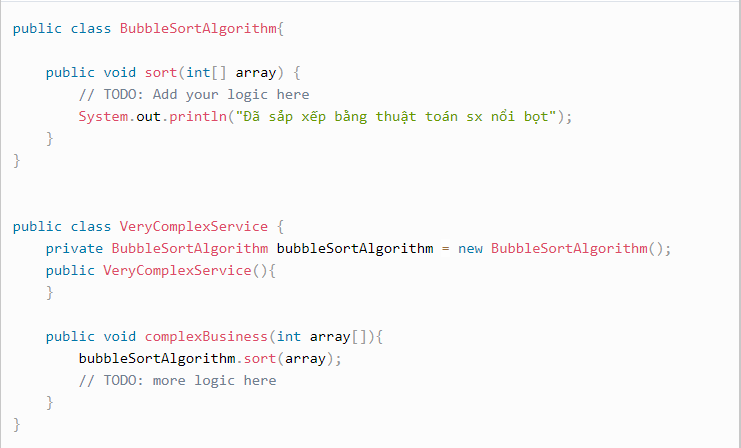
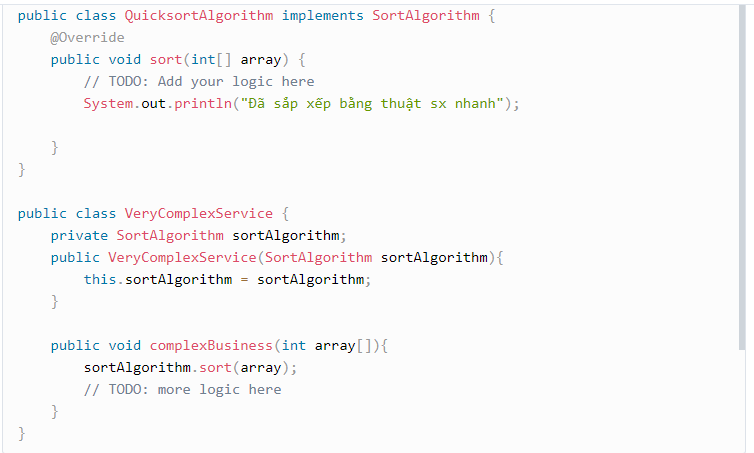
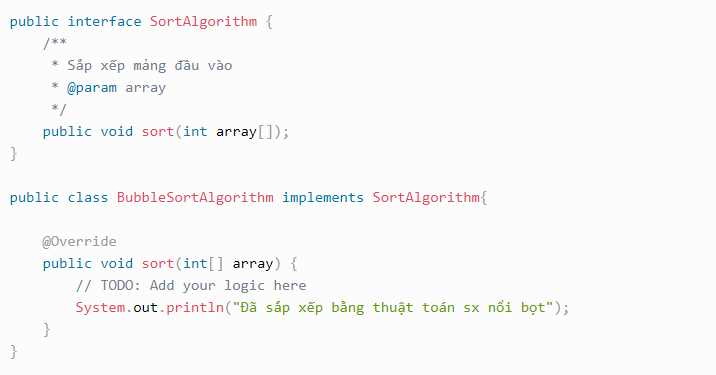
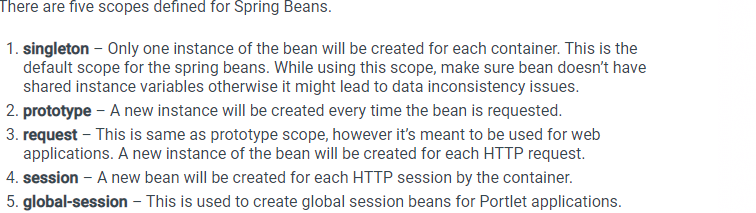
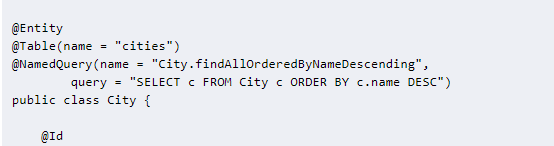
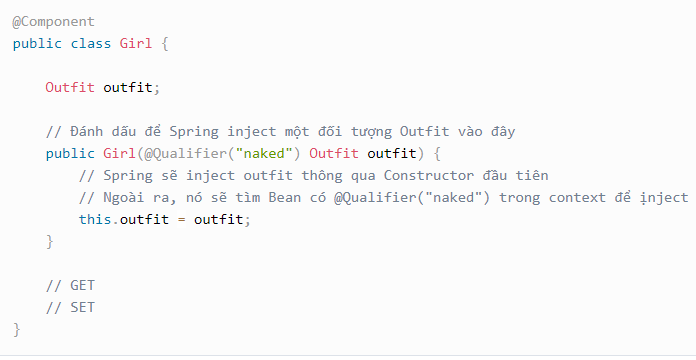
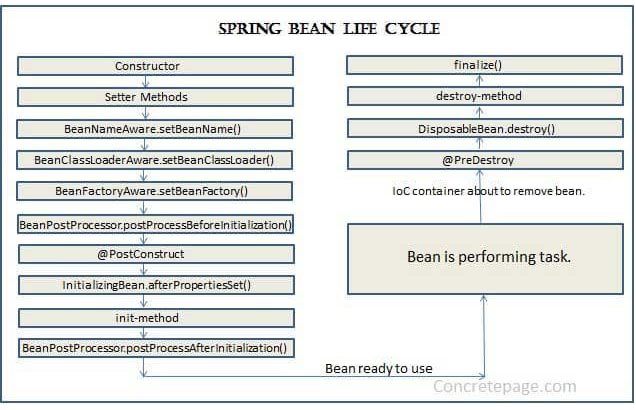
# Dependency Injection

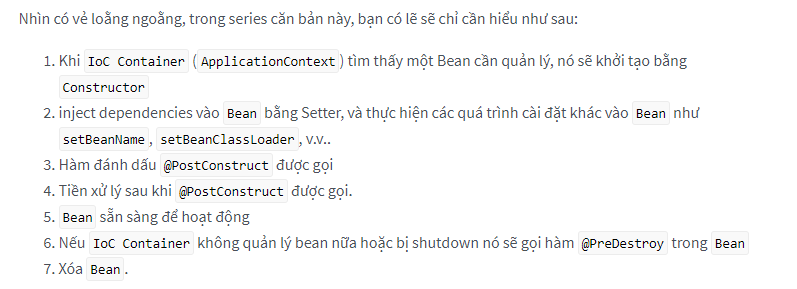
* Wrong: 
* When they want u QuickSort instead of BubbleSort :
  + VeryComplexService a = new VeryConplexSerivce();
  + a.complexBusiness(array);
* Have to create new method to do QuickSort and new instance of QuickSortAlgorithm
* Dependency Injection: Object should depend on Abstract class an Abstract class’s implementation is injected to object in runtime: 2ways: setter injection, constructor injection(this image)
* Inversion of Control: Define all dependencies used in project and put it in a container, and framework has a responsibility to control it. When a class needs, framework will inject it.
* *Inversion of Control is a programming principle. flow of control within the application is not controlled by the application itself, but rather by the underlying framework.*
* In Spring: container is ApplicationContext, dependency is Bean (bean is singleton): @Component.

# Basic concepts

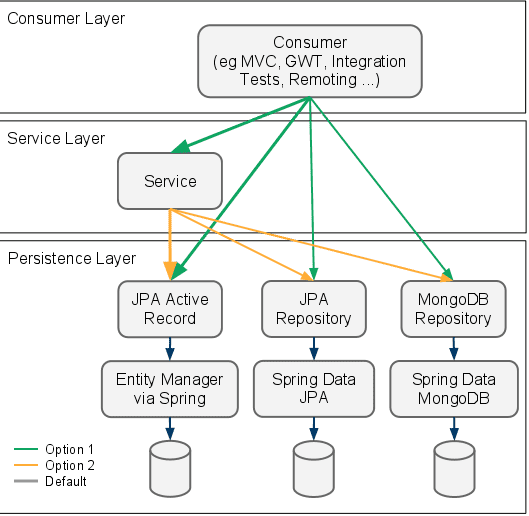
* Spring bean scope: 
* Spring bean configuration: Annotation, xml
* @ComponentScan: normal, spring container search all beans , but if u want container search a specific package => use this 

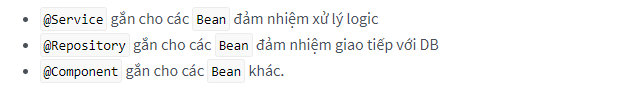
# Annotation

* @NamedQuery: is used on model class, used when it has few custom query
* @Modifying: As EntityManager might contain outdated entities => so use it to clear. (used in insert, update, delete)
* Use DeleteInBulk when delete large number of records. Coz DeleteById deletes the returned instances one by one.
* @Query: is used on repository class, when has many custom queries
* Inject by @Autowired.
* @Primary is when 2 or more @Components implement an interface. So when we @Autowired => Spring cant know which @component to inject=> @Primary give highest priority to choose.
* @Bean in method(Must have @Configuration in classs). @Component in Class (Controller, repository, service are specialization of @Component).
* @SpringbootApplication: in main class.
* @Qualifier: To solve problem above.  
* Life cycle of Bean: 



# SpringBoot architecture



* Consumer Layer is a layer which communicates with outside and handles requests from outside to the system
* Service Layer: handles business and logical process
* Repository layer: handle query, communicate with db, return data requested by SeviceLayer. 

# @Controller

# 

