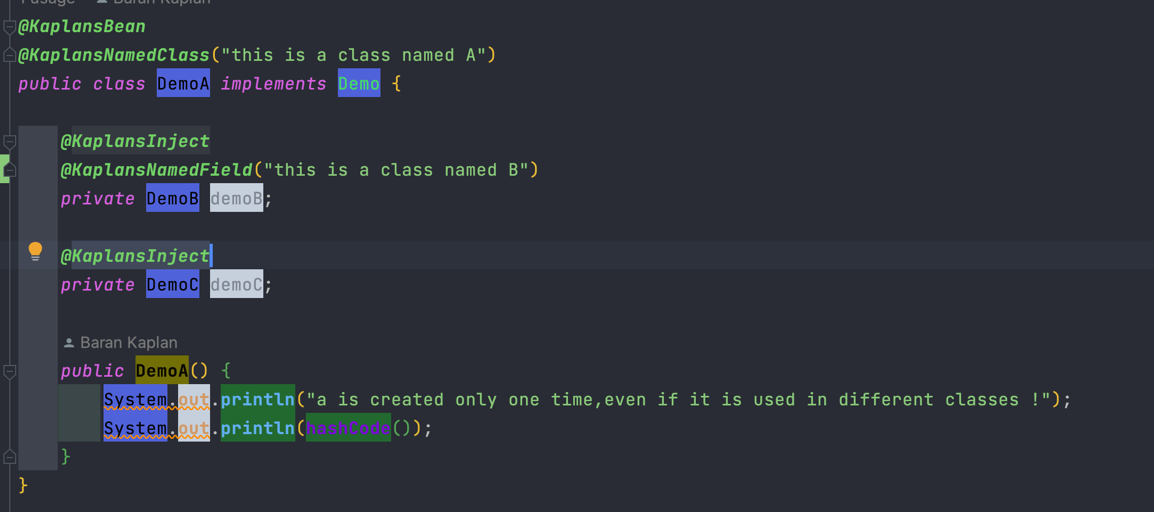
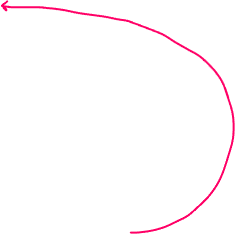
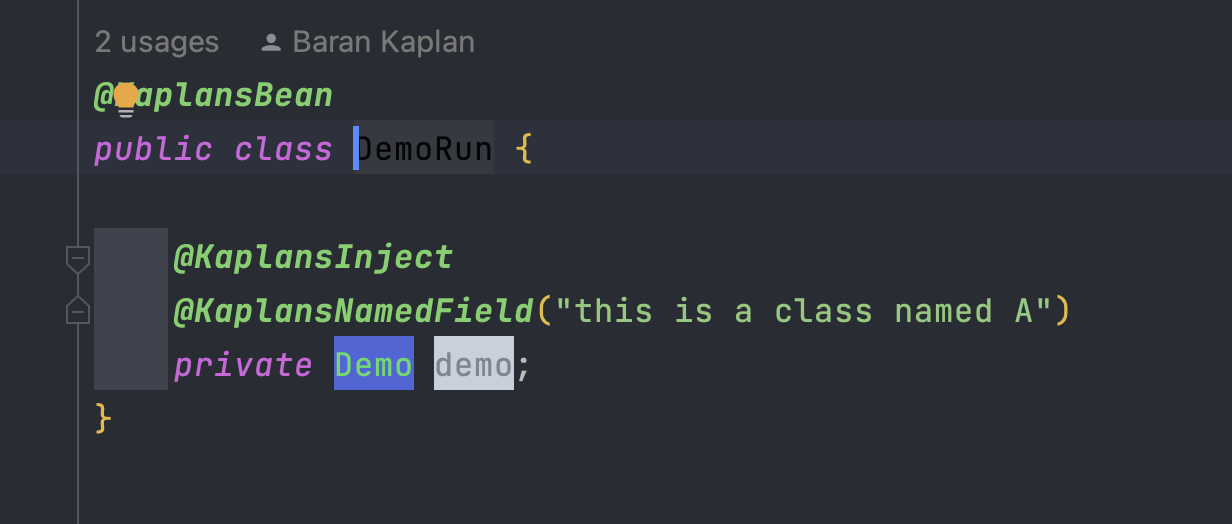
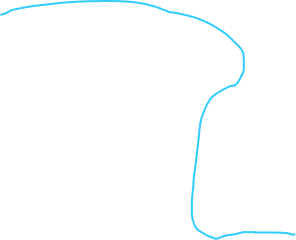
step 1 – scan and determine directories then find location of classes - ClassLocator. This finds all available classes.

step 2 - collecting metadata information - ServicesScanning service returns ServiceDetails class from the model’s package which stores all kinds of useful info.

step 3 - creating instances - ServicesInstantiation and DependencyResolver services. These use the result from the previous step and update the ServiceDetails with more data like instance and maybe few other things like resolved field and constructor parameters.

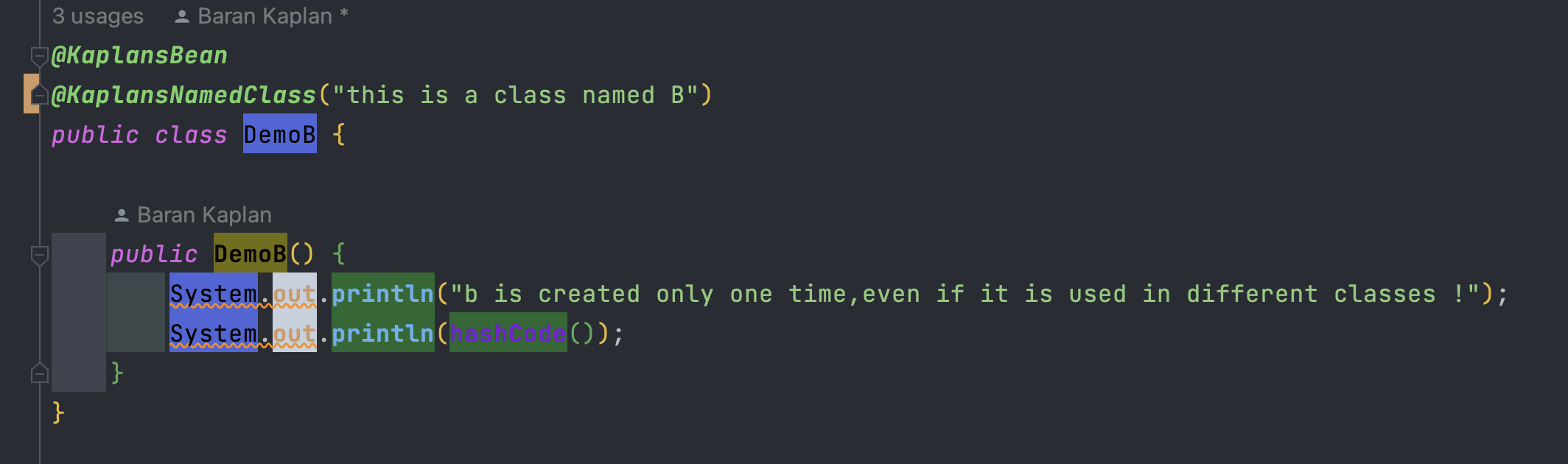




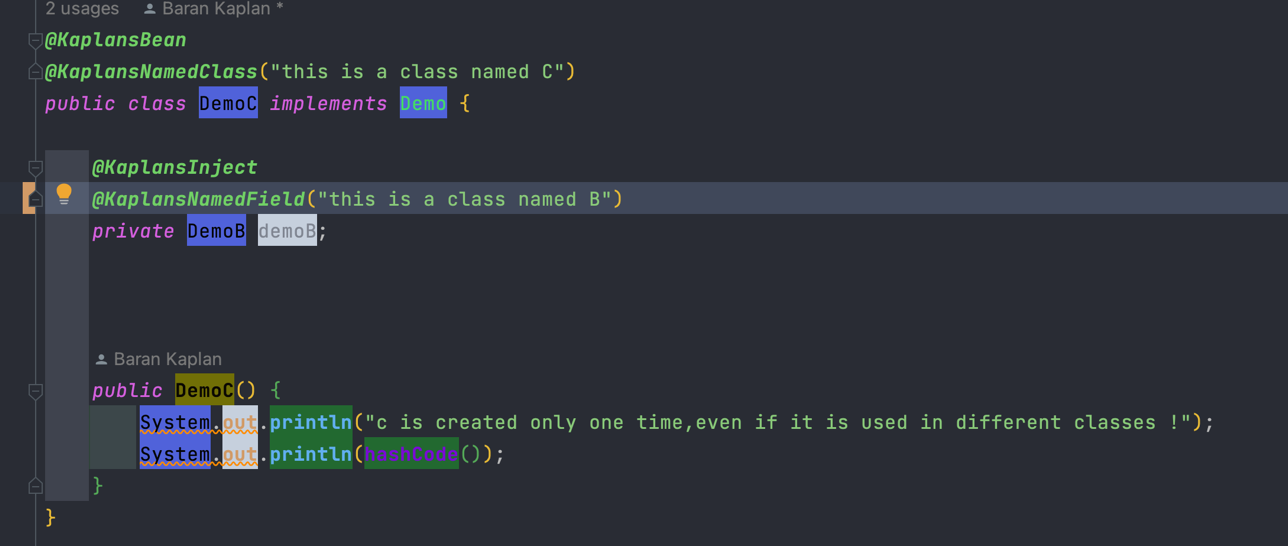


When the container runs the DemoRun class, thanks to @KaplansNamedField("this is a class named A"), it will create only the DemoA class object within the interface implemented by the 2(DemoA and DemoC) classes. Then the field object references within this demoA object will be created automatically (DemoB and DemoC). DemoB has dependencies in 2 separate classes (DemoA and DemoC) will only be created once, even though it has a dependency relationship in 2 classes .(Singleton)









test expectations

-are objects created with a singleton understanding with using @KaplansBean annotation?

initAndObserveSingleton,checkIfDemoAIsSingleton

-Is field injection done in demoA? (If it is , that will be also valid for other classes!)

checkIfDemoAHasFields

-are these field object references annotated with @KaplansInject and/or @KaplansNamedField?

checkIfFieldsIndDemoAareAnnotatedByKaplansInject , checkIfFieldsIndDemoAareAnnotatedByKaplansKaplansNamedField

-Does a class Named @KaplansNamedClass("this is a class named A") match when it is called as a field in another class after it is marked with an @KaplansNamedField annotation?

checkIfDemoAHasName,checkIfDemoAMatchesInDemoRunAsNamed

- Are all classes marked with @KaplansBean really defined within the service model?

checkIfAllAnnotatedClassesFetched, checkIfAllAnnotatedClassesFetchedUsingNotAnnotatedClass