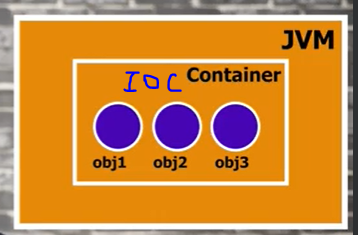
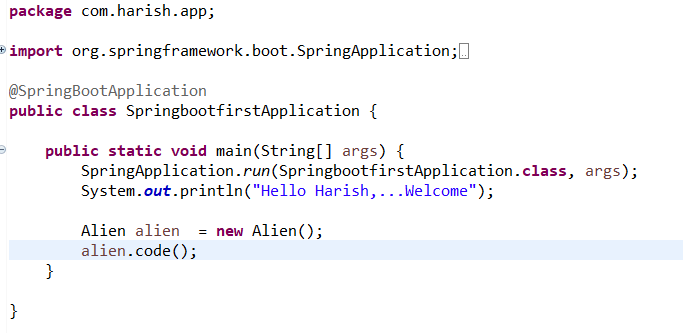
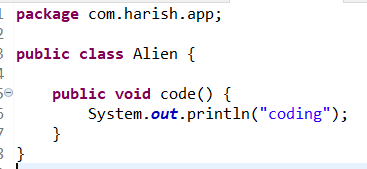
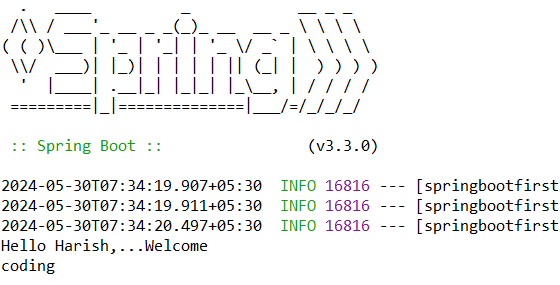
**May 20, 2024**  
Spring is a j2ee for building large scale applications.

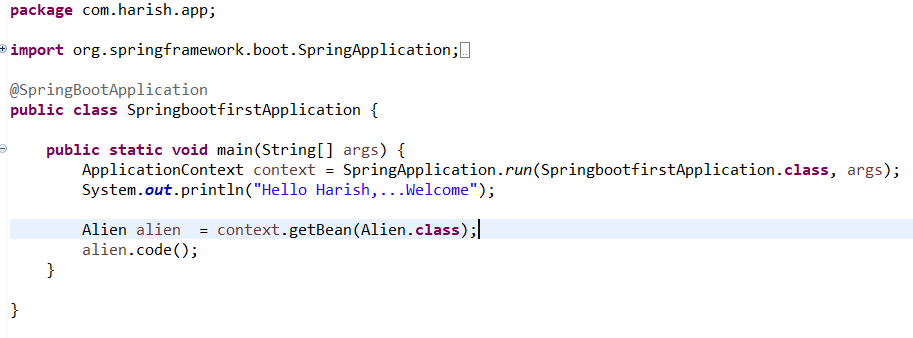
**IOC** – Inversion of Control -> Instead of we managing the object creation, maintaining and destruction we can’t focus on business logic. So we handover the control to others regarding object life cycle so that we can focus on business logic.

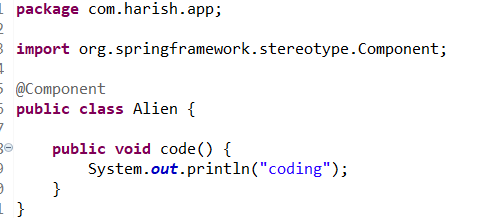
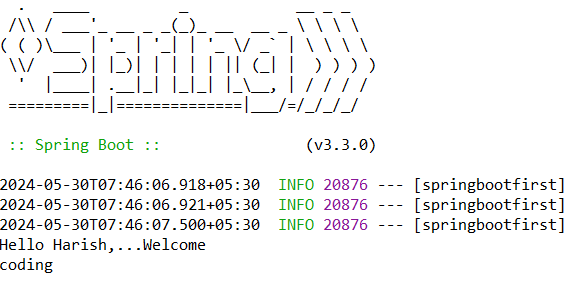


Spring will create objects for you and it will have objects inside IOC container.  
**IOC is a principle**.  
**DI(Dependency Injection) is a design pattern by which we can implement IOC**.  
For making IOC to work we use DI. You have objects in IOC container but some one needs to inject right, so **DI will do injection**.  
Spring is a **opinionated framework**.  
An opinionated [person](https://dictionary.cambridge.org/dictionary/english/person) is [certain](https://dictionary.cambridge.org/dictionary/english/certain) about [their](https://dictionary.cambridge.org/dictionary/english/their) [beliefs](https://dictionary.cambridge.org/dictionary/english/belief), and [expresses](https://dictionary.cambridge.org/dictionary/english/express) [their](https://dictionary.cambridge.org/dictionary/english/their) [ideas](https://dictionary.cambridge.org/dictionary/english/idea) [strongly](https://dictionary.cambridge.org/dictionary/english/strongly) and often.  
For example if you include the spring boot starter pom for jpa, you'll get autoconfigured for you an in memory database, a hibernate entity manager, and a simple datasource. This is an example of an opinionated (Spring's opinion that it's a good starting point) default configuration that you can override.  
**Spring created objects are called beans.**  
So in spring object is called bean.  
  
Demo on DI   
without spring how it is ?  




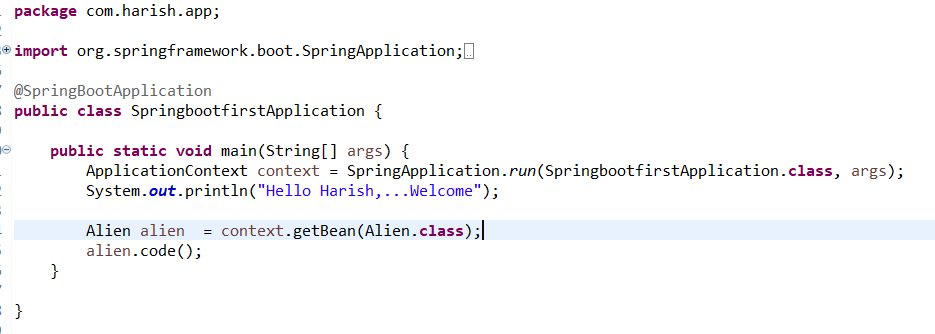


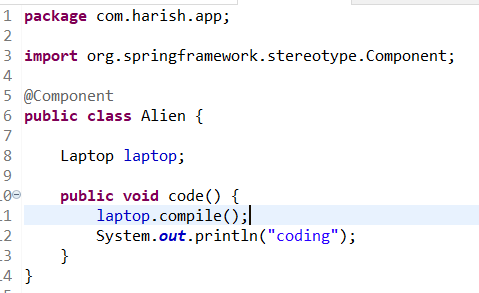
**Using DI**   
Using ApplicationContext you can be able to access objects inside IOC container.  
By default spring will not create objects for you for all classes created, only if you mention only it will create. We can mention annotations **like @Component, @service,@Repository, @Controller**.  


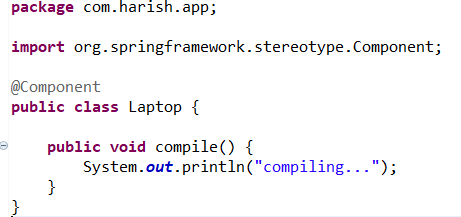
  


**Autowiring** : Earlier using application context we can able to access object in the container. If I want to access object in another class then I need to use autowired. If I don’t use autowire annotation then the object created will be null inside another class.

Without autowire how it is ?



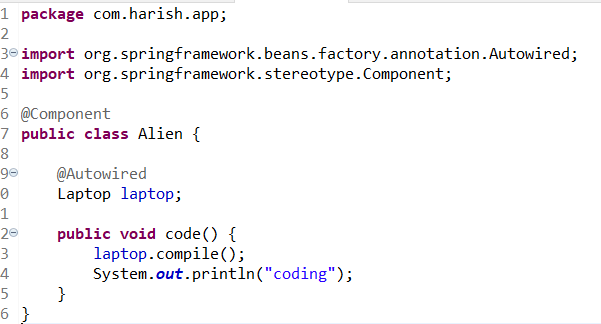
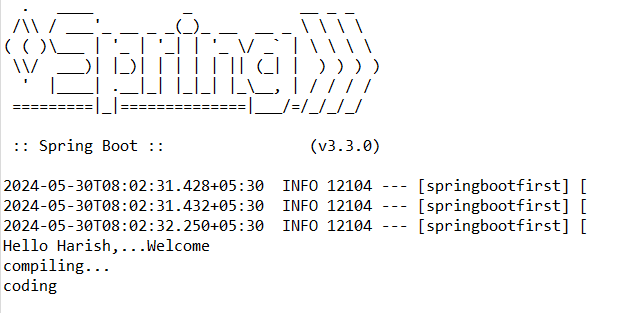




Output :



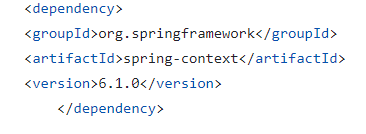
By default calling just laptop inside Alien class will not bring object from IOC container. To bring it from **IOC container we need to inform spring by using @Autowired annotation.**

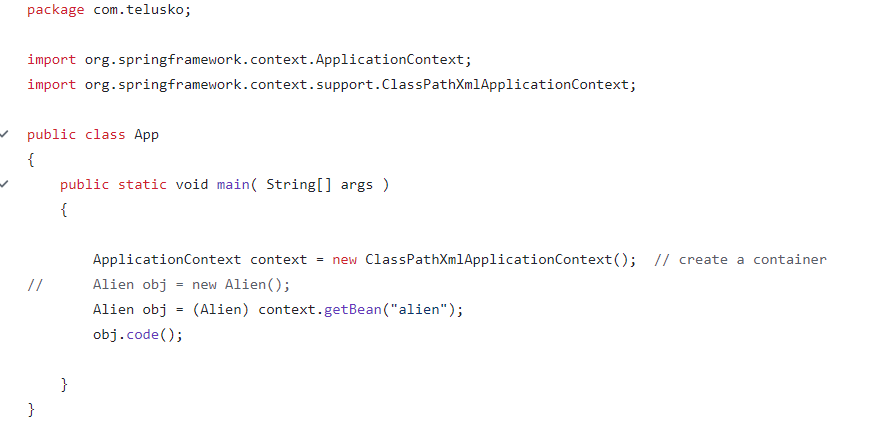
Using now @Autowired  
  
  
output:  


May 31,2024

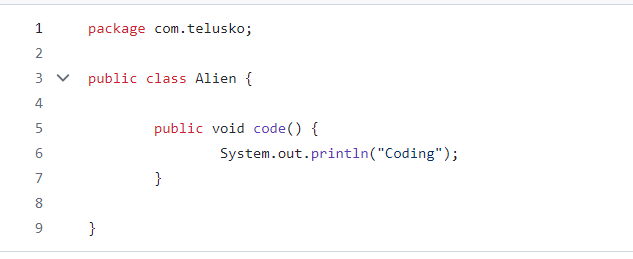
**Spring Framework**

**XML based configuration :**

**We need application context to access beans from container.  
so to include it we need to add a dependency.**

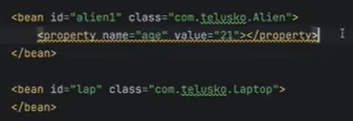


**By default application context will return object, so we need to type cast it.**





Under src/main/resources you need to have spring.xml with the above content.  
by default the scope of the bean is singleton and its get loaded in the container when the application starts.  
So even if you don’t call any getbean in main file it will execute the constructor.  
For the single ton scope bean, if you create multiple beans for same class it will point to the same object which you have created first.  
If I mention scope = prototype in xml for particular bean then on application start it will not load, only when get bean method is called it will load and execute the constructor. And in this case two different beans has been created for us.



This will assign age with 21, by using setter method.

@Qualifier annotation takes precedence over @Primary annotation.