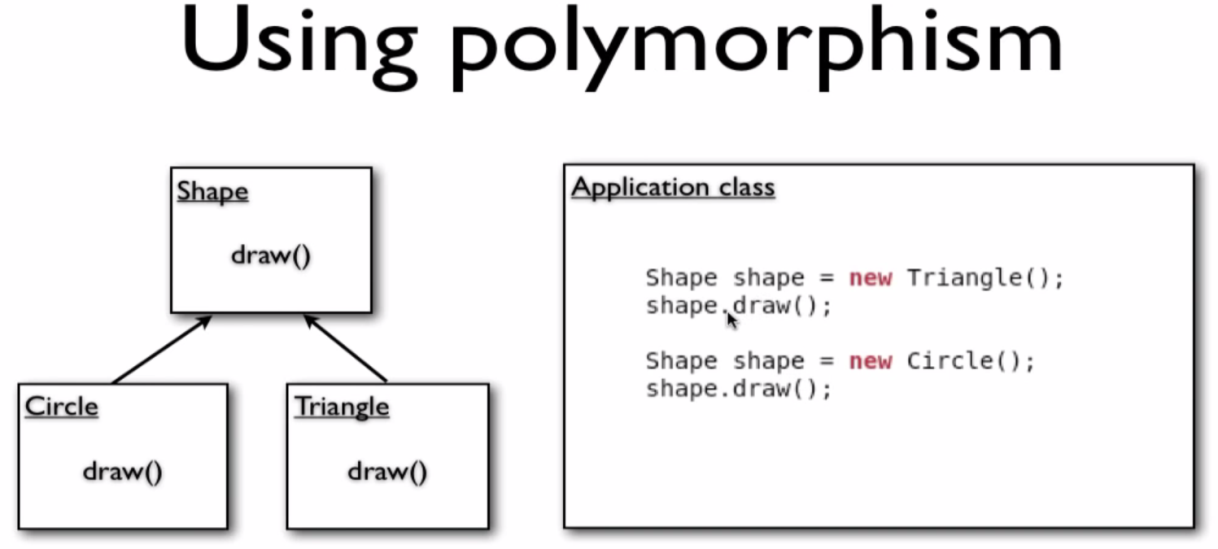
**Springs Framework**

-Decouple the dependency so as to make the object implementation is not tight to each other

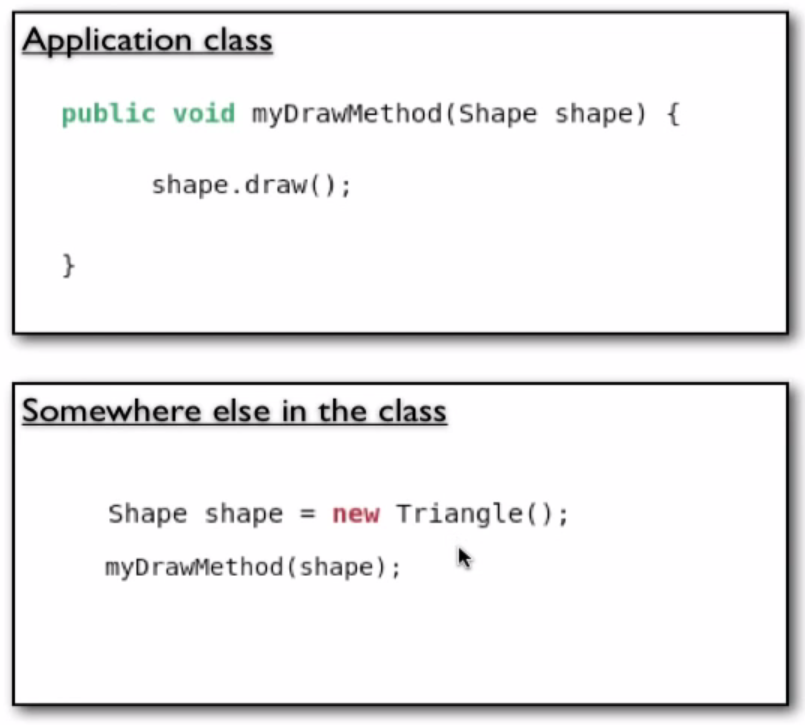
-after the notes made in ThinkPad

-Use of polymorphism



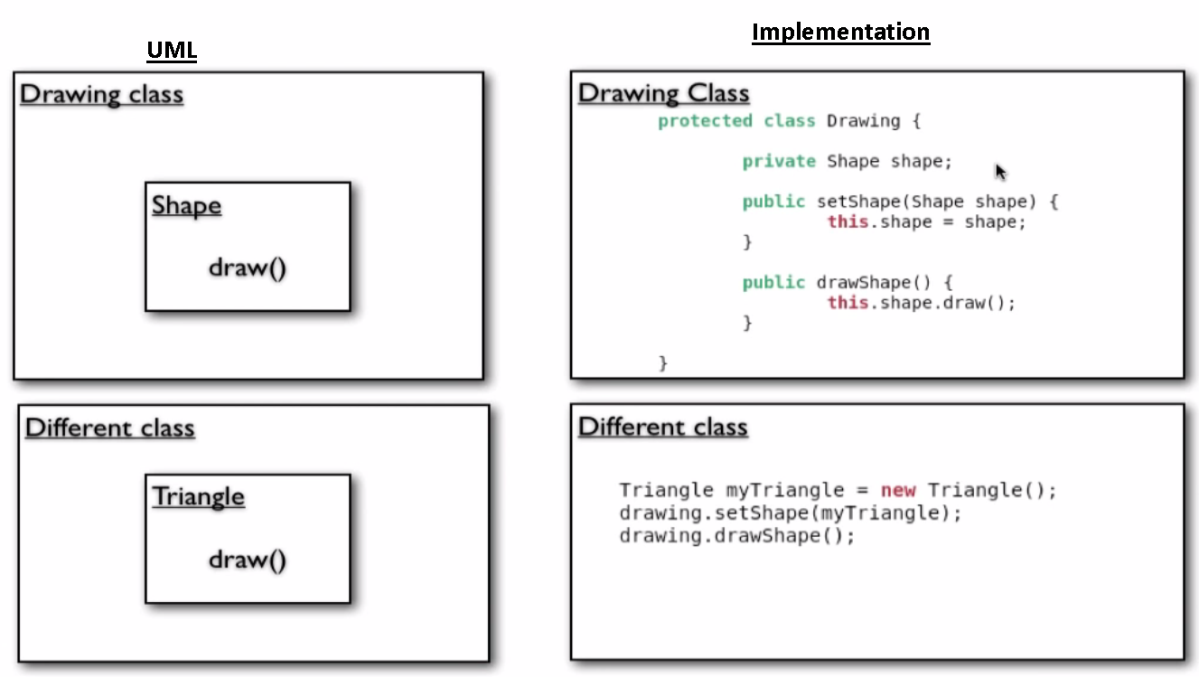
-But the above code object is dependent on the implementation.

-In the below code i will try to remove dependency.



-To make myself clear here is the diagram.

-I will use the concept of "class member variable"



-Separate the dependency out of the class. We got spring to do that.

-The advantage of above code is even the requirements change, 'Drawing Class' don't need to be changed.

-In the above code 'Different class ' is injecting a shape object.

-Spring will automatically takes care of this senario.

-Home Work time --Create a simple Spring main class which prints "HELLO WORLD!!"--either of the two ways

**Class-2**

-Spring Factory Bean

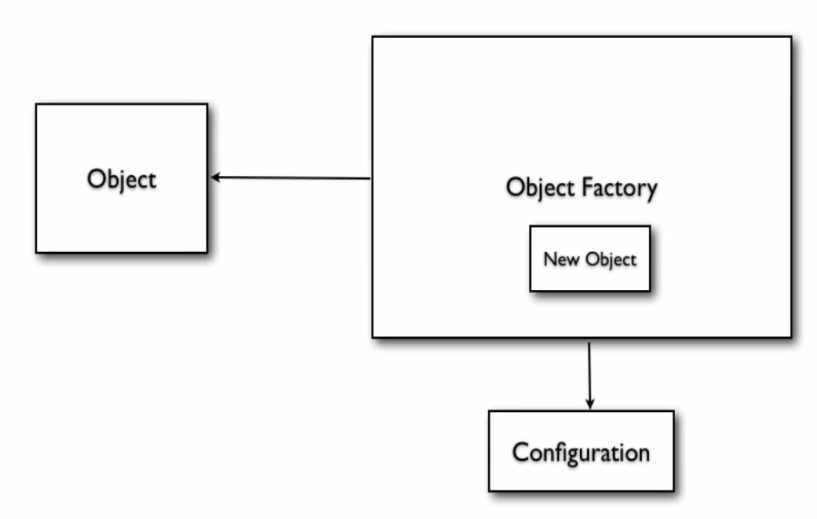
-Spring provides Dependency injection or inversion by the help of the concept called "Container".

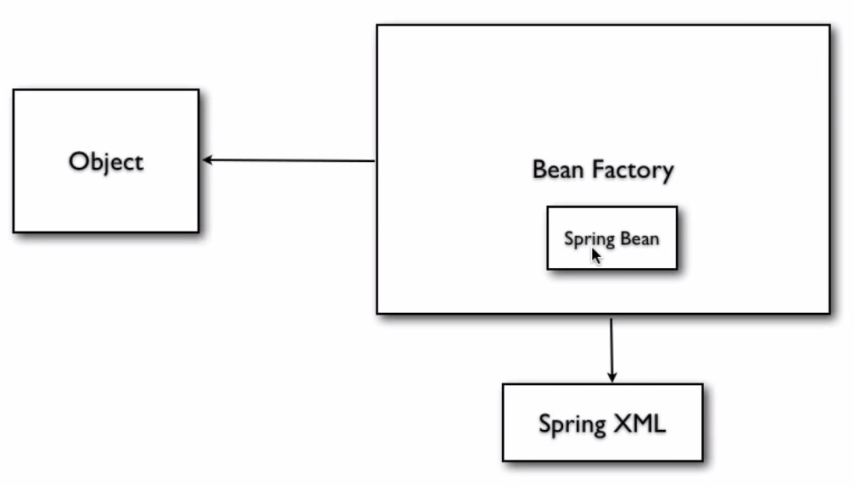
* Spring Container:
  + it's the container of BEANS.
  + it's a box of objects where its maintains lifecycle of each object defined in it.
  + creation of the object is done by the container.
  + Container works as factory pattern.

TODO:

Factory design pattern from new think tank

* Factory Pattern
  + Study what is factory design pattern.





**Class-3**

* BeanFactory
* XmlBeanFactory
* FileSystemResource
* ApplicationContext
* ClassPathXmlApplicationContext

-property initialization: to set some value to a variable.

-constructor injection:

-injecting object

-inner bean aliasing

-collection

-beanAutowiring

autowire="byName"(here autowire happens by name)

or "byType"(here autowire happens by type)

or "constructor"(here autowire by constructor) if default the autowire is off

-beanScope: There are two Bean Scopes

* Singleton-Only once per spring container
* Prototype-New bean is created for every request or reference

other scopes: Web aware Context Bean Scopes

* Request - New Bean per servlet request
* Session - New Bean per session
* Global Session- New Bean per global HTTP session(portlet Context)

syntax for bean is 'scope="singleton"'

ApplicationContextAware: So all the dependent class(points)in triangle are made singleton because Triangle class is singleton

-ApplicationContextAware gives information about bean context

BeanDefination Inheritance: if we have a lot of Bean Definition and wants to implements a common set of values this can be possible by BeanDefinitionInheritance.

Lifecycle Callback: AbstractApplicationContext is making use of life cycle of a bean

registerShutdownHook();

Here the triangle class implements InitializingBean ,DisposableBean.

you can also declare init-method followed by the first method that is called in applicationcontext.xml

likewise you can declare destroy-method followed by the last method that is called in applicationcontext.xml