# IoC

Inversion of Control.

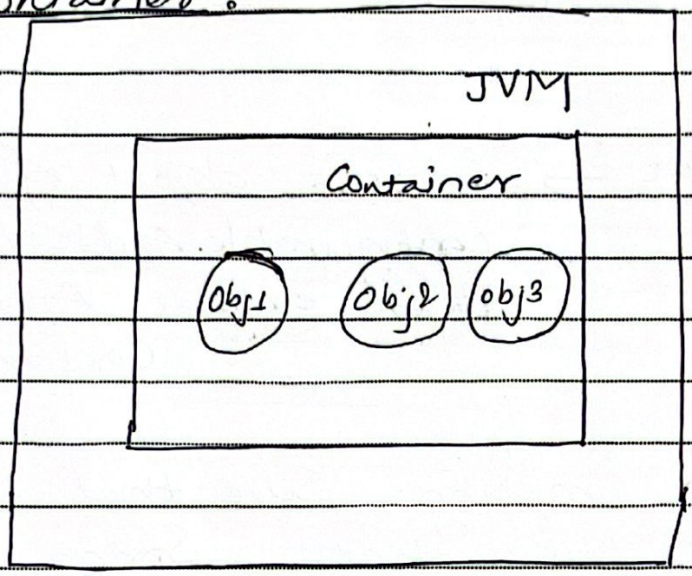
As a programmer -
- i) Creating the object
- X ii) Maintaining the object
- iii) Destroying the object

**Main Role:**

Business logic. ✓✓

User don't have to create the object, but need to focus on business logic

**IoC Container:**




Spring will create object for us in IoC container

Dependency Injection.
- to implement IOC principle.
- injecting objects in application.
- It is a design pattern.

Any object which is created, managed by spring is known as Beans.

I want Spring to get me object.

SpringApplication.run( , args)

Every time Spring creates objects for us, it will be available in the container.

Using ApplicationContext.

run → returns object of configurable ApplicationContext that extends ApplicationContext.

@Component. ⇒ making sure that Spring knows that Spring has to manage this particular object.

Create obj, assemble obj, and manage it.
everything will be done by Spring.

# Scope in Spring



scope = "singleton"

- Spring will create the object by default
- The moment we load the application, it will load the container and it will get the object and we can use that object any number of times. (same object)

scope = "prototype"

- object will be created only when getBean() is used. (new objects) =

lazy-init true : object will be created only when we want it but it will be singleton. =

I want this object to be created only when I call it for the first time

⇒ When we have non-lazy bean dependent on a lazy bean, still it will create object for a lazy bean. because someone wants it.

getBean() returns object of type Object.
and we need to do type casting.
But not if we use,

getBean("id", classname)

autowire = "byType" → if same type then, goes for primary = "true" bean.
If same types of bean better to go for name

If we want certain bean only for particular bean than we put that inside property field instead of ref = " ".