Course :: spring basics, spring boot, spring cloud and microservices **Aug 26 Intro 1**
pre-requisite :: core Java oops
duration :: 140 to 150 sessions
timings :: 7:30 pm to 9pm /9.15 pm Completely online course
weekly 6/7 classes

Faculty Details ::
name:: nataraj
FB Group name :: natarazjavaarena
FB group url :: https://www.facebook.com/groups/388095825162910
email id :: natarazjavaarena@gmail.com
admin details ::
Mr.Srikanth: +91 6302968665 (whats up only)

Course Fee :: 8`000/- with ReactJS or Angular or HB with JPA
5000 /- (with out free topic)

earlier days: ReactJS / Angular by sudhakr sharma sir.
1. spring course (90 sessions , fee : rs:3500)
2. spring boot and MicrosServices (80 session ,rs:4000)

Spring basics, spring boot , springcloud and microservices (spring stack Course)

free :: angular or reactjs or hiberante with JPA
free :: 15+ realtime java tools like Maven,Gradle, GIT, SVN,
log4j, slf4j, jenkins, docker , ...
free :: 10 + commons topics (resume preparation, interview tips,
Dos and do nots after joining in the s/w company,
AGILE-SCRUM, JIRA and etc..
free :: 7 to 8 Mini Projects.

free :: Job guarantee program by Naresh IT (u need to clear screening test)

Spring Modules covered (Spring stack)
==========
spring core
spring jdbc
spring aop Recorded vedios download course content from
spring TxMgmt https://www.facebook.com/groups/388095825162910/files
spring data jpa
spring data mongoDB
spring mvc
spring security
spring oauth
spring batch
spring rest
spring boot actuators
microservices
spring mail + dynamic content...
spring cloud
and etc...

Running batch time is :: 7 am to 8:30 /8:40 am

notes /materials
==========
=> old batch pdfs will be shared
=> old batch Worskpaces will be shared
=> u r batch running notes/material will be shared using FB group /Google classroom
=> u r batch Calss Room Apps , Mini Projects will be shared using GIT / GITHUB
=> Recorded session using www.youtube.com/nareshit

Who needs this course

Fresher :: spring basics ,spring boot (mandatory)
other advanced topics are optional (recomanded to show them added advantage)

=> 1+ , 2+ ,3+ 4+ ,5+ (complete spring stack course)

=>pkg for Freshers :: 4 to 6 lacs
=> pkg for 1+ ,2+ :: 5 to 7 lacs
=> pkg for 3+ , 4+ :: 6 to 12 lacs
=> pkg for 5+ :: 8 to 15 lacs

3+ years experience :: 16 lacs
(record package so far)

---

to

Do i need learn xml and webSErvices course seperately?

Ans) No ... not required becoz spring Rest is there web service implemenation

DO i need to learn Adv .java course (jdbc,servlet,jsp) ?

Ans) yes required ... we should have servlet,jsp knowlege before
it start spring MVC topic here.

Do i need to learn Hiberatne with JPA course ?

Ans) we can manage with out that course .. but learning is added advantages
=> spring data jpa internally uses hibernate... So learning hibernate makes
spring data jpp module learning process quite easy.

What else i should learn to become java full stack developer after this course?

Ans) spring stack course (current course)
+
angluar or reactjs (UI Technology)
+
oracle Db s/w
+
slenimum (Testing)
+ Design patterns
+
DevOps (Deployment)
+
AWS / Azzure /Google cloud

Java full stack developer pkg for 3+/4+/5+ :: 20 to 25 lacs.

Q) This is my first framwork course , is there any problem?
Ans) No Problem .. we will learn from 0% (right from what is framework?)

Q) can i learn adv.java course and framework course parellel?

Ans) yes u can

Q) My Core java is just started , can i sit in this course ?

Ans) No.

spring basics, spring boot, spring cloud and micro services

Aug 27 intro 2

(spring stack course or complete spring course or end to end learning of spring framework)

This course details with

spring basics
spring boot
(multiple module of spring will be discussed in spring boot mode)
|-->spring mail, spring mvc, spring security ,spring oauth ,spring data jpa, spring mongoDB and etc..)
spring Rest (For restfull web services)
spring batch
spring cloud
MicroServices

15+ tools
10 + common topics
7 to 8 Mini Projects..

free courses
->React JS or
-> Angular
-> Hibernate with JPA
(by nataraj)

by sudharkar sharma

For class room material :: https://www.facebook.com/groups/388095825162910/files
ur batch code :: NTSPBMS615

=>c,c++,java ,c# and etc.. programming languages (like raw materials)
=>JDBC, Sevlet,Jsp, EJB, JMS and etc.. are called Java Technoloies (like semi-finished products)
=> Spring,struts ,hibernate,JSF and etc.. called java frameworks.. (like fully finished products)

Can u explain the differences among programming languges, technologies and frameworks?

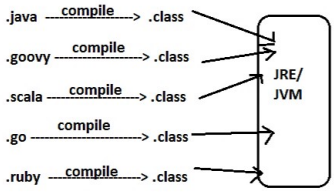
Programming language

syntax: rules of coding
semantics : structure of the code.

=>It is directly installable s/w acting as raw material providing basic features for developing s/w Applications.
=>It defines the syntaxes, semantic of programming by supplying compilers and interpeters.
=>Programming languages are base to create s/w technologies , tools, frameworks, Operating system,Db s/ws and etc.

eg: c,c++, java , java script ,c# , vc++ and etc..

JVM based languages
java , scala, groovy ,koltin, go ,ruby and etc..



Technologies

=>Technology is a s/w specification providing set of rules and guidelines in the form apis to create implementation softwares using one or another programming language.

=> Technology is not installable.. but the technology based implementation s/w is installable. Working with Technology based implementation s/w is nothing but working with technology.

eg:: jdbc , servlet, jsp , EJB,JMS , JAAS ,java Mail and etc..
Jsp :: java server pages
EJB :: Enterprise Java Beans
JMS : Java Messaging service
JAAS : Java Authentication and Authorization service

=>JDBC is a java technology that gives rules and guidlines in the form of jdbc api to create JDBC driver s/w using java programming language.
=>JDBC technology is not installable.. But jdbc technology based JDBC driver s/w is installable or arrangable .. Working withb JDBC Driver s/w (like ojdbc6.jar) is nothing but working with JDBC Technology.

API :: Application Programming interface

In "C" language API means set of functions given in the form of header files
In "C++" language API means set of functions,classes given in the form of header files
In "Java" language API means set of classes,interfaces,enum,annotations given in the form of packages

Upto java7

=> The Technology API package interfaces represent rules
=> The Technology API package abstract classes represent rules and guidelines
=> The Technology API package concrete classes represent guidelines

Normal java class for which object creation is possible is called concreate class.

From Java8 onwords

=>The Technology API Packages interfaces, abstract classes represent both rules and guidelines
=>The Technology API Packages classes represent guidelines

note:: Every technology does not given rules and guidelines in the form of english statements.. they given them as java api having packages with classes, interfaces and etc..

Two types of Technologies

1.Open Technologies

=>Here the technology rules and guidlines (apis) are open to all the vendor companies to create implementation s/ws

eg:: All JAva , JEE Technologies like JDBC,JNDI,EJB,SErvlet,Jsp, JMS and etc..

2.propritiery Technologies

=>Here the technology rules and guidlines (apis) are specific to vendor and only that vendor company is allowed to create implementation s/ws

eg: All MicroSoft technologies (asp.net, vb.net and etc..)

Frameworks Introduction

Technologies

=>Technology is a s/w specification providing set of rules and guidelines in the form apis to create implementation softwares using one or another programming language.

=> Technology is not installable.. but the technology based implementation s/w is installable. Working with Technology based implementation s/w is nothing but working with technology.

eg:: jdbc , servlet, jsp , EJB,JMS , JAAS ,java Mail and etc..
Jsp :: java server pages
EJB :: Enterprise Java Beans
JMS : Java Messaging service
JAAS : Java Authentication and Authorization service

=>JDBC is a java technology that gives rules and guidlines in the form of jdbc api to create JDBC driver s/w using java programming language.
=>JDBC technology is not installable.. But jdbc technology based JDBC driver s/w is installable or arrangable .. Working withb JDBC Driver s/w (like ojdbc6.jar) is nothing but working with JDBC Technology.

API :: Application Programming interface

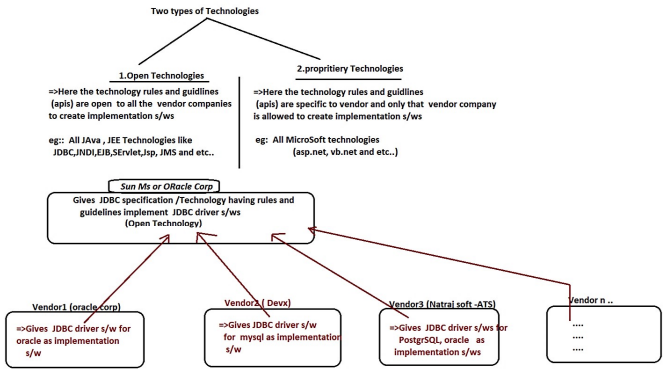
In "C" language API means set of functions given in the form of header files
In "C++" language API means set of functions,classes given in the form of header files
In "Java" language API means set of classes,interfaces,enum,annotations given in the form of packages

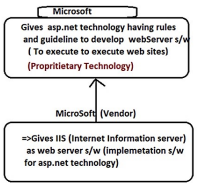
Upto Java7
==========
=> The Technology API package interfaces represent rules
=> The Technology API package abstract classes represent rules and guidelines
=> The Technology API package concrete classes represent guidelines

Normal java class for which object creation is possible is called concreate class.

From Java8 onwords
==========
=>The Technology API Packages interfaces, abstract classes represent both rules and guidelines
=>The Technology API Packages classes represent guidelines

note:: Every technology does not given rules and guidelines in the form of english statements.. they given them as java api having packages with classes, interfaces and etc..





=>Since all vendors are giving their implementation s/ws based on the common rules and guidelines software technology ..the way we work with implementation softwares of all the vendors is going to be much similar.
i.e if we know how to work with one implemetation s/w,we can use that knowledge to work with another implemetation s/w of same technology

=> All JDBC driver s/ws are given based on the common rules and guidlines of JDBC technology..So tha way we work with all the implementation s/ws (JDBC driver s/ws)is goining to be much similar.

Framework
=========
Def1:: Framework is installable s/w that is built on top of 1 or more technologies having capability to generate the common logics of the App/project dynamically ..and letting programmer to focus only on application specific logics devleopment.

Def2:: Framework is installable s/w that is built on top of 1 or more technologies providing abstraction on Technologies to simplify the Application/Project devleopment process.
hiding the implemetations/internals.

example of JAva framworks struts , JSF, spring , spring boot hibernate, Eclipse Link , Toplink, Sring Rest , Axis , Apache CFX and etc..
example of Java script frameworks /Libraries/Toolkits :: angular , reactJS and etc..
example of .net frameworks :: asp.net mvc ,aps.net core and etc..
example of php frameworks :: Drupal , Code Iginator , WordPress, Larvel and etc..
example of python frameworks :: DJango , flask and etc..
example of BigData framework :: Hadoop , Spark and etc..

=>While working with Technologies , makes the programmer to write both common logics and application specification logic (improves the burden on the programmer)
=>While working with Framework makes the programmer to write only the application specific logics becoz the the framework will take care of generating application specific logics devleopment.

JDBC Aplplication (Java Technology App)
==========
=>regster JDBC driver s/w (by loading jdbc driver class)
=> Establish the connection with DB s/w from Java App
=> create JDBC Statement object
Common logics

=> send and execute SQL queries in Db s/w (inputs) using Statement object
=> Gather SQL query results from Db s/w and process them using Statement object
application specific logics.

=> Exception handling
=> close connection with Db s/w
Common logics

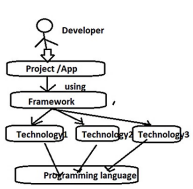
note: Here Programmer needs to take care of both commong logics and application specific logics
Common logics= biolerplate code (problem)

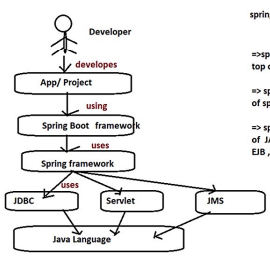
=>The code that repeates across the multiple parts of the project either with no changes or with minor changes is called biolerplate code (common logics) , i.e while working with technologies we have boilerplate code problem.

Spring JDBC App (Framework App) (note:: spring JDBC is part of spring framework)
==========
=>create JdbcTemplate class obj (pre-defined class given by spring framwork)
(Takes care of common logics internally by using technology)

=> send and execute SQL queries to Db s/w
=>Gather SQL query results and process the results
application specific logics

note ::Here programmer needs to take care of only application specific logic becoz the common logics will be generated by the framework dynamically .. i.e no boilerplate code problem.





.... techonologies

spring boot = spring ++

=>spring boot is framework that built on the top of spring framework

=> spring boot avoids boilter plate code of spring framework programming.

=> spring framework avoids boilter plate code of JAVA,JEE technologies like JDBC, servlet,jsp, EJB ,JMS and etc.. programming

Advantages with Frameworks
==========

Aug 31 Types of Frameworks

a) Simplifies application development by providing abstraction on one or more technologies

b) Avoids the boiler plate code by generating common logics of the application internally and dyanamically.

c) Improves the productivity becoz programmers just needs to develop only Application specific logics

d) Lots of framework are giving ready made apis to directly implement realtime scenarios or uses-cases.

e) Now a days frameworks have become industry defacto standard to develop medium scale and larget scale projects.

f) Some frameworks like spring boot are giving built-in servers , DB s/ws and etc.. So that we need not arrange them seperately in the devleopment mode and testing mode of the project..

(This feature is very useful while developing applications by taking s/w and hardware setup from cloud env.. like AWS,Google Cloud and etc.. on rental basis.

and etc..

Types of java frameworks (Based on the kind of Apps we can develop)
==========

a) ORM frameworks
b) Web application frameworks/MVC frameworks
c) JEE Frameworks/ Application frameworks
d) WebServices frameworks/ Distributed App Frameworks
e) Big Data frameworks (Not required for Java developer or full statck java developer. Required only for BigData Analyst or Data Scientist)

1) ORM Frameworks (ORM :: Object relational mapping)
==========

=> Provides abstraction on plain JDBC Technology and simplifies the Objects based DB s/w independent Persistence logic developement with out using any SQL queries.

=>The code that performs insert,update,delete ,select operations db s/w tables is called Persinstence logic (eg: jdbc code, hibernate code, eclipse link code and etc..)

=>JDBC code (persistence logic) is DB s/w dependent becoz it uses the Db s/w dependent SQL queries
=> ORM frameworks code (like hibernat code) is portable across the multiple Db s/ws becoz no SQL queries will be used and entire persistence logic will be develope using java objects.

eg::
hibernate ----------> from softTree /redhat (1)
Eclipse link ----------> from Eclipse (2)
OJB ----------> from apache OJB >> Object Java Beans
iBatis ----------> from apache (3)
Toplink ----------> from oracle corp
and etc..

2) Web application frameworks/ MVC frameworks
==========

=> provides abstration on servlet,jsp technologies and simplifies MVC architecture based web application devleopment..

MVC :: Model View Controller (Developing the web application as layered Application)

=>MVC architecture servlet,jsp web applicaiton devleopment (1000 Lines of code)
=>Spring MVC based MVC architectecture web application devleopment (500 lines of code)
=>Spring Boot MVC based MVC architectecture web application devleopment (200 lines of code)

Struts ----------> from apache
JSF ----------> from sun Ms /oracle corp (3) (JSF :: JAva Server Faces)
webwork -----> from Open symphony
ADF ----------> from oracle corp (ADF: Application Development framework)
Spring MVC ----> Interface 21 (team name : pivotal team) (2)
Spring boot MVC ----> Interface 21 (team name : pivotal team) (1)
and etc...

=>Spring framework /spring boot framework are having multiple modules like core , AOP, JDBC, Tx Mgmt , MVC, Rest, security , Batch ,mail and etc.. So we can spring MVC is module of spring framework and spring boot MVC is module of spring boot MVC framework

Fee details
==========

spring statck course + Free React JS or angular or hibernate with JPA :: 8000/-
only spring stack course :: 6000/-

only angular or react JS :: 5000/-
only hibernate with JPA :: 3000/-

Sep 01 Types of Frameworks

Types of java frameworks (Based on the kind of Apps we can develop)

==========

a) ORM frameworks
b) Web application frameworks/MVC frameworks
c) JEE Frameworks/ Application frameworks
d) WebServices frameworks/ Distributed App Frameworks
e) Big Data frameworks (Not required for Java developer or full statck java developer. Required only for BigData Analyst or Data Scientist)

JEE Frameworks/Application frameworks

=> provides abstraction on multiple Java ,JEE technologies and simplifies different Java ,JEE application devleopement process.

eg:: spring , spring boot

=> spring/spring frameworks are JEE frameworks that provides abstraction JAva technologies like JDBC,JNDI,RMI and et.. and also on JEE technologies like servlet,jsp,jms,ejb,java mail and etc.. to simplify all kinds of java ,jee applications development.

=>Using spring /spring boot we can devleop standalone Apps, web applications, distribited Apps and etc..

spring boot =spring++

WebServices framework / distributed Application framework

==========

Distributed App

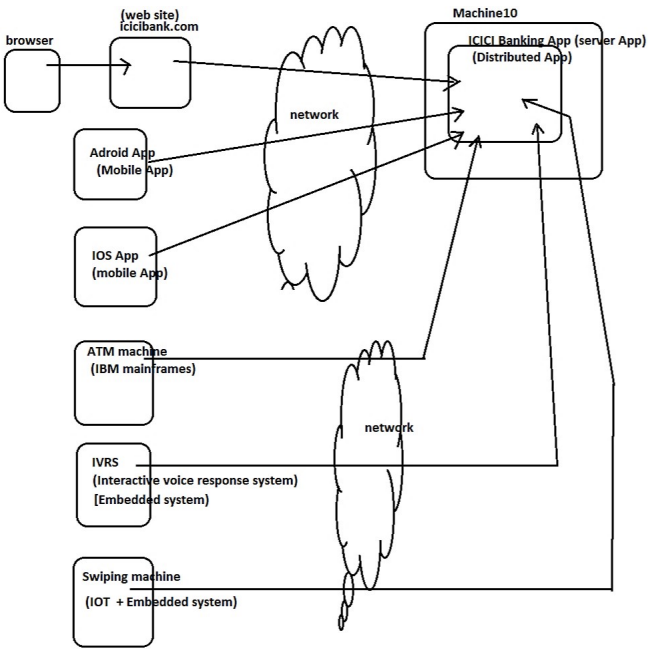
==========

=>The App that allows to have different types of Local or Remote clients to access and execute the logics is called Distributed Application.

eg:: Google Pay App , Phone Pay , PayPall (payment broker), Banking server Apps and etc.

note:: if Server App and client App are using same JVM then that client App is Local Client App called

note:: if Server App and client App are using two different JVMs of same computer or different computer then that client App is Remote Client App called

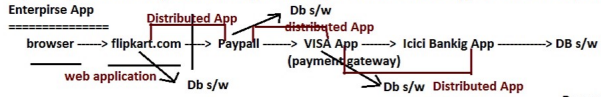


=>To develop distributed App we need to use webServices tehcnologies and frameworks

=>JaxWs , Jax-RPC and etc.. web service technologies
=>Spring Rest , Jersry , Axis , Apache cfx , Rest easy and etc..are web service framworks.

only web application or only destributed App or web application +distributed App is called an Enterprise App

Enterpirse App
==========



Payment gateways
VISA
Master
Mastrero
Rupay
and etc..

What is the diffence web application(website) and Distributed App?

a) It client -server App where Client is a browser and server is s/w App	a) It is Client -server App where both client and server Apps s/w Apps
b) allows only browser as the client	b) allows differen types of client Apps
c) It is Thin Client -Fat Server App	c) It is Fat client - Fat Server App
d) communication model request-response model	d) Communication model is method calls model
e) use servlet,jsp technologies to develop web sites in java	e) use jax-ws, jax-rpc ,jax-rs and etc. technologes to develope web service based distributed Applicaitons
f) use struts, JSF, spring MVC, spring boot MVC and etc.. f/w to devleop java web applications	g) use Spring Rest, Rest easy, jersy , Axis, Apache CFX and etc.. to develope web services based distributed Apps..

=>Standalone App is the App that is specific one to computer and allows only user at time to operate the App
eg:: class with main(-) , calculator App ,core Java Apps

=> The client -Server App where client is browser and server App is s/w App interacting with each other using http protocl in request-response model
eg: nareshit.com , flipkart.com

=> The client -Server App where both client and server Apps s/w Apps talking with each other through method calls sinvocation .. the Server app can have diffrent types of local or remote clients
eg:: UPI Apps, IRCTC App , BSE App, NSE App ,Weather Report App and etc..

Types of java frameworks (Based on the kind of Apps we can develop)

Sep 02 Types of Frameworks -Java Bean

a) ORM frameworks
b) Web application frameworks/MVC frameworks
c) JEE Frameworks/ Application frameworks
d) WebServices frameworks/ Distributed App Frameworks

part of spring/spring boot framework

e) Big Data frameworks (Not required for Java developer or full statck java developer.
Required only for BigData Analyst or Data Scientist)

BigData Frameworks (Required only for data scientist or BigData analyst)

=>The data that is beyond processing and storing capacity using the regular Db s/ws and computers is called Big Data (Generally talks PB,TB, XB,YB of data)

1024 bytes = 1kb
1024 kbs = 1 mb
1024 mbs = 1 gb
1024 gbs = 1 tb
1024 tbs – 1 pb
1024 pbs = 1 xb
1024 xbs = 1 yb

=> This store and process this kind of big data by using ordinary computers of LAN or cloud we need Bigdata frameworks

eg:: Hadoop (java) , spark (java)

=> spring ORM or spring data jpa module is buit on the top of hibernate f/w (ORM f/w)
=> spring MVC module is given as web application f/w
=> spring Rest module is given as webService f/w (Restfull websservices)
So we can spring or spring boot framework are JEE frameworks/ Application framworks/All rounder frameworks

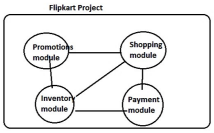
Using spring or spring boot we can develop two types Apps

a) Monolithic Architecture based Apps
b) MicroService Architecture based Apps

Monolithic Architecture

Each each service will be devleoped as each module and all the modules integrated module in a Project i.e we can not use each module as seperate service/project.

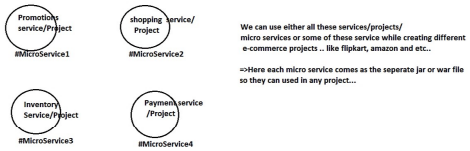
Flipkart Project



=>Here we need to use all modules to gather with in a project i.e we can not use each module as seperate independent service /project.
=> We get reusability or modurality with in project.. not acorss the multiple projects.

=>All modules togather will be released as single jar file /war file

MicroService Architecture Project



We can use either all these services/projects/ micro services or some of these service while creating different e-commerce projects .. like flipkart, amazon and etc..

=>Here each micro service comes as the seperate jar or war file so they can used in any project...

Can u explain JAva Bean class, POJO class, POJI, Bean class/Component class and spring bean?

Java Bean

=> It is a java class that is developed by following some standards
=> It is always used as helper class to pass multiple values as single object from one class to another class of same project or different projects.

The standards are

=>class must be publi class
=>Recomanded to implement java.io.Serialzable(I)
=> All member variables must be taken as private and non-static (bean properties)
=> Make sure that one Zero param constructor is going to come in the class directly or indirectly.
(given by developer) (compiler given)
=> Every bean property (non-static variable) should have setter method and one getter method (setter method is useful to set /modify data of bean property and getter method is useful to read data from bean property)

Example

```
public class CreditCardDetails implements java.io.Serializable{
  //bean properties
  private long cardNo;
  private String cardType;
  private String gateway;
  private int cvv;
  private Date expiryDate;

  //setter and getter methods
  public void setCardNo(Long cardNo){
    this.cardNo=cardNo;
  }
  public long getCardNo(){
    return cardNo;
  }
  ....
  .... //more setter and getter methods ( 4 + 4)
  .....
  .....
}
```

In Java Bean we do not place methods having b.logic.. becoz java bean is just data carrier.

problem:

```
public class StudentProgressReportService{

  public String generateResult(int sno,String sname, String addrs, int m1,int m2,int m3){
      ..... //calculate total, avg
      .... //generate result
      ....
    reuturn "pass"/"fail";
  }
}
```

Client App

```
public calss ClientApp{

  public static void main(String args[]){

    StudentProgressReportService servlce= new StudentProgressReportService();
      String result=service.generateResult(101,"raja","hyd",70,88,99);
      System.out.println(result);
    }
  }
```

Designing java method having more than 3 params is bad pratice , the reasons

a) we must remember the order of args while calling the method
b) Though we do not one or another arg value.. we still need to pass meaning full dummy value.
c) Remembering lengthy signature of the method is very complex.

Solution :: take Java Bean type parameter for java method

```
//Java Bean
public class Student{
  private int sno;
  private String sname;
  private String sadd;
  private int m1, m2,m3;
  //setter and getter methods
    ..... 6 setter , 6 getter methods
    .....
}
```

```
//serlvce class /Bussiness class
public class StudentProgressReportService{

  public String generateResult(Student st){
    .... //read data from "st" obj using
    .... //getter methods and calculate total,avg
    .... and generate reuslt;
    ....
    return "pass"/"fail"
  }
```

Client App

```
public calss ClientApp{

  public static void main(String args[]){

    StudentProgressReportService service= new StudentProgressReportService();

    Student st=new Student();
    st.setSno(101);
    st.setM1(90);
    st.setM2(30);
    st.setSname("raja");

    String result=service.generateResult(st);
      S.o.println(result);
  }
}
```

Advantages of working java bean

a) method signature becomes simple signature to remember
b) While setting data java bean class obj we need not to follow the order
c) the names of setter methods helps a alot while setting data to bean properties of java bean class obj
d) we can ignore to set one or two values which unknow us.

Sep 03 Types of Frameworks -POJO class, POJI, Spring Bean ,Bean class

POJO class
=========
=>Plain Old Java Object class
=> The ordinary java class with out specialities is called POJO class
=> The ordinary java class that compiled only using Jdk libraries is called POJO class
=> The Orindary java class that is not extending and implementing technology, framework apis classes and interfaces is called POJO class.

note: POJO class is not againt of inheritence or implementing interfaces .. It is aganist of inheriting and implementing technology,framework api classes and interfaces.

=> oops, exception handling, awt,swing, collection api , reflectin api, networking api , applets ,multithreading and etc..part java language.
=> Jdbc, servlet,jsp,jndi, jsp ,ejb and etc.. called java techologies
=> struts, spring, hibernate, spring rest and etc.. called java frameworks..

```
class Test{
  ...
  ..
}
POJO class

class Demo extends Test{
  ...
  ...
}
POJO class

class Demo extends Thread{
  ....
  ...
  ...
}
POJO class
(part of java lang api)

class Demo extends HttpServlet{
  ...
  ...
  ...
}
Not a POJO class
(given by servlet api)
```

```
class Demo implements java.rmi.Remote{
  ....
  ...
}
Not a POJO class
(part of RMI technology)

public class Demo1 implements java.sql.Connection{
  ....
  ....
}
not a POJO class
(part of JDBC technology)
```

```
class Test extends Test1{
  ....
  ....
}

class Test1 implements java.sql.Statement{
  ...
  ...
}

Test,Test1 are not POJO classes..
```

```
class Test extends Test1{
  ...
  ...
}

class Test1 implements java.io.Serializable{
  ....
  ...
  ...
}
(part of IO streams (java language))

Test,Test1 are POJO classes..
```

```
public class Demo{
  public void m1(){
    ....
    ... //hibernate API
  }
}
It is POJO class ..but can not be compiled just having jdk libraries in the class path (Exceptional case)

@Entity --> hibernate api annotation
class Demo{
  ...
  ..
}
It is POJO class ..but can not be compiled just having jdk libraries in the class path (Exceptional case)
```

=> POJO class need not follow java bean standards..
=>Every Java bean class is POJO class .. but every POJO class is not Java bean class
=> POJO classes can have b.methods with complex b.logics..

=>Spring supports POJO and POJI model programming i.e the main classes/resources of the spring projects /Apps can be taken as ordinary classes (POJO classes) and ordinary interfaces (POJI) .. with out having depedency with spring APIs .

POJI (Plain Old Java Interface)
====
=> An ordinary interface with out specialaties is called POJI
=> An ordinarly inteface that can be compiled only using jdk libraries is called POJI
=> The interface that is not extending from Technology , framework api interfaces is called is called POJI.

```
interface Test1{
  ...
  ...
}
It is POJI

inteface Test1 extends java.lang.Cloneable{
  ...
  ...
}
(java lang api)
It is POJI
```

```
Interface Test2 extends java.rmi.Remote{
  ....
  ...
  ...
}
(RMI Technology API)
It is not a POJI

interface Test3 extends java.sql.Connection{
  ....
  ...
}
Not a POJI
```

```
interaface Test4 extends java.lang.Runnable{
  ....
  ....
}
It is POJI

interface Test5 extends Test6{
  ...
}
interface Test6 {
  ...
}
"Test5","Test6" are POJIs
```

```
@Remote --> given EJB API
interface Test7{
  ...
  ..
}
not
=>POJI but can be compiled only having jkd libraries
(Exceptional case)
```

=> If project supprots POJO ,POJI model programming while devleoping main classes where b.logics and persistence logics.. then we can move those other frameworks having loose coupling classes to

note:: POJO classes , POJIs are very much specfic to programmer devleoped user-defiend classes.. we can not use these terminologies while refering technology ,framework apis supplied pre-defined classes..

Bean Class /Component class
=========
=>The java class that is havin state (member variables) and behaviour (mehtods) and state is used inside the behaviour having reusability is called Bean class /component class..
=> The methods of component/Bean class contains b.logic with reusability ..
=> This class can be developed as POJO class or not POJO class
=>This class not be devleoped as JAva Bean..
=> In real projects service class (contains b.logic) , DAO class (persistence logic), controller class (monitoring logic) are component /Bean classes

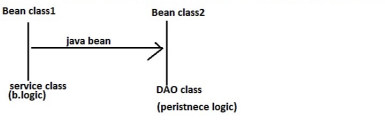
=>Servlet classes are component classes

example
==========

```
public class BankingService{
 //state
 private String branchName;
 private String ifscCode;
 private String location;
 private long mid;
 private long phoneNo;
```
member variables

```
 public String transferMoney(long srcAcno,
                             long destAcno,
                             double amount){
   .....
   ... //b.logic to transfer money
   ...    (state will be used in b.logic)
   }
}
```

=>Java bean class is different from Bean class /component..
=>Java bean is helper class .. acting as data carrier
=>Component /bean class is main class having main logics like b.logics, Persisntence logics and etc...
=>if needed u can use java bean as data carrier b/w two bea classes/component classes



note:: JAva Bean :: Java class acting data carrier having setter and getter methods
POJO class :: Ordinary java class supporting portability across the multiple java projects
POJI :: Ordinary java Interface supporting portability across the multiple java projects
Componet/Bean calss :: class having state , behaviour with reusability .. can be devleoped as POJO or as Non POJO clss.

What is the difference b/w tool and framework?

Spring Bean class
==========

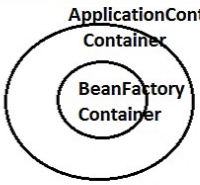
Sep 04 Spring Bean and Spring Core module

=>The java class whose object is created managed by spring Containers (part of spring framework) is called Spring bean i.e the whole life cycle spring bean class will taken care by spring containers.

=>A container is s/w program that manages whole life cycle of given resource/class i.e takes cares of all activities from birth to death (Object creation to object destruction).

=> Container is like an aquirioum taking care of fishes called comps/resources/classes.

=> Servletcontainer takes care of the servlet comps life cycle
=> Jsp Container takes care of the jsp comps life cycle
=> Spring Containers takes cares of spring beans life cycle.
note: spring containers no way related to servlet container and jsp container .In fact spring containers not alternate servlet container, jsp container.
=> Spring f/w given two built-in spring Containers
a) BeanFactory Container (Basic Container)
b) ApplicationContext container (advanced container)
ApplicationContext container =BeanFactory container ++



=> Spring bean class can be a user-defined class or pre-defined class or third party supplied java class but its life cycle management (object creation to object destruction) must be done by Spring containers.

=>Spring Bean class can be developed as POJO class or Non POJO class (POJO class is recomanded)
=>Spring Bean class can be a Java bean class or component /Bean class.

=> Generally , we will not take abstract classes ,interfaces as the spring beans becoz they can not be instantiated (object creation is not possible)

=> To make java class spring bean class we need to <bean> tag in xml file or @Component annotation.

=> We can give instructions/inputs to Springcontainers either using xml file(spring bean cfg file) or using annotations (best).

using xml cfgs

```
<bean id="bankService"  class="com.nt.beans.BankServiceMgmt"/>
```

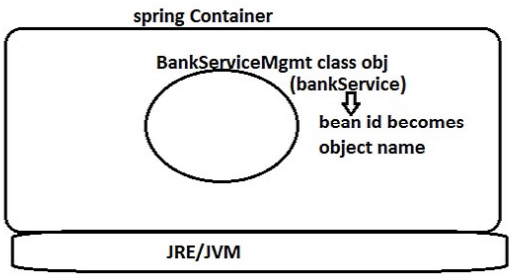
- id="bankService": Internally becomes spring bean class object name when spring container creates the object.
- class="com.nt.beans.BankServiceMgmt": Fully qualified java class to take as spring bean
  - pkg: com.nt.beans
  - class name: BankServiceMgmt

using xml cfgs (Giving details to Spring container using xml file and xml tags)

(or)

```
package com.nt.beans;

@Component("bankService")
public class BankServiceMgmt{
   ....
   ....
   ...
}
```

=> Annotation cfgs (Giving details to spring container using annotations)



=>Generally in core java standalone Apps we manually create object using "new" operator becoz we need to create less no.of objects as standalone App is speciific to computer and will be operated by 1 user at time..

=> While dealing web applicaitons(websites), distributed Apps, enterprise Apps .. we will be having lacs of users using multiple services simultaenously.. So huge no.of obejcts creation ,management , destruction is required continuosly 24/7 For this we take the support containers for automating the whole process.

Q) every JAva bean class is POJO class? (true)
Q) every POJO class is a Java bean class? (false)
Q) Spring Bean class must be a POJO class? (false)
Q) Spring Bean class can be a POJO class? (true)
Q) Thrird paty supplied java class can be taken as spring bean? (true)
Q) we create objects for spring bean classes? (false)
Q) spring containers takes all java classes of jdk library as spring bean calsses automatically? (false)
Q) Java bean can not be taken as spring bean? (false)
Q) The id of spring bean becomes spring bean class object intenrally (true)
Q) Every Spring bean class must component class ? (false)
Q) Component class can be taken as spring bean class ? (true)

Two types of frameworks based on mode of the Application development
==========

a) Invasive Framework
b) Non-Invasive Framework

Sep 06 Spring Core Module

a) Invasive Framework

=>Here the resources/classes framework based App devleopment are tightly coupled with Framework APIs i.e the classes of the App development should implement or should extend from Framework API interfaces or classees.
=> HEre the App classes developeed progammer for certain Framework .. should always be executed in the same framework and can not be moved to another frameworks.

eg: Struts Framework

=> Invasive frameworks do not support POJO and POJI model programming

b) Non-invasive Framework
==========

=>Here the resources/classes framework based App devleopment are loosely coupled with Framework APIs i.e the classes of the App development need not implement or extend from Framework API interfaces or classees.

=>HEre the resources/classes of the App develeopment will be developed as POJO classes i.e it support POJO/POJI model programming..

=>HEre the App classes develped from certain framework can be moved another frameworks dynamically becoz they are POJO classes

eg: Hibernate, Spring , spring boot, JSF and etc..

---

spring
=====
type :: Application framework or JEE framework
vendor :: Interface21 /pivotal team
open source framework
Creator :: Mr.Rod jhonson

Repository means :: small perminent storage place

spring libaries (jars) :: earlier they used give in form of zip file .. now they stopped giving..
asking to use Maven /Gradle dependencies mechanism (download jar files from internet repositories dynamically)

How an Open Source company survive financially?

Ans) a) By conducting corporate training on new technologies
b) By conducting certification exams
c) By conducting certification exams traning programs
d) By providing services to s/w companies if they struckup in the middle of the Project devleopment.
and etc..

Q) Is spring is alternate for EJB?

Ans) No, EJB is just distributed Technology to develop distributed Apps where as Spring is all-rounder or JEE framework which can be used to devlep standalone Apps,web applications, distributed Apps and etc..

Q) Is spring alternate to Struts ?

Ans) NO, Struts is just for web application development where as Spring is all-rounder or JEE framework which can be used to devlep standalone Apps,web applications, distributed Apps and etc..

Q) Is spring alternate for JEE Technologies?

Ans) No, Spring is a framework internally using JEE technologies to simplify the Application devlelopment.. i.e spring is altenrnate JEE technologies rather it compliments them by internally using them,

Q) Is spring boot is alternate spring ?

Ans) No, spring boot internally uses spring framework and simplifies spring Apps development.
note: spring boot extension spring to simplfiy spring Apps development.

note:: Boilerplate code (comnon logics) of JAva, JEE TEchologies programming will be avoided by spring frameworknote::
note:: Boilerplate code (comnon logics) of spring programming will bed avoided by spring boot framework

---

Spring/spring boot modules: core, aop, jee, Txmgmt, jndi,orm, jdbc, mvc, mail , jms and etc..
sprjng/spring boot extension modules :: security , batch, oauth , social , spring data jpa, spring mongo db ,sping cloud,spring rest and etc..

spring micro services concepts :: eureka server and client,
Load Balancer,
Flyway Database,
ZUUL API gateway,
WebClient VS FeginClient,
Spring config server & client,
circuit breaker,
Retry limiter pattern,
rate limiter pattern,
and et...

Why spring or spring boot framework libraries are given in the form modules?

Ans) Generally we do not or we can not use all spring/spring boot modules in one Project development.. we use different sets of modules in different projects.. So they releases spring /spring boot libraries (jar files) as modules.. So that we can add only thosse libraries of modules as per the project requirement.

Why the name spring is given for spring framework?

Ans) The creator liked spring season of his nepal trip ...so he named framework as spring boot

Spring core module
==========
=>Base module for other other modules
=>if this module is used alone ..we can develop only standalone Apps in spring
=> This modules gives Tow spring containers(BeanFactory ,AppicationContext) to perform spring beans life cycle management and Dependency Management

we developing spring Apps in 4 approaches
==========
approach1) Using Xml driven cfgs
approach2) Using Xml + Annotation driven cfgs ✓2
approach3) Using 100% code driven/Java Config cfgs
approach4) Using Spring Boot ✓1

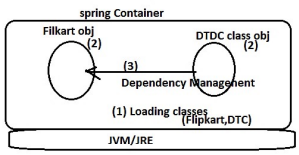
spring bean life cycle management
==========
=>if we given java class to Spring Containers .. it will care of total life cycle of that spring Bean like Loading bean class, creating object ,management object, destorying obejct.

Dependency Management
==========
=> Assigning/Arranging Dependency class object to target class object dynamically is called Dependency Management.

=>The spring bean/class that uses other spring services is called target class /target spring bean /main class
=>The spring bean/class that acts as helper class for other spring beans/classes is called Dependent class/helper class

=>Flipkart need DTDC services for courier activies So flipkart is target class ,DTDC is dependent class.
=>Vehicle needs Engine services for moving/driving activies So Vechile is target class ,Engine is dependent class.
=>Student needs CourseMaterial services for prepration activies So Student is target class ,CourseMaterial is dependent class.

=>if we given target and dependent class to spring containers.. it not only manages their life cycle.. it also assingns /arranges to Depedent class obj to target class obejct..So that target class can use dependent class services directly with out arranging it seperately..



(2) creating objects for spring bean clases

(3) assigning depenent class object to target class obj

(1)+(2)+(3) :: spring container performing spring bean life cycle management and Dependency management.

Dependency Mgmt

Spring core module

=>Base module for other other modules
=>if this module is used alone ..we can develop only standalone Apps in spring
=> This modules gives Tow spring containers(BeanFactory ,AppicationContext) to perform spring beans life cycle management and Dependency Management

spring bean life cycle management

=>if we given java class to Spring Containers .. it will care of total life cycle of that spring Bean like Loading bean class, creating object ,management object, destorying obeject.

we developing spring Apps in 4 approaches

approach1) Using Xml driven cfgs
approach2) Using Xml + Annotation driven cfgs ✓2
approach3) Using 100% code driven/Java Config cfgs
approach4) Using Spring Boot 1
(Approach 3+ AutoConfiguration)

(Based on the jar files added to the project certain pre-defined classes automatically become spring beans and also Dependency Management)

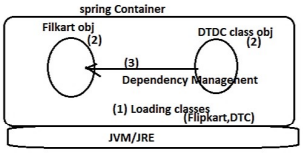
Dependency Management

=> Assigning/Arranging Dependency class object to target class object dynamically is called Dependency Management.

=>The spring bean/class that uses other spring services is called target class /target spring bean /main class
=>The spring bean/class that acts as helper class for other spring beans/classes is called Dependent class/helper class

=>Flipkart need DTDC services for courier activies So flipkart is target class ,DTDC is dependent class.
=>Vehicle needs Engine services for moving/driving activies So Vechile is target class ,Engine is dependent class.
=>Student needs CourseMaterial services for prepration activies So Student is target class ,CourseMaterial is dependent class.

=>if we given target and dependent class to spring containers.. it not only manages their life cycle.. it also assingns /arranges to Depedent class obj to target class obejct..So that target class can use dependent class services directly with out arranging it seperately..



(2) creating objects for spring bean clases
(3) assigning depenent class object to target class obj

(1)+(2)+(3) :: spring container performing spring bean life cycle management and Dependency management.

Dependency Management

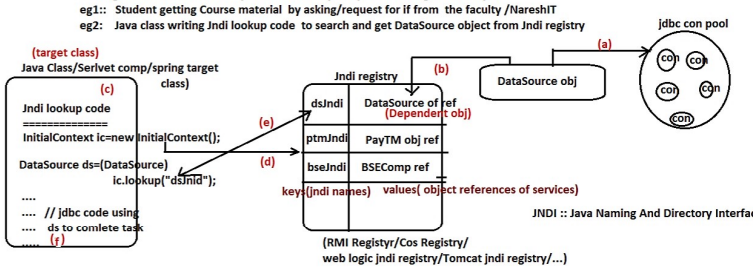
=> It is all about keeping Dependent class object ready for target class object ,So the target class can the use the services of dependent class while executing its own services.

Two types Dependency Management (is called as IOC :: Inversion of Control)

1.Dependency Lookup
2.Dependency Injection

1.Dependency Lookup

=> Here the target calss should spend time and should write some logic to search and get Dependent class obj from various resources/places.
=> target class searches for Dependent class obj and pulls it to target class object
eg1:: Student getting Course material by asking/request for if from the faculty /NareshIT
eg2: Java class writing Jndi lookup code to search and get DataSource object from Jndi registry



JNDI :: Java Naming And Directory Interface.

=>To provide global visibility and accessiblity to any java object we keep its reference in the Jndi registry
=> In Jndi registry we can keep only java objects references
=> The process searching and getting object references from Jndi registry is called Jndi lookup operation
eg:: Cos registry, RMI registry, weblogic registry and etc..
note:: InitialContext object establisht the connection b/w Java App and Jndi registry and provides base to perform lookup operation.

pros of Dependency Lookup

=> The target class can serach and get only required dependent class objects

cons

=>The target class should spend time and should write logic to search and gets Dependent class objs.

2.Dependency Injection

=> HEre the underlying container/server /framework/ JVM/JRE /another service dynamically assigns Dependent class object to target class obejct at runtime.

=>HEre the undelrying server/container/.... pushes dependent class obj to target class object

eg1:: Student getting Course material dynamically facutly the moment he joins the course
eg2:: The way JVM assings default values(dependent values) to object (like 0,0.0,null and etc..) the moment JVM creates the obejct of java class (target class)
eg3:: The way Servlet Container assigns Serlvetconfig object to Servlet class object that moment it creates Serlvet class obj

pros

=>Target class can directly use depedent class object and its services
=> Target class need not to spend time or need not write logics to search and get dependent class obj becoz the underlying container/server /.. takes cares of this work

cons

=> The underlying server/container/... may inject both neccessary and unnecessary objects.

=>spring containers support both Dependency lookup and Depedency Injection.. but realtime projects we use depedency injection.

Depedency Management/IOC in spring

1.Dependecny Lookup
2.Depedency Injection

a) setter Injection
b) constructor Injection
c) Aware Injection /Interface injection/Contexual Injection
d)Lookup Method Injection
e) Method Injection/Method Replacer
f) Field Injection (only in annotation driven cfgs)

What is the difference b/w IOC and Depenency Injection?

=>spring containers support both Dependency lookup and Depedency Injection.. but realtime projects we use depedency injection.

Spring Core Dependency Management types

Depedency Management/IOC in spring
==========

1.Dependecny Lookup

2.Depedency Injection
==========

a) setter Injection
b) constructor Injection
c) Aware Injection /Interface injection/Contexual Injection
d)Lookup Method Injection
e) Method Injection/Method Replacer
f) Field Injection (only in annotation driven cfgs)

What is the difference b/w IOC and Depenency Injection?

IOC : Inversion of Control

=>IOC is specification providing rules and guidelines to manage dependency between target and dependent classes .. It is like a Plan.

=>Dependency lookup and Dependency Injection are two implementation models of IOC /Dependency Management (It is like execution of a plan)

Spring Container are designed to perform Dependency Management /IOC .. So they are also called as IOC containers..

Why it is named as IOC ?
(or)
Why Dependency Management is named as IOC?

=>IOC means Inversion of Control i.e reversing the control/process that is actually happening..
=>Generally programmers hold control to assign /arrange depenedency class object to target class obj.. In IOC implementations that control/power is totally taken by the underlying server/container/framework/JVM.. i.e it is reverse/inversion of the regular process ,So it is named as IOC (given by Martin Flower)

=>Spring Containers/IOC container are light weight containers becoz
a) They can created in any kind java code by just instantiating one pre-defined java class given spring API (spring framework packages)
b) Spring Container/IOC container can be created directly on top of JVM i.e no need of having any heavy webServers(Tomcat) and Application servers(widlfy)

note: Spring containers are not alternate for servlet container, jsp Contaienrs
note:: In Xml driven cfgs based spring App development .. we need to pass spring bean cfg file(xml file) as input file while creating spring Cotnainer/IOC contaienrs..

Creating BeanFactory container in spring App
==========

=> By creating object for a class implemeting BeanFactory(I) .. we can create BeanFactory Container.. spring api provides multiple classes implementing this BeanFactory Interface.. One of them is "XmlBeanFactory" class.

org.springframework.beans.factory

Interface BeanFactory

All Known Subinterfaces:

ApplicationContext, AutowireCapableBeanFactory, ConfigurableApplicationContext, ConfigurableBeanFactory, ConfigurableListableBeanFactory, ConfigurableWebA... calBeanFactory, ListableBeanFactory, WebApplicationContext

Interface in org.springframework.context

All Known Implementing Classes:

AbstractApplicationContext, AbstractAutowireCapableBeanFactory, AbstractBeanFactory, AbstractRefreshableApplicationContext, AbstractRefreshableConfigApplicationContext, AbstractRefreshableWebApplicationContext, AbstractXmlApplicationContext, AnnotationConfigApplicationContext, AnnotationConfigWebApplicationContext, ClassPathXmlApplicationContext, DefaultListableBeanFactory, FileSystemXmlApplicationContext, GenericApplicationContext, GenericGroovyApplicationContext, GenericWebApplicationContext, GenericXmlApplicationContext, GroovyWebApplicationContext, ResourceAdapterApplicationContext, SimpleJndiBeanFactory, StaticApplicationContext, StaticListableBeanFactory, StaticWebApplicationContext, XmlBeanFactory, XmlWebApplicationContext

```
//Locate and hold Spring bean cfg file
FileSystemResource res=new FileSystemResource("...../applicationContext.xml");
   This class internally uses
     java.io.File class to locate
     hold
   and spring bean cfg name and location

//create BeanFactory IOC container
XmlBeanFactory factory=new XmlBeanFactory(res);
```

=>The above code can be placed in any kind of Java App to create BeanFactory contnainer

=>any <filename>.xml can be given as spring bean cfg file .but recomanded to take applicationContext.xml as the spring bean cfg file as the spring Container name "ApplicationContext".
=>spring bean cfg file contains the following cfgs (instructions) to Spring containers/IOC container

a)spring bean cfgs
b)depenency management cfg
c) spring bean life cycle cfgs
d) spring bean ids alia names cfgs
and etc..

if code is placed in standalone App

Standalone Java App
BeanFactory Container
applicationContext.xml
data..
JRE/JVM

(we use this model in spring's standalone Apps)

if code in placed in servlet comp of web application deployed Tomcat server

Tomcat server
Servlet Container
BeanFactory container
applicationContext.xml
data...
JRE/JVM

(we use this model in spring MVC, Spring Rest and MicroServices env..)

---

setter Injection -- Dependency Injection
==========

=> if the Spring Container uses setter method of target class to inject/assign Dependent class object to target class object then it is called setter Injection..
=>For this we need to given instructions to IOC container using <property> in springBean cfg file (applicationContet.xml) and we should also place setter method in target class.

com.nt.beans.WishMessageGenerator (target class) → java.util.Date (dependent class)

This class uses in the injected Date class obj having system date and time to get current hour of the day to generate wish message based on the current of the day.

WishMessageGenerator.java (target class)
==========

```
package com.nt.beans;
      import.java.util.*;
public class WishMessageGenerator{
      //HAS- A property (Composition)
     private Date date;

      //setter method supporting setter injection
     public  void setDate(Date date){
       this.date=date;
     }

    //b.method
    public String generateMessage(String user){
     //get current of the day
       int hour=date.getHour(); // current hour of day in 24 hours format (0 to 23)

     if(hour<12)
       return "GM:"+user;
     else if(hour<16)
       return "GA:"+user;
     else if(hour<20)
        return "GE:"+user;
     else
       return "GN:"+user;
     }
}
```

applicationContext.xml (com/nt/cfgs)

```
<beans....>
    <!-- spring beans cfgs -->
   <bean id="dt"  class="java.util.Date"/>
          bean Id   fully qualified  class name
   <bean id="wmg"   class="com.nt.beans.WishMessageGenerator">
        <property name="date"  ref="dt"/>
   </bean>
              property name   dependent class bean id to injection depedent sprjng bean class obj to target clas sobj
</beans>
```

Dependent clss
for setter injection
target class

Client App (SetterInjectTest .java)
==========

```
package com.nt.test;
   ...  ...  //pkg imports

public class SetterInjection{
    p  s  v main(String args[]){
```

Spring Core Module 1st app development

setter Injection -- Dependency Injection
==========

=> if the Spring Container uses setter method of target class to inject/assign Dependent class object to target class object then it is called setter Injection..
=>For this we need to given instructions to IOC container using <property> in springBean cfg file (applicationContet.xml) and we should also place setter method in target class.

com.nt.beans.WishMessageGenerator (target class) → java.util.Date (dependent class)

This class uses in the injected Date class obj having system date and time to get current hour of the day to generate wish message based on the current of the day.

WishMessageGenerator.java (target class)
==========

```
package com.nt.beans;
    import.java.util.*;
public class WishMessageGenerator{
     //HAS- A property (Composition)
    private Date date;

       //setter method supporting setter injection
    public  void setDate(Date date){
      this.date=date;
    }

    //b.method
    public String generateMessage(String user){
     //get current of the day
       int hour=date.getHour(); // current hour of day in 24 hours format (0 to 23)

     if(hour<12)
       return "GM:"+user;
     else if(hour<16)
       return "GA:"+user;
     else if(hour<20)
        return "GE:"+user;
     else
       return "GN:"+user;
      }

}
```

applicationContext.xml (com/nt/cfgs) (spring bean cfg file)

```
<beans....>
   <!-- spring beans cfgs --> (7)
   <bean id="dt"  class="java.util.Date"/>
          bean id        fully qualified  class name
                                                          Dependent clss
   <bean id="wmg"  class="com.nt.beans.WishMessageGenerator">   target class
(5)  <property name="date"  ref="dt"/>   for setter injection
   </bean>
          (8) property name   (6) dependent class bean id to injection depedent sprjng bean class obj to target clas sobj
</beans>
```

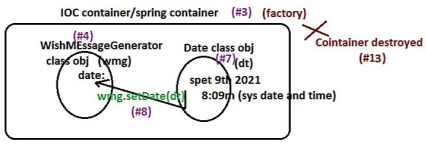
(4)

=>Spring Container /IOC container creates spring Bean class object with bean id (takes bean id as the object name /refereence variable name)..

=>Spring Cotnainer identities spring bean with its bean id if want to get springbean class from spring container and if want to inject one spring bean class object to another spring bean class object we need to use bean ids.

Client App (SetterInjectTest .java)
==========

```
package com.nt.test;                (0) Run the App
   ...  ...  //pkg imports
public class SetterInjection{        (1)
   p  s  v main(String args[]){

     //Locate and hold spring cfg  file name and location
      FileSystemResource res=new FileSystemResource(".../com/nt/cfgs/applicationContetx.xml");
(2)  //create IOC Cotainer
     XmlBeanFactory factory=new XmlBeanFactory(res);

     //Ask SpringContainer/IOC container to give Target spring bean class obj
(9)    Object obj=factory.getBean("wmg");   (3)

      //invoke b.method
      //   String result=obj.geneateMessage("Raja");
              (if Super class ref variable is pointing to sub class obj. then
              can not call direct methods of sub class .. So the above throws
              compile time error.. To make it possbile use downcasting/typecasting)
       //type casting
                                  (10)
       WishMessageGenerator generator=(WishMessageGenerator)obj;

        //invoke b.method
                                  (11)
           String result= generator.generateMssage("Raja");

        System.out.println("Wish MEssage is::"+result);   (12)

      }  (13)
  }
```



w.r.to code
==========

3) Client App ccall factory.getBean(-) method
4) IOC cintaienr searches for sprigng bean cfg whose gean bean id is "wmg" and loads class and creates the object
5 ,& 6) Based <property name="date" ref="dt"> IOC container becomes ready to perform setter injection "date" property
7) takes ref="dt" and seraches for spring bean class cfg with bean id is "dt" and finds java.util.Date class, So Loads and creates the object
8) Spring container creates calls wmg.setData(dt) method to complete setter injection
9) factory.getBean("wmg") returns WIshMssageGenerator class obj having Date class obj in it back to Client App and we are refering that object with java.lang. Object class ref variable.

Setup up required for 1s tApplication development
==========

a) Jdk any version (jdk1.8+) (jdk 16)
b) Eclipse IDE (2019+) (eclipse 2021-06)
|—>contains built-in maven support

download eclipse (as zip file)::
https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2021-06/R/eclipse-jee-2021-06-R-win32-x86_64.zip

---

Procedure to devlope spring First App (setter Injection using Eclipse IDE)
==========

step1) Lanuch eclipse IDE by choosisng workspace folder name and location. (G:\Worskpaces\Spring\NTSPBMS615)
The Folder where projects of eclipse IDE will be saved..

step2)

maven /gradle —>Build tools Simplifying all aspects project devleopment, executing ,Testing, release and etc..

=>Building the Project/App is nothing keeping the Project/App ready for execution /release.

maven/gradle features
==========

a) gives standard project directory strcutures (archetypes) for different types apps
b) Allows to donwload jar files/libraries/Dependencies dynamically to projects from Internet reposorities
c) Gives built-in decompilers to see the source code of api.
d) Allows to pack the code in different formats like jar file ,war file and etc..
e) Can run JunitTest cases and can generate Test report
f) if we add main jar files/libraries ..it automatically downloads dependent jars/libaries..
and etc..

maven /gradle -->Build tools Simplifying all aspects project devleopment,
executing ,Testing, release and etc..

=>Building the Project/App is nothing keeping the Project/App ready for execution /release.

Spring Setter Injection exaxmple App

maven/gradle features
=====================
a) gives standard project directory strcutures (archetypes) for different types apps
b) Allows to donwload jar files/libraries/Dependencies dynamically to projects from Internet reposorities
c) Gives built-in decompilers to see the source code of api.
d) Allows to pack the code in different formats like jar file ,war file and etc..
e) Can run JunitTest cases and can generate Test report
f) If we add main jar files/libraries ..it automatically downloads dependent jars/libaries..
and etc..

Setup up required for 1s tApplication development
==========
a) Jdk any version (jdk1.8+) (jdk 16)
b) Eclipse IDE (2019+) (eclipse 2021-06)
|--->contains built-in maven support

download eclipse (as zip file):: https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2021-06/R/eclipse-jee-2021-06-R-win32-x86_64.zip

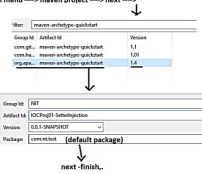
Procedure to devlope spring First App (setter injection using Eclipse IDE)
==========

step1) Lanuch eclipse IDE by choosisng workspace folder name and location. (G:\Worskpaces\Spring\NTSPBMS615)
The Folder where projects of eclipse IDE will be saved..

step2) create Maven Project by taking "maven-archetype-quickstart" as the archetype (Project Template for standalone App)

=>maven is providing 2000+ ready made archetypes as readymade Project templates .. the popular archetypes are
a) maven-archetype-quickstart ----> for standalone Apps
b) maven-archetype-webapp ------> for web applications/ Distributed Apps

File menu ----> maven project ----> next ---->



next -finish,.

In Maven Project/jar file/plugin are identified with 3 details

group id :: Company name
artifactid :: Project name/jar file name /plugin name
version :: Projection version/ jar file version/ plugin version

step3) chanage java version of the project to latest version
in pom.xml

```
16
<maven.compiler.source>1.7</maven.compiler.source>
<maven.compiler.target>1.7</maven.compiler.target>
16
```

POM : Project Object Model

Right click on project ---->maven --update.

step4) Add the following <depedency>tag in pom.xml file to download spring core module jar file to the classpath of project (build path)

=>maven dependencies means the libariries nothing but jar files

In <dependencies> section of pom.xml file
==========

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context-support -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context-support</artifactId>
  <version>5.3.9</version>
</dependency>
```

save

To collect this tags go to mvnrepository.com ---> search for spring context support--> select version (5.3.9) ---> go to maven tab ---> copy xml code ---> paste in <dependencies> section of pom.xml

=>This code not only gvies spring-context-suppot-5.3.9.jar file and also gives its dependant jar files.

By default every Project these two jar files.. main jar file

dependent jar file

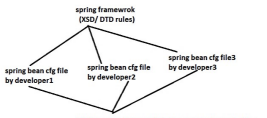
step5) Understand The directory stucture of the Project?

Programmer's testing on his own piece code is called unit testing. we can do that unit testing by using the tool called junit.

step6) Develope the packages having source code in "src/main/java" filder..

=>To make all spring pgogrammers developing spring bean cfg file by fllowing same set of rules and guidelines (i.e same tags and and attributes) we need to import DTD /XSD rules given by spring framework in spring bean cfg file..

DTD :: Document Type Definitation (old)
XSD :: Xml schema Definitation (new)



All these developers will devleop spring bean cfg files by using tags and attributes specified in XSD /DTD rules..

//WishMessageGenerator.java
==========

```
//WishMessageGenerator.java
package com.nt.beans;

import java.util.Date;

public class WishMessageGenerator {
	//HAS-A property (supporting composition)
	private Date date;

	public WishMessageGenerator() {
		System.out.println("WishMessageGenerator:: 0-param constructor");
	}

	//setter method for setter Injection (Alt+shift+s , r -->select setDate(-) ---> ok)
	public void setDate(Date date) {
		System.out.println("WishMessageGenerator.setDate(-)");
		this.date = date;
	}

	//B.method
	public String generateMessage(String user) {
		System.out.println("WishMessageGenerator.generateMessage()");
		//get current hour of the day
		int hour=date.getHours(); // 24 hours format (0 to 23)
		//generate wish messsage
		if(hour<12)
			return "Good Morning::"+user;
		else if(hour<16)
			return "Good AfterNoon::"+user;
		else if(hour<20)
			return "Good Evening::"+user;
		else
			return "Good Night::"+user;
	}//method
}//class
```

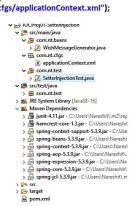
applicationContext.xml (right click on com.nt.cfgs -->new ---> file -->search xml file ---> xml from template --> ....

Collect it from Internet by searching for same spring cfg file

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- spring bean cfgs -->
    <bean id="dt" class="java.util.Date"/>

    <bean id="wmg" class="com.nt.beans.WishMessageGenerator">
      <property name="date" ref="dt"/> <!-- Setter Injection -->
    </bean>

</beans>
```

SetterInjectionTest.java

```
package com.nt.test;

import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.FileSystemResource;

import com.nt.beans.WishMessageGenerator;

public class SetterInjectionTest {

	public static void main(String[] args) {
		//Hold spring bean cfg file name and location (ctrl+shift+o :: To import pkg)
		FileSystemResource res=new FileSystemResource("src/main/java/com/nt/cfgs/applicationContext.xml");
		//create IOC containe (BeanFactory container)
		XmlBeanFactory factory=new XmlBeanFactory(res);
		//get Target spring bean class object
		Object obj=factory.getBean("wmg");
		//typecasting
		WishMessageGenerator generator=(WishMessageGenerator)obj;
		//invoke the b.method
		String result=generator.generateMessage("raja");
		System.out.println("Wish Message is ::"+result);

	}//main
}//class
```



step7) Run the Client App

Go go Client App's .java file ---> run as---> Java App

(or)

ctrl +F11

Xml file /document Syntax rules are

Sep 13 First App internals

a) xml tags and attributes are case-sensitive
b) xml docs are strictly typed docs (rules must be followed)
c) every Open tag should have closing tag
d) Tags must be nested in proper order (In which order they are opened .. they must be closed in reverse order)
e) Attribute values must be quoted (must be there either in single quotes or double quotes)
f) The First tag in xml is called root tag and once is closed .. we should not place any other content in xml file.

What is main difference b/w HTML and XML?

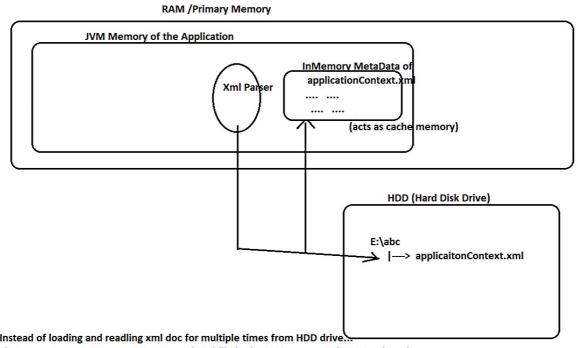
Ans) HTML is given to display to data/documents by applying styles where as XML is given to construct Data by describing it.
=>HTML is loosely typed .. where as XML is strictly typed.
=>HTML tags and attributes are pre-defined.. where XML tags and attributes are user-defined.

=> if Xml doc /file is statisfying its xml syntax rules then is called "well-formed" xml document.
=> if Xml doc /file is statisfying the imported DTD/XSD rules then is called "Valid-formed" xml document.

=> We can Xml Parser (software program) to check wheather given XML doc is well-formed or not , valid or not and also to read and process the xml documents.

Java based Xml parsers are
a) SAX parser (Simple API for xml processing)
b) DOM Parser (Document Object Model)
c) JDOM parser (Java Document Object Model)
d) DOM4J Parser (DOM for Java)
and etc..

=>java Xml parser loads the given XML doc, checks wellformed, valid or not , reads the document and prepares In Memory Metadata in the memory (JVM memory)where the current Application is of RAM is running..

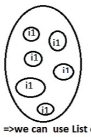


=> Instead of loading and readling xml doc for multiple times from HDD drive.. it is better to create InMemory MetaData of Xml file in the JVM memory of RAM and use its content for multiple times for better performence..

=> Spring Containers/IOC containers are having built-in Xml parser called SAX Parser.. So Spring containers also prepares InMemory MetaData of spring bean cfg file and uses that content for multiple times as discussed above.

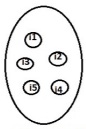
What is the difference b/w pool and cache?

Pool gives reusability of same items



=>we can use List collection to create our own pool

cache gives reusability of different items



=>we can use Map Collection to create our own cache ..

Problems with "new" operator

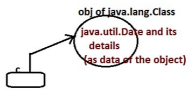
=> Test t=new Test();
=> "new" operator creates the object of java class at runtime.. but expects the presence of java class from compile time onwords i.e we can not use new operator to create object java class whose class name comes to our App dynamically at runtime either through xml file or properites file or cmd line args and etc.. For this we need to "newIsntance()" method of relfection API

note: spring container also uses "reflection API" newInstance() method to create spring bean class objs becoz spring cotnainer gets the class names of spring bean from spring bean cfg file dynamically at runtime.

Relfection API newInstance() method code

a) Load java class into App dynamically at runtime

```
Class c=Class.forName(args[0]);
```
assume the class name is "java.util.Date"



b) create the object of Loaded class using "c".

```
Object obj=c.newInstance();
```
creates the object for the Loaded class (java.util.Date) using 0-param constructor Internally..

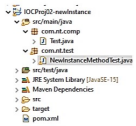
java.lang.Class is pre-defined class and forNAme(-) is static method of java.lang.Class

from Java 11 newInstance() of java.lang.Class is deprecated and recomading to use newIstance() of java.lang.reflect.Consturctor

```
System.out.println(obj.toString());
```

example App

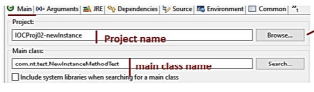
```
//Test.java
package com.nt.comp;

public class Test {
	private int a=10;
	private String b="hello";

	public Test() {
		System.out.println("Test:: 0-param constructor");
	}

	//alt+shift+s , s
	@Override
	public String toString() {
		return "Test [a=" + a + ", b=" + b + "]";
	}
}
```

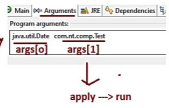



```
//NewInstanceMethodTest.java
package com.nt.test;

public class NewInstanceMethodTest {

	public static void main(String[] args) throws Exception{
		//Load classes
		Class c1=Class.forName(args[0]);
		Class c2=Class.forName(args[1]);

		//create the objects Loaded classes
		Object obj1=c1.newInstance();
		Object obj2=c2.newInstance();

		System.out.println(obj1.toString());
		System.out.println(obj2.toString());


	}

}
```

Passing cmdline arg for eclipse Java App

Right clik on client App --->Runs as ---> run configurrations ---->





apply ---> run

Working with newInstance() method of java.lang.reflect.Constructor class for dynamic object creation

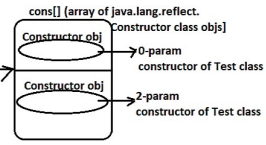
Sep 16 Reflection Api

=>This method can use either 0-param constructor or parameterized constuctor for object creation where as the newInstance() method of java.lang.Class can use only 0-param constructor for object creation.

sample code on newInstance() of java.lang.reflect.Constructor class

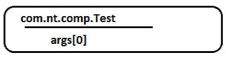
```
//Test.java
package com.nt.comp;

public class Test {
        private int a=10;
        private String b="hello";


        public Test() {
                System.out.println("Test:: 0-param constructor");
        }

        public Test(int a,String b) {
                System.out.println("Test:: 2-param constructor");
                this.a=a;
                this.b=b;
        }



        //alt+shift+s , s
        @Override
        public String toString() {
                return "Test [a=" + a + ", b=" + b + "]";
        }
}
```

```
// NewInstanceMethodTest1.java
package com.nt.test;

import java.lang.reflect.Constructor;

public class NewInstanceMethodTest1 {

        public static void main(String[] args) throws Exception{
                //Load classes
                Class c1=Class.forName(args[0]);
                //get All constructor of the Loaded class
                Constructor cons[]=c1.getDeclaredConstructors();
                //Dynamic object using 0-param constructor
                Object obj1=cons[0].newInstance();
                System.out.println(obj1);
                System.out.println("......................");
                //Dynamic object using 2-param constructor
                Object obj2=cons[1].newInstance(100,"india");
                System.out.println(obj2);
        }

}
```



[Elements in this array represents gives Test class constructors in order they are present in the source code "Test" class]

While running code ..Run As -->Run cfgs --->arggguments tab



-->apply ---> run.

First spring App Internal Flow (Complte flow end to End)

a) Enduser runs the Client App and man(-) method executes

b) //Hold spring bean cfg file name and location (ctrl+shift+o :: To import pkg)
FileSystemResource res=new FileSystemResource("src/main/java/com/nt/cfgs/applicationContext.xml");

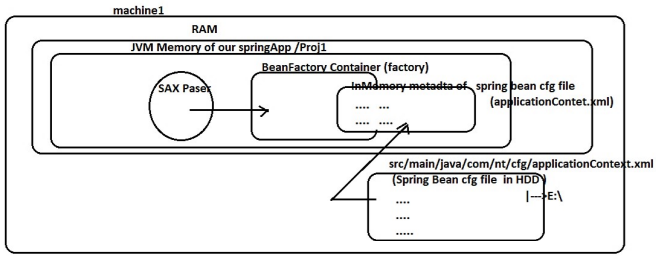
=>Internally uses java.io.File class to hold the name and location of given spring cfg file.. It does not actullay locate the given file.. it just holds name and location of the file.

c) //create IOC container (BeanFactory container)
XmlBeanFactory factory=new XmlBeanFactory(res);

these 3 operations are done by Xml parser of IOC container

i) takes given "res" object (FileSystemResource obj) .. to get the name and location spring bean cfg file
ii) Loads the spring bean cfg file (applicationContext.xml) and wheater it is well-formed or not and valid or not .. if not thorws exception. (checks)
iii) if the spring bean cfg file is well-formed and valid .. then InMemory Metadata of spring bean cfg file will be created in the JVM Memory of the RAM where the spring app is executing.

iv) create IOC container/spring container of type BeanFactory having the above created InMemory MetaData of spring bean cfg file and returns XmlBeanFactory object ref (factory) representing IOC container



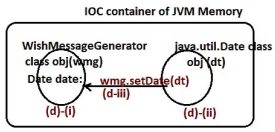
d) Object obj=factory.getBean("wmg");

i) factory.getBean("wmg") method called on IOC container object (factory) ,makes the IOC container to search "wmg" bean id spring bean class cfg in the InMemory MetaData of spring cfg file(applicationContext.xml file's InMemory MetaData) and finds com.nt.beans.WishMessageGenerator having setter injection cfg (becoz of <property>).. So its loads the class cretes the object using reflection api.

```
//Loading class
Class c1=Class.forName("com.nt.beans.WishMessageGenerator");
//get all constructors
Constructor cons[]=c1.getDelcaredConstructors();
//create dynamic object
  Object obj=cons[0].newInstance();
  WishMEssageGenerator wmg=(WisMessageGenerator)obj;
```

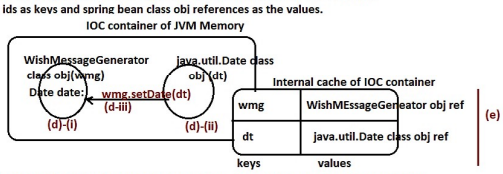
ii) Goes to <property> tag ref="dt" attribute and searches for the spring bean class cfg whose bean id is "dt" in the InMemory MetaData of applicationContext.xml and finds "java.util.Date" class cfg. Notices that no injects are configured ,So loads the class and creates the object using reflect api.

```
//Loading class
Class c2=Class.forName("java.util.Date");
//get all consturctors
  Constructor cons1[]=c2.getDeclaredConstructors();
// create dynamic object
   Object obj2=cons1[0].newInstance();
   java.util.Date dt=(java.util.Date) obj2;
```



iii) completes the setter Injection by calling wmg.setDate(dt) method based <property name="date" ref="dt"> tag of InMemory MEtadata of spring bean cfg file (setter Injection)
(target obj, dependent obj)

e) Spring Container /IOC container keeps spring bean calss objects in the Internal cache of IOC container for reusability having bean ids as keys and spring bean class obj references as the values.



f) Client App gets WishMEssageGeneator class obj ref as java.lang.Object class ref and it uses it for typcasating .

```
//typecasting+
WishMessageGenerator generator=(WishMessageGenerator)obj;
```

g) invocation of b.method getting the result and display the results..

```
//Invoke the b.method
String result=generator.generateMessage("raja");
System.out.println("Wish Message is ::"+result);
```

h) End of the main(-) ..ends the app's eexeuction by vanising the objects .. In that process the IOC contianer(factory) and its spring bean objets (wmg,dt) , its InMemory MetaData and , Xml praser , Intenal cache and etc.. will also be vanished.

Sep 17 constructor Injection

constructor Injection

=>If the spring container/IOC container is using paramterized constructor to create spring bean class obj and also inject dependents then it called constructor injection.
=>For this we need to place <constructor-arg> tags under <bean> tag where we cfg java class as the spring bean.
=> if we place "n" <constructor-arg> tags under <bean> tag then then the spring container looks for n-param consturctor for consturctor injection ,For example if we place <constructor-arg> tags for 3 times then it looks for 3 param consturctor for spring Bean class instantation and injection.

=> spring container performs setter injection after creating spring bean class object , it seperately calls setter methhod for setter injection

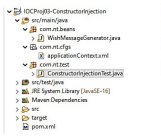
injection

=> spring container performs constructor while instantiating the spring bean class using parameterized constructor.

on

Example App constructor Injection

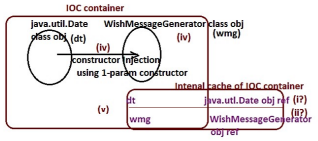
To disable block commenting in eclipse
window menu-->preferences --> java...code style..formatter..new....give name --> disable block commenting.




```
//WishMessageGenerator.java
package com.nt.beans;

import java.util.Date;

public class WishMessageGenerator {
      //HAS-A property (supporting composition)
      private Date date;


      // for constructor injection
      public WishMessageGenerator(Date date) {
            System.out.println("WishMessageGenerator:1-param constructor");
            this.date = date;
      }


      //B.method
      public String generateMessage(String user) {
            System.out.println("WishMessageGenerator.generateMessage()");
            //get current hour of the day
            int hour=date.getHours(); // 24 hours format (0 to 23)
            //generate wish messsage
             if(hour<12)
                  return "Good Morning::"+user;
             else if(hour<16)
                  return "Good AfterNoon::"+user;
             else if(hour<20)
                  return "Good Evening::"+user;
             else
                  return "Good Night::"+user;
      }//method
}//class
```

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="
      http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

      <!-- spring bean cfgs --> (ii?)
      <bean id="dt" class="java.util.Date"/> <!-- dependent class -->
                                  (i?)
      <bean id="wmg"  class="com.nt.beans.WishMessageGenerator"> <!-- target class -->
(i?)    <constructor-arg name="date" ref="dt"/> <!-- constructor injection -->
      </bean>
                                  (ii)

</beans>
```




```
//ConstructorInjectionTest.java
package com.nt.test;

import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.FileSystemResource;

import com.nt.beans.WishMessageGenerator;

public class ConstructorInjectionTest {

      public static void main(String[] args) {
            //Hold spring bean cfg file name and location  (ctrl+shift+o :: To import pkg)
            FileSystemResource res=new FileSystemResource("src/main/java/com/nt/cfgs/applicationContext.xml");
            //create IOC container (BeanFactory container)
            XmlBeanFactory factory=new XmlBeanFactory(res);
            //get Target spring bean class object
(vi)        Object obj=factory.getBean("wmg");  (i)
            //typecasting+
            WishMessageGenerator generator=(WishMessageGenerator)obj;
            //invoke the b.method
            String result=generator.generateMessage("raja");
            System.out.println("Wish Message is ::"+result);

      }//main
}//class
```

when factory.getBean("wmg") is called in Constructor Injection

i) Takes given bean id "wmg" , check in the internal cache of IOC container for Object ref availability .. since not not available it goes to InMemory MetaData of spring bean cfg file (applicationContext.xml) and finds "com.nt.beans.WishMEssageGenerator" class as spring bean and notices constructor injection cfg using <constructor-arg> tag.

note:: In setter Injection model, the IOC container first creates target class obj then creates Dependent class obj later assigns dependent class object to target class object by setter(-) on target class object
In constructor Injection model, the IOC container first creates depedent class obj and use that Dependent class obj as the constructor arg value while creating target class object.

ii) Since the enabled injection is constuctor Injection , it takes ref="dt" from <constructor-arg> tag and searches "dt" bean id spring bean class obj ref in the Internal Cache. since not available the IOC container searches in the InMemory MetaData of applicationContext.xml for spring bean cfg whose bean id "dt" and finds "java.util.Date" class with out having any injections cfg.

iii) IOC container loads "java.util.Date" class and creates the object for it using 0-param constructor and reflection API.

```
//Load the spring bean class
Class c1=Class.forName("java.util.Date");
//get all the constructors the loaded spring bean class
Constructor cons[]=c1.getDeclaredConstructors();
//Dynamic object creation
   Date dt=(Date)cons[0].newInstance();
```

iv) Completes constructor injection on target class by loading and instantiating target class (WishMessageGenerator) using 1-param constructor through reflection API as shown below

```
//Load class
Class c2=Class.forName("com.nt.beans.WishMessageGenerator");
//get All constructors of the loaded class
Constructor cons[]=c2.getDeclaredConstructors();
//Dynamic object creation.
  WishMessageGenerator generator=(WishMessageGenerator)cons[0].newInstance(dt);
                                                              Object creation and injection using 1-param
                                                              constructor of WishMEssageGenerator class
```

v) IOC container keeps both target and dependent class class obj in the Internal cache of IOC container taking 'bean ids as the keys and bean class obj refs as values..

vi) factory.getBean("wmg") returns "WishMessageGenerator" class obj ref back to Client App as java.lang.Object class ref.

---

if we perform both setter injection and construction injection cfgs on the same bean property of target class with different dependent values .. can u tell which will be taken as the final value?

Ans) since the setter method executes after constructor execution .. so the value injected by constructor Injection will be overridden with the value injected by the setter injection.

sample code
==========

in applicationContext.xml

```
<bean id="dt1"  class="java.util.Date">
      <property name="year" value="110"/> <!-- add 1900 to given year so it becoems 2010 -->
      <property name="month" value="4"/> <!-- 0 to 11-->
      <property name="date" value="20"/> <!-- 1 to 31-->
</bean>

<!-- spring bean cfgs -->
<bean id="dt"  class="java.util.Date"/> <!-- dependent class -->

<bean id="wmg"  class="com.nt.beans.WishMessageGenerator"> <!-- target class -->
   <constructor-arg name="date" ref="dt"/> <!-- constructor injection -->
   <property name="date" ref="dt1"/>  <!--setter injection -->
</bean>
```

note:: To inject one spring bean class object to bean property of another spring bean class use "ref" attribute specifying other spring bean id

```
<property name="date" ref="dt1"/>
```

To inject simple or string value to bean property of spring bean class use "value" attribute specifying hardcoded value

```
<property name="date" value="20"/>
```

=> we can cfg one java class as multiple spring beans with diffent bean ids (nothing diffent object names will be created)

```
<bean id="dt1"  class="java.util.Date">
      <property name="year" value="110"/> <!-- add 1900 to given year so it becoems 2010 -->
      <property name="month" value="4"/> <!-- 0 to 11-->
      <property name="date" value="20"/> <!-- 1 to 31-->
</bean>

<!-- spring bean cfgs -->
<bean id="dt"  class="java.util.Date"/> <!-- dependent class -->
```

=> IOC container identifies spring beans using its bean ids (object names/ reference variable names pointing spring bean class objects).So they must be unique with in a spring container.

=>The "name" attribute in <property> tag does not actually looks to take spring bean class property name.. It is actually the "xxx" word of setXxx(-) method that is taken for the setter Injection. Generally the xxx word setXxx(-) method matches with bean property name .. if not matched we need give "xxx" word in the name attribute of <property> tag.

Improving the Client App

spring bean class

```
public class WishMessageGenerator {
    //HAS-A property (supporting composition)
    private Date date;

    // for constructor injection
    public WishMessageGenerator(Date dat) {
        System.out.println("WishMessageGenerator:1-param constructor");
        this.date = dat;
    }

    public void setDate1(Date date) {
        System.out.println("WishMessageGenerator.setDate(-)");
        this.date = date;
    }
}
```

In applicationContext.xml

```
<!-- spring bean cfgs -->
<bean id="dt" class="java.util.Date"/> <!-- dependent class -->

<bean id="wmg" class="com.nt.beans.WishMessageGenerator"> <!-- target class -->
    <property name="date" ref="dt"/> wrong
    <property name="date1" ref="dt "/> valid

</bean>
```

=> In the "name" attribute of <constructor-arg> we should give constructor param name .. not the spring bean class property nname.. generlly both names match.. if not match we should give param name not bean property name.

```
//spring bean calss
public class WishMessageGenerator {
    //HAS-A property (supporting composition)
    private Date date;

    // for constructor injection
    public WishMessageGenerator(Date dat) {
        System.out.println("WishMessageGenerator:1-param constructor");
        this.date = dat;
    }

    .....
}
```

applicationContext.xml

```
<!-- spring bean cfgs -->
<bean id="dt" class="java.util.Date"/> <!-- dependent class -->

<bean id="wmg" class="com.nt.beans.WishMessageGenerator"> <!-- target class -->
  <constructor-arg name="date" ref="dt"/> //invalid
  <constructor-arg name="date1" ref="dt"/> <!-- constructor injection -->
</bean>
```

---

important statatents on spring bean instantiation done spring container

(a) if spring bean class is cfg with out any injections cfg then the IOC container creates that spring bean class object using 0-param constructor

(b) if spring bean class is cfg only having setter injections cfgs, then the IOC container creates that spring bean class object using 0-param constructor

(c) if spring bean class is cfg having constructor injection cfgs, then the IOC container creates that spring bean class object using parameterized constructor.

Q) Can I place only private constructor in spring bean class?

Ans) yes , we can place .. becoz spring container internally uses reflection api to get access to private constructor to perform spring bean instantiation and the necessary constructor injections. (Not a good pratice to keep private constructor in spring bean classes)

note:: we can not private setter methods for setter Iniection

Core Java sample Application to create the objects by accessing private constructor of java class outside of that java class

```
//Test.java
==========
/Test.java
package com.nt.comp;

public class Test {
    private int a=10;
    private String b="hello";


    private Test() {
        System.out.println("Test:: 0-param constructor");
    }

    private Test(int a,String b) {
        System.out.println("Test:: 2-param constructor");
        this.a=a;
        this.b=b;
    }


    //alt+shift+s , s
    @Override
    public String toString() {
        return "Test [a=" + a + ", b=" + b + "]";
    }
}
```

```
//NewInstanceMethodTest1 .java
package com.nt.test;

import java.lang.reflect.Constructor;

public class NewInstanceMethodTest1 {

    public static void main(String[] args) throws Exception{
        //Load classes
        Class c1=Class.forName(args[0]);
        //get All constructor of the Loaded class
        Constructor cons[]=c1.getDeclaredConstructors();
        //Dynamic object using 0-param constructor
        cons[0].setAccessible(true); //gives access to private constructor
        Object obj1=cons[0].newInstance();
        System.out.println(obj1);
        System.out.println(".....");
        //Dynamic object using 2-param constructor
        cons[1].setAccessible(true); //gives access to private constructor
        Object obj2=cons[1].newInstance(100,"india");
        System.out.println(obj2);
    }

}
```

---

we can use either FileSystemResource or ClassPathResource obj to locate and hold spring bean cfg file

What is the difference b/w both?

Ans) FileSystemResource obj makes the spring container to locate the given spring bean cfg file from the specified path of file system (all drives of computer managed by OS)..we can pass either relative path(good) or absolute path of spring bean cfg file

eg:: FileSystemResource res=new FileSystemResource("src/main/java/com/nt/cfgs/applicationContext.xml");
relative path w.r.t Project folder

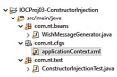
(or)

eg: FileSystemResource res=new FileSystemResource("G:\\Worskpaces\\Spring\\NTSPBMS615\\IOCProj03-ConstructorInjection\\src\\main\\java\\com\\nt\\cfgs\\applicationContext.xml");
absolute path (not a good pratice)

ClassPathResource obj makes the spring container to locate the given spring bean cfg file from the directories and jar files added to CLASSPATH or BuidPATH.

=>In MAVEN/Gradle Project "src/main/java" and "src/test/java" folders will be there in the build path /classpath by default.

eg:: ClassPathResource res=new ClassPathResource("com/nt/cfgs/applicationContext.xml");



(or)

eg: ClassPathResource res=new ClassPathResource("applicationContext.xml");

For this we need to make "com/nt/cfgs" folder as source folder to keep that folder in the buildpath of project.

right click on "com/nt/cfgs" folder ->builpath ->use as the source folder

after moving to classpath

=>In a running java app to hold class name , interface name, annotation name, enum name and their details we can use the object of java.lang.Class (c).

=> To hold numeric values in java app, we use numeric data type variables
=> To hold text values in java app, we use String class objects
=> To hold class name or interface name or annotation name or enum name and their details (metadata) we can take the support of java.lang.Class object. This object does not hold names as Stirng values..it holds them with out loosing the java meanings..

=>The easiest way to create the object of java.lang.Class holding any class/interface/enum/annotation name and details is ".class" property.

```
Class c1= java.util.Date.class;
Class c2=Demo.class;
Class c3=Test.class;
```

Built-in property of java class

object of java.lang.Class

Test class name and its details

c3

```
STring s1="Test";   //Here "Test" text content , so we can only String manipulations
Class c3=Test.class; // Here "Test" treated as java class ..So using c3 we can get all the details
                     "Test" class like super class name, constructor details, method details an etc.. and we use them
                     for different operations/invocations.
```

=>During the compilation of every java class ,the java compiler adds the default property called "class" as the public static property of type java.lang.Class ..

```
Test.java
=========
class Test{
}
```

javac Test.java

Compiler improvised code in Test class during the compilation

```
class Test extends java.lang.Object
  public static Class class;
  public Test(){
    super()
  }
```

gives Test.class file (physical file)

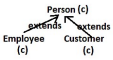
```
Class c1=Test.class;
    Class c2=java.util.Date.class;
    System.out.println("c1 obj data::"+c1);  //gives Test
    System.out.println("c2 obj data::"+c2);  //gives java.util.Date
```

=>Built-in properties in Java class are :: .class , .length
=>Built-in reference variables in Java App are :: this, super
=>Built-in Threads in Java App are :: main, gc
=>Built-in streams in Java App are :: System.in, System.out, System.err

Need of Generics
==========

Problem::
=========

Sep 21 Client App by class path



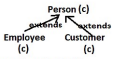

```
public   Person   getDetails(String type){
   if(type.equals("cust"))
      return new Customer();
   else if(type.equals("emp"))
      return new Employee();
   else
     return throw new IllegalArgumentException("Invalid PersonType");
  }
```

Testing code
==========

```
  Person per=getDetails("cust"); //valid
    //using "per" we can call only common of Person and class Customer class.. To call
   //  Direct methods of Customer class we need type casting..
     Customer cust=(Customer)per;
   //usign cust we can directly methods of "Customer"  class
        ....

     Person per=getDetails("emp"); //valid
      //using "per" we can call only common of Person and class Customer class.. To call
     //  Direct methods of Employee class we need type casting..
      Employee emp=(Employee)per;
     //usign emp we can directly methods of "Employee"  class
        ....
```

note:: when we do type casting there is possiblity of getting Type casting related exception that is ClassCast Exception. Code is not type safe ... To make code as type safe code take the support of Generics.

Solution using Genrics
==========




```
public  <T>  T  getDetails(String type, Class<T> clazz){
     if(type.equals("cust"))
       return   clazz.newInstance();
     else if(type.equals("emp"))
       return   clazz.newInstance();
     else
      return throw new IllegalArgumentException("Invalid PersonType");
   }
```

Here can any class bsed java.lang.Class obj

Testing code
==========

```
  Employee e=getDetails("emp",Employee.class);
```

Gives object of java.lang.Class having Employee class details as data

[Here no typecasting is required becoz of generics.. So we can say ClassCastException is avoided and we can say code is type safe code]

```
  Customer  cust=getDetails("cust",Customer.class);
```

More improved Code
==========

```
   public  <T extends Person>  T  getDetails(String type, Class<T> clazz){
        if(type.equals("cust"))
          return   clazz.newInstance();
        else if(type.equals("emp"))
          return   clazz.newInstance();
        else
         return throw new IllegalArgumentException("Invalid PersonType");

    }

  Employee e=getDetails("emp",Employee.class);
```

must be Person and its sub class based java.lang.Class obj

Gives object of java.lang.Class having Employee class details as data

[Here no typecasting is required becoz of generics.. So we can say ClassCastException is avoided and we can say code is type safe code]

```
  Customer  cust=getDetails("cust",Customer.class);
```

Old getBean(-) of BeanFactory (spring 1.0 onwords)

signature

```
public  Object getBean(String beanid)  throws BeansException
```

```
  eg::
    Object obj=factory.getBean("wmg");
     //using obj we can not call direct methods of "WishMessage" Generator class
     //for Go for Type casting
     WishMessageGenerator generator=(WishMessageGenerator)obj;
     //using "generator" we can call direct method WishMessageGenerator class.
       ...
       ..
```

note: Type casting required, So there is possbility of getting ClassCastExceptionException i.e code is not type safe.

Improvised getBean(-,-) method of BeanFactory (Using Generics from spring 2.5)
==========

Signature ::

```
  public  <T>  T  getBean(String bean id , Class<T> clazz)throws BeansException

  eg::
    WishMEssageGenerator generator=factory.getBean("wmg",WishMessageGenerator.class);
       //No Type casting required , So code becomes type safe code..
  eg::
    Date  d=factory.getBean("dt",Date.class);
       //No Type casting required , So code becomes type safe code..
```

passing WishMessageGenerator class as the the object of java.lang.Class

Improved factory.getBean(-,-) in the Client App
==========

```
package com.nt.test;

import java.util.Date;

import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.ClassPathResource;

import com.nt.beans.WishMessageGenerator;

public class ConstructorInjectionTest1 {

      public static void main(String[] args) {
              ClassPathResource res=new ClassPathResource("applicationContext.xml");
            //create IOC container (BeanFactory container)
            XmlBeanFactory factory=new XmlBeanFactory(res);
            //get Target spring bean class object
            WishMessageGenerator  generator=factory.getBean("wmg",WishMessageGenerator.class);
            //invoke the b.method
            String result=generator.generateMessage("raja");
            System.out.println("Wish Message is ::"+result);

      }//main
}//class
```

STS (Spring ToolSuite)
===

```
   =>STS is an IDE which exclusively given for  spring/spring Boot apps development.
   =>This IDE is based  Eclipse IDE
   => Instead of STS IDE .. we add  STS plugin to Eclipse ,So we can use
   Eclipse features and STS IDE features..
```

=>Plugin
==========

```
   => A plugin is patch s/w  that provides addtional features on the top of existing features.. in the
  software or  sofware  App.

            Eclipse having two type plugins
              a) Eclipse supplied plugins
                    => Can be added using  help menu => Install new software option
                    eg:: Gui builder, GlassFish Tools,  Jboss tools and etc..
              b) Third party plugins
                   =>Can be added using help menu=>Eclipse market place
                    eg::  STS plugin, sonar cube , jasper soft and etc..

   note:  From eclipse 2019  Gradle, maven , GIT, SVN and etc.. plugins are built-in plugins  in basic
   Eclipse Installation..

  STS plugin features
  ==========
  a) Allows to crete spring bean cfg file  by importing selected XSD names spaces
  b) helps to finish  spring cfg file develpment very fastly
  c) simplifies  spring boot App development by lots of starters..
  d) allows to create spring project and spring boot projects easilty..
  and etc..

  Process to STS Plugin
  ==========
     Help Menu ===>  Eclipse Market place ==> search STS --> go  --->
       selecct 3.9.18  version -->install --->confirm ---> next ----> accept terms conditions --> restart
     eclipse IDE
```

Limitations of creating BeanFactory IOC container using XmlBeanFactory class

Sep 22 DefaultListable BeanFactory & DesignPatterns

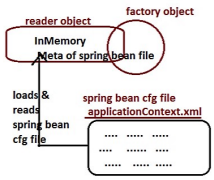
a)XmlBeanFactory class is Deprecated from the version spring 3.1
b)XmlBeanFactory does not allow to working with multiple spring bean cfg files at a time
c) We can not pass spring bean cfg file name and location as string values.. it is expecting them as Resource object (FileSystemResoruce or ClassPathResource obj)
d) The spring Beans given to XmlBeanFactory will be passed to DefaultListableBeanFactory class for registration .. So better to use DefaultListableBeanFactory as directly

```
java.lang.Object
    org.springframework.core.SimpleAliasRegistry
        org.springframework.beans.factory.support.DefaultSingletonBeanRegistry
            org.springframework.beans.factory.support.FactoryBeanRegistrySupport
                org.springframework.beans.factory.support.AbstractBeanFactory
                    org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory
                        org.springframework.beans.factory.support.DefaultListableBeanFactory
                            org.springframework.beans.factory.xml.XmlBeanFactory
```

To overcome the above problems .. prefer using DefaultLisableBeanFactory and XmlBeanDefinitationReader togather to create BeanFactory IOC container.

Code to create BeanFactory Container using DefaultListableBeanFactory and XmlBeanDefinitionReader

```
//create IOC container
DefaultListableBeanFactory factory=new DefaultListableBeanFactory();
XmlBeanDefinitionReader reader=new XmlBeanDefinitionReader(factory);
reader.loadBeanDefinitions("com/nt/cfgs1/applicationContext.xml");
```



Advantages of working DefaultListableBeanFactory

a) Not a Deprecated java class
b) Can take multiple xml files as the spring bean cfg files at a time (var args)
c)we can spring cfg file names and locations as string values. (No need of Resource objs)
d) The java classes become spring beans directly with out passing them to other classes..

DesignPatterns

=> These are set of rules that are given best solutions for reoccuring problems of application/project development.

=>Design patterns helps the developers to use programming languages, technolgoies and and frameworks more effectively in the Application development ..

=> Design Patterns are not part of Project designing .. they are part of Project Implementation (coding)

=>Desing Patterns acts best solutions for solving memory , perfomences , tight coupling related problems..

=> Every Object Oriented Programming language gives support to implement Design Patterns.

Different Types of Desing Patterns

a) GOF Pattterns (total 23 patterns)
=>Can be implemented in any oop language including java
=>Generally these are desing patterns related to standalone Apps devleopment

eg:: singleton class, factory pattern, abstract factory pattern, bridge pattern, strategy pattern and etc..

b) JEE Patterns (25 patterns)
=> Given by Sun Ms
=> useful to address the problems related to JAva Layred Application Development
=>Can be implemented only JAva ,JEE,java frameworks env..

eg:: DAO class , FrontController , ViewHelper ,Composite View and etc..

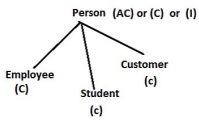
c) Micro services Patterns

=> Can be used while developing applications having micro services architectures
=> addresses problems related to MicroService architecture based application development.
eg: SAGA Pattern, SOLID principles pattern and etc..

Factory Pattern

=> Factory Pattern creates and returns one of several related related classes obj based on the data that is supplied by providing abstraction (hiding the details) on object creation process.

=> related classes means , the classes having common super class or commonly implementing interface.



if Factory pattern is not given .. The Client App fellow should know object creation process entirely.. sometimes it may be more complex becoz one object creation may need multiple other objects as dependents.

eg1:: Car Factory provides abstaction on Car Manifactring
eg2:: DriverManager.getConnection(-,-,-) provides abstraction on Jdbc con object creation based on arg values (jdbcurl, db user, dbpwd) details we supply)

```
Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
                                           "system","manager");
```
gives oracle Db s/w connectivity related JDBC con obj

(or)

```
Connection con=DriverManager.getConnection("jdbc:mysql:///<logicalDB>","root","root");
```
gives mysql Db s/w connectivity related JDBC con obj

note:: All Jdbc statements that are given to create Statement obj,ResusltSet obj and etc.. are also based Factory Pattern.

```
Statement st=con.createStatement();
ResultSet rs=st.executeQuery("SELECT * FROM STUDENT");
```

eg3: spring containers (both BF ::BeanFactory and AC: ApplicationContext) are given based on factory Pattern.

```
Date dt=factory.getBean("dt",Date.class);

WishMessageGenerator generator=factory.getBean("wmg",WishMEssageGenerator.class);
```

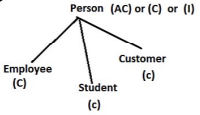
=>Factory pattern implementaton needs factory method .. The method that can returns either its own class object or related class obj or irrelavent class obj called factory method. The factory method used in Factory Pattern generally returns one of several related classes object based on the data /inputs the that are supplied.

The factory method of Factory pattern can static or non-static .. generally we take it is as static method having the common super class or common interface as the return type

```
public class PersonFactory{
    //Factory method having Factory pattern logic
    public static Person getPerson(String type){
        if(type.equalsIgnoreCase("emp"){
            return new Employee();
        else if(type.equalsIgnoreCase("cust"){
            return new Customer();
        else if(type.equalsIgnoreCase("stud"){
            return new Student();
        else
            throw new IllegalArguementExcpetion("invalid person type");
        }
    }
```

Common class or interface as the return type

Clients call method with out object



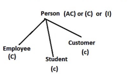
note:: Always assume Factory class and and ClientApps that usisng factory calsses are develped two different programmers..

Sep 23 Factory Pattern

Factory Pattern
================

=> Factory Pattern creates and returns one of several related related classes obj based on the data that is supplied by providing abstraction (hiding the details) on object creation process.

=> related classes means , the classes having common super class or commonly implementing interface.



if Factory pattern is not given .. The Client App fellow should know object creation process entirely.. sometimes it may be more complex becoz one object creation may need multiple other objects as dependents.

eg1:: Car Factory provides abstaction on Car Manifactring
eg2:: DriverManager.getConnection(-,-,-) provides abstraction on Jdbc con object creation based on arg values (jdbcurl, db user, dbpwd) details we supply)

```
Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
                                            "system","manager");
```

gives oracle Db s/w connectivity related JDBC con obj

(or)

```
Connection con=DriverManager.getConnection("jdbc:mysql:///<logicalDB>","root","root");
```

gives mysql Db s/w connectivity related JDBC con obj

note:: All Jdbc statements that are given to create Statement obj,ResusltSet obj and etc.. are also based Factory Pattern.

```
Statement st=con.createStatement();
ResultSet rs=st.executeQuery("SELECT * FROM  STUDENT");
```

eg3: spring containers (both BF ::BeanFactory and AC: ApplicationContext) are given based on factory Pattern.

```
Date dt=factory.getBean("dt",Date.class);

WishMessageGenerator generator=factory.getBean("wmg",WishMEssageGenerator.class);
```

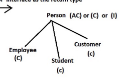
=>Factory pattern implementaton needs factory method .. The method that can returns either its own class object or related class obj or irrelavent class obj called factory method. The factory method used in Factory Pattern generally returns one of several related classes object based on the data /inputs the that are supplied.

The factory method of Factory pattern can static or non-static .. generally we take it is as static method having the common super class or common interface as the return type

```
public class PersonFactory{
   //Factory method having Factory pattern logic
  public  static Person  getPerson(String type){
     if(type.equalsIgnoreCase("emp"){
       return new Employee();
     else if(type.equalsIgnoreCase("cust"){
       return new Customer();
     else if(type.equalsIgnoreCase("stud"){
       return new Student();
     else
       throw  new IllegalArguementExcpetion("invalid person type");
  }
}
```

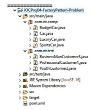
Clients call method with out object (static)

Common class or interface as the return type (Person)



note:: Always assume Factory class and and ClientApps that usisng factory calsses are develped two different programmers..

Problem code
==============



Car.java

```
package com.nt.comp;

public abstract class Car {
   public abstract  void drive();
}
```

```
//BudgetCar.java
package com.nt.comp;

public class BudgetCar extends Car {
  private String regNo;
     public BudgetCar(String regNo) {
          System.out.println("BudgetCar::1-param constructor");
          this.regNo=regNo;
     }

     @Override
     public void drive() {
       System.out.println("Driving Budget Car");

     }
```

```
//LuxoryCar.java
package com.nt.comp;

public class LuxoryCar extends Car {

     private String regNo;
     public LuxoryCar(String regNo) {
          System.out.println("LuxoryCar::1-param constructor");
          this.regNo=regNo;
     }

     @Override
     public void drive() {
       System.out.println("Driving Luxory Car");

     }

}
```

```
////SportsCar.java.java
package com.nt.comp;

public class SportsCar extends Car {
     private String regNo;

     public SportsCar(String regNo) {
          System.out.println("SportsCar::1-param constructor");
          this.regNo=regNo;
     }

     @Override
     public void drive() {
       System.out.println("Driving Sports Car");

     }

}
```

Client Apps

```
//BusinessManCustomer3 .java
package com.nt.test;

import com.nt.comp.BudgetCar;
import com.nt.comp.Car;
import com.nt.comp.LuxoryCar;
import com.nt.comp.SportsCar;

public class BusinessManCustomer3 {
     public static void main(String[] args) {
          Car car=new LuxoryCar("TS11 AB 5151");
          car.drive();

     }

}
```

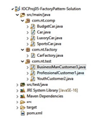
```
//ProfessionalCustomer1.java
package com.nt.test;

import com.nt.comp.BudgetCar;
import com.nt.comp.Car;

public class ProfessionalCustomer1 {
     public static void main(String[] args) {
          Car car=new BudgetCar("TS09EN 5656");
          car.drive();

     }

}
```

```
//YouthCustomer2.java
package com.nt.test;

import com.nt.comp.BudgetCar;
import com.nt.comp.Car;
import com.nt.comp.SportsCar;

public class YouthCustomer2 {
     public static void main(String[] args) {
          Car car=new SportsCar("TS10 KK 5656");
          car.drive();

     }

}
```

HEre the multiple client Apps should know following things
a) How to create obejcts for different car classes
b) if need they should know how to create the required depeddent class objs
c) They should good command on the hierarchy of classes

=>To this solve this problem for clients provide then one factory class having factory pattern which takes care of object creation process.. by hiding details from clients..

Solution using Factory Pattern
==========================




```
//CarFactory.java
package com.nt.factory;

import com.nt.comp.BudgetCar;
import com.nt.comp.Car;
import com.nt.comp.LuxoryCar;
import com.nt.comp.SportsCar;

public class CarFactory {

     //Factory having  factory pattern logic  (Logic to create and return one of several related classes obj)
     public static  Car  createCar(String type, String regNo) {
          if(type.equalsIgnoreCase("sports"))
               return new SportsCar(regNo);
          else if(type.equalsIgnoreCase("budget"))
               return new BudgetCar(regNo);
          else if(type.equalsIgnoreCase("luxory"))
               return new LuxoryCar(regNo);
          else
               throw new IllegalArgumentException("Invalid car Type");
     }

}
```

```
//YouthCustomer2.java
package com.nt.test;

import com.nt.comp.BudgetCar;
import com.nt.comp.Car;
import com.nt.comp.SportsCar;
import com.nt.factory.CarFactory;

public class YouthCustomer2 {
     public static void main(String[] args) {
          //get Car obj using factory
          Car car=CarFactory.createCar("sports", "TS08 EN 6688");
          car.drive();
          System.out.println("================");

     }

}
```

```
//ProfessionalCustomer1.java
package com.nt.test;

import com.nt.comp.BudgetCar;
import com.nt.comp.Car;
import com.nt.factory.CarFactory;

public class ProfessionalCustomer1 {
     public static void main(String[] args) {
          //get Car obj using factory
               Car car=CarFactory.createCar("budget", "TS09 EN 5688");
               car.drive();
               System.out.println("================");
     }

}
```

```
//BusinessManCustomer3 .java
package com.nt.test;

import com.nt.comp.BudgetCar;
import com.nt.comp.Car;
import com.nt.comp.LuxoryCar;
import com.nt.comp.SportsCar;
import com.nt.factory.CarFactory;

public class BusinessManCustomer3 {
     public static void main(String[] args) {
          //get Car obj using factory
          Car car=CarFactory.createCar("luxory",
"TS10 EN 5688");
          car.drive();
          System.out.println("================");
     }

}
```

Strategy Pattern
================

=>Stategy means planing

=> It is a GOF(Gang of Four) Pattern and can be implemented in any oop language
=> It has become very popular after the arrival spring framework .. but it is not spring framework pattern.
=> This patterns given set of rules/principles to designing classes having dependency as target and dependent classes.
=> This design pattern says develop the target and dependent classes of Dependency Management as loosely copuled interchangable parts.
(i.e we can change one dependent class to another dependent class with out effecting the the source code target class)

=>if the degree of dependency is more b/w two comps/classes then they are called tightly coupled classes/comps
eg:: CPU Box and Console/Display Unit
eg:: FAn and switch

=>if the degree of dependency is less b/w two comps/classes then they are called loosely coupled classes/comps
eg:: TV and Remote
eg: Laptop and Mouse

The 3 principles /rules of strategy Pattern
====================================

a) Favour Composition over inheritence
b) Code to interfaces .. Never code to Concrete classes /abstract classes
c) Code must be open for extension and must be closed for modification.

a) Favour Composition over inheritence

Inheritence (IS-A relation)

```
class  A{          class  B extends A{
  ....               ....
  ..                 ...
}                  }
```

B is sub class of A class

Composition (HAS- Relation)

```
class A{           class  B{
  ...                private A a=new A();
  ...}                ...
                   }
```

B has got A class object.

=> if class1 wants to use entire state and behaviour of class2 then keep them in inheritence relationship.
=> if class1 wants to use specific state and behaviour of class2 then keep them in composition relationship.

Limitations of Inheritence (while using it in deepndency management)
======================

(i) Few languages do not support inheriting from multiple classes (including java)
(ii) Code becomes fragile (Easy breakable)
(iii) Testing of code becomes very complex.

(i) Few languages do not support inheriting from multiple classes (including java)

problem ::

```
class A{     class B{          class C extneds  A,B{
  ...          ....              ....
}            }                   ...
                               }
```

Not allowed in most of the OOP languages.

=>Do not look at interfaces in the angle of inheritence..becoz they does not contain any thing exclusively to inherit.. Always look at them in the angel of polymorphism.

Solution::

```
class A{     class B{        class   C{
  ...          ....            private A a=new A();
}            }                 private B b=new B();

                             }
```

Can work with multiple other classes objects using composition.

Strategy Pattern
================

Sep 24 Stategy Pattern

=>Stategy means planing

=> It is a GOF(Gang of Four) Pattern and can be implemented in any oop language
=> It has become very popular after the arrival spring framework .. but it is not spring framework pattern.
=> This patterns given set of rules/principles to designing classes having dependency as target and dependent classes.
=> This design pattern says develop the target and dependent classes of Dependency Management as loosely copuled interchangable parts.
(i.e we can change one dependent class to another dependent class with out effecting the the source code target class)

=>if the degree of dependency is more b/w two comps/classes then they are called tightly coupled classes/comps
eg:: CPU Box and Console/Display Unit
eg:: FAn and switch

=>if the degree of dependency is less b/w two comps/classes then they are called loosely coupled classes/comps
eg:: TV and Remote
eg: Laptop and Mouse

The 3 principles /rules of strategy Pattern
=====================================

a) Favour Composition over inheritence
b) Code to interfaces .. Never code to Concrete classes /abstract classes
c) Code must be open for extension and must be closed for modification.

a) Favour Composition over inheritence (strategy Pattern rule no:1)

Inheritence (IS-A relation)

```
class A{          class B extends A{
 ....               ....
  ..                ...
}                  }
```

B is sub class of A class

Composition (HAS- Relation)

```
class A{          class B{
  ...               private A a=new A();
  ...}              ...
                  }
```

B has got A class object.

=> if class1 wants to use entire state and behaviour of class2 then keep them in inheritence relationship.
=> if class1 wants to use specific state and behaviour of class2 then keep them in composition relationship.

Limitations of Inheritence (while using it in deepndency management)
======================
(i) Few languages do not support inheriting from multiple classes (including java)
(ii) Code becomes fragile (Easy breakable)
(iii) Testing of code becomes very complex.

(i) Few languages do not support inheriting from multiple classes (including java)

problem ::

```
class A{     class B{        class C extneds A,B{
  ...          ....            ....
}            }                 ...
                             }
```

Not allowed in most of the OOP languages.

=>Do not look at interfaces in the angle of inheritence..becoz they does not contain any thing exclusively to inherit.. Always look at them in the angel of polymorphism.

Solution::

```
class A{     class B{       class C{
  ...          ....           private A a=new A();
}            }                private B b=new B();

                            }
```

Can work with multiple other classes objects using composition.

(ii) Code becomes fragile (Easy breakable)

```
class A{
        float
 public ~~int~~ m1(){
   ....
   return 100;
 }
}

class B extends A{
   public int m1(){
    ...
    return 1000;
   }
}

class C extends B{
   ....
   ....
}

class D extends B{
   ....
   ..
}
```

problem :: if we modify the return type of m1() in "A" class from int to float the all classes that inheriting from "A" class directly or indirectly will be distrubed..and we can get lots of inconsistency problem.

note:: if the signature of java.lang.Object class methods are modified bit .. then all all java apis will be collapsed ..becoz java.lang.Object is the top most class in the inheritence hierarchy of every java class.

Solution:: Use the composition , then we can restrict the problem to certain 1 or 2 methods.

```
class A{
        float
public ~~int~~ m1(){
  ..
  return 100;
 }
}

class B{
 private A a=new A();

 public int m1(){
  ~~int x=a.m1();~~
 int   x=Math.round(a.m1());
        (or)
  var x=a.m1();  //from java 10
  ...
  return x+1000;
 }

class C extends B{
  ....
  ...
}
class D extends B{
  ....
  ....
}
```

=>Here B class has the object of A class (composition and no inheritence b/w A and B classes), So if we change the return type of m1() in A class from int to float .. we just need to modify the code in B class or other classes when that m1() of A class is being invoked i.e we can restrict the code change problem propagation to certain method call area .. All classes in the hierarchy will not be distrubed.

=> declarting local variable having type "var" in given from Java 10 ..
In Java 10, the var keyword allows local variable type inference, which means the type for the local variable will be inferred by the compiler based on the value that is assigned.

(iii) Testing of code becomes very complex.

=>Programmer's testing on his own piece of code is called unitTesting.
=> We can use Junit tool to perform UnitTesting through coding

Problem::

```
class A{
 public int m1(){
   ...
   return 100;
  }
 public int m2(){
  ...
  return 200;
  }
}

class B extends A{
  public int m3(){
   ...
   return 300;
  }
  public float m4(){
   ....
   return 45.77;
  }
}

class C extends B{
  ...
  ... //more 3 methods
  ...
}

class D extends C{
  ...
  ... //more 3 methods
  ...
}

B b=new B();
b.m1(); b.m2(); b.m3(), b.m4();

D d=new D();
 //we can invoke total 10 methods on this object.
```

UnitTesting deals with Testcases where expected results will be compared with generated Results.

While performing unit Testing of any sub class .. we need to write test cases for the direct methods of the sub class and also for the inherited method to sub class .. This improves burden on the programmer.

Solution
=======

```
class A{
  public int m1(){
    ...
    return 100;
   }
  public int m2(){
   ...
   return 200;
   }
 }

class B{
 private A a=new A();

 public int m3(){
   int x=a.m1();
   return x+200;
  }

 public float m4(){
  return 200.6f;
  }
}

class C {
  ...
  ... //more 3 methods
  ...
}

class D        {
  ...
  ... //more 3 methods
  ...
}

B b=new B();
 b.m3();
 b.m4();
```

D d=new D();
we can only 3 methods on this object of "D" class.

=> Here while doing unit testing on any class.. we just need to write test cases only for that class methods.. So burden on the programmer will be decreased.

b) Code to interfaces .. Never code to Concrete classes /abstract classes (strategy Pattern rule no:2)

problem:: (Tight coupling)
========================

```
//target class
class Flipkat {  //HAS-A Property
 ~~private DTDC dtdc =new DTDC();~~
  private BlueDart bDart =new BlueDart()
public String shopping(
     String item[],float prices[]){
   .....
   ..... //calc b.method
   .....
   ------
   ..... //logic for paymemt
   .....
    String status=bDart.deliver(oid);
   ~~String status=dtdc.deliver(oid);~~

   return status +" "+billAmt;
  }
}

//dependent class
public class DTDC{

 public String deliver(int oid){
     ....
     ...
  }
}

//dependent class
public class BlueDart {

 public String deliver(int oid){
     ....
     ...
  }
}
```

=>To change Depedent "DTDC" to "BlueDart" we need to modify the source code target class every time , So we can say there is tight coupling.

solution :: Loose Coupling through interfaces/abstract classes
========================

=>Make all related Dependent classes implementing the common Interface or extending from Common super class /abstract class
=> Take common Interface or super class or abstract class type HAS-A property in the Target class and use the support setter method or parameter constructor for assinging dependent to target externally.

Soultion code
============

```
     //Common interface
interface Courier{
  public String deliver(int oid);
 }

//Target class
=============
class Flipkart {
 //HAS- A property
 private Courier courier;
      Coding to interfaces
 public void setCourier(Courier courier){
    this.courier=courier;
 }

 public String shopping(
      String item[],float prices[]){
   .....
   ..... //calc b.method
   .....
   ------
   ..... //logic for paymemt
   .....
  String status=courier.deliver(oid);

   return status +" "+billAmt;
 }

//Dependent class1
class DTDC implements Courier{

 public String deliver(int oid){
    .....
    .....
  return "....";
 }
}

//Dependent class2
class BlueDart implements Courier{

 public String deliver(int oid){
    .....
    .....
  return "....";
 }
}
```

Client App
===========

```
Flipkart fpkt=new Flipkart(); //target class obj
Courier courier=new DTDC(); //Dependent class obj
 flpk.setCourier(courier); // assinging dependent class obj
                              to target class obj
 String resultMsg=fpkt.shopping(new Stirng[]{"shirt","trouser"},
                                new float[]{5000.0f,4000.of} );
 S.o.p(resultMsg);
```

Client App
===========

```
  Flipkart fpkt=new Flipkart(); //target class obj
Courier courier=new BlueDart();    //Dependent class obj
  flpk.setCourier(courier); // assinging dependent class obj
                               to target class obj
  String resultMsg=fpkt.shopping(new Stirng[]{"shirt","trouser"},
                                 new float[]{5000.0f,4000.of} );
  S.o.p(resultMsg);
```

In the Client Apps we can from one Dependent courier and another Dependent courier with out changing the code of target class becoz the coding is done to interfaces.. to achieve loosely coupling.

c) Code must be open for extension and must be closed for modification. strategy pattern rule no:3

=>This rule is extension of rule2 .. if we enhance rule number2 , automatically the rule number3 will be implemented..

=> We can take more dependent classes like DHL, FirstFlight and etc.. implementing the common Interface i.e our code is open for Extension..
=> We can take both target and dependent classes as the final classes or their methods as final methods to make their code "Closed for modifition" (Final classes can not have sub classes , Final mehods can not be overriden in sub classes)

Soultion code
============

```
        //Common interface
  interface Courier{
    public String deliver(int oid);
   }

  //Target class
  =============
final  class Flipkart {
  //HAS- A property
  private Courier courier;
       Coding to interfaces
  public void setCourier(Courier courier){
     this.courier=courier;
  }

  public String shopping(
       String item[],float prices[]){
    .....
    ..... //calc b.method
    .....
    ------
    ..... //logic for paymemt
    .....
   String status=courier.deliver(oid);

    return status +" "+billAmt;
  }

    //Dependent class1
final class DTDC implements Courier{

   public String deliver(int oid){
      .....
      .....
    return "....";
   }
  }

final //Dependent class2
  class BlueDart implements Courier{

   public String deliver(int oid){
      .....
      .....
    return "....";
   }
  }

final //Dependent class3
  class DHL implements Courier{

   public String deliver(int oid){
      .....
      .....
    return "....";
   }
  }
```

Client App
===========

```
Flipkart fpkt=new Flipkart(); //target class obj
Courier courier=new DTDC(); //Dependent class obj
 flpk.setCourier(courier); // assinging dependent class obj
                              to target class obj
 String resultMsg=fpkt.shopping(new Stirng[]{"shirt","trouser"},
                                new float[]{5000.0f,4000.of} );
 S.o.p(resultMsg);
```

Client App
===========

```
  Flipkart fpkt=new Flipkart(); //target class obj
Courier courier=new BlueDart();    //Dependent class obj
  flpk.setCourier(courier); // assinging dependent class obj
                               to target class obj
  String resultMsg=fpkt.shopping(new Stirng[]{"shirt","trouser"},
                                 new float[]{5000.0f,4000.of} );
  S.o.p(resultMsg);
```

Sep 25 Stategy Pattern DP

c) Code must be open for extension and must be closed for modification.

strategy pattern rule no:3

=>This rule is extension of rule2 .. if we enhance rule number2 , automatically the rule number3 will be implemented..

=> We can take more dependent classes like DHL, FirstFlight and etc.. implementing the common Interface i.e our code is open for Extension..

=> We can take both target and dependent classes as the final classes or their methods as final methods to make their code "Closed for modifition" (Final classes can not have sub classes , Final mehods can not be overriden in sub classes)

Soultion code

```
//Common interface
interface Courier{
  public String deliver(int oid);
}

//Target class
final class Flipkart {
  //HAS- A property
  private Courier courier;
          Coding to interfaces
  public void setCourier(Courier courier){
    this.courier=courier;
  }

  public String shopping(
        String item[],float prices[]){
    .....
    ..... //calc b.method
    .....
    ------
    ..... //logic for paymemt
    .....
   String status=courier.deliver(oid);

    return status +" "+billAmt;
  }
```

```
//Dependent class1
final class DTDC implements Courier{

  public String deliver(int oid){
    .....
    .....
   return "....";
  }
}

//Dependent class2
final class BlueDart implements Courier{

  public String deliver(int oid){
    .....
    .....
   return "....";
  }
}
```

```
//Dependent class3
final class DHL implements Courier{

  public String deliver(int oid){
    .....
    .....
   return "....";
  }
}
```

Client App

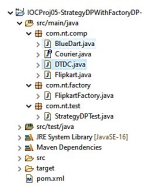
```
Flipkart fpkt=new Flipkart(); //target class obj
Courier courier=new DTDC(); //Dependent class obj
flpk.setCourier(courier);  // assinging dependent class obj
                              to target class obj
String resultMsg=fpkt.shopping(new Stirng[]{"shirt","trouser"},
                               new float[]{5000.0f,4000.of} );
S.o.p(resultMsg);
```

=>To provide abstraction to Client App developers .. It is better to provide Factory to them

Client App|

```
Flipkart fpkt=new Flipkart(); //target class obj
Courier courier=new BlueDart();    //Dependent class obj
flpk.setCourier(courier);  // assinging dependent class obj
                              to target class obj
String resultMsg=fpkt.shopping(new Stirng[]{"shirt","trouser"},
                               new float[]{5000.0f,4000.of} );
S.o.p(resultMsg);
```

example App Strategy DP




```
//Fliplkart.java (target class)
package com.nt.comp;

import java.util.Arrays;
import java.util.Random;

public final class Flipkart {
    //HAS -A  property of type interface
    private Courier courier;

    public Flipkart() {
        System.out.println("Fliplkart:: 0-param constructor");
    }

    //setter method for setter injection
    public void setCourier(Courier courier) {
        System.out.println("Fliplkart.setCourier(-)");
        this.courier = courier;
    }

    public  String shopping(String items[], float prices[]) {
        //caculate billAmt  (b.logic)
        float billAmt=0.0f;
        for(int i=0;i<items.length;++i)
            billAmt=billAmt+prices[i];
        //generate order id
        int oid=new Random().nextInt(100000);
        //use courier for shipping
        String status=courier.deliver(oid);
        String finalMsg=Arrays.toString(items)+"are purchsed with prices"+Arrays.toString(prices)+". The geneated billAmount is::"+billAmt;

        return finalMsg+" :::"+status;
    }


}
```

```
package com.nt.comp;

public interface Courier {
    public String  deliver(int oid);
}
```

```
//BlueDart.java (Dependent class1)
package com.nt.comp;

public final class BlueDart implements Courier {

    public BlueDart() {
        System.out.println("BlueDart:: 0-param consturctor");
    }

    @Override
    public String deliver(int oid) {
      return oid+" order id order is delivered by BlueDart";
    }

}
```

```
//DTDC.java (Dependent class1)
package com.nt.comp;

public final class DTDC implements Courier {

    public DTDC() {
        System.out.println("DTDC:: 0-param consturctor");
    }

    @Override
    public String deliver(int oid) {
      return oid+" order id order  is delivered by DTDC";
    }

}
```

```
//FlipkartFactory.java
package com.nt.factory;

import com.nt.comp.BlueDart;
import com.nt.comp.Courier;
import com.nt.comp.DTDC;
import com.nt.comp.Flipkart;

public class FlipkartFactory {

    //static factory method supporting Factory pattern
    public static  Flipkart  createFlipkart(String courierType) {
        //create target class obj
        Flipkart fpkt=new Flipkart();
        Courier courier=null;
        //create Depennt class obj based on given courier type
        if(courierType.equalsIgnoreCase("dtdc"))
            courier=new DTDC();
        else if(courierType.equalsIgnoreCase("bDart"))
            courier=new BlueDart();
        else
            throw new IllegalArgumentException("Invalid courier type");
        //set  Dependent class object to target class obj
        fpkt.setCourier(courier);
      return fpkt;
  }

}
```

```
//StrategyDPTest.java
package com.nt.test;

import com.nt.comp.Flipkart;
import com.nt.factory.FlipkartFactory;

public class StrategyDPTest {

    public static void main(String[] args) {
        try {
            //get target class object form Factory
            Flipkart fpkt=FlipkartFactory.createFlipkart("bDart");
            // invoke the b.method
            String result=fpkt.shopping(new String[] {"shirt","trouser","mobile"},
                                        new float[] {3400.0f , 5600.0f, 60000.0f});
            System.out.println(result);

        }//try
        catch(Exception e) {
            e.printStackTrace();
        }
    }//main
}//class
```

=>The above code is developed by using strategy DP and Factory DP and getting loose coupling with respect to target and dependent classes i.e we can change from depedent class to another dependent class with out changing the source code of target class, But this not 100% loosely coupled code..

Why the above code is not 100% Loosely coupled?

a) afer adding 1 more courier class by implementing Courier(I) ...we need to modify the source code of Factory class

b) if Client App wants to change from 1 courier to another courier then we need to modify the source code of Cleint App.

Read following topics and come for next class

a) Properties file

b) java.util.Properties

c) IO streams

d) static block

Why the above code is not 100% Loosely coupled?

Sep 27 Stategy Pattern DP Using Spring

a) afer adding 1 more courier class by implementing Courier(I) ...we need to modify the source code of Factory class
b) if Client App wants to change from 1 courier to another courier then we need to modify the source code of Cleint App.

To make initial Strategy DP application as 100%Looselycoupled App , we need to use the proeprties file or xml file

properties file
==========
=>The text file that maintain the entries as key=value pairs is called properties file. It is recomanded to take .properties extension for the properties file.

info.properties (com/nt/commons pkg)

```
# Personal info
per.name=raja
per.addrs=hyd
per.age=40
```
Keys values

=>In keys we can take "."," ","_","-" as the seperators ..not the other characters..
=>Keys must be unique.
=>'#' symbol comments the line

=>java.util.Properties is the Map Collection and the sub class of java.util.Hashtable. we can store the keys and values of properties file into java.util.Properties collection using load(-) of same class.

//sample code
==========
```
//Locate proeprties file
FileInputStream fis=new FileInputStream("src/main/java/com/nt/commons/info.properties");
//Load properties file content into java.util.Properties class object
Properties props=new properites();
 props.load(fis);
```

props(java.util.Properties class obj)

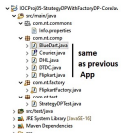
per.name	raja	element1
per.addrs	hyd	element2
per.age	40	element3
keys	values	

```
System.out.println(props);
System.out.println("per.age key value"+props.getProperty("per.age"));
```
gives the value 40

=>java.util.Proepeties Collection elements ,the key and values must be Strings

---

Making Core java based StrategyDP App as loosely coupled App using the support of Properites file
==========



info.properties

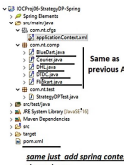
```
#Specify Dependent class
dependent.comp=com.nt.comp.DHL
```

```
//FlipkartFactory.java
package com.nt.factory;

import java.io.FileInputStream;
import java.lang.reflect.Constructor;
import java.util.Properties;

import com.nt.comp.Courier;
import com.nt.comp.Flipkart;

public class FlipkartFactory {
   private static Properties props;
   static {
      System.out.println("FlipkartFactory.static block");
  try{//Load Properties file
            FileInputStream fis=new FileInputStream("src/main/java/com/nt/commons/info.properties");){
            //load properties file content to java.util.Properties class obj
            props=new Properties();
            props.load(fis);
      }//try
      catch(Exception e) {
            e.printStackTrace();
      }
   }

      //static factory method supporting Factory pattern
      public static  Flipkart  createFlipkart()throws Exception {
            //create target class obj
            Flipkart fpkt=new Flipkart();

            // Load Dependent class
            Class c=Class.forName(props.getProperty("dependent.comp"));
            //create object using reflection object
            Constructor cons[]=c.getDeclaredConstructors();
            //create object
            Courier courier=(Courier) cons[0].newInstance();
            //set  Dependent class object to target class obj
            fpkt.setCourier(courier);
         return fpkt;
      }

   }
```

```
//StrategyDPTest.java
package com.nt.test;

import com.nt.comp.Flipkart;
import com.nt.factory.FlipkartFactory;

public class StrategyDPTest {

   public static void main(String[] args) {
      try {
            //get target class object form Factory
            Flipkart fpkt=FlipkartFactory.createFlipkart();
            // invoke the b.method
            String result=fpkt.shopping(new String[] {"shirt","trouser","mobile"},
                                        new float[] {3400.0f , 5600.0f, 60000.0f});
            System.out.println(result);
      }//try
      catch(Exception e) {
            e.printStackTrace();
      }
   }//main
}//class
```

Problems in Strategy DP impl using core Java setup even after improvising with properties file
==========
a) We need to implement factory pattern explicitly
b) No internal cache by default.. If needed we should write those logics explicitly
c) To make code of 100% Loosely coupled code , we need to take properties file and xml file but reading from properties file or from xml file needs lots of api knowledge .. This improves burden on the programmer

To overcome these problem implement strategy DP using Spring framework



same last add spring context support dependency

applicationContext.xml

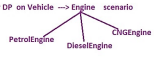
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">

   <!-- cfg dependent classes -->
   <bean id="dtdc" class="com.nt.comp.DTDC"/>
   <bean id="dhl" class="com.nt.comp.DHL"/>
   <bean id="bDart" class="com.nt.comp.BlueDart"/>

   <!-- cfg Target class -->
   <bean id="fpkt" class="com.nt.comp.Flipkart">
    <property name="courier"  ref="bDart"/>
   </bean>

</beans>
```

```
//StrategyDPTest.java
package com.nt.test;

import org.springframework.beans.factory.support.DefaultListableBeanFactory;
import org.springframework.beans.factory.xml.XmlBeanDefinitionReader;

import com.nt.comp.Flipkart;

public class StrategyDPTest {

   public static void main(String[] args) {
         //create IOC container
         DefaultListableBeanFactory factory=new DefaultListableBeanFactory();
         XmlBeanDefinitionReader reader=new XmlBeanDefinitionReader(factory);
         reader.loadBeanDefinitions("com/nt/cfgs/applicationContext.xml");
         //get Target class obj
         Flipkart fpkt=factory.getBean("fpkt",Flipkart.class);
         //invoke the b.method
         String resultMsg=fpkt.shopping(new String[] {"shirt","trouser","glasses"},
                                        new float[] {5679.60f,5557.55f,4545.77f} );
         System.out.println(resultMsg);


   }//main
}//class
```

Advantages of developing strategy DP application using spring framework
==========
a) No need of developing Factory pattern ..becoz spring Container/IOC container itself acts factory Patttern
b) Keeps Spring bean class objs in the built-in internal cache of IOC container giving the reusability of spring bean class obj
c) Allows to give Dependency Mamagement inputs in multiple ways like xml file, annotations, java classes
d) Improves the readability of the Application.
e) 100% Loosely copuling promised.
f) It is industry standard to use Spring Framework for Dependency Manageemnty.
and etc..

=>A perfectly develop spring App contains the following DesignPatterns
a) Factory pattern
b) strategy DP
c) Flyweight pattern (keeping objects in the internal cache and reusing the objs is activity of Flyweight patter)

Assignment:: Implement strategy DP on Vehicle --> Engine scenario



Sep 28 Setter Injection Vs Constructor Injection

Q) Where should use setter Injection and where should we contructor Injection?

Ans) if all properties spring bean are mandatory to parcipate in Dependency Injection then go for Constructor Injection .. if the proeprties of spring bean are optional to participate in dependency Injection then go for setter Injection.

=>Spring Bean class with 4 properties
a, b, c, d are bean propeties

Let us assume 4 param constuctor is placed in spring bean class
=>If we do constuctor injection using 4-param constructor then all params must partipate in the injection

=> Using constructor Injection , if want to involve our choice no.of properties in the Injection process then we need multiple overloaded constructors that is 4! no.of constructors (24), if the bean properties count is 4. 10! no.of constructors if the bean properties count 10(36,28,800)

Conclusion:: Performinng constructor Injection using overloaded constructors on our choice no.of bean properties is very complex process. So prefer constructor injection only when all bean proeprties mandatory to particapte in costructor injection.. For this only one parameterized constructor is enough

eg: spring bean with 10 properties
=> contstrctor injection on all bean properties ==> one 10 param consturctor is enough
=> contstrctor injection on my choice bean properties ==> we need 10! no.of overloaded constructors (36,28,800)

=>While workign setter methods seetter Injection we can involve our choice no.of bean proepeties fo setter Injection.

10 spring bean properties in spring bean calss
==========
=>To invole our choice no.of bean properties in setter Injection we just need 10 setter methods
=> All properties must participate in dependency Injection :: Not possible with setter Injection.

Final Conclusion::
==========
spring bean class with 10 bean properties
==========
=>all bean properties must participate in the Dependency Injection then go for one 10 param constructor supporting constructor Injection
=>all bean properties are optional to participate in the Dependency Injection then go for 10 setter methods supporting seetter Injection.

=>In spring bean class few properites are mandatory to participate in dependency Injection another few propeerties optional to participate in the dependency Injection then prefer using both setter and constructor Injection
=>For mandatory properties :: cosntructor Injection
=> for optional oprperties :: setter Injection.

=>Spring's dependency Injection is not given to Inject the enduser non-technical input values to spring bean properties like name,age, addrs and etc.. It is always given to Inject programmer supplied technical input values (liked jdbc properties) or IOC container created spring bean class objs (like DTDC obj to Flipkart)

jdbc properties :: jdbc driver class name, jdbc url, db user, db pwd.

//Employee1.java

```
package com.nt.beans;

public class Employee1 {
    //all are mandtory to participate
    private int eno;
    private String ename;
    private  float billAmt;

    public Employee1(int eno, String ename, float billAmt) {
        System.out.println("Employee1:: 3-param constructor");
        this.eno = eno;
        this.ename = ename;
        this.billAmt =billAmt;
    }


    @Override
    public String toString() {
        return "Employee1 [eno=" + eno + ", ename=" + ename + ", billAmt=" + billAmt + "]";
    }

}
```

Customer.java

```
package com.nt.beans;

public class Customer {
    //cno,cname,billAmt are mandatory properties
    private  int cno;
    private String cname;
    private  float billAmt;
    //caddrs,mobileNo are optional properities
    private String caddrs;
    private long  mobileNo;

    public Customer(int cno, String cname, float billAmt) {
        super();
        this.cno = cno;
        this.cname = cname;
        this.billAmt = billAmt;
    }

    public void setCaddrs(String caddrs) {
        this.caddrs = caddrs;
    }

    public void setMobileNo(long mobileNo) {
        this.mobileNo = mobileNo;
    }

    @Override
    public String toString() {
        return "Customer [cno=" + cno + ", cname=" + cname + ", billAmt=" + billAmt + ", caddrs=" + caddrs
                + ", mobileNo=" + mobileNo + "]";
    }
```

//Student.java

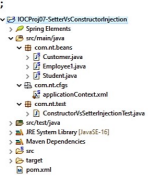
```
package com.nt.beans;

public class Student {
    //let us assume optional to provide
    private  String sname;
    private  String  collegeName;
    private  int age;
    private  String qlfy;

    public void setSname(String sname) {
        this.sname = sname;
    }
    public void setCollegeName(String collegeName) {
        this.collegeName = collegeName;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public void setQlfy(String qlfy) {
        this.qlfy = qlfy;
    }

    @Override
    public String toString() {
        return "Student [sname=" + sname + ", collegeName=" + collegeName + ", age=" + age + ", qlfy=" + qlfy + "]";
    }



}
```

applicationContet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">

  <bean id="emp1" class="com.nt.beans.Employee1">
    <constructor-arg name="eno" value="1001"/>
    <constructor-arg name="ename" value="raja"/>
    <constructor-arg name="billAmt" value="545.45f"/>
  </bean>

  <bean id="stud1" class="com.nt.beans.Student">
    <property name="sname" value="raja"/>
    <!-- <property name="collegeName" value="CBIET"/>
    <property name="qlfy" value="B.Tech"/> -->
  </bean>

  <bean id="cust1" class="com.nt.beans.Customer">
    <constructor-arg name="cno" value="1001"/>
    <constructor-arg name="cname" value="rajesh"/>
    <constructor-arg name="billAmt" value="67.88f"/>
    <property name="caddrs" value="hyd"/>
  </bean>

</beans>
```

Client App

```
package com.nt.test;

import org.springframework.beans.factory.support.DefaultListableBeanFactory;
import org.springframework.beans.factory.xml.XmlBeanDefinitionReader;

import com.nt.beans.Customer;
import com.nt.beans.Employee1;
import com.nt.beans.Student;

public class ConstructorVsSetterInjectionTest {

    public static void main(String[] args) {
        DefaultListableBeanFactory factory=new DefaultListableBeanFactory();
        XmlBeanDefinitionReader reader=new XmlBeanDefinitionReader(factory);
        reader.loadBeanDefinitions("com/nt/cfgs/applicationContext.xml");
        //get Spring bean class obj
        Employee1 emp1=factory.getBean("emp1",Employee1.class);
        System.out.println(emp1);
        System.out.println("====================");
        Student  st1=factory.getBean("stud1",Student.class);
        System.out.println(st1);
        System.out.println("====================");
        Customer  cust1=factory.getBean("cust1",Customer.class);
        System.out.println(cust1);



    }

}
```



Cyclic Dependency Injection/ Circular Inejction

Sep 29 Setter Injection Vs Constructor Injection -part 2

=> Here Target and dependent classes maintain HAS-A property having Circular Dependency i.e Both are dependent to each other.

=>If A,B are the spring Beans the A dependent B and B is dependent to A
=> In real Projects , we do not see the need of Cyclic DI.. It is purely POC (knowledge)

=>setter injection support Cyclic DI and constructor injection does not Support Cyclic DI

Cyclic DI using setter injection

A.java

```
package com.nt.beans;

public class A {
        private  B b;

        public A() {
              System.out.println("A:: 0-param constructor");
        }

 public void setB(B b) {
        System.out.println("A.setB()");
              this.b = b;
        }

@Override
public String toString() {
      return "A [b=]";
}


}
```

B.java

```
package com.nt.beans;

public class B {
 private A a;


  public B() {
        System.out.println("B: 0-param constructor");
  }


public void setA(A a) {
        System.out.println("B.setA()");
        this.a = a;
}


@Override
public String toString() {
      return "B [a=]";
}


}
```

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans https://www.springframework.org/schema/beans/spring-beans-3.1.xsd">

 <!--Spring beans cfg  -->
 <bean id="a1" class="com.nt.beans.A">
   <property name="b"  ref="b1"/>
 </bean>

 <bean id="b1" class="com.nt.beans.B">
     <property name="a"  ref="a1"/>
    </bean>
</beans>
```

Client App

```
package com.nt.test;

import org.springframework.beans.factory.support.DefaultListableBeanFactory;
import org.springframework.beans.factory.xml.XmlBeanDefinitionReader;

import com.nt.beans.A;

public class CyclicDITest {

      public static void main(String[] args) {
            //create IOC container
            DefaultListableBeanFactory factory=new DefaultListableBeanFactory();
            XmlBeanDefinitionReader reader=new XmlBeanDefinitionReader(factory);
            reader.loadBeanDefinitions("com/nt/cfgs/applicationContext.xml");

            //get Spring bean calss obj
            A  oa=factory.getBean("a1",A.class);
            System.out.println(oa);

      }

}
```

Cyclic DI using constructor injection (not possible .. throws Exception org.springframework.beans.factory.BeanCurrentlyInCreationException: Error creating bean with name 'a1': Requested bean is currently in creation: Is there an unresolvable circular reference?)

A.java

```
package com.nt.beans;

public class A {
        private  B b;

        public A(B b) {
              this.b=b;
              System.out.println("A:: 1-param constructor");
        }


@Override
public String toString() {
      return "A [b=]";
}


}
```

B.java

```
package com.nt.beans;

public class B {
 private A a;


  public B(A a) {
        this.a=a;
        System.out.println("B: 1-param constructor");
  }



@Override
public String toString() {
      return "B [a=]";
}


}
```

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
https://www.springframework.org/schema/beans/spring-beans-3.1.xsd">

 <!--Spring beans cfg  -->
 <bean id="a1" class="com.nt.beans.A">
   <constructor-arg name="b"  ref="b1"/>
 </bean>

 <bean id="b1" class="com.nt.beans.B">
     <constructor-arg  name="a"  ref="a1"/>
    </bean>
</beans>
```

```
//Client App
==============
package com.nt.test;

import org.springframework.beans.factory.support.DefaultListableBeanFactory;
import org.springframework.beans.factory.xml.XmlBeanDefinitionReader;

import com.nt.beans.A;

public class CyclicDITest {

      public static void main(String[] args) {
            //create IOC container
            DefaultListableBeanFactory factory=new DefaultListableBeanFactory();
            XmlBeanDefinitionReader reader=new XmlBeanDefinitionReader(factory);
            reader.loadBeanDefinitions("com/nt/cfgs/applicationContext.xml");

            //get Spring bean calss obj
            A  oa=factory.getBean("a1",A.class);
            System.out.println(oa);          throws  Exception::
                                             org.springframework.beans.factory.BeanCurrentlyInCreationExcept
      }                                      ion: Error creating bean with name 'a1': Requested bean is
                                             currently in creation: Is there an unresolvable circular reference?
}
```

Cyclic DI using **one side** ***constructor injection*** **and another side Setter injection**

Possible but we need to pass that class beanid in factory.getBean(-) where setter injection is enabled.

A.java

```
package com.nt.beans;

public class A {
        private  B b;

        public A() {
              System.out.println("A:: 0-param constructor");
        }

        public  void setB(B b) {
              System.out.println("A.setB()");
              this.b=b;
        }


@Override
public String toString() {
      return "A [b=]";
}


}
```

B.java

```
package com.nt.beans;

public class B {
 private A a;


  public B(A a) {
        this.a=a;
        System.out.println("B: 1-param constructor");
  }



@Override
public String toString() {
      return "B [a=]";
}


}
```

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
https://www.springframework.org/schema/beans/spring-beans-3.1.xsd">

 <!--Spring beans cfg  -->
 <bean id="a1" class="com.nt.beans.A">
   <property name="b"  ref="b1"/>
 </bean>

 <bean id="b1" class="com.nt.beans.B">
     <constructor-arg  name="a"  ref="a1"/>
    </bean>
</beans>
```

Cleint App
==========

```
package com.nt.test;

import org.springframework.beans.factory.support.DefaultListableBeanFactory;
import org.springframework.beans.factory.xml.XmlBeanDefinitionReader;

import com.nt.beans.A;
import com.nt.beans.B;

public class CyclicDITest {

      public static void main(String[] args) {
            //create IOC container
            DefaultListableBeanFactory factory=new DefaultListableBeanFactory();
            XmlBeanDefinitionReader reader=new XmlBeanDefinitionReader(factory);
            reader.loadBeanDefinitions("com/nt/cfgs/applicationContext.xml");

            //get Spring bean calss obj  (works)
            A  oa=factory.getBean("a1",A.class);
            System.out.println(oa);
            /*System.out.println("=======");
            B  ob=factory.getBean("b1",B.class);  //throws exception
            System.out.println(ob); */

      }

}
```

What is difference b/w setter injection and constutor injection?

Setter Injection	Constructor Injection
a) IOC container uses setter method to perform this injection	a) IOC container uses parameterized constructor to perform this injection
b) use <property> tag to cfg this injection	b) use <constructor-arg>tag to cfg this injection
c) place @Autowired on the setter Method to perform this injection *	c) place @Autowired on the parameterized constructor to perform this injection *
d) IOC container first creates target class obj ,then creates Depedent class obj	d)IOC container first creates Dependent class obj ,then creates Target class obj
e) If All bean properties of spring bean class are enabled with setter injection then IOC container creates spring bean class object using 0-param constructor	e) If any bean property of spring bean is enabled with constructor injection then IOC container creates spring bean class object using parameterized constructor
f) It is good to use this setter injection if the bean properties of spring bean class are optionai participare in dependency injection..	f) It is good to use this constructor injection if all the bean properties are mandatory to participate in the injection.
g) To Perform injection on n properties in all permitation and combination we generally need max "n" setter methods	g) To Perform injection on n properties in all permitation and combination we generally need max "n!" overloaded constructor (partically impossible)
h) we can use P name space tags and attributes for simplifying setter injection cfgs **	h) we can use C name space tags and attributes for simplifying constructor injection cfgs **
(i) Supports Cyclic DI	i) Does not support Cyclic DI
j) Bit slow in the injection process becoz injection takes place after creating sprign bean class obj	j) Bit faster compare to setter injection becoz the injection takes place while creating object itself

Realtime DI Background preparation (Mini Project Development) sep 30 Realtime DI

=> A typical project contains the following logics
a)presentation logics (Logic giving user-interface)
b) business logic/ Service logic (Logic performing calculations, analyzations, filtering ,sorting and etc..)
c) Persistece logic performing CURD/CRUD Operations on s/w
(Insert,update,delete and select operations)
and etc..

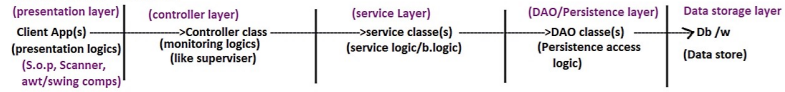
Limitations of keeping multiple logics in single java class (It is like bachelor single room)

=> Multiple logics will be mixedup , So we can say there is no clean seperation b/w logics
=> Code becomes very clumzy
=> The modifications done in one kind of logics effects other kind of logics
=> Maintainace and the enhancement of the Project becomes complex
=>Parellel development is not possible , So the productivity is poor.
=> It is not indurstry standard approach to develop the s/w Apps.

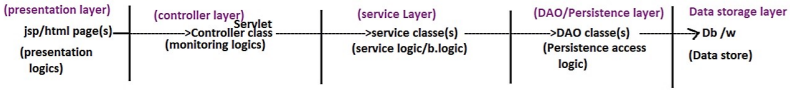
=>To overcome these problems .. Develop the App/project as the Layered App/Project where multiple category logics will be placed in multiple logical or physical partiations (nothing but layers) of the Project.

=> A Layer is logical or physical partiation of the Project having 1 more resources(files/classes) representing certain category logics.
=>Presentation layer (1 or more classes/files) represents presentation logics
=>Service layer (1 or more classes/files) represents service logics
=Persistence layer/DAO Layer (1 or more classes/files) represents Persistence logics
=>Controller layer (generally 1 class) represents controlling or monitoring logics

Standalone Layered Application (It is like 3BHK Flat)



Web based Layered Application



DAO class (Data Access Object class)

=>The java class whose code is written using one or another DataAccess tehchnologies or frameworks to seperate persitence logics from other logics of the Project to make the Persistence logics as the reusable logics and flexible to modify is called DAO class.

=>DAO class should have only persistence logic peforming CURD operations .. it should not have even c=a+b kind of b.logic

=>DAO class can use either direct JDBC con object or Pooled JDBC object for interacting with Db s/w..

=> If project is having <100 db tables then take 1 DAO class per 1 db table
if project is having >=100 db tables then take 1 DAO class per related 4 to 5 db tables

=> Can have perisitence logic calling PL/SQL procedures and functions

=> Evey DAO class contains two parts
a) Query part (All SQL queries will be placed at top of the class in upper case letters as private string constant values (private static final String variable)

eg: private static final String GET_ALL_EMPS="SELECT * FROM EMP";

To make query specifc current class
To access query with out object
To make sure that query is not going be modified in rest of prg
Recomanded to take in upper case to differentiate from the regular java code.

Code part
Java method having persistence logic devleoped using persistence technologies /frameworks like JDBC, hibernate, spring JDBC, spring ORM, spring data jpa.

=>DB s/ws like oracle ,mysql ,mongoDB and etc.. called Data storage technologies

=> JDBC,hibernate, spring JDBC, spring ORM, spring data jpa and etc.. are given to access and manipulate the data of Db s/ws.. So they are called Data Access technologies or frameworks

=>insert ,update, delete and select perations are called CURD Operations

Two types of JDBC con objects
1.Direct JDBC con obj — created by the progammer manually
2.Pooled JDBC con obj — Collected from JDBC con pool (ready made jdbc con obj)

Service class

=>The java class that contains b.logic or service logic representing calculations,analyzations, filtering, sorting and etc..
=> It is 1 class per module
=> purely contains b.logics /service logics

University Project
|---> Admissions module
|--->Examinations module
|--->Academics module
|--->Sports module
|---> Payroll module
|--->Co-Circular activities module
|--->Placements module
|---->HR department module

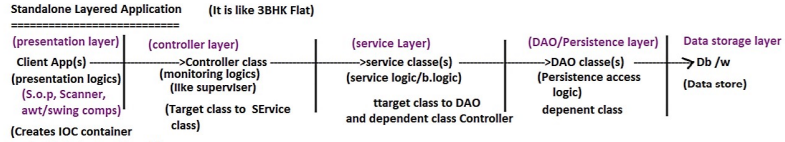
Controller class

=>The java class contains purely monitoring logic
=>This class code controls all activities of the Application
=> It is 1 class Per Project.
=>Makes Approprivate Client Apps/ presentation comp/view comp taking approprivate SErivce class and vice-versa.
=>It is like superviser
=>It contains addtional logics like logging,, auditing and etc.. to enable more control and monitoring on the Project.
=>All the activities of the App must take place under the monitoring Controller.

Client App / View Comps/ html,jsp comps

=>Contains presentation logics providign user-interface to enduser
=>Responsible to gather inputs from enduser and to display results for enduser
=> Any exception raised in any layer during the excution of the Project should be propagated/passed to Client App to display to enduser..

Standalone Layered Application (It is like 3BHK Flat)



use
Where did u Realtime DI in your project?

Ans) My Project is Layered App .. In the Project multuple layer classes are there like DAO class, service class,Controller class and etc.. By cfg these classes as spring beans for IOC container we make the IOC container injecting DAO class object to service class obj and Service class object Controller object..
note:: The Client App gets Controller object having injected Service,DAO class objs by calling factory.getBean(-) method