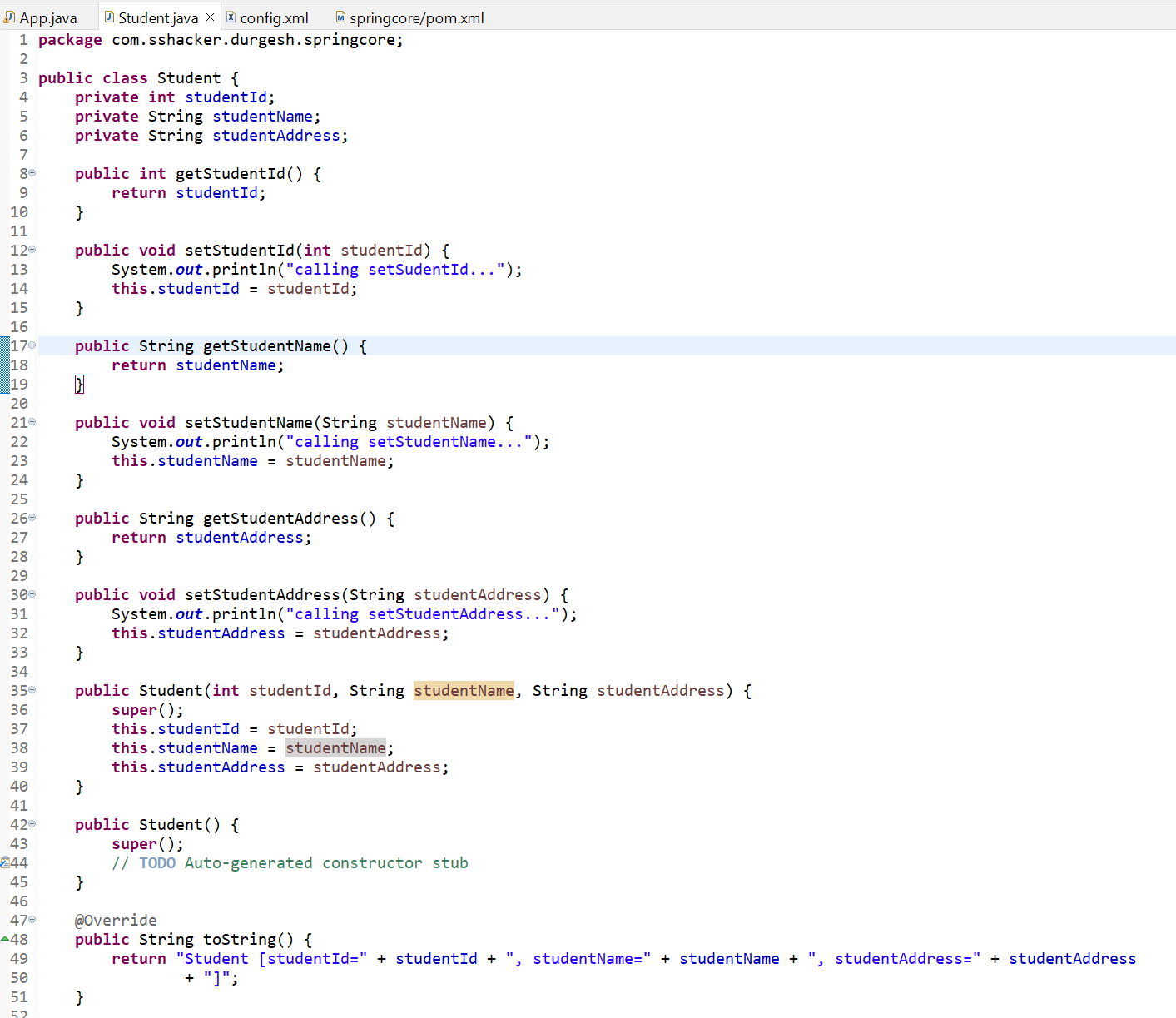
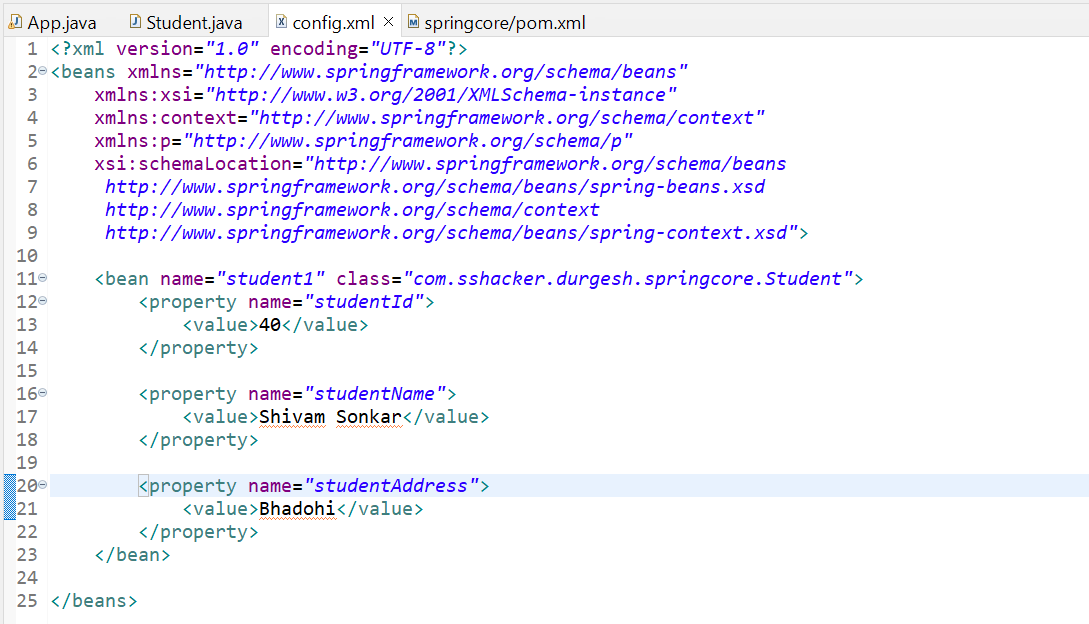
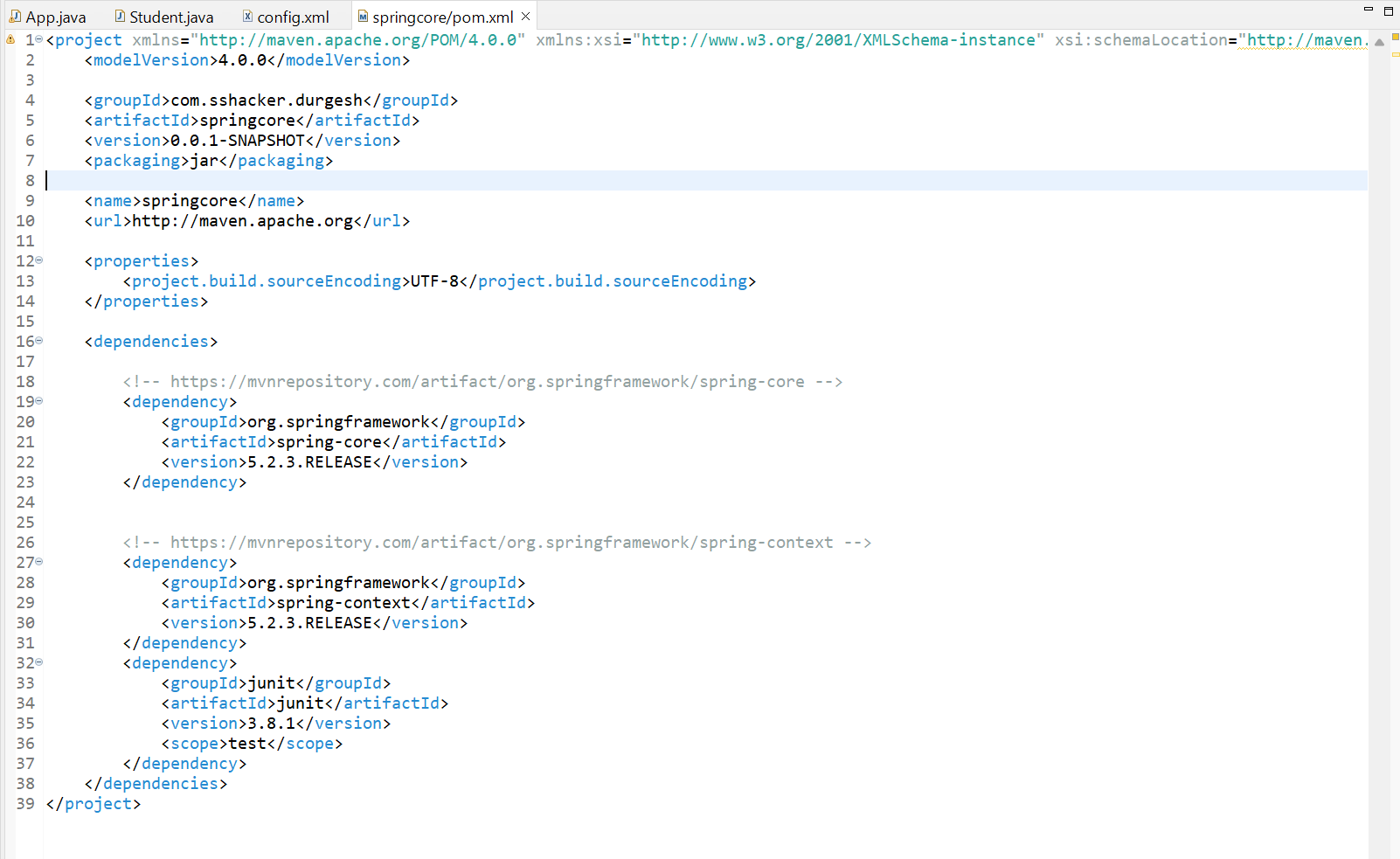
1. New Maven Project | Adding Spring Dependencies | Create Config File | Setter Injection | Practical



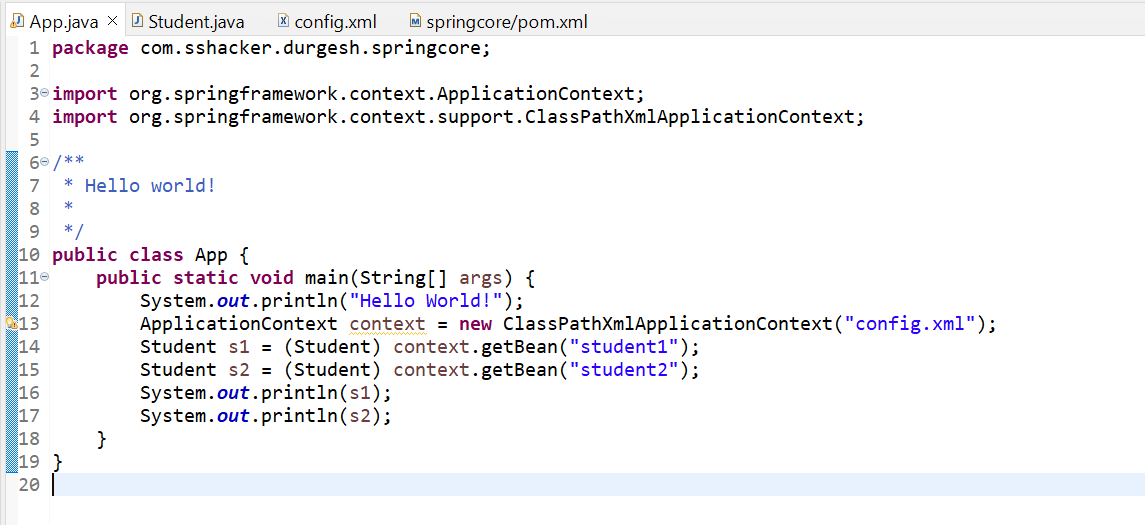


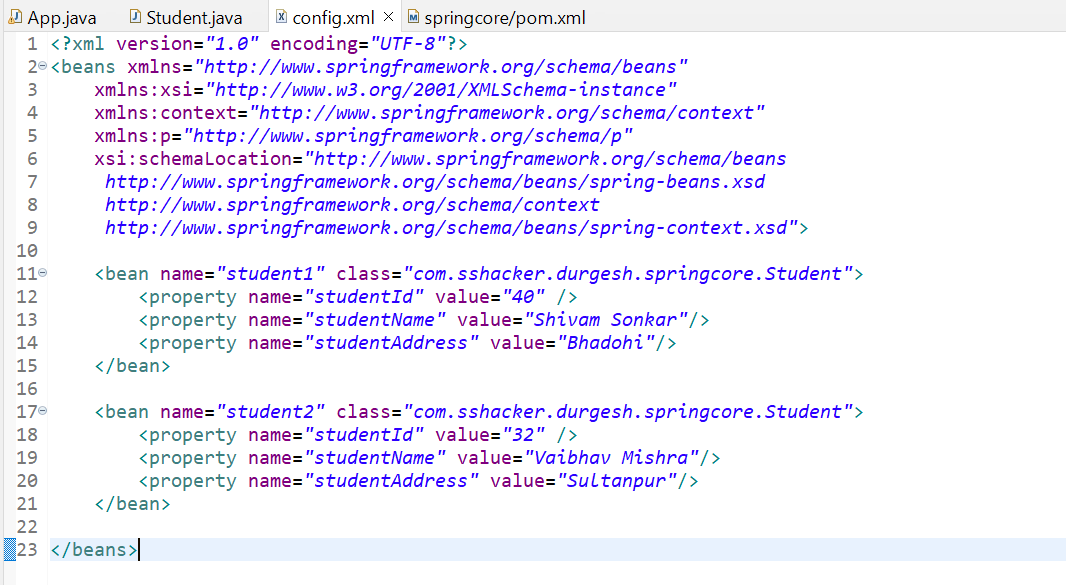




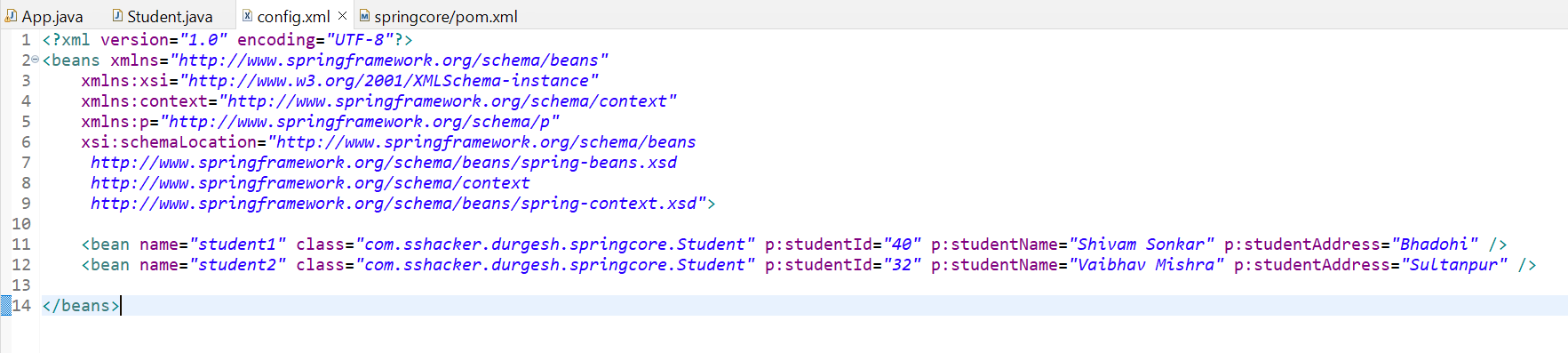
# Property injection using p Schema and using value as attribute | Spring tutorial

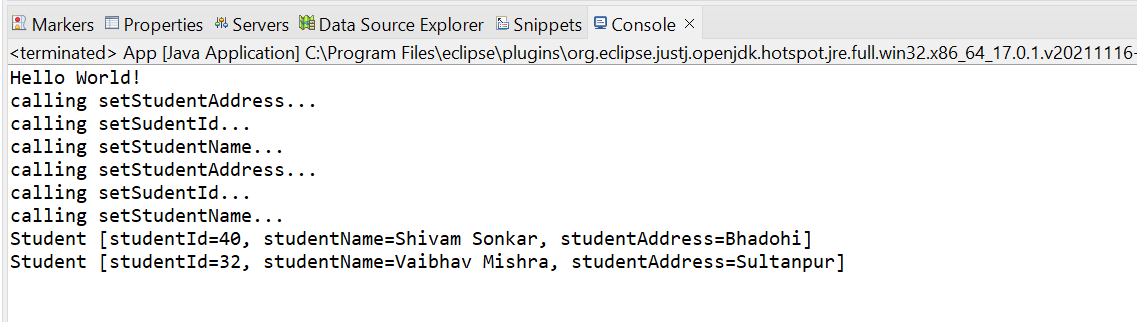
1. Using value as attribute: -



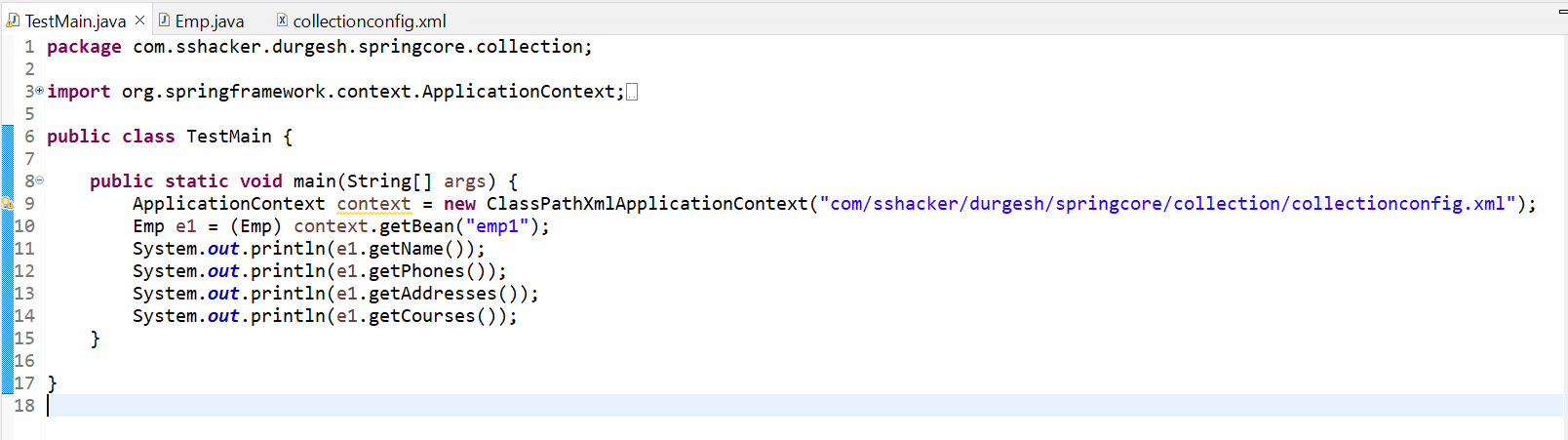


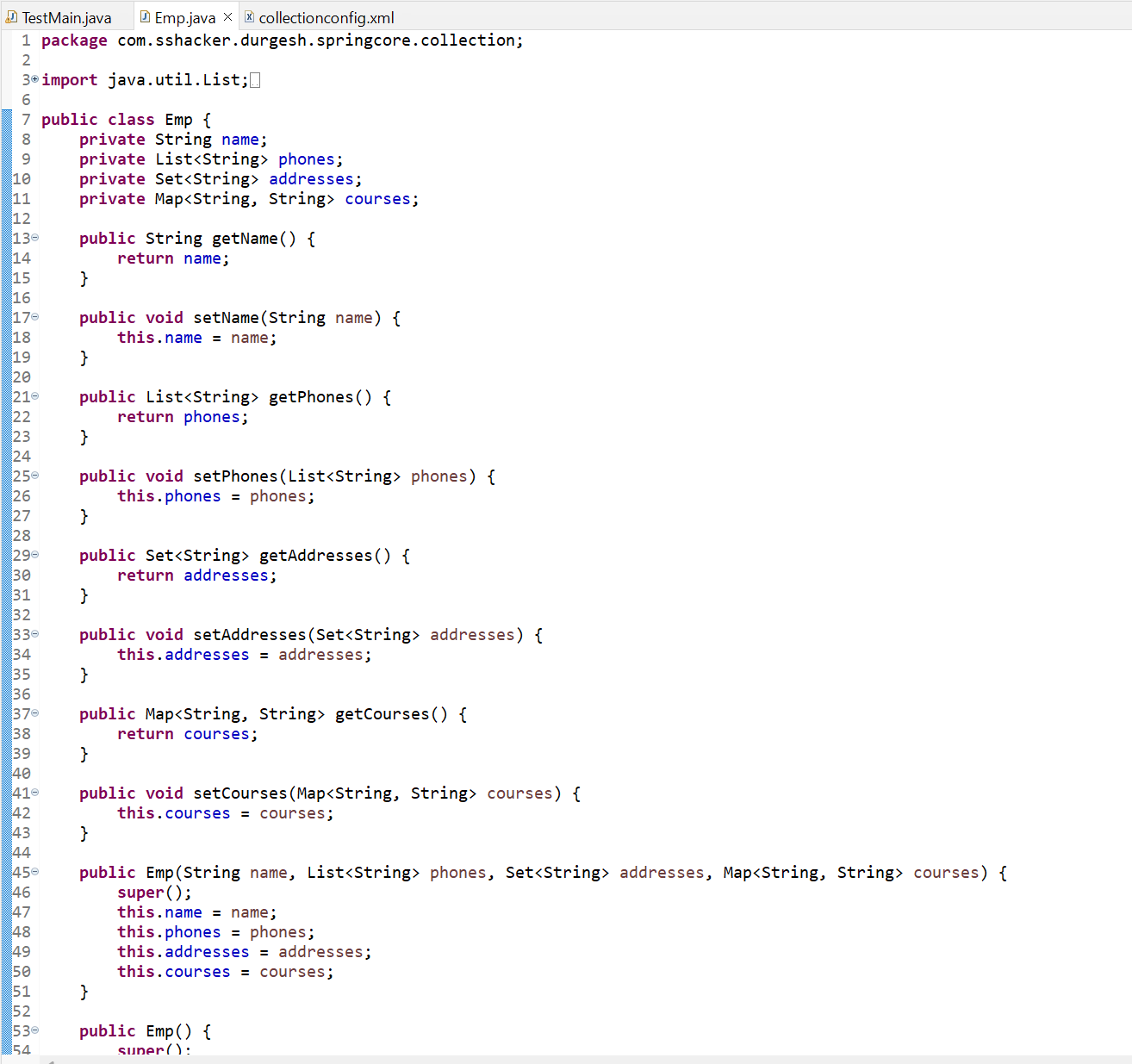
1. Using p schema: -



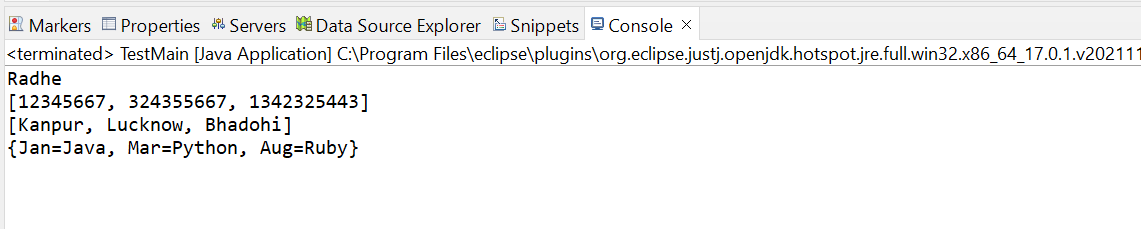


# 9. Injecting Collection Types[ List , Set , Map , Properties ] | Practical | Spring Framework Tutorial





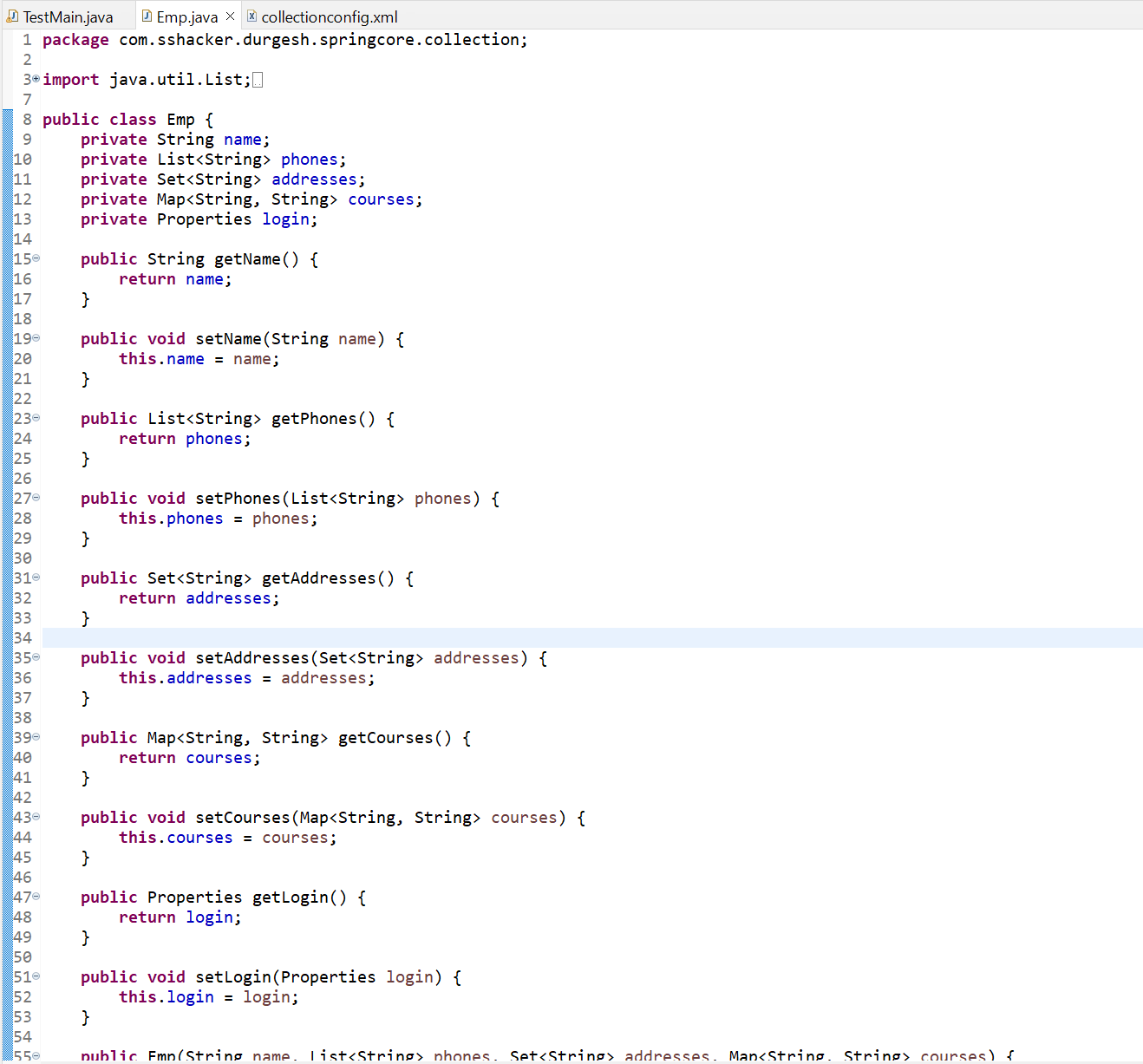




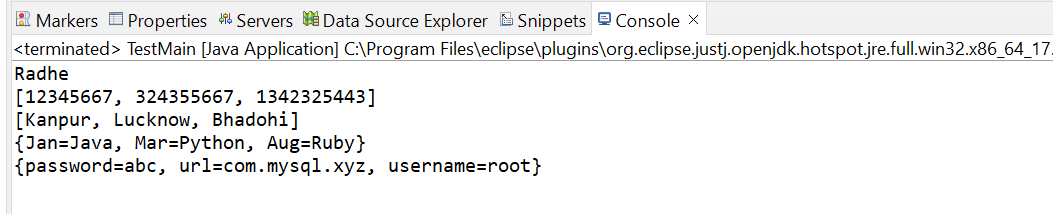


**For Properties Collection: -**

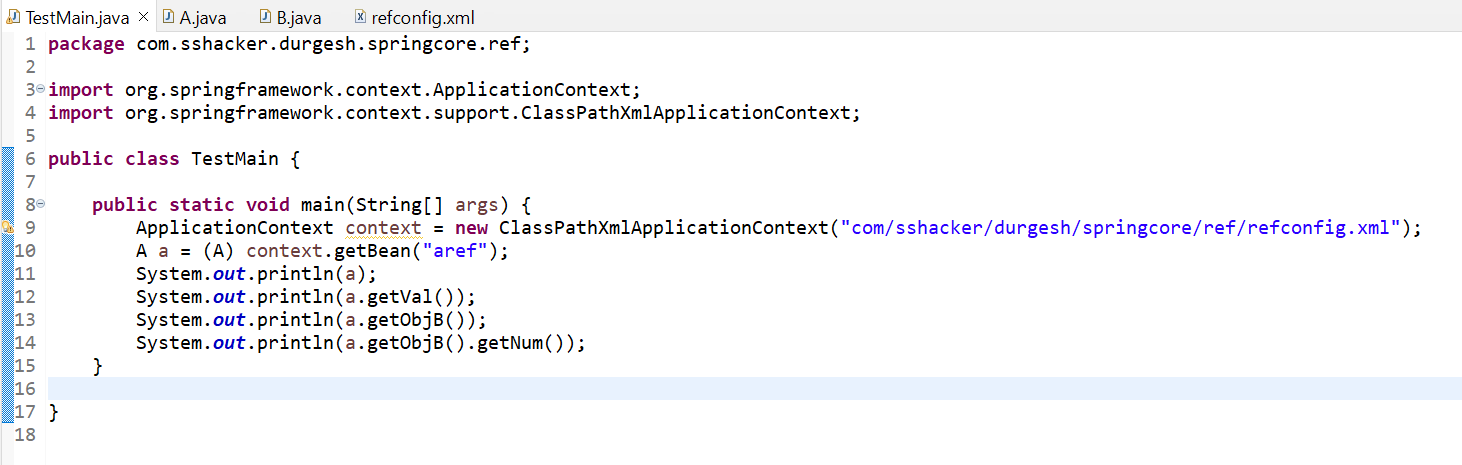


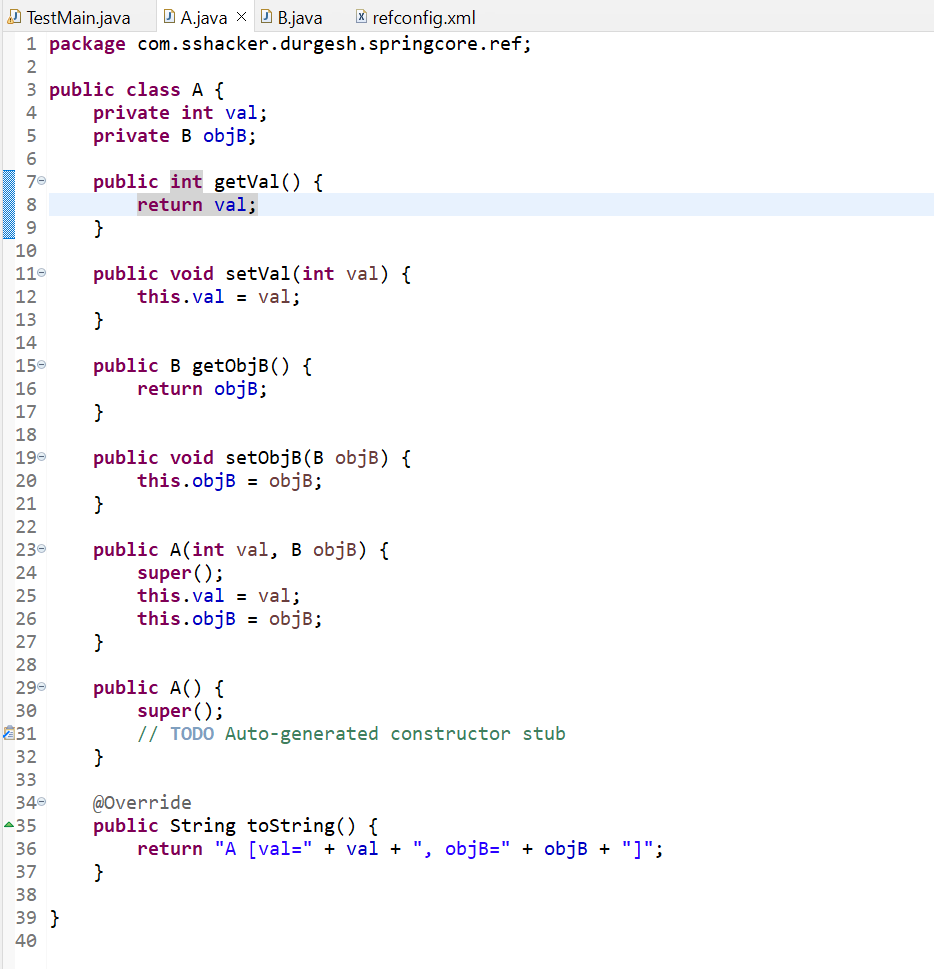


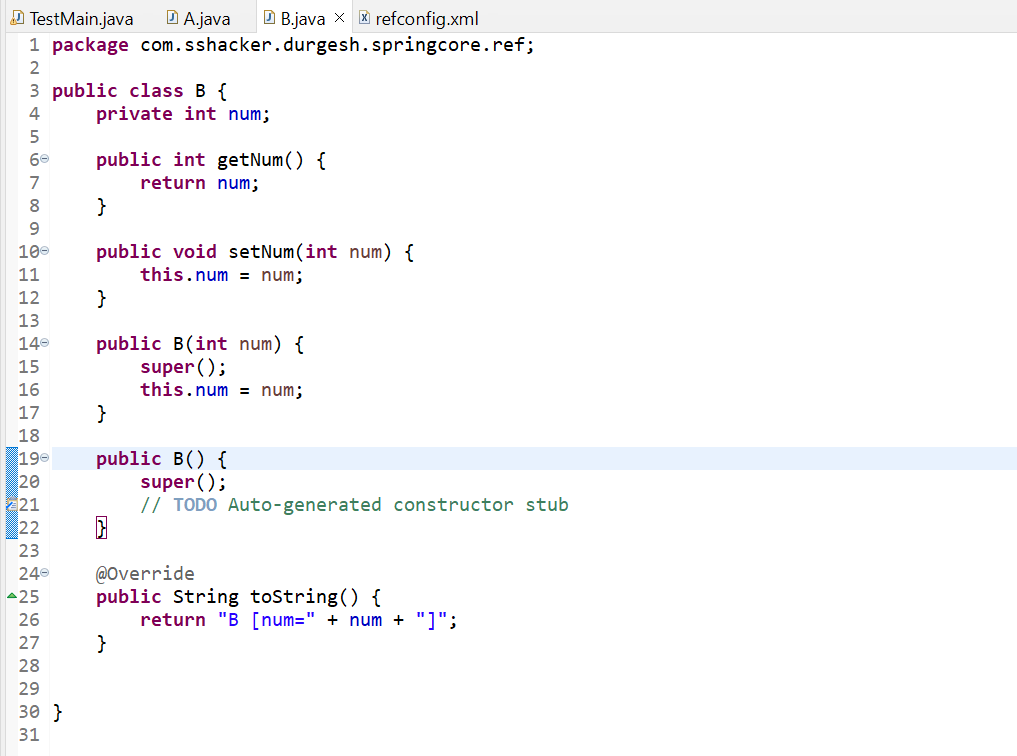


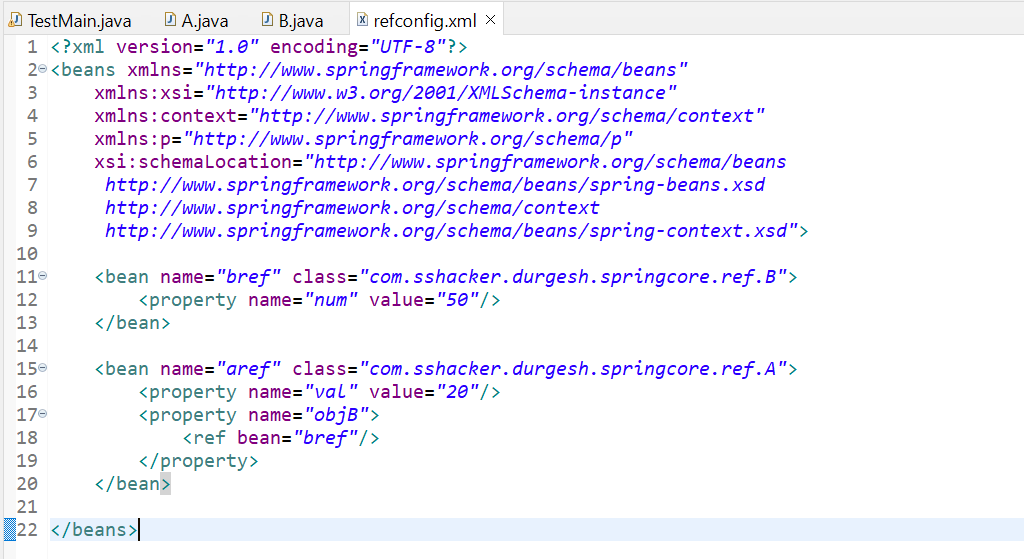


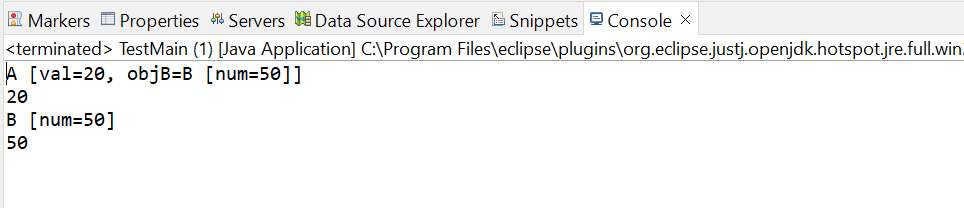
# 10. Injecting Reference Type Practical Video | Spring Framework Tutorial



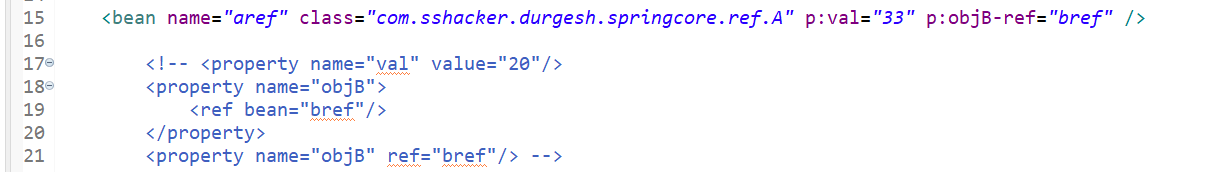




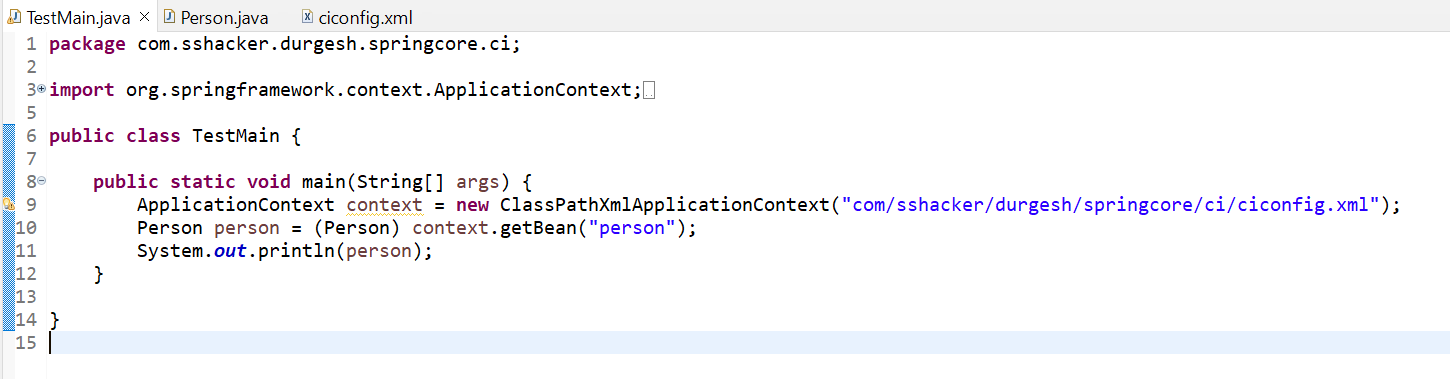


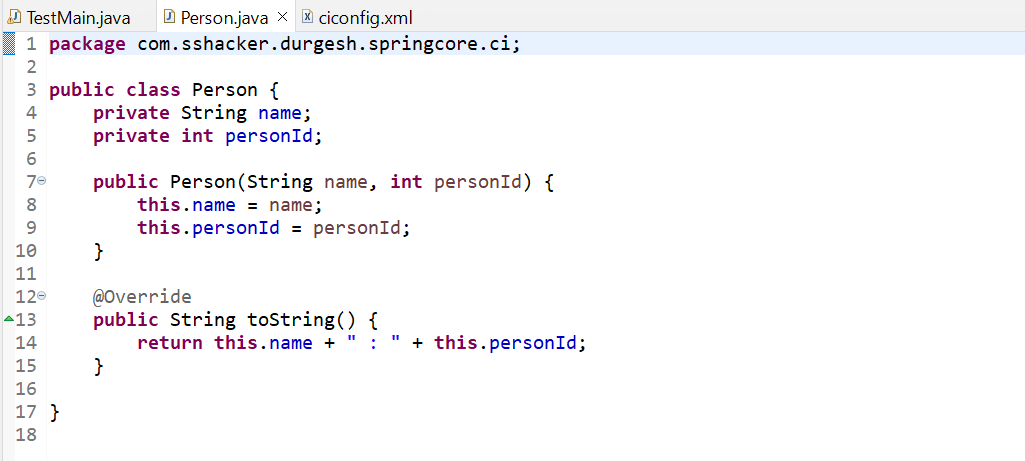


**Using ref as attribute and p schema**

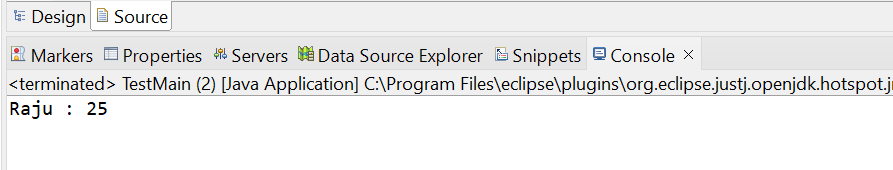


# 11. Constructor Injection Complete Explanation | Spring Framework Tutorial

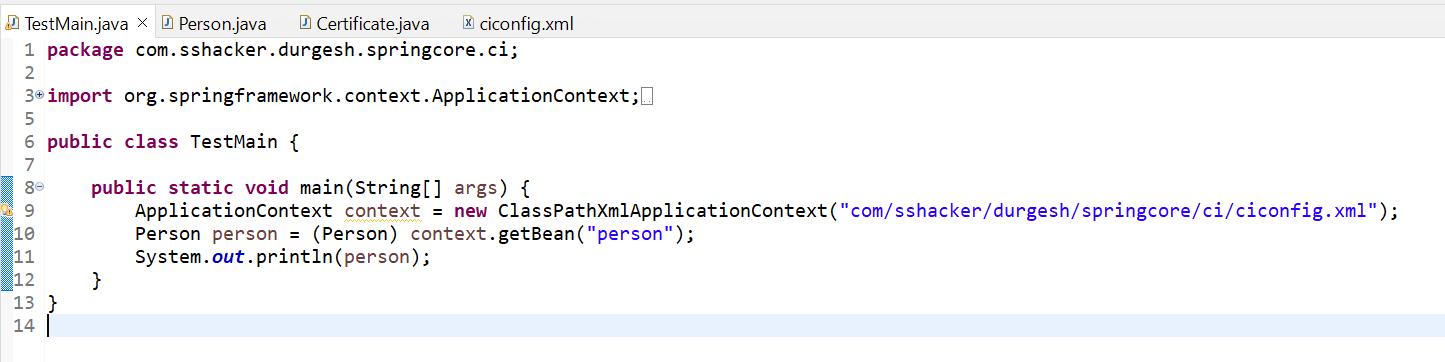


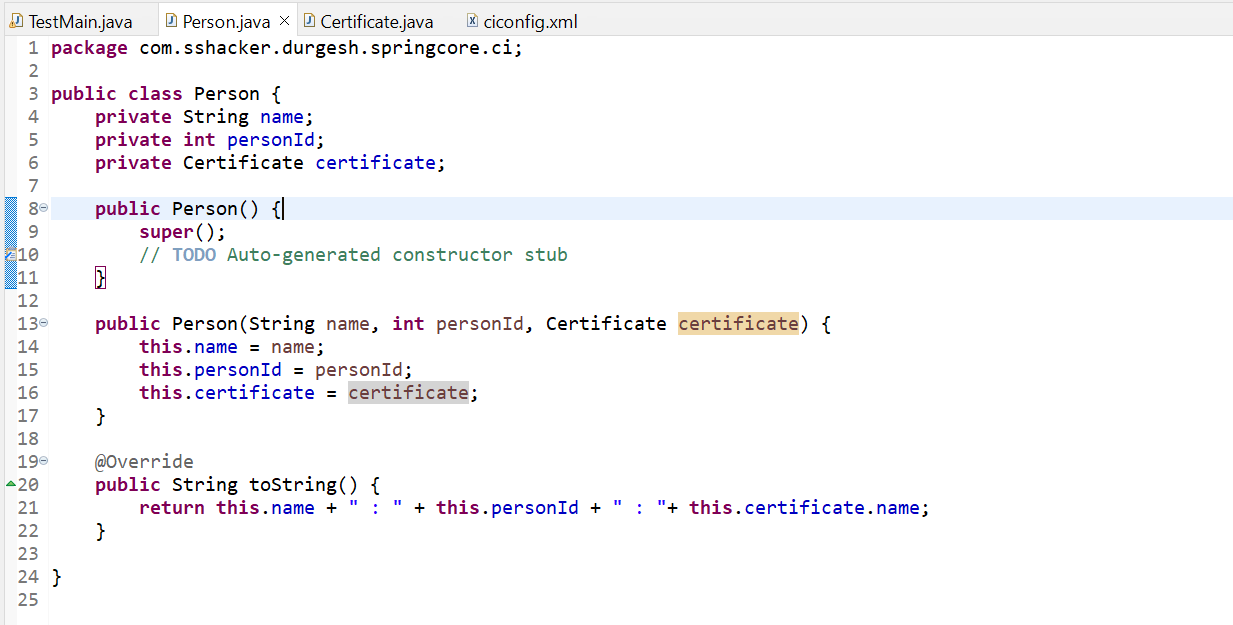


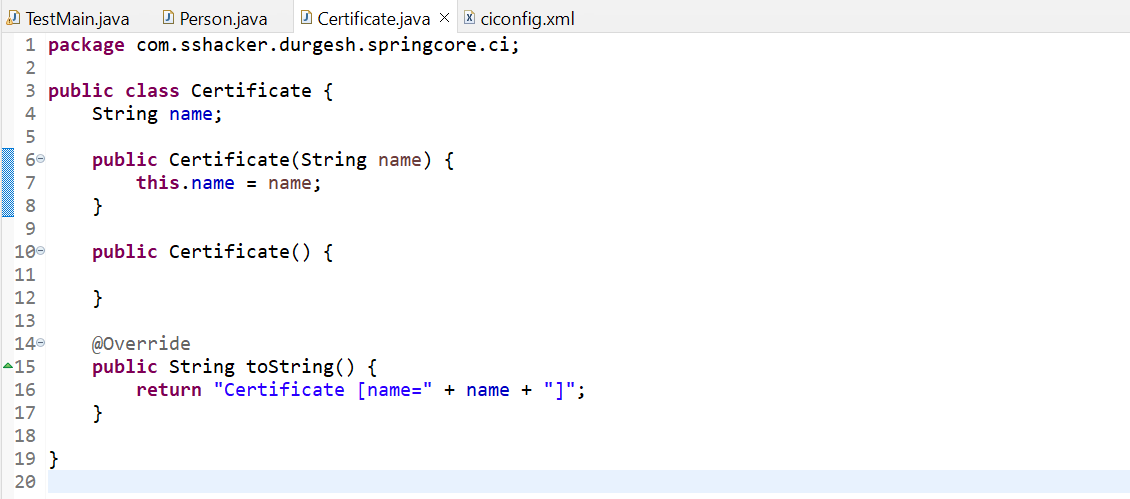


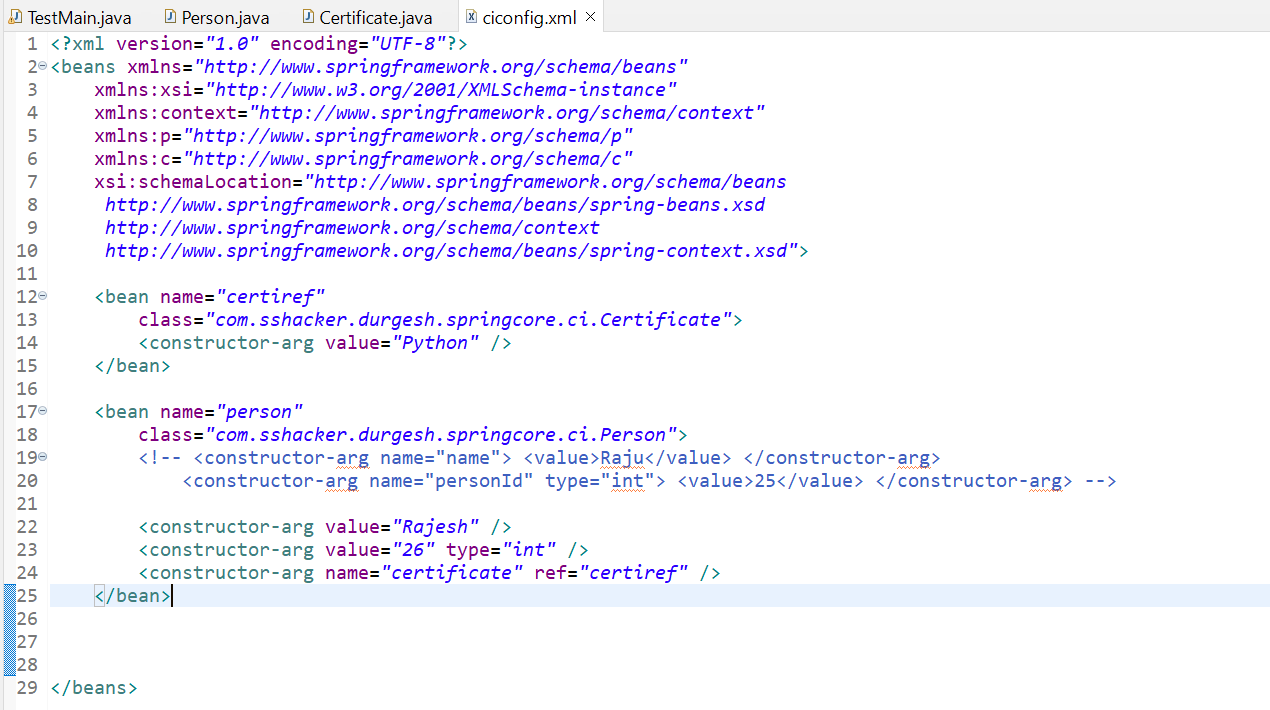


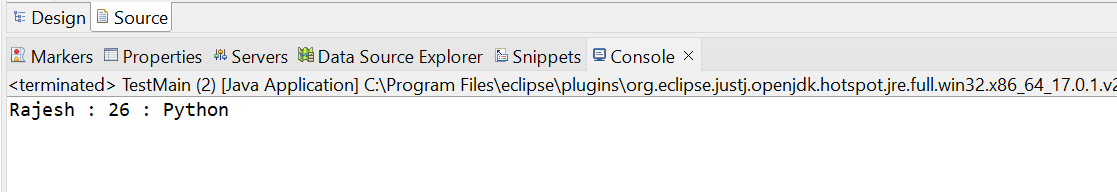
**Constructor Injection with ref**

****

****

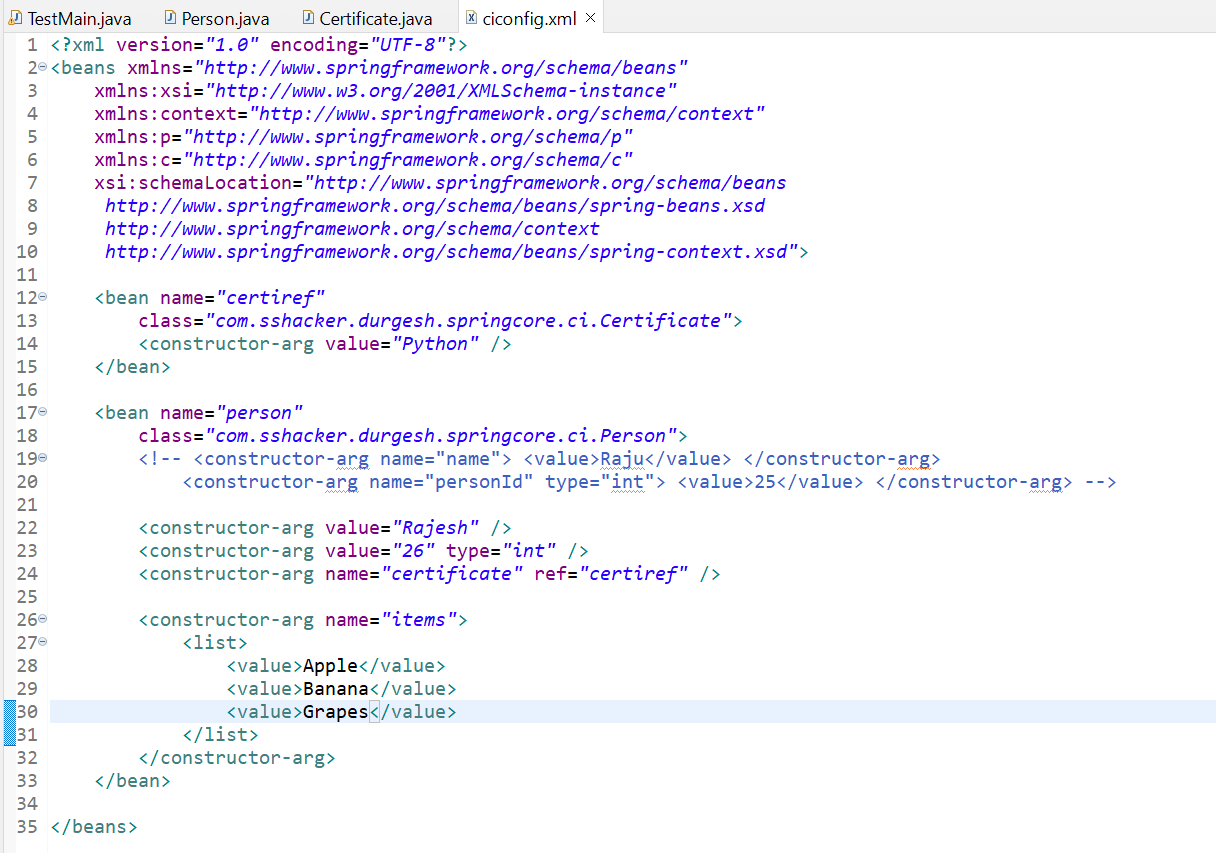
****

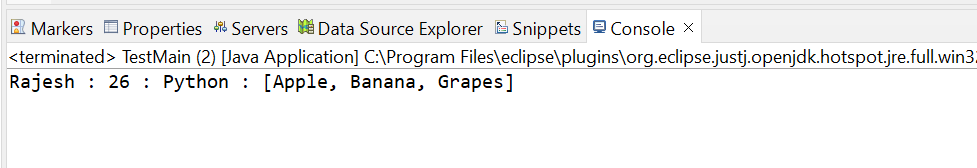




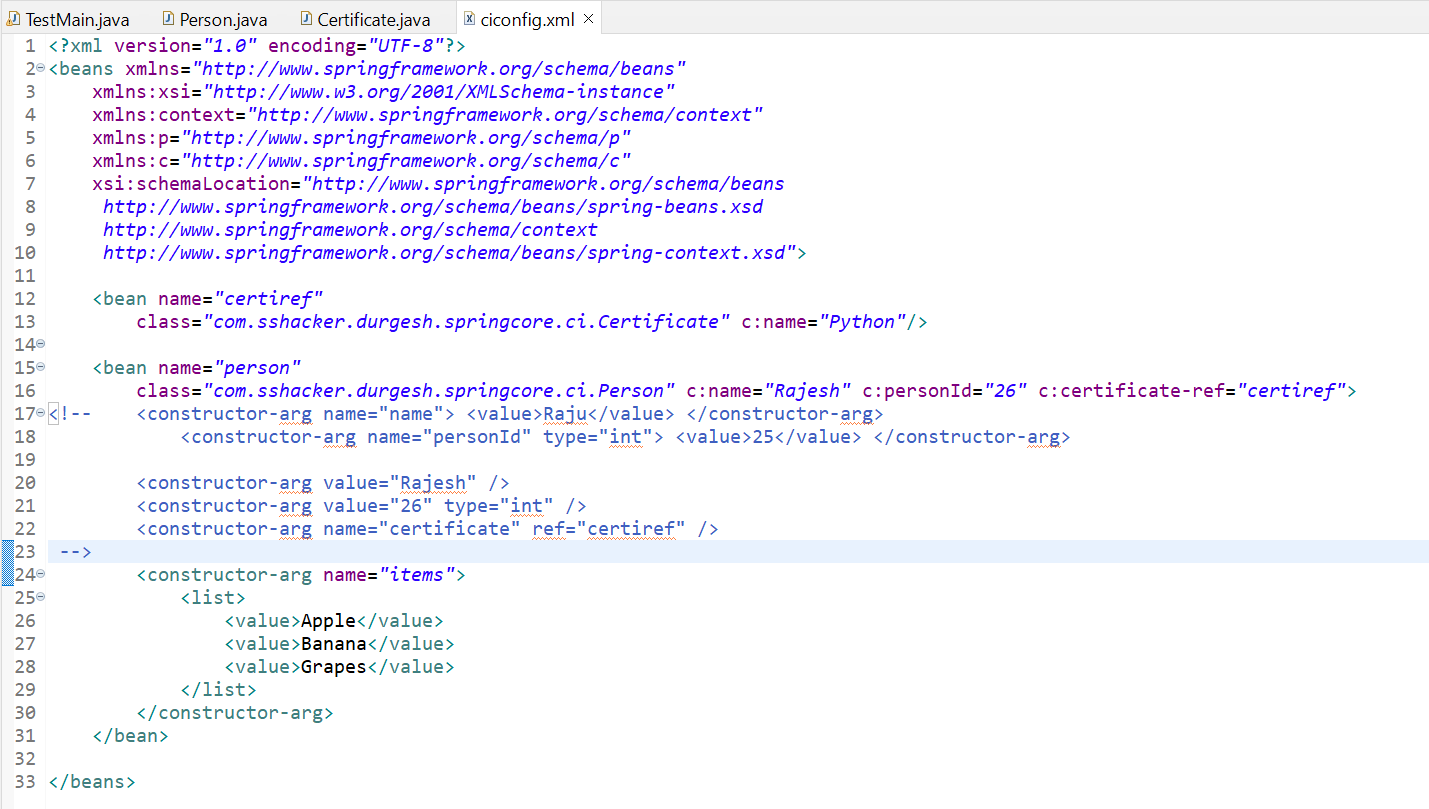
Passing list using Constructor injection: -







**Using c schema for Constructor injection: -**

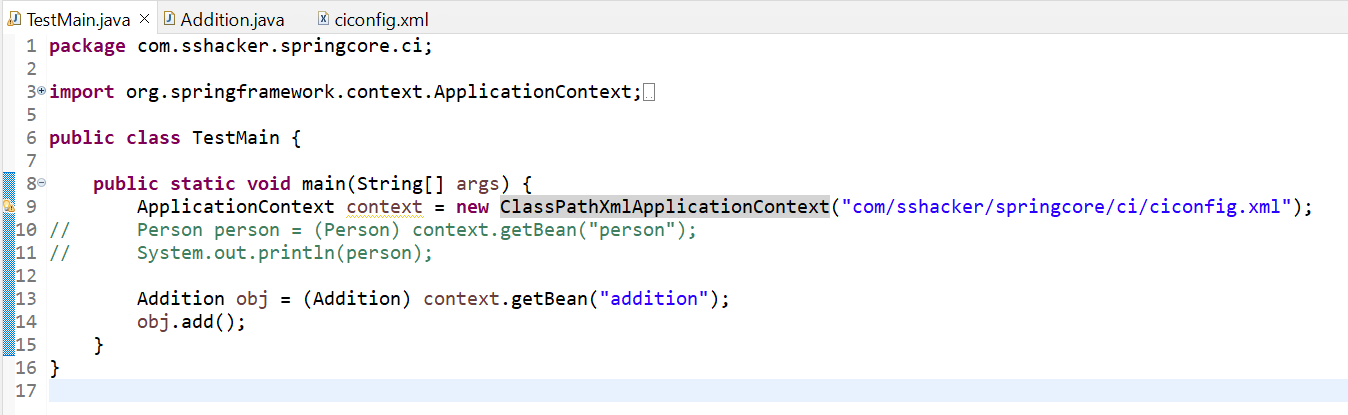


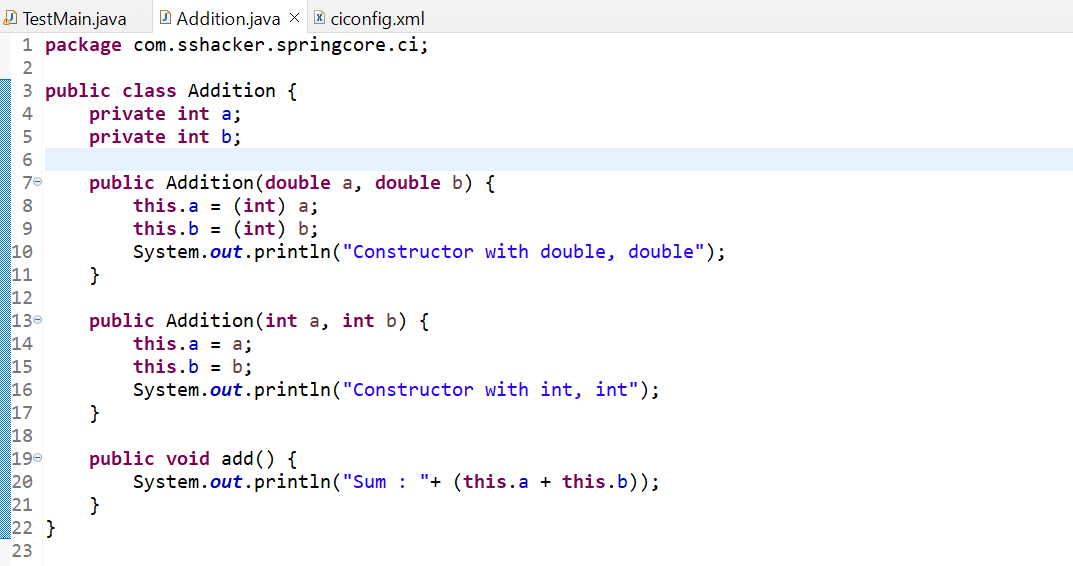
# 12. Ambiguity Problem and its Solution with Constructor Injection | Spring Framework Tutorial

Sequence Of Resolution: -

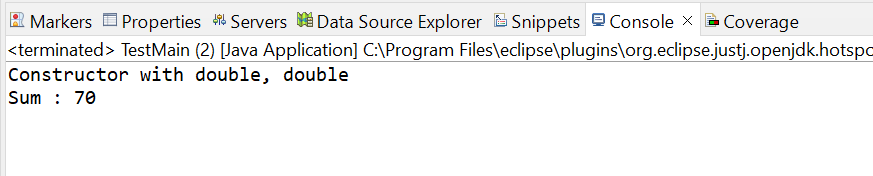
1. By exact match
2. By Type
3. By Index
4. By Order

**By Order: -**

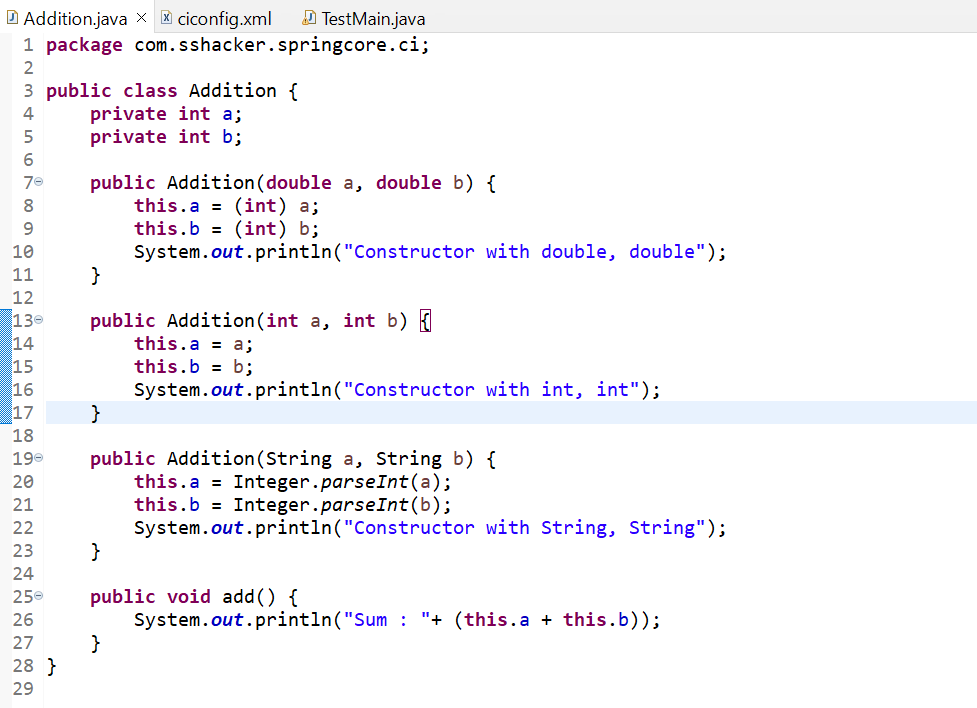


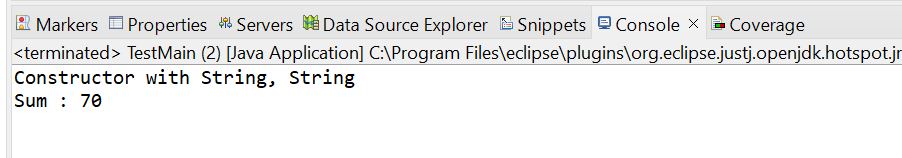






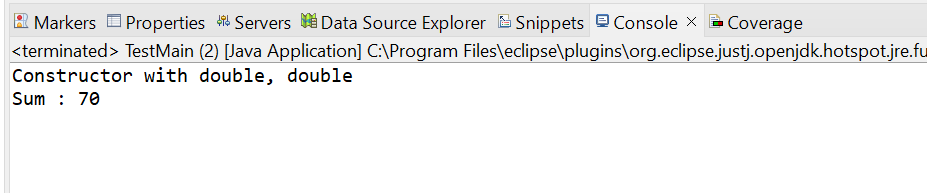
**By exact match: -**



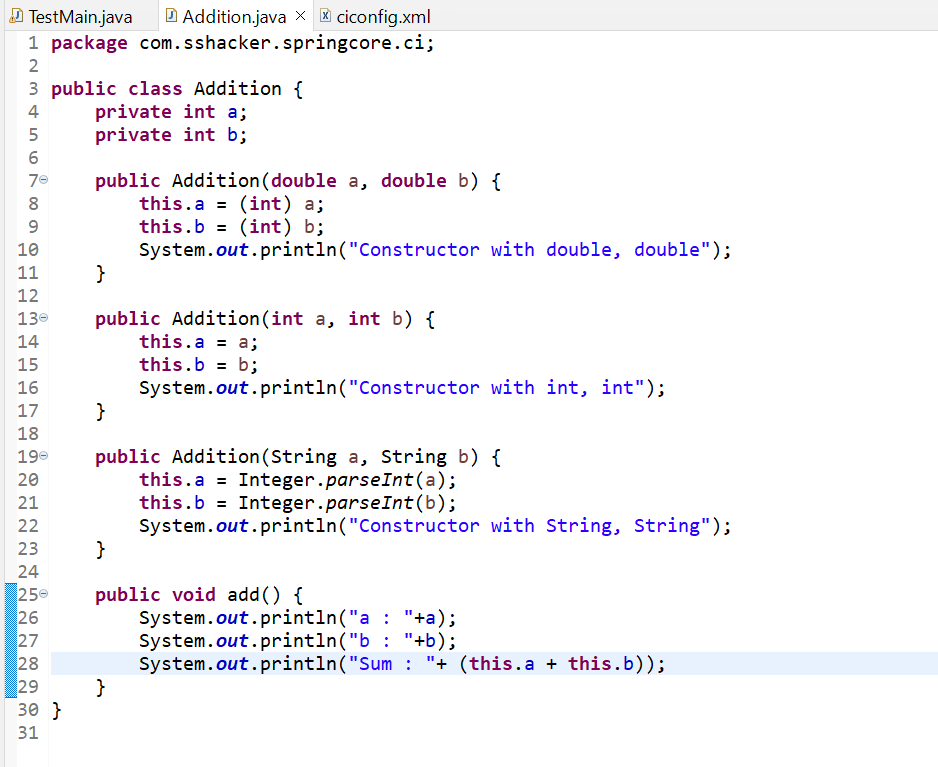


**By Type: -**

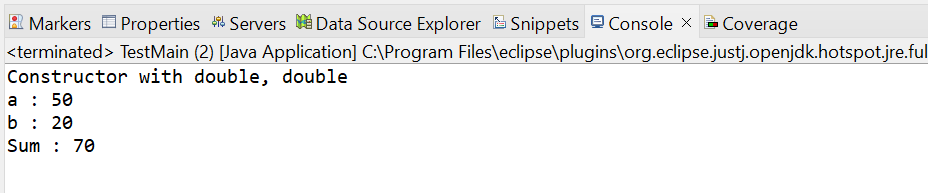




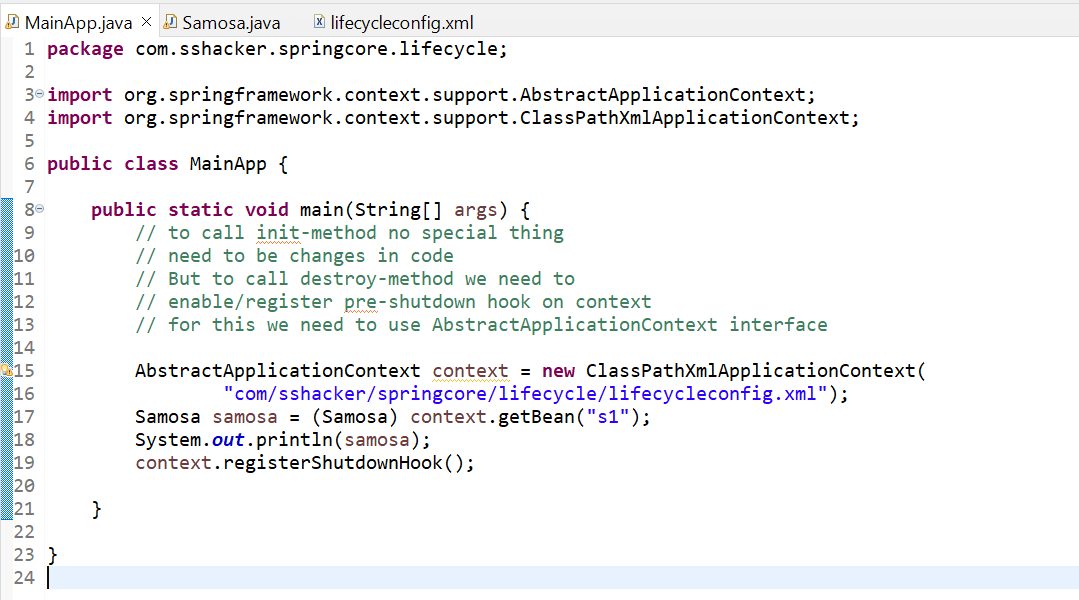
**By Index: -**

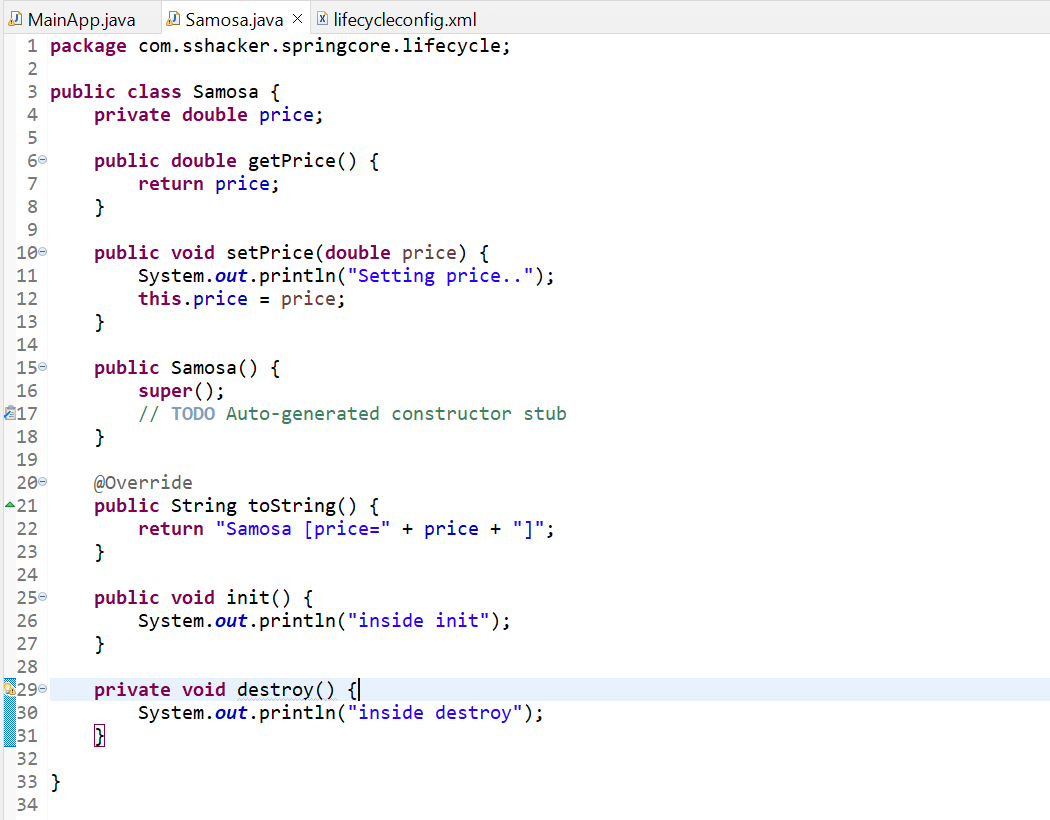
****

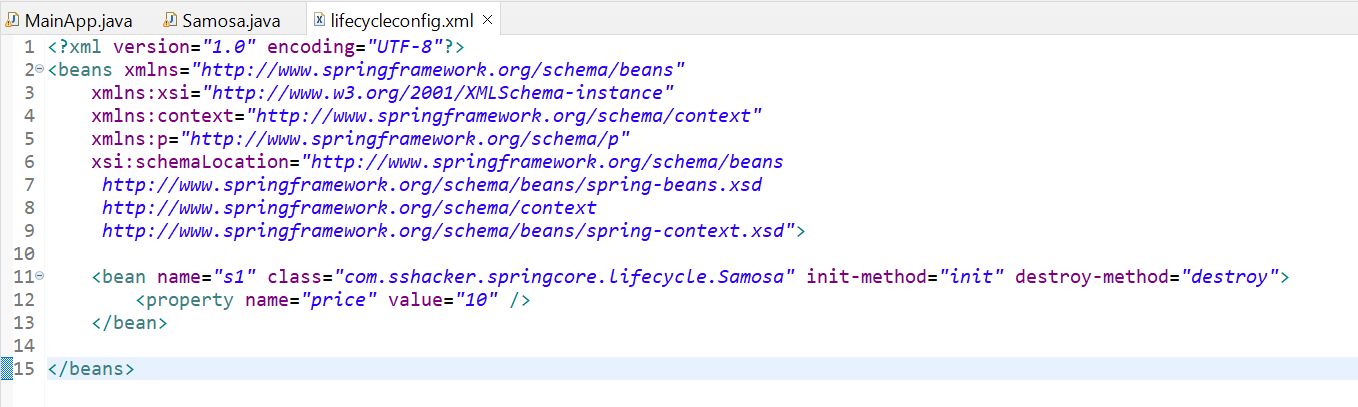
****

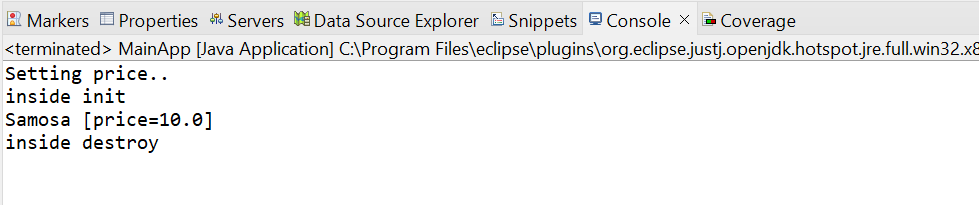
****

# 14. Implementing Lifecycle methods Using XML | Spring Bean Lifecycle methods | Spring Framework Tut

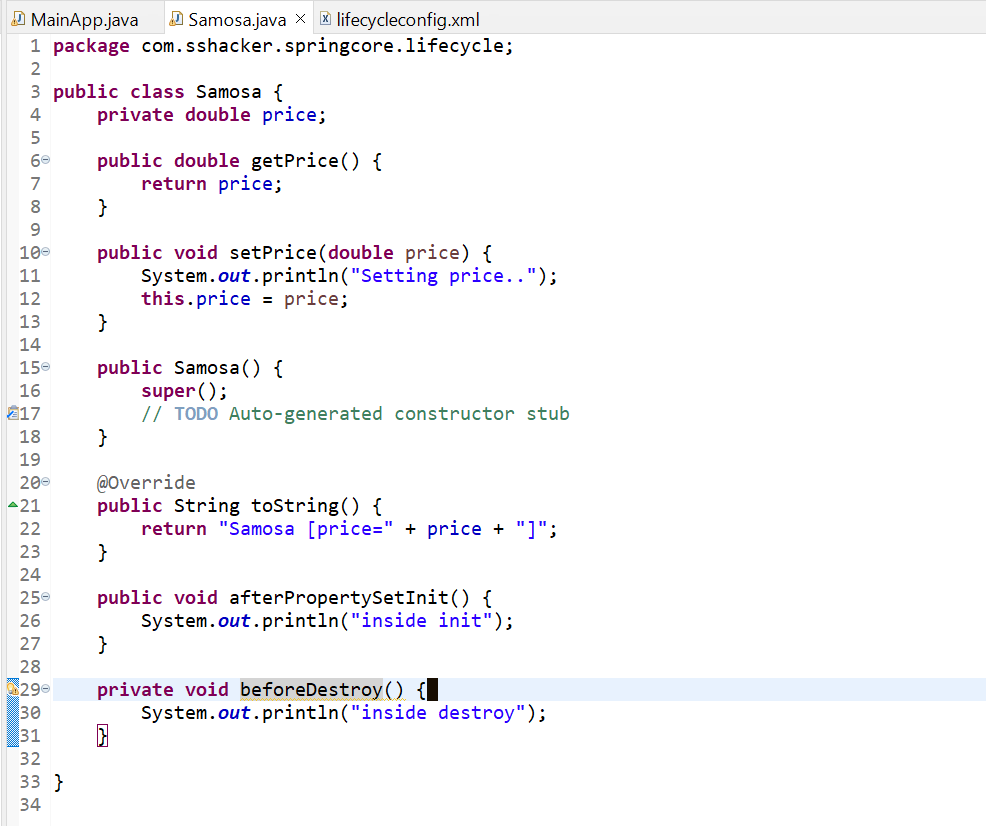
****

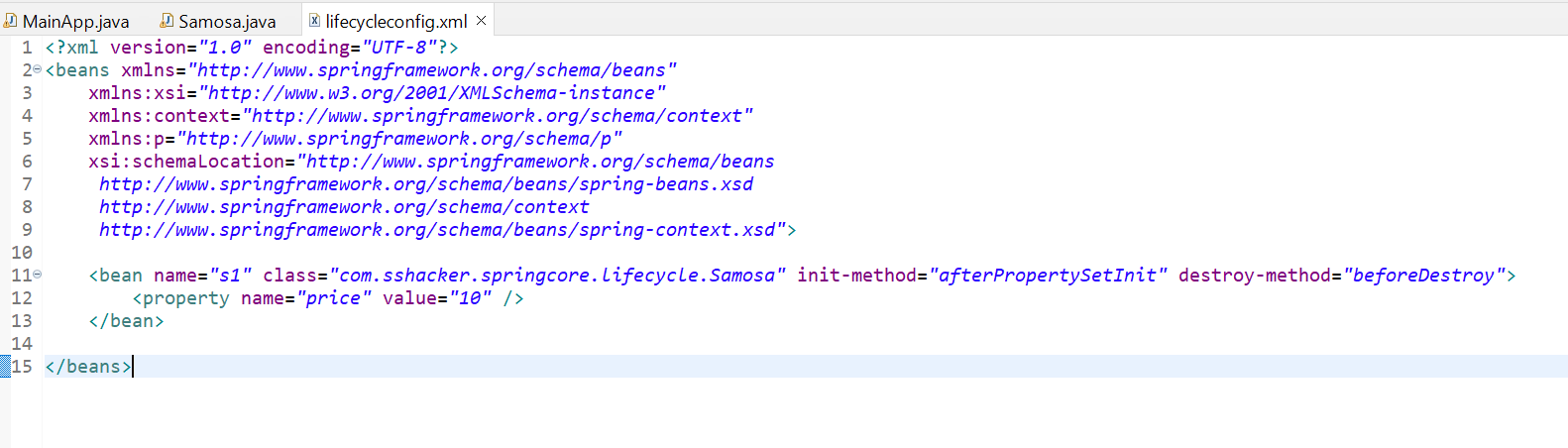
****

****

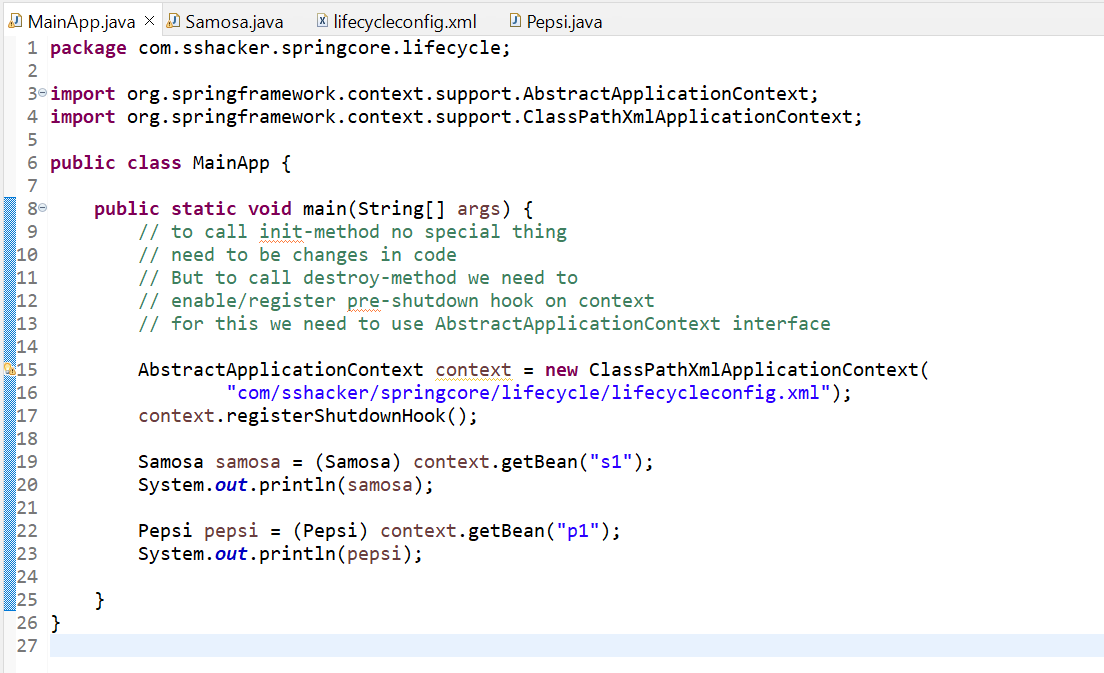
****

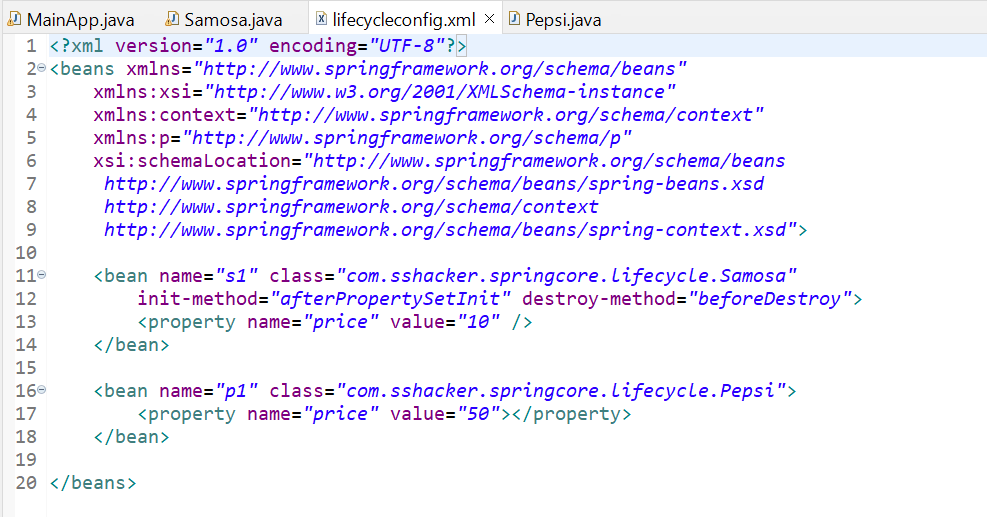
**By taking any name for init and destroy method: -**

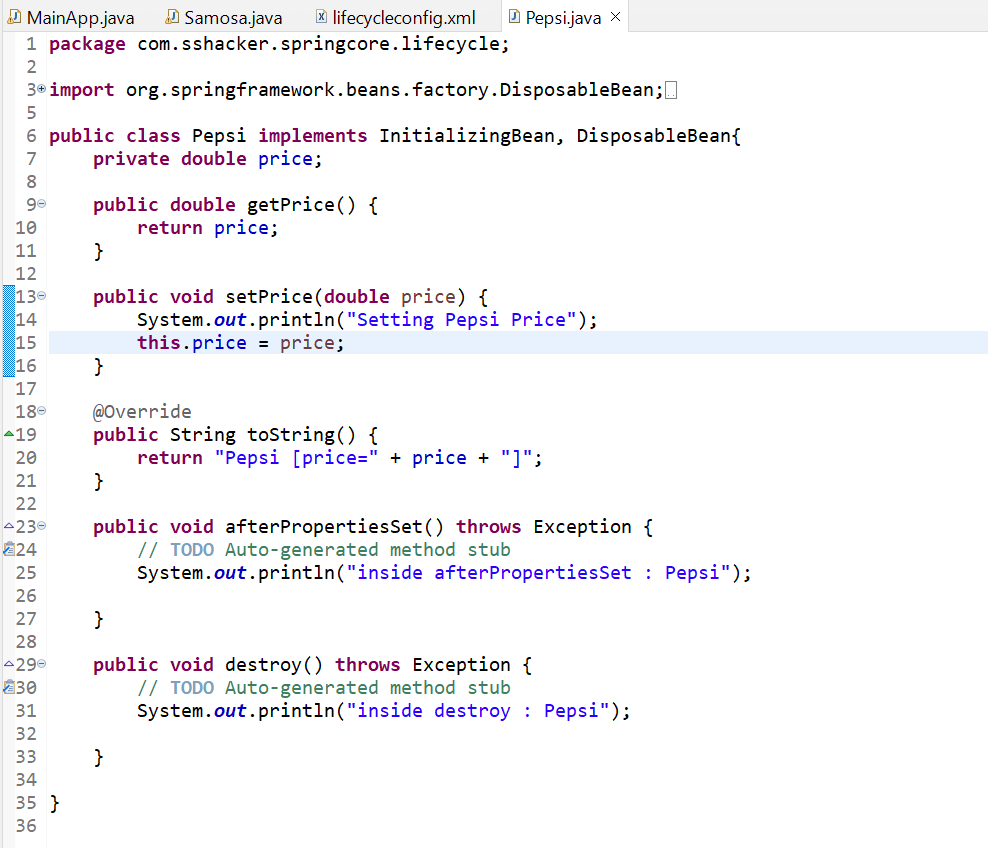
****

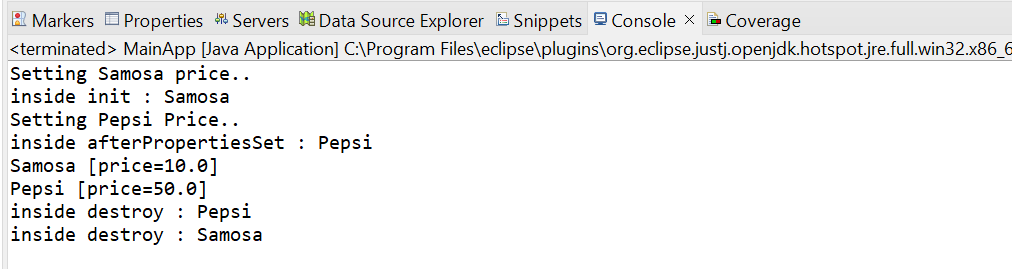
****

# 15. Implementing bean life cycle using interfaces | IntializingBean | DisposableBean | Spring Tutorial

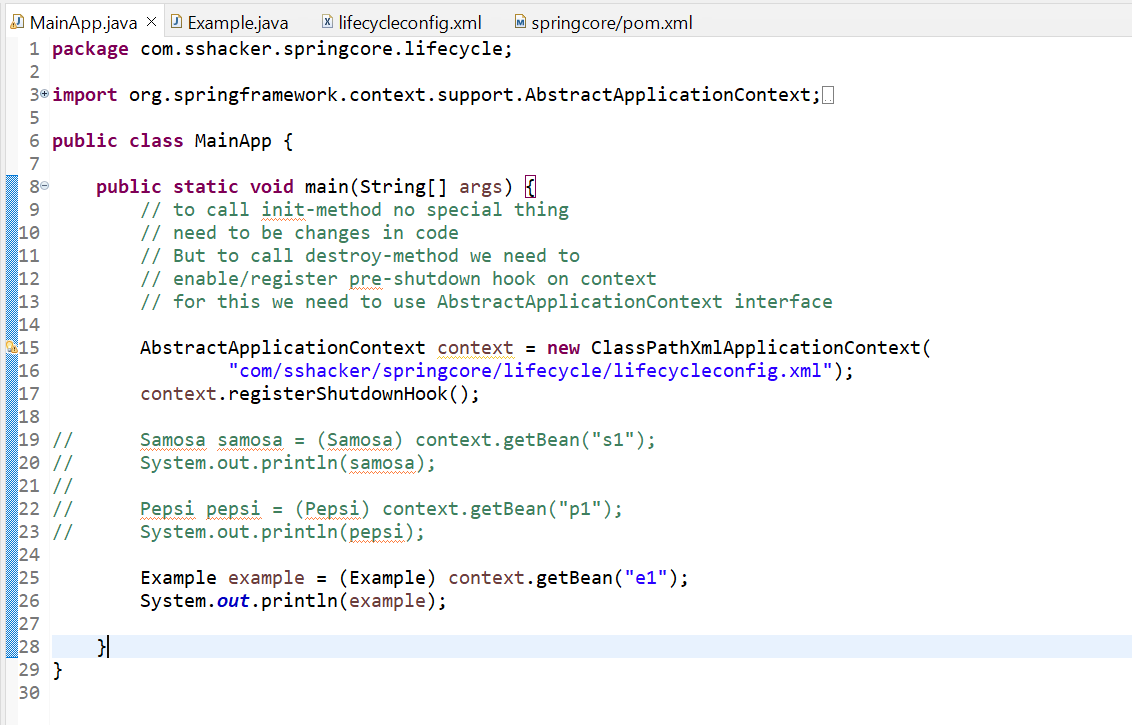




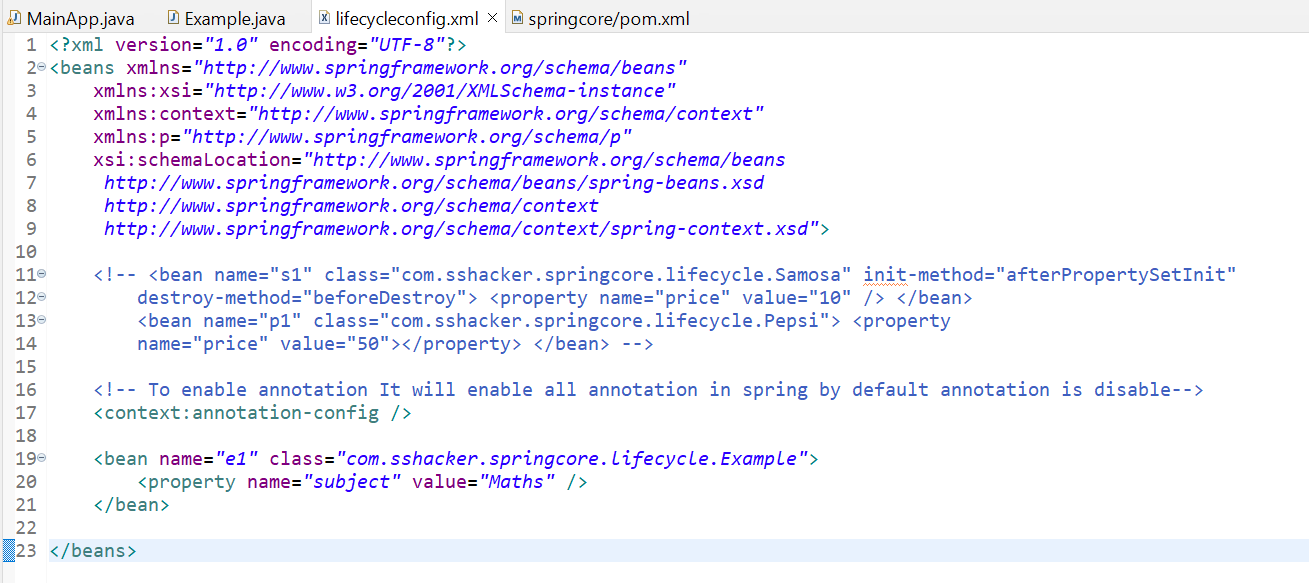


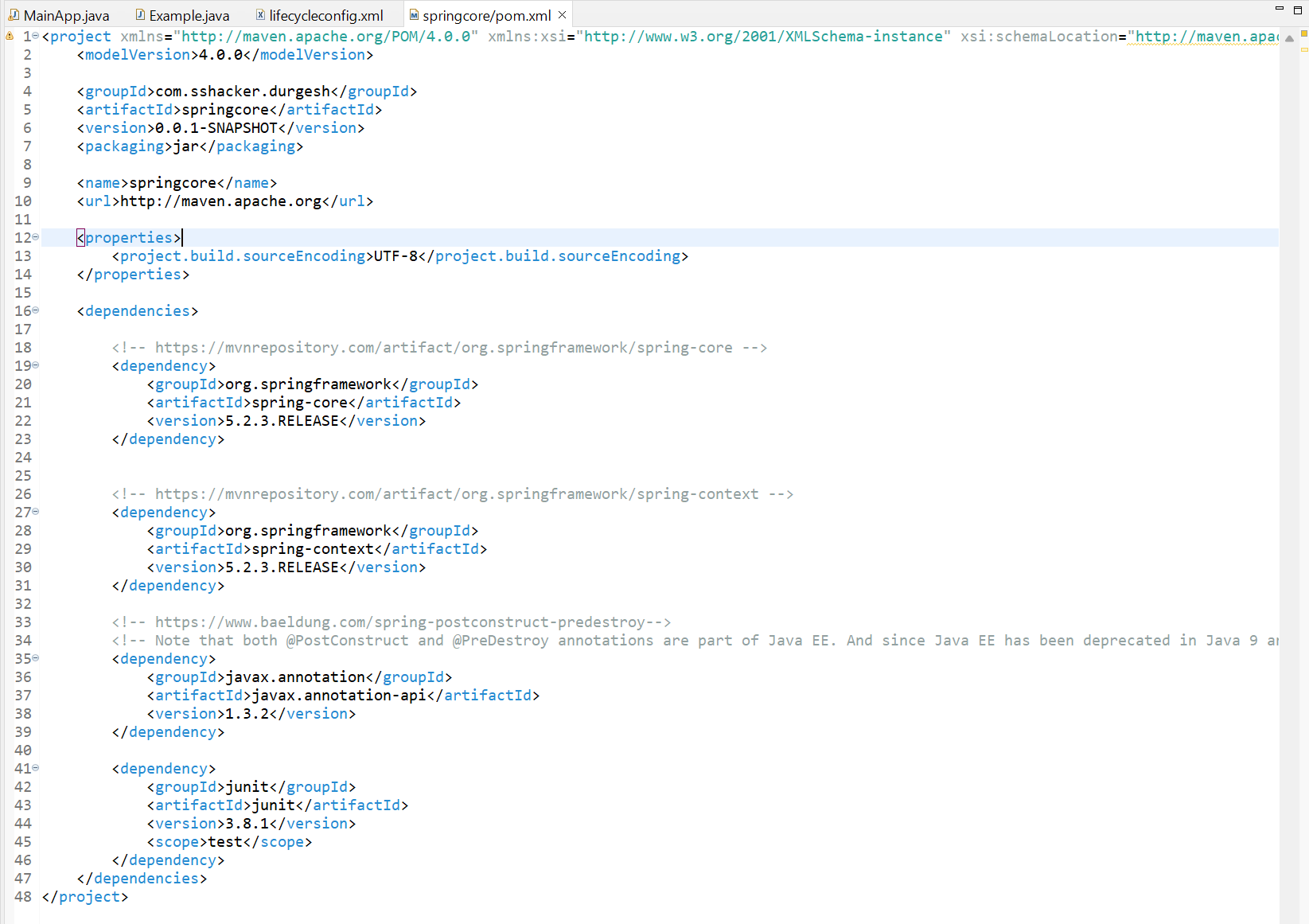


# 16. Implementing Bean LifeCyle using Annotations | @PostConstruct | @PreDestroy | Spring Tutorial

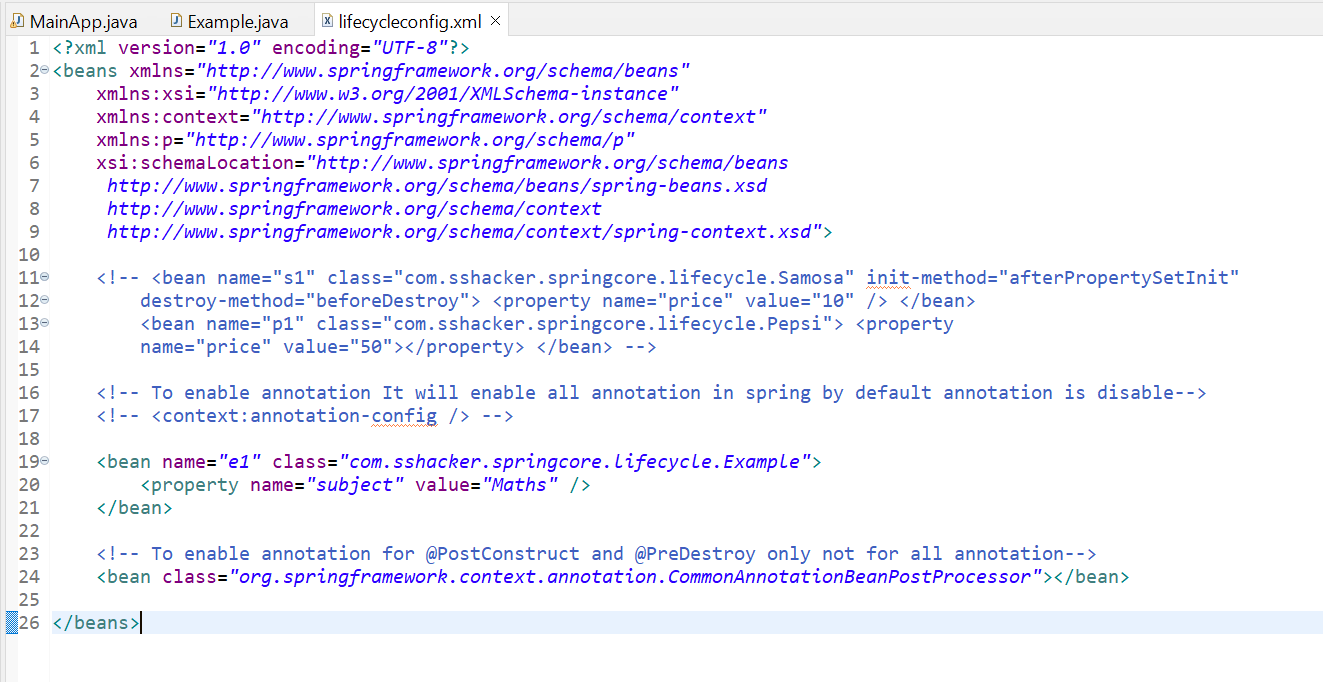








To enable annotation for @PostConstruct and @PreDestroy only not for all annotation: -

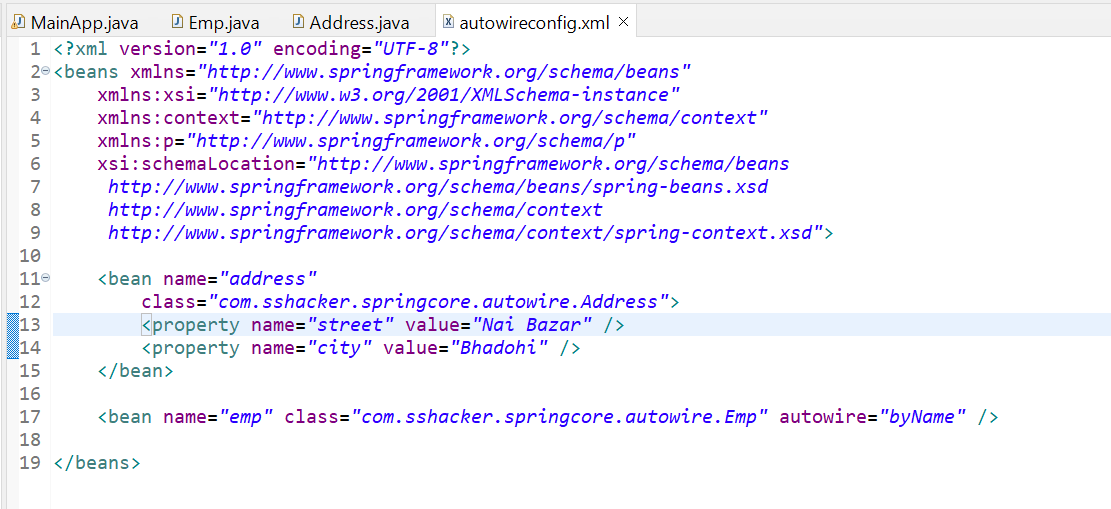


# 18. Autowiring using XML complete Session | byName | byType | byConstructor | Spring Tutorial





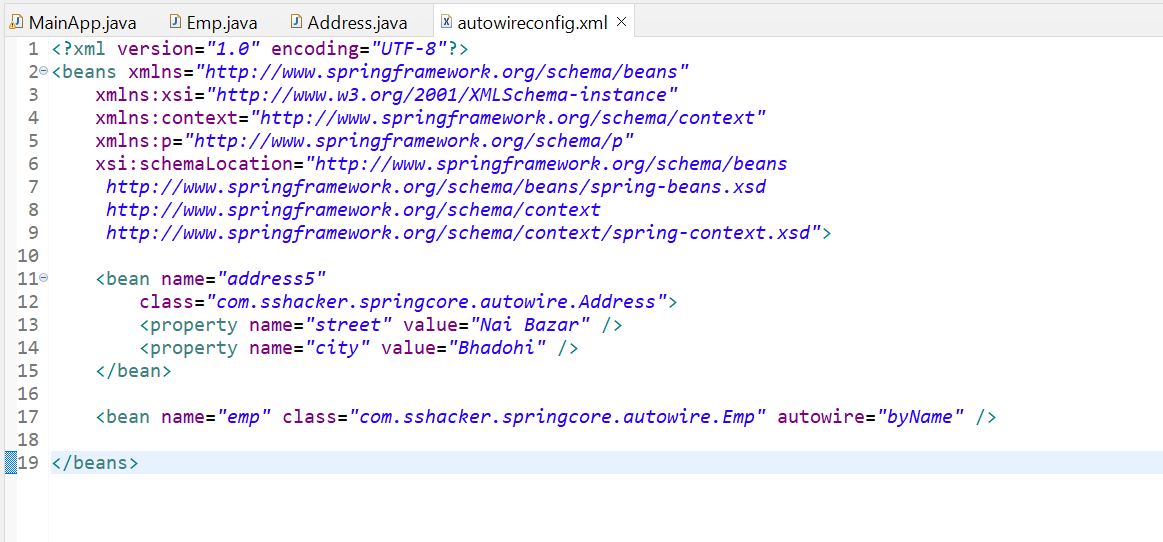


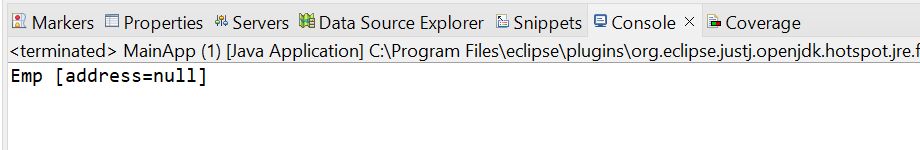




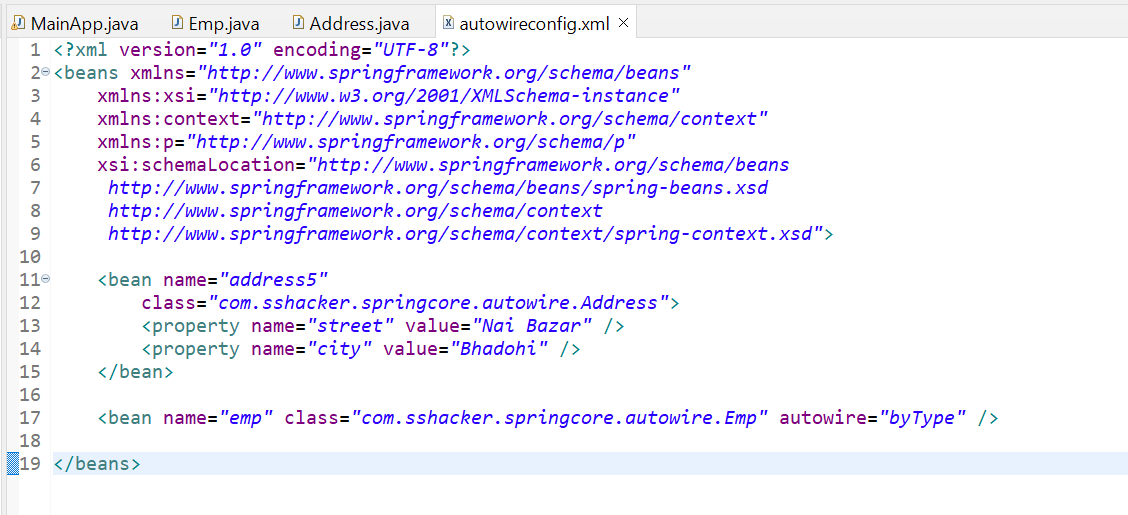
If bean name is not match with variable reference name there will be no error it just returns null.

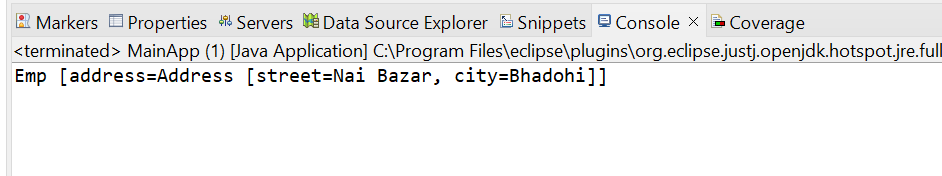






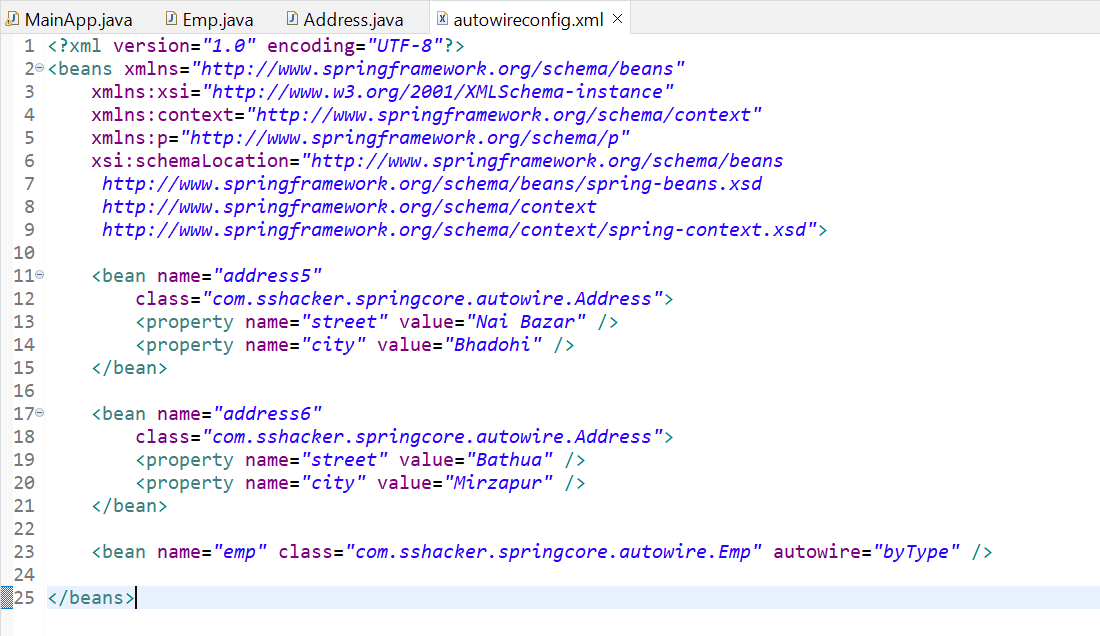
Autowire=”byType” : -





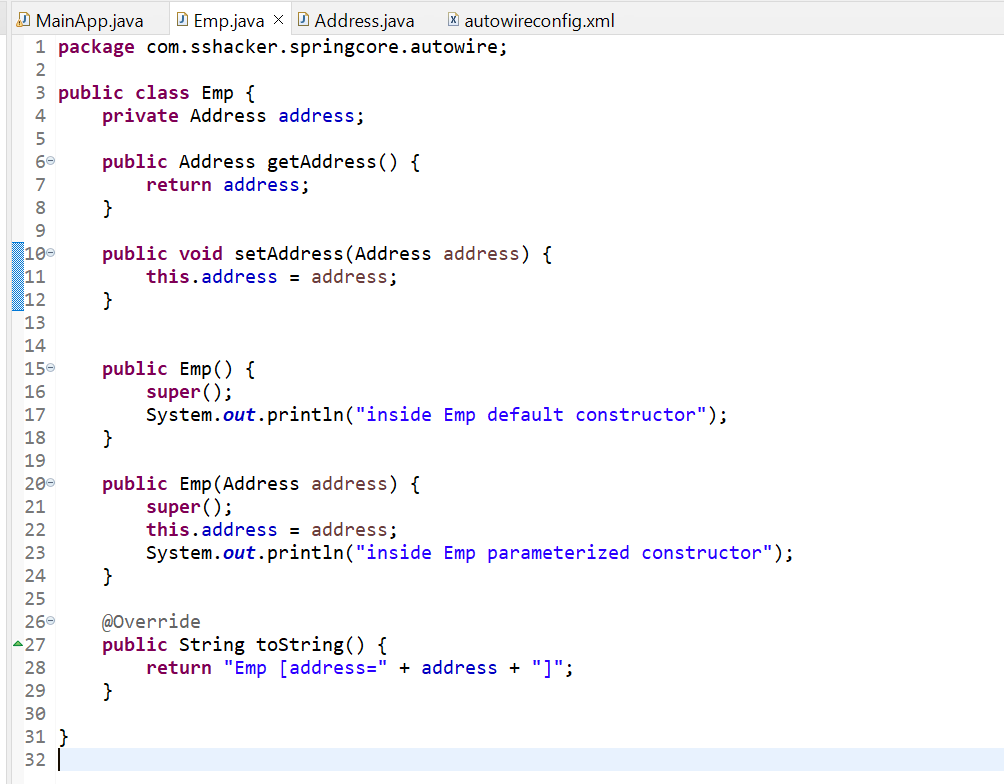
Conflict issue due to byType: -

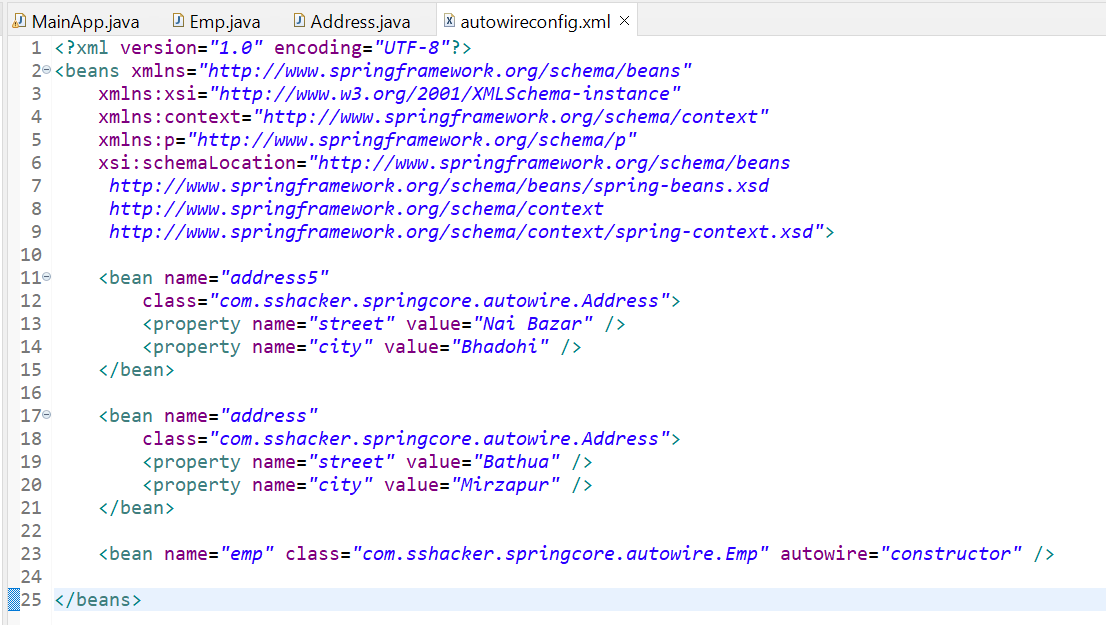
NoUniqueBeanDefinitionException

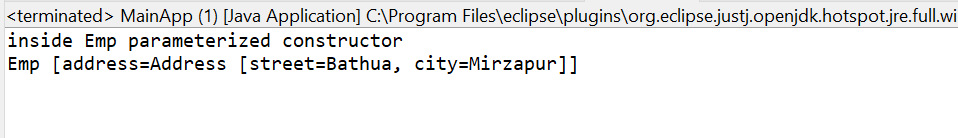


Constructor: -

It will check byName and call parameterized constructor also.

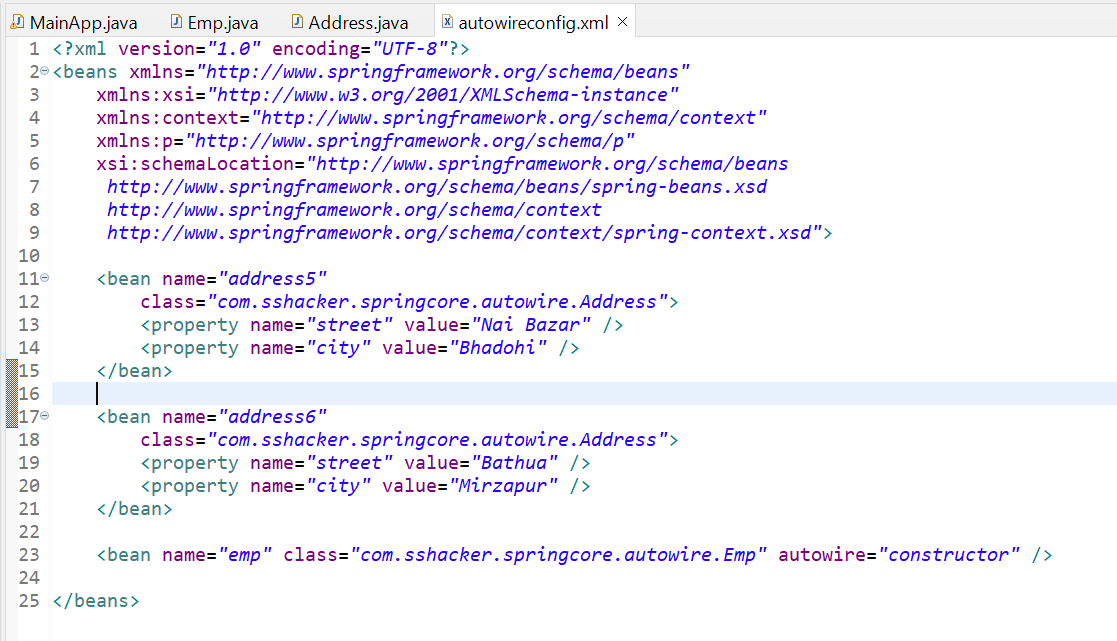


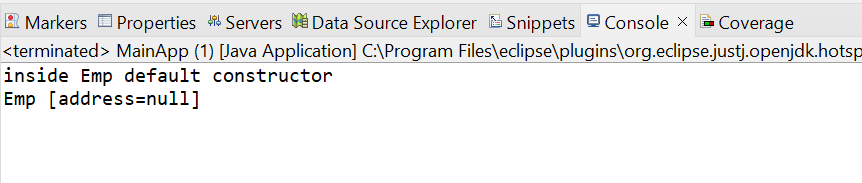




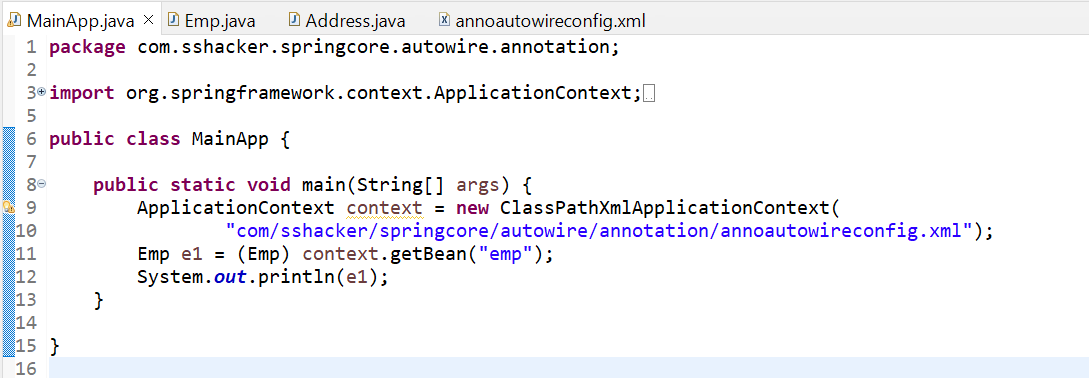
Constructor: -

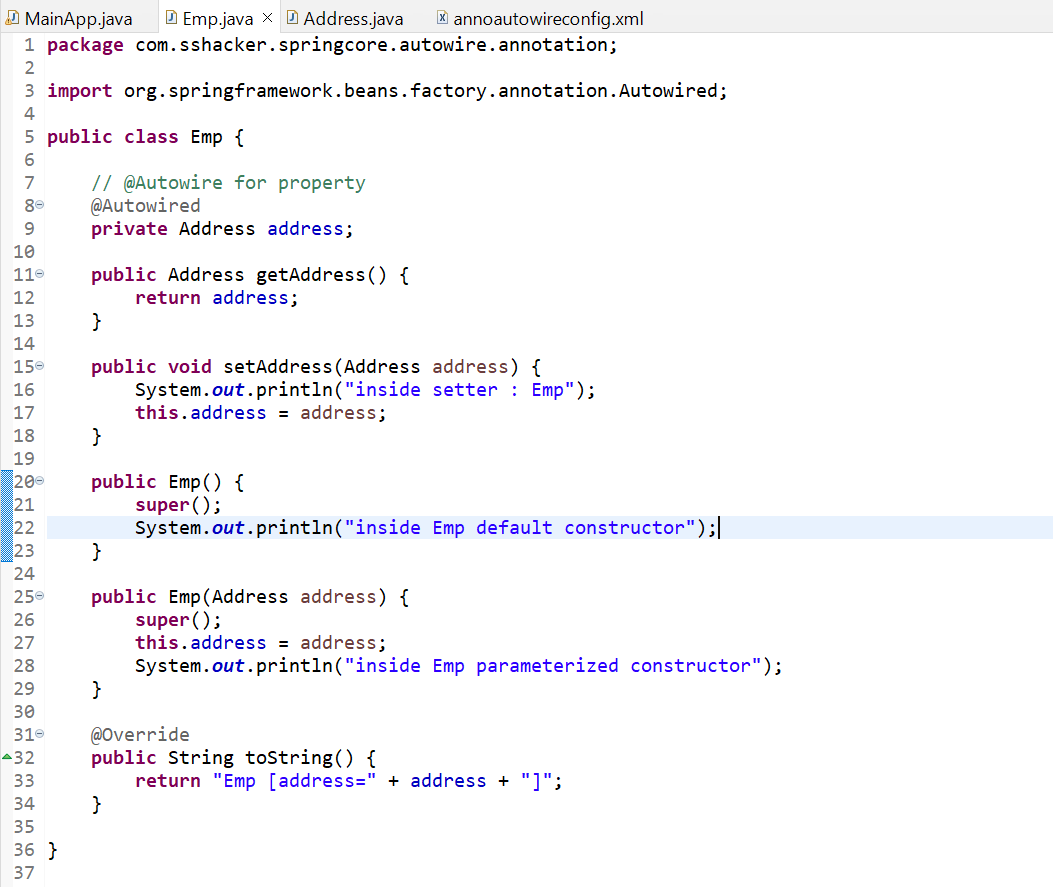
It will check byName and call parameterized constructor also. If byName no bean found then it return null (no exeption or error raise).

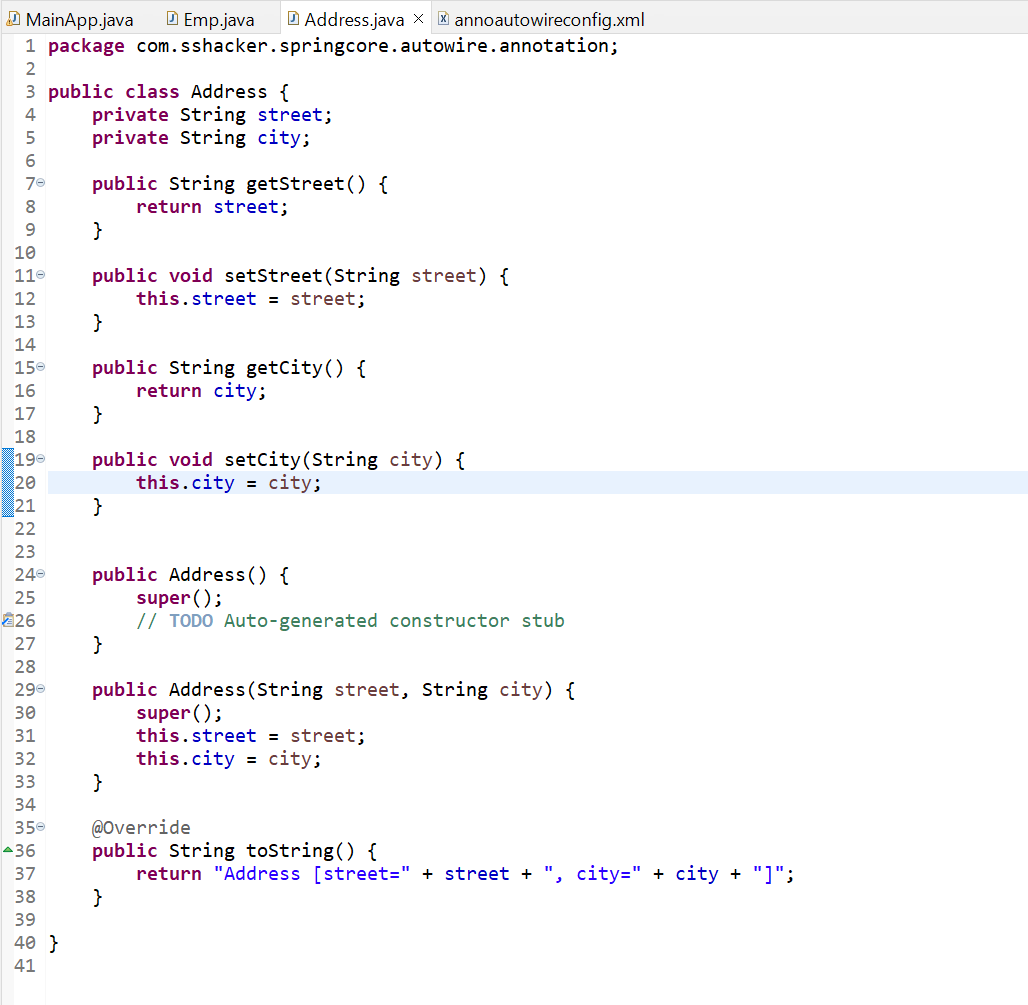


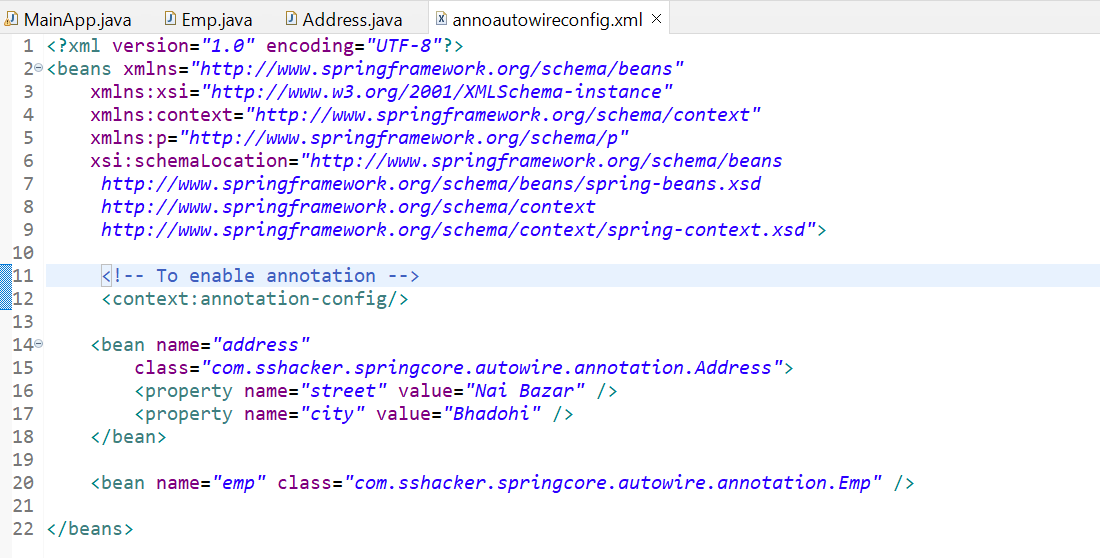


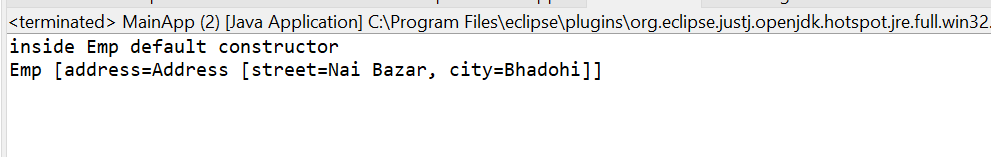
# 19. @Autowired Annotation for Autowiring | Autowiring in Spring Framework | Spring Framework Tutorial



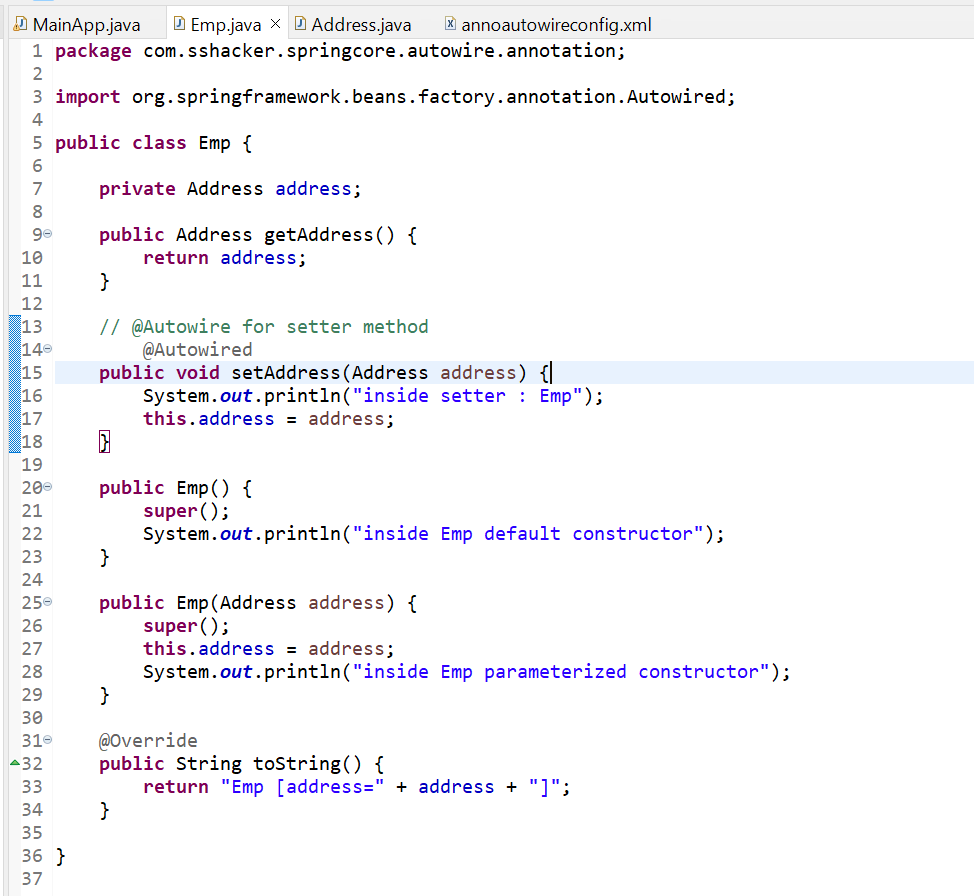


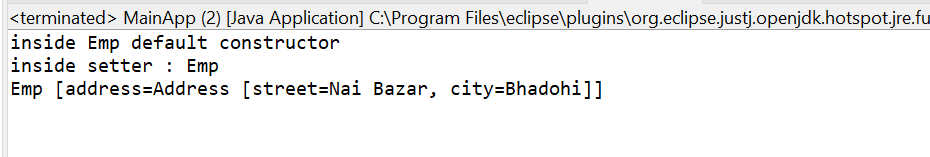






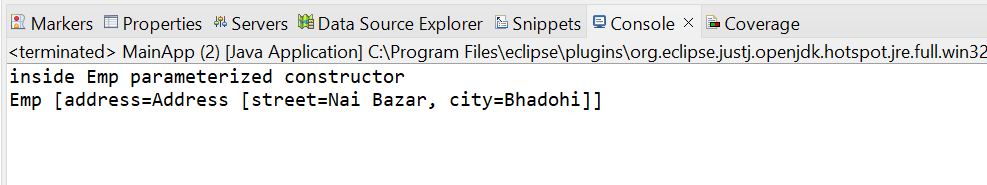
For setter: -



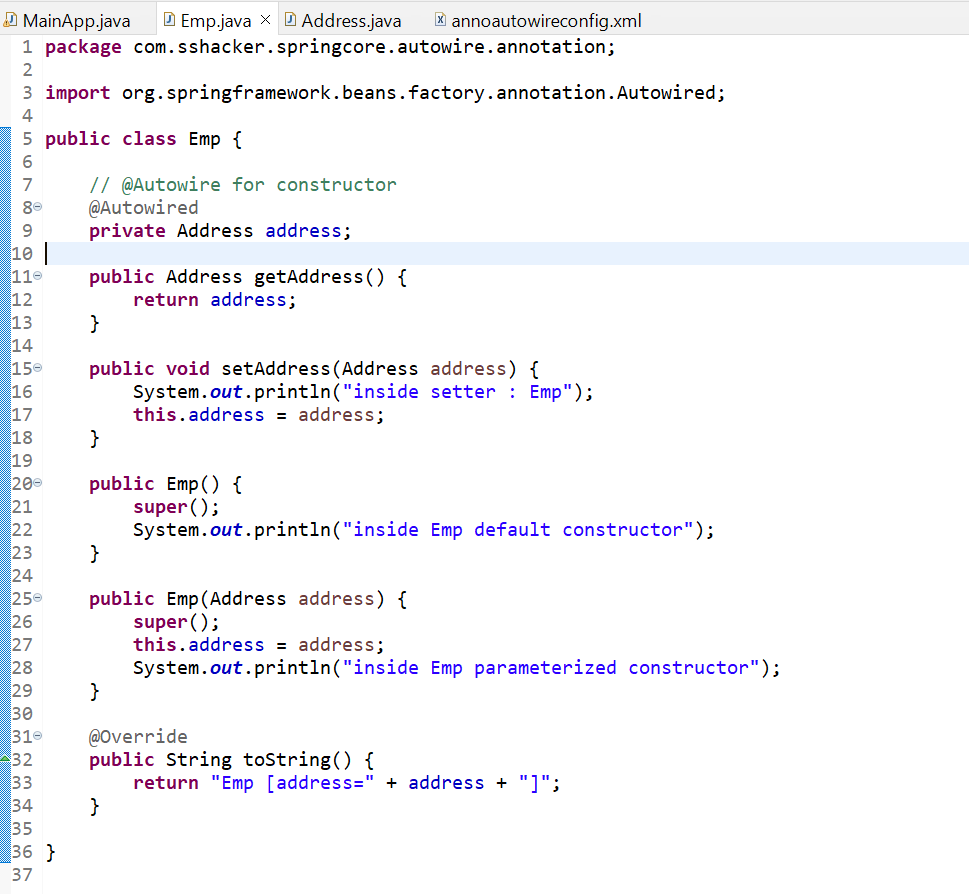


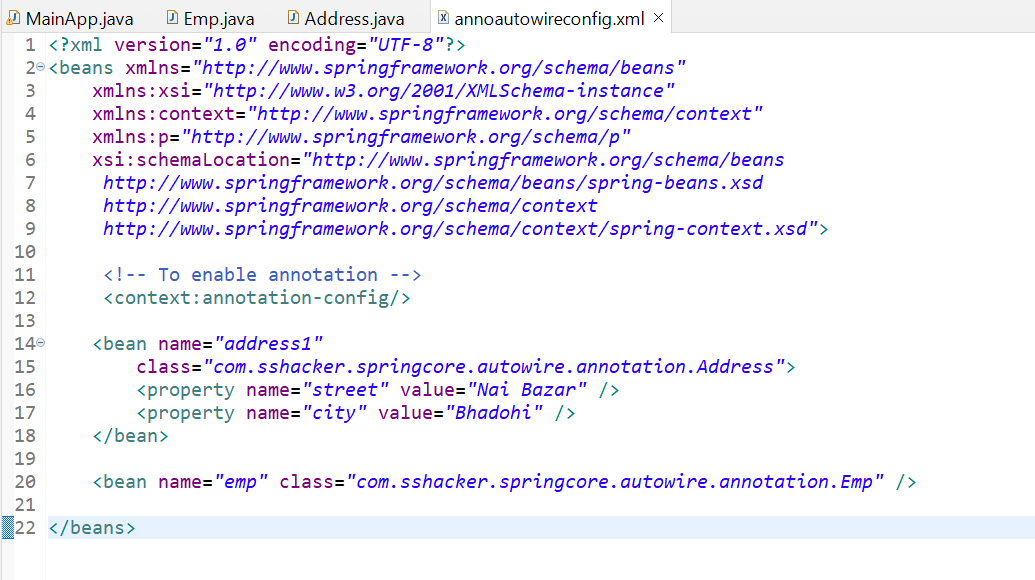
For constructor: -

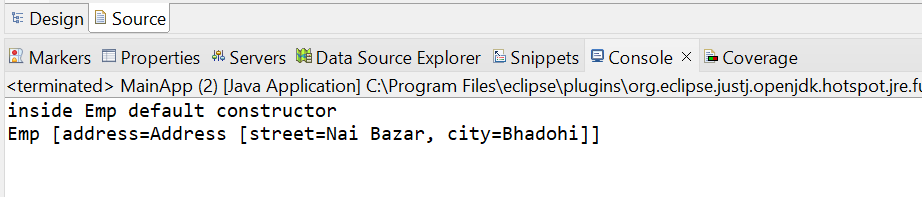




@Autowire is by default byType

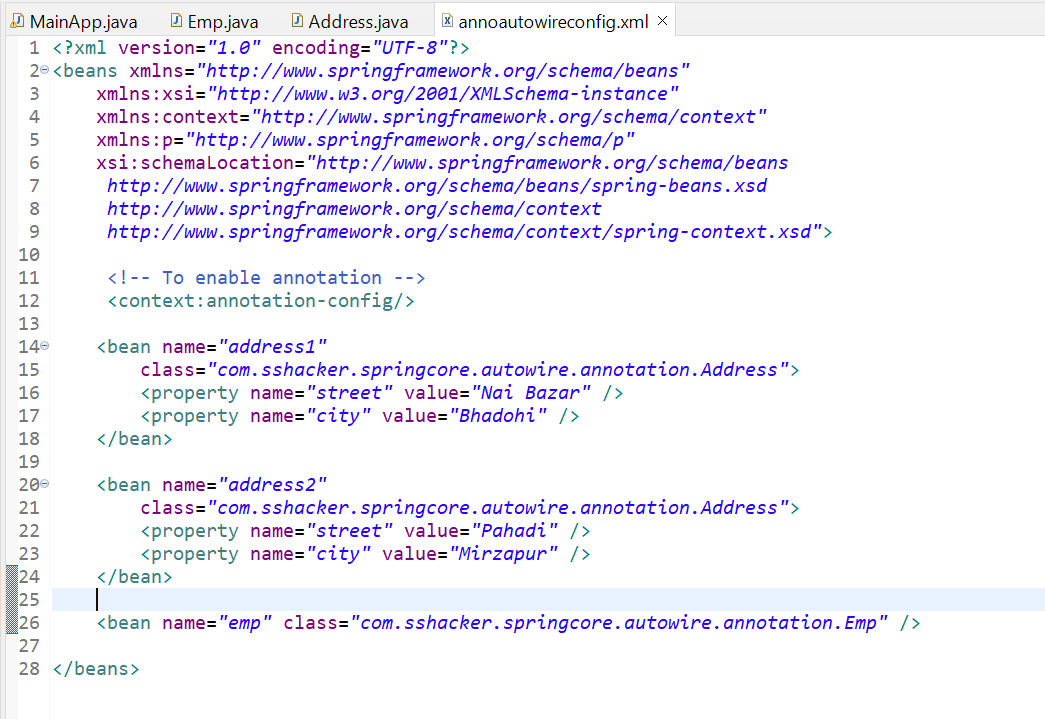


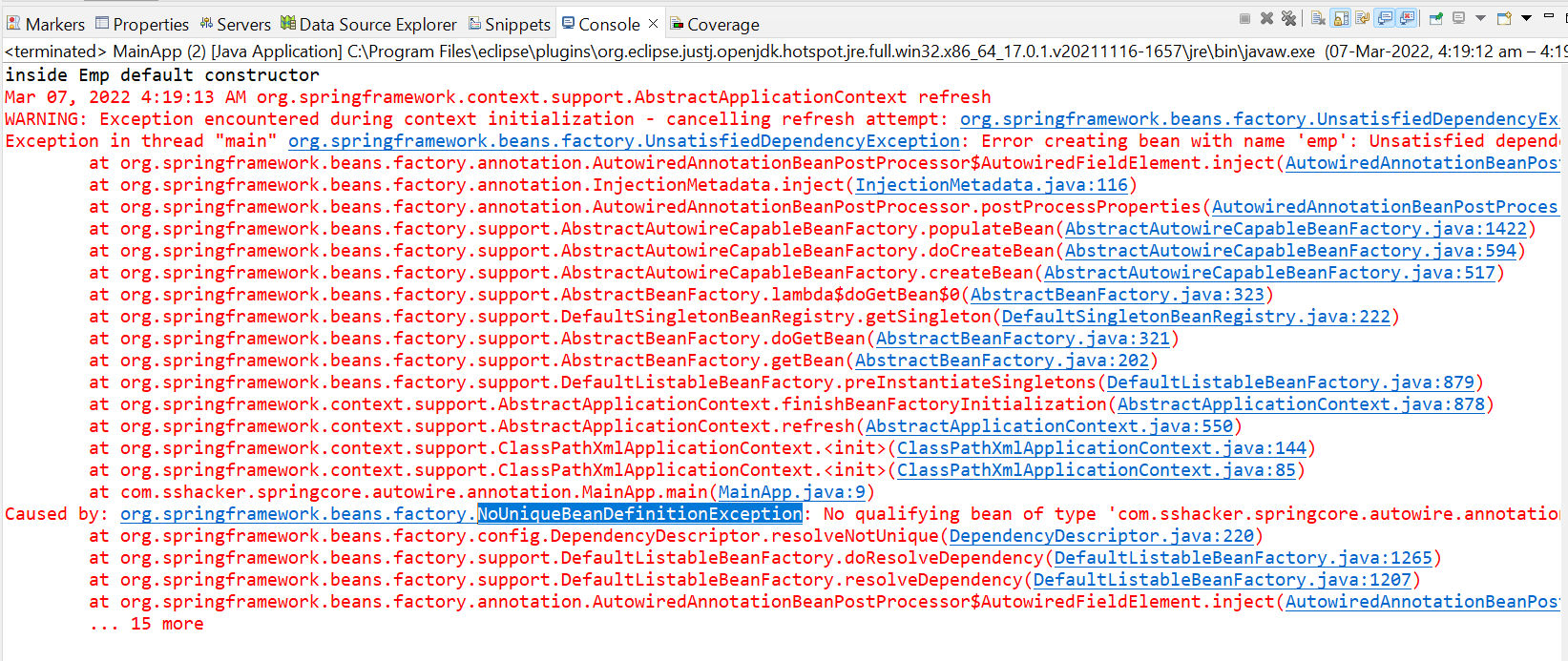




If same type 2 bean present in config.xml it will raise exception:-

NoUniqueBeanDefinitionException





# 20. @Qualifier Annotation with Autowiring | Spring Framework Tutorial

It is use with @Autowire when 2 or more than two same type of beans present so @Quilifier annotation needed to qualify a unique bean.

