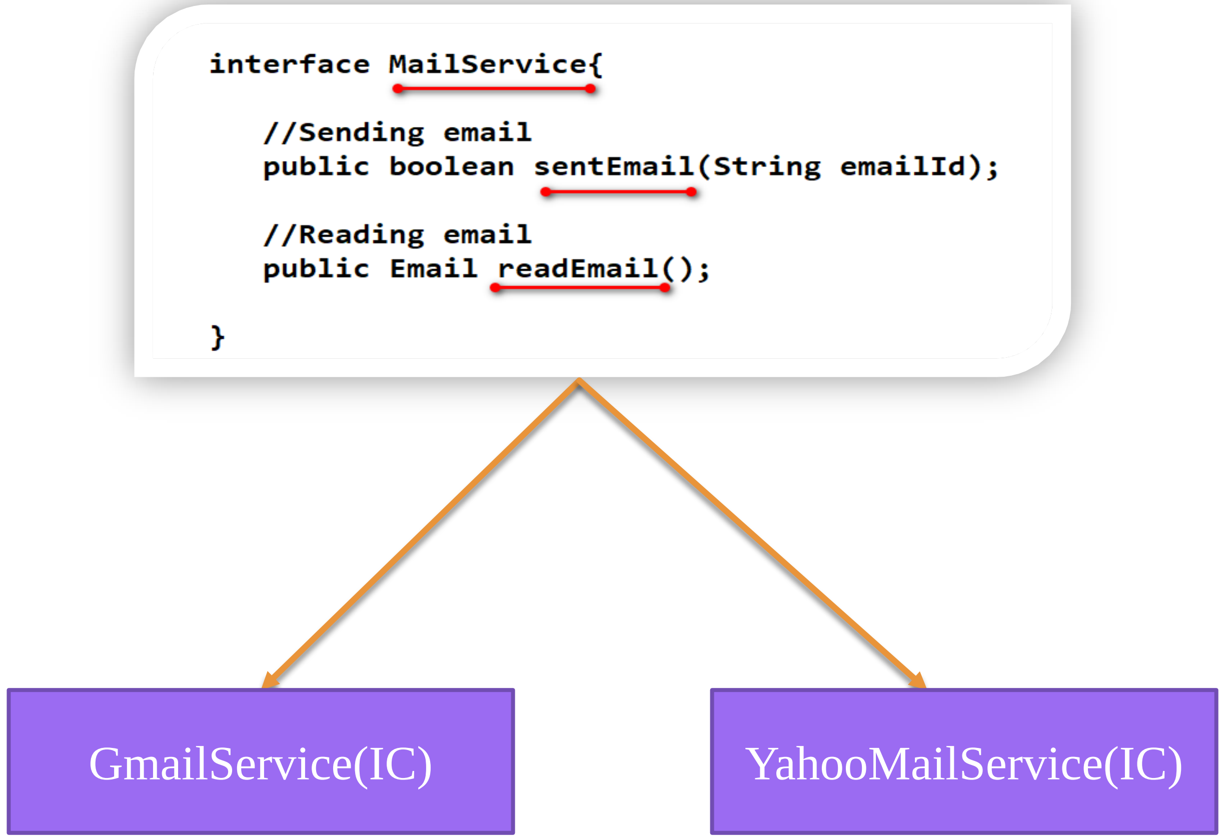


1. **Setter Injection**
2. **Constructor Injection**
3. **Interface Injection**

Multiple Classes of an application providing the similar functionality we can apply IOP.

It is technique of injecting dependent object into main object at runtime by spring container.

Interface Orientation Programming



```
interface MailService{

   //Sending email
   public boolean sentEmail(String emailId);

   //Reading email
   public Email readEmail();

}
```

Dependency Injection(DI)

❖ Dependency Injection is a fundamental concept of Spring framework.

❖ Spring container will inject the dependencies required for class at runtime through Dependency Injection Mechanism.

❖ Spring framework will provide three kinds of Dependency Injections(Setter, Constructor, Interface)

Setter Injection

```
Class Employee{

    private Address address;

    public void setAddress(Address address){
         this.address = address;
    }
}
```

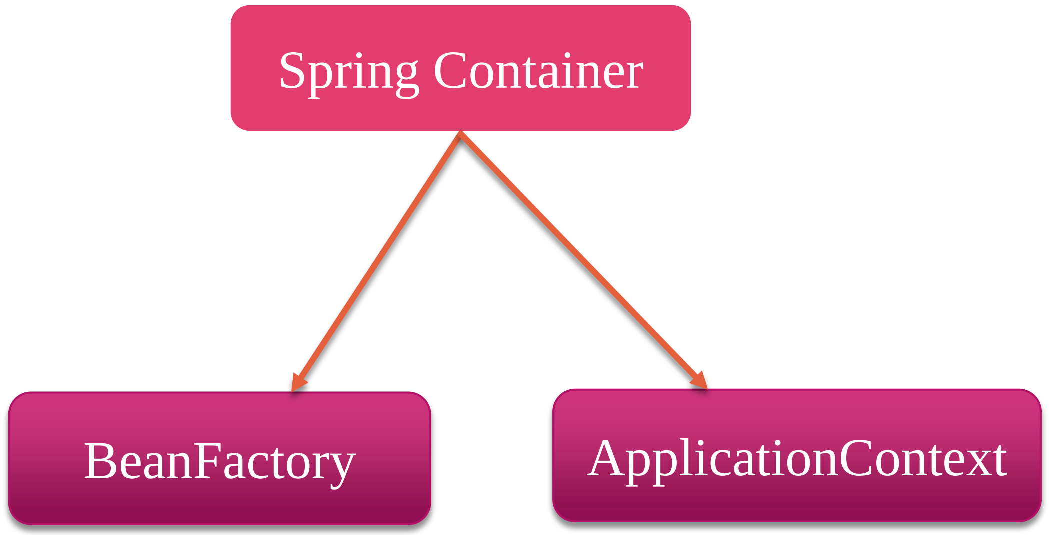
Constructor Injection

```
Class Employee{

    private Address address;

    public Employee(Address address){
         this.address = address;
    }
}
```

Spring Container

❖ Container is a software application (or) Java Class that can take care of the whole life cycle of given resource.

❖ Spring container is nothing but a Java Class which is provided by spring framework but it is not .exe file (or) .bat file (or) not a setup file.



✓ Bean Factory can only provides "**Dependency Injection**"

✓ Application Context can provide "**Dependency Injection, AOP, I18N,EventHandling**" etc.,

✓ BeanFactory & ApplicationContext are interfaces from Spring library.

Spring Container Classes

❖ BeanFactory(I) and its implementation Classes.
1) XMLBeanFactory
2) SimpleJndiBeanFactory
3) DefaultListableBeanFactory

❖ ApplicationContext(I) and its implementation Classes.
1) ClassPathXmlApplicationContext
2) FileSystemXmlApplicationContext
3) XMLWebApplicationContext

❖ Activating the Spring container in spring application is nothing but creating object an implementation class for either of any Spring containers.

❖ Most frequently used implementation class are “XMLBeanFactory, ClassPathXMlApplicationContext” in spring application development.

❖ We can't activate the Servlet Container, JSP Container by creating objects for certain classes so they are heavy weight containers these containers will activate during server starup time itself.

Spring Configuration

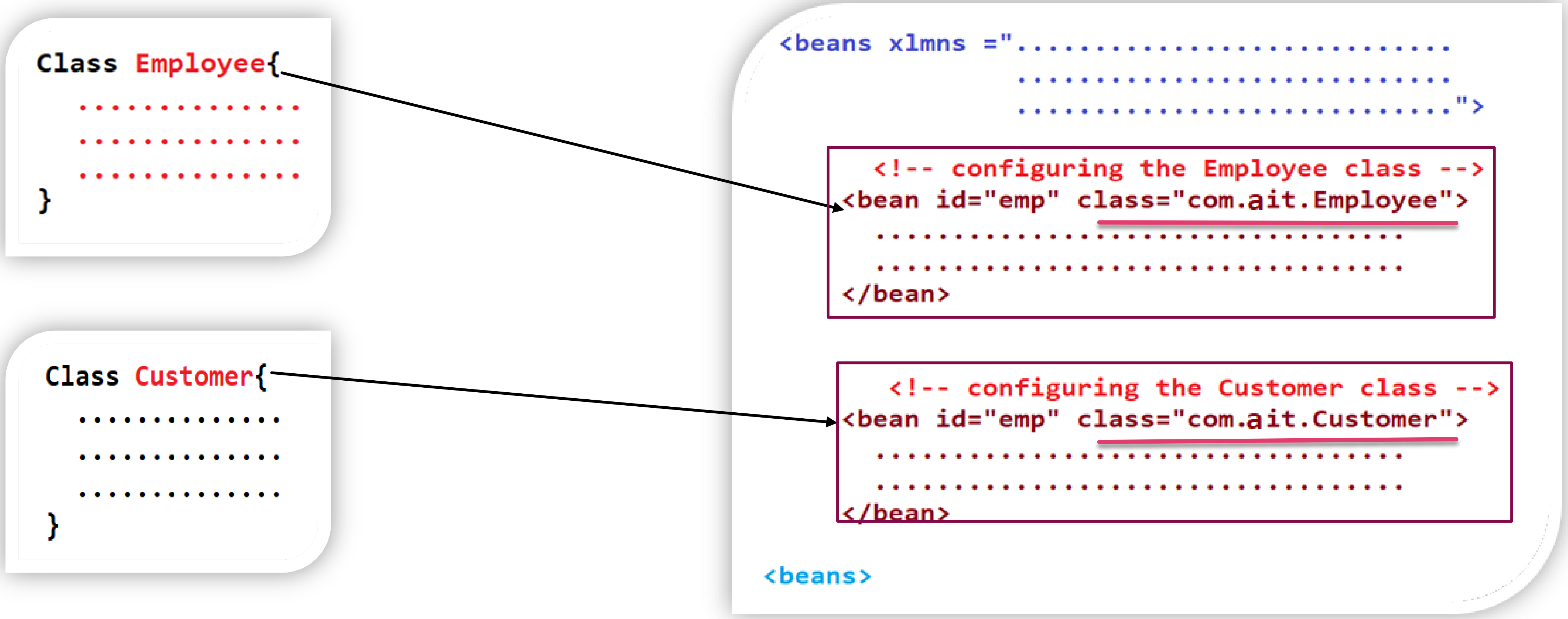
- Spring Container creates spring bean object, performs life cycle operations, inject dependencies and finally destroys spring bean object.
- Every Spring bean in Spring application has to be configure in **Spring configuration file**.
- Spring configuration file is nothing but an xml file created by programmer/Developer. It should be configured every spring bean in spring application.
- We can provide any file name to Spring configuration file i.e., anyname.xml.
- We can create more than one spring configuration file in spring application.
- Spring configuration File is a simple xml file with beans and their dependencies configuration.

anyname.xml

XML

Spring Bean Configuration

Spring Bean Configuration



```
Class Employee{
    ..............
    ..............
    ..............
}
```

```
Class Customer{
    ..............
    ..............
    ..............
}
```

```
<beans xmlns ="...........................
              ...........................
              ...........................">

  <!-- configuring the Employee class -->
<bean id="emp" class="com.ait.Employee">
  ..................................
  ..................................
</bean>

  <!-- configuring the Customer class -->
<bean id="emp" class="com.ait.Customer">
  ..................................
  ..................................
</bean>

<beans>
```

Spring Bean

❖ Spring bean is a key concept of the Spring Framework.

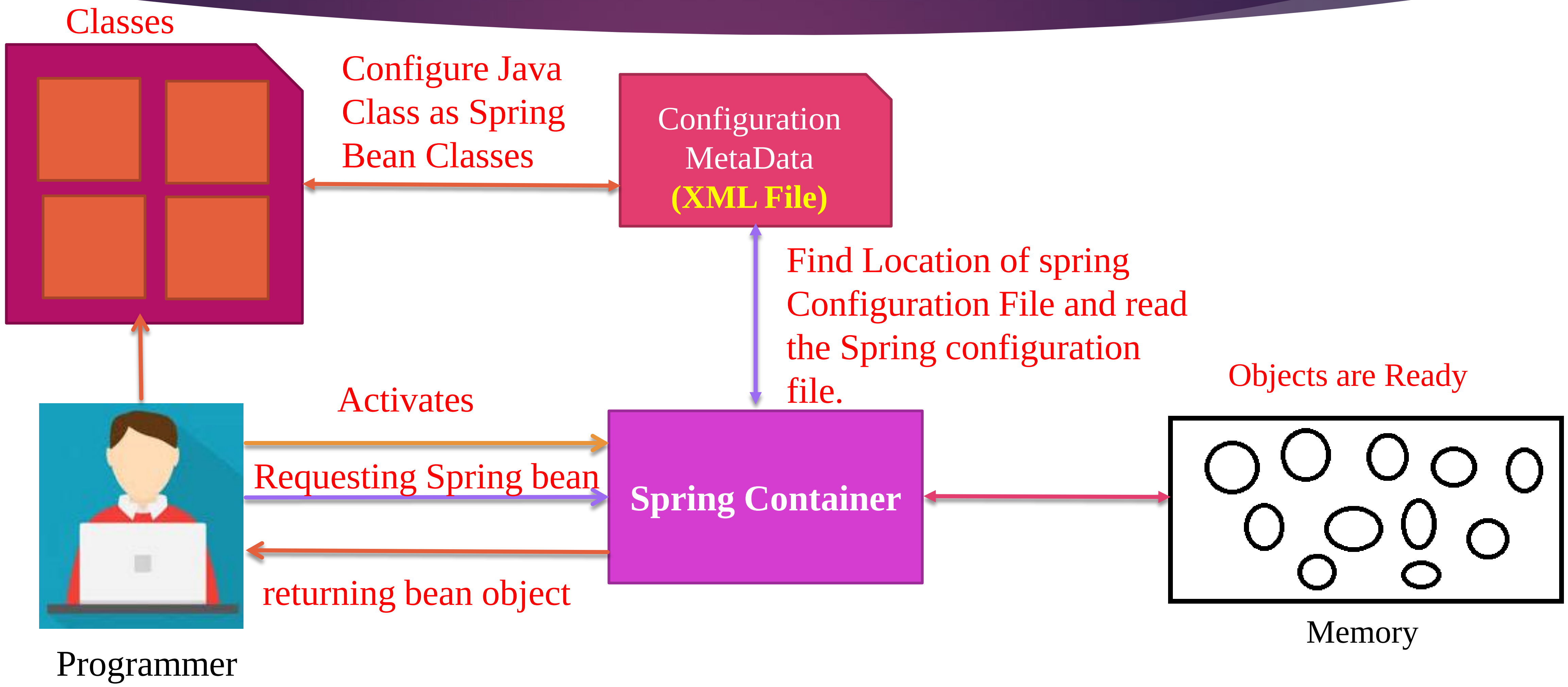
❖ Spring beans are nothing but Simple Java Class which doesn't extends (or) implements third party related classes & interface.

❖ Spring beans are always managed by Spring IOC Container mean that Till Object Creation to Object Destruction will be taking care of everything by Spring Framework.

❖ We can represent the Java Class as Spring Bean below ways
1) XML Configuration
2) By Using Stereo Type Annotation(@Component, @Service, @Repository, @Controller)
3) Java Based Configuration (@Bean)

❖ Every Spring Bean in Spring framework represented as "Singleton" by default.

Basic Flow

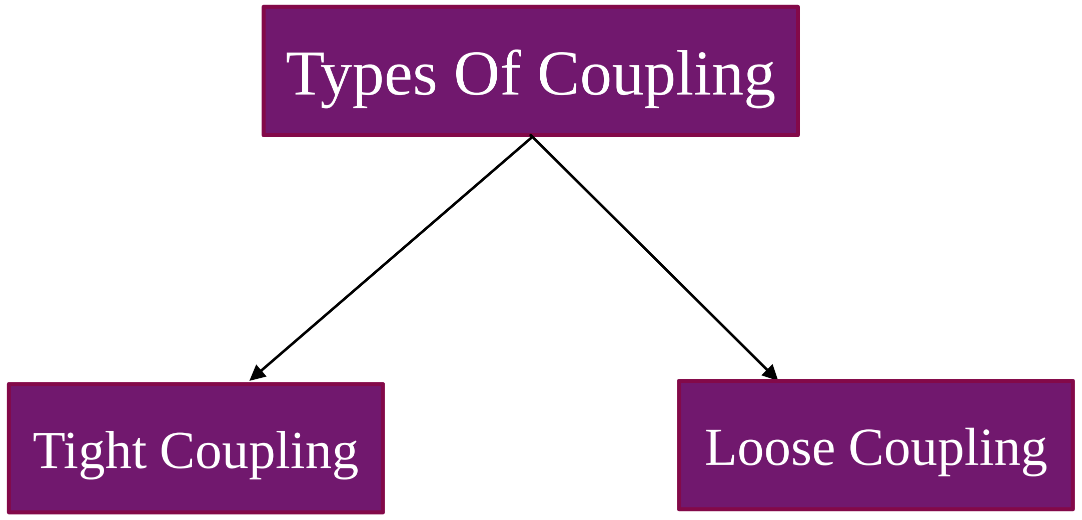

Classes
Configure Java Class as Spring Bean Classes
Configuration MetaData
(XML File)
Find Location of spring Configuration File and read the Spring configuration file.
Objects are Ready
Activates
Requesting Spring bean
Spring Container
returning bean object
Memory
Programmer

Coupling

- ❖ Java is an Object-oriented programming language. Coupling in Java plays an important role when you work with Java Classes and Objects.

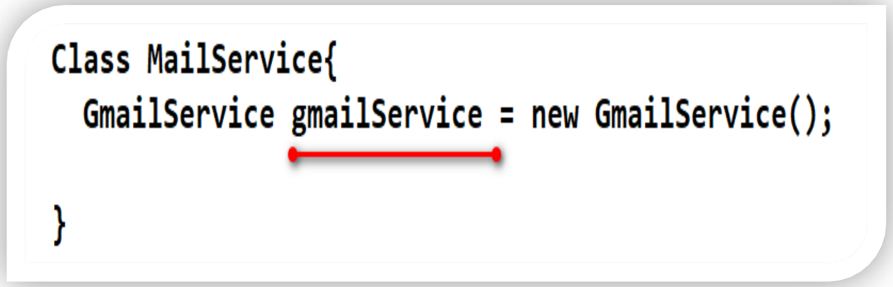
- ❖ It basically refers to the extent of knowledge one class knows about the other class.

- ❖ If one class calls another class / one class communicating with another Class called as “**Collaboration**”

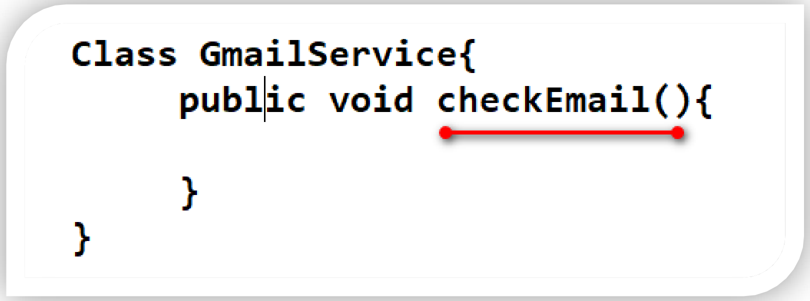


Tight Coupling

Scenario-1

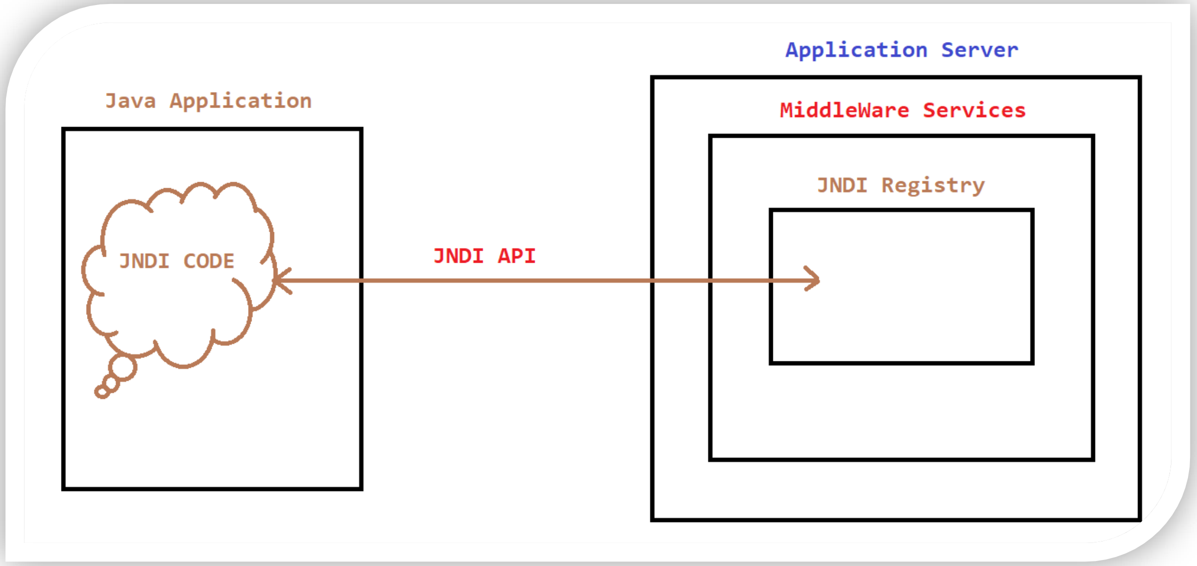


```
Class MailService{
  GmailService gmailService = new GmailService();

}
```



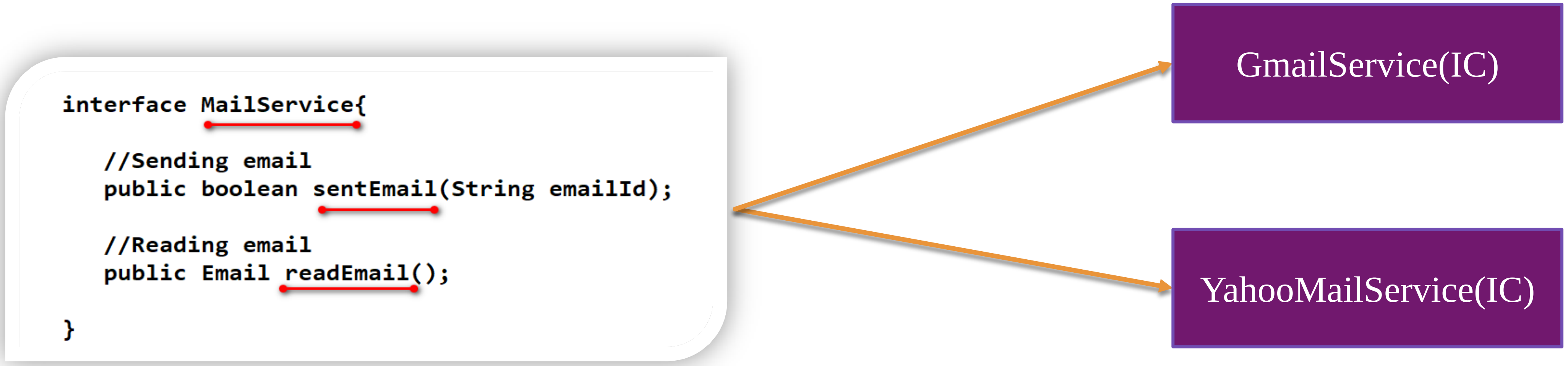
```
Class GmailService{
     public void checkEmail(){

     }
}
```

Scenario-2



Loose Coupling

- In Java Loose Coupling means that the classes are independent of each other.
- In Loose Coupling the knowledge one class has about the other class is what the other class has exposed through its interfaces.
- We can implement the Loose Coupling between Java Objects through Abstract classes & Interfaces.



```
interface MailService{

   //Sending email
   public boolean sentEmail(String emailId);

   //Reading email
   public Email readEmail();

}
```

Dependency Injection

- ❖ Dependency Injection is a fundamental concept of Spring framework.
- ❖ Spring container will inject the dependencies required for class at runtime through Dependency Injection Mechanism.
- ❖ Spring framework will provide three kinds of Dependency Injections(**Setter, Constructor, Interface**)

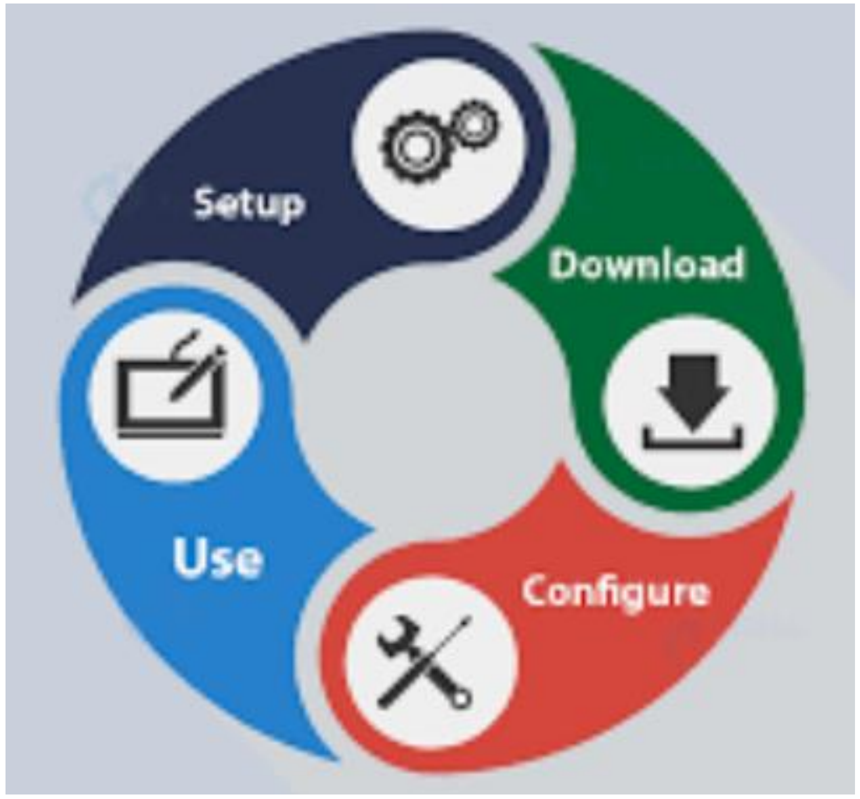
Setter Injection

```
Class Employee{

    private Address address;

    public void setAddress(Address address){
         this.address = address;
    }
}
```

Constructor Injection

```
Class Employee{

    private Address address;

    public Employee(Address address){
         this.address = address;
    }
}
```

Spring Environmental Setup



- ✓ Java Software with Minimum 1.8 Version
- ✓ https://www.eclipse.org/downloads/
  - **Eclipse IDE**
- ✓ https://spring.io/tools
  - **STS IDE**
- ✓ https://repo.spring.io/ui/native/release/org/springframework/spring/
  - **Spring Software Repo**
- ✓ **Maven Build Tool for dependency Management**
- ✓ **More Information About Spring**
  https://spring.io/projects/spring-framework/#learn

Spring First Application

1. Creating Spring Bean Class.
2. Creating Spring Configuration File.
3. Creating Spring Client Class with main method.
   - ❖ Creating Resource(I) Object to hold the spring configuration file.
     - ClassPathResource(IC)…..FileSystemResource(IC)
   - ❖ Activates the Spring Container Object by Programmer
   - ❖ Request the Spring Bean Object from Spring container.
   - ❖ Call the Spring Bean Service Methods…

Q & A Session

For More Information : https://ashokitech.com/