

Title :: spring basics/core, spring boot and Micro Services (End to End learning of spring and spring boot)


pre-requisites :: core Java (strong), JDBC (awareness)  
duration :: 120 to 140 sessions (weekly 6/7 sessions) duration :: 120 to 140 Session (single slot) (5:30 pm to 7 pm)  
course fee :: 8K (INR), 5.5K for Naresh IT fullstack java students 60 to 70 Sessions (two slots)

Faculty Details ::  
Name :: Mr. nataraj  
FaceBook(FB) Group name :: natarajjavaarena  
FB group url :: <https://www.facebook.com/groups/388095825162910> (private group)  
To collect material :: <https://www.facebook.com/groups/388095825162910/files> (batch code :: NTPSPBM5 516)

email id :: natarajjavaarena@gmail.com  
for new Batches info :: <https://nareshit.com/new-batches-hyderabad/>  
youtube url :: <https://www.youtube.com/NareshIT>  
admin details ::  
Mr.Srikanth+91 6302968665

Course materials /Class room notes and Applications::  
Initially through FB group , later using Google Class room / Google drive and GIT HUB

Modules covered in this course ( This course deals with end to end learning of spring, spring boot and MicroServices)

=>spring core/basics  
=> spring boot  
=> spring jdbc  
=> spring msp  
=> spring boot data jpa  
=> spring boot mongoDB  
=> spring boot mail  
=> spring boot MVC  
=> spring boot security  
=> spring boot oauth 2.0  
=> spring boot JWT  
=> spring boot batch  
=> spring boot scheduling  
=> spring boot Rest (for Restful webservices)  
=> spring boot Cloud (MicroServices programming)  
( 45 sessions)  
=> spring boot JMS  
=> spring boot Kafka  
=> spring boot actuators  
=> Integration with Angular/ React Js  
=> Spring boot web flux ( Reactive programming)  
and etc...  
  
=> 5/6 Mini Projects  
|----> by applying realtime condning standards  
|----> by applying design pattern (best practices)  
|----> by dealing challenging problems and solutions  
  
+ 15+ Java real time tools  
|----> maven, gradle, log4j, alfa, code debugging, code coverage (jacoco), junit, mockito, http unit,  
postman, swagger api, github, Agile-JIRA, Jenkins CI/CD, docker and etc..  
  
+ 3 IDEs ( Eclipse JEE, STS IDE, IntelliJ IDE)  
  
+ 10+ common topics  
|----> annotations , java beans, lombok api  
|----> Java 8 features  
|----> resume preparation  
|----> Interview tips  
|----> how to show gap as experience  
|----> Project release process  
|----> TDD (Test Driven Development)  
|----> Do's and Do not's after joining in the company  
and etc..  
  
Spring boot = spring ++  


Who needs this course?

Ans) Every Java Job seeker (both fresher and experienced)

What are the other course that i can do parallelly with this course?

Ans) Core Java (make sure that u already completed oops, exception handling, collections topics)

adv.java  
hibernate  
design patterns  
devops  
Aws  
UI Technologies

What is the difference b/w Java Developer / Server Side Java Developer and Java FullStack Developer?

Ans) Java Developer / Java Server side Developer

(Expertized only Java JEE technologies and Java frameworks)

Fresher

Core java, adv.java, oracle , html , css, Java script , spring, spring boot

+ CRT (Campus recruitment Training ) /Softskills

Experienced

=> Fresher topics - CRT

+ hibernate, design patterns , microServices, web services

=> java realtime tools {10+}

Java Full stack Developer

( He is all rounder to take care of every thing related project development , testing , release and support)

( Full stack developer must be expertized in 6 domains )

Domain1 (java)

core java, adv.java , spring, spring boot, microServices, hibernate, webservices, design patterns and etc..

Domain2 (Database)

=> SQL DB s/w :: oracle/mysql/postgresql/SQLServer  
=> No SQL DB s/w :: MongoDB, Cassandra , couch base and etc..

Domain3 (Testing/QA - Quality Assurance)

=> manual testing  
=> Selenium  
=> appnium (for automated testing)

Domain4 (UI Technologies /Front End Technologies)

=> html ,css, java script, jquery/angular/reactjs , bootstrap and etc...

Domain5 ( DevOps)

(Tools simplifying the project development, testing , release and maintainace operations)

github, maven, gradle , jenkins , docker , ansible , kubernetes and etc...

Domain6 ( Cloud)

=> Provides hardware,softwares , Operating system , networking Infrastructure on rental basis through internet which used by developer to develop , test and to release the projects

eg: aws (amazon webservices)  
google cloud  
Microsoft Azure  
and etc..

What is the road map to become java fullstack developer?

Ans) If u r fresher or trying to show 1/2 years gap as experience then first prefeere to join in the company as java developer and keep the target of becoming java fullstack developer with in next 2-3 years

If ur showing 3/4/5 experience (real or fake) better to join in the company as a full stack developer  
If u can take the load otherwise follow the 1/2 years formulae.

Java developer is x  
Full stack develop salary is 1.5x or 2.x

Q) Is Chat GPT or any other AI tool kills the jobs of Java developers?

Ans) No , but kills the jobs of content writers , BPO sector (call centers), KPO (support centers)  
becoz chat GPT or any other AI can not generate code for complex requirement like e-commerce App code having UPI Payment integration

Q) Is the python good or Java Good to build career in software Industry?

Ans) Java is used and proved in  
-> website development  
-> distributed Apps development (like gpay apps)  
-> mobile Apps  
-> BigData Apps (like hadoop, sparks)  
-> DataScience domain (It is combination 32+ technologies  
|----> AI,ML Including java, python  
and etc...

python is used/proved only in  
=>IoT  
=> DataScience domain  
note:: Industry is not trusting  
to use python in  
websites, distributed Apps  
mobile apps , bigdata aapps development  
becoz of security and performance issues.

What are packages we can expect after completion this course?

Ans) As Java Developer (fresher)	As a Full stack Java Developer (fresher)
3.5 Lacs 6.4 Lacs	5Lacs to 10 Lacs
As Java Developer (1+/2+)	As a Full stack Developer (1+/2+)
4.5 Lacs to 10 lacs	7Lacs to 13 Lacs
As a Java Developer (3+/4+/5+)	As a FullStack Developer (3+/4+/5+)
7.5L to 15 Lacs	10 lacs to 22 Lacs

What is the Path that we should choose to build strong IT Career in Java?

Ans) First join the in company as Java developer and target to become full stack java developer in the time span of next 2-3 years.

What is the difference among programming language, software technology and software framework?

Ans) C++, Java, C#, Javascript and etc.. called programming languages (these are raw materials of programming eg:: rice granuals, wheat granuals)

JDBC, Servlet, JSP, EJB, JMS and etc... are called Java Technologies (these are semi-finished products of the programming eg: wheat flour(ata), rice flour)

JDBC :: Java DB connectivity (old)  
JDBC is a trademark (new)

JSP :: Java Server Pages  
EJB :: Enterprise Java Beans  
JMS :: Java MessagingService

Struts, spring, spring boot, hibernate and etc.. are called Java frameworks (these are called fully finished products almost every thing is every, we need to give final touch eg: Maggie Noodle, baba noodles)

Course in JAVA domain

Core Java ----> deals with Programming language learning  
Adv.java ----> deals with Programming technologies learning  
spring/spring boot/hibernate/... ----> deals with Programming frameworks learning

note:: 80% projects of JAVA are there in Java frameworks (especially in spring, spring boot)  
20% Projects of JAVA are there in Java technologies

note:: MVC, WebServices, MicroServices are not frameworks/technologies/Programming languages  
these are architectures or mechasims or methodologies or specifications (plans) to develop Projects using any language or technology or framework

=> MVC (Mode-View-Controller) Architecture is given to develop the webapplications/websites ( browser to App interaction)  
eg: browser to flipkart.com, browser to amazon.in

=> WebServices, MicroServices architecture is given to develop the Distributed Apps /Remoting Apps

eg:: flipkart.com to UPI Payment App (PhonePe/Gpay/BharatPe /...)  
eg :: amazon.in to PaymentGateways (VISA/Master/Mastreo /...)

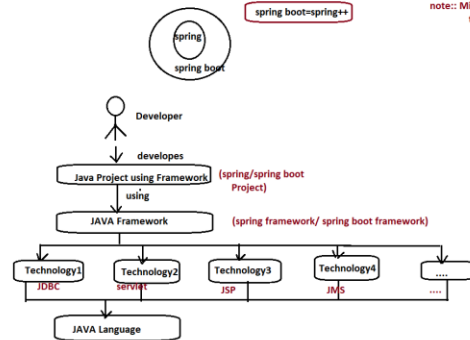
=> Java Frameworks are built on the of java tehcnologies  
=> Java Technologies are built on the of java Language

note:: some frameworks like spring boot are built on the top of other framework

spring boot= spring ++

spring boot= spring++

note:: MicroService architecture means extension of webservices architecture to develop distributed App in a better way.



=> E-commerce App with UPI Payment provision



WS arch ----> webServices architecture  
MS arch ----> MicroService architecture

## Programming Language Vs Software Technology Vs Software framework

### Programming Language

=> Programming language is installable software that provides basic features like raw materials of programming that are required to develop the software apps

=> Programming languages define the syntaxes (rules), semantics (structure) of the programming by supplying compilers and interpreters.

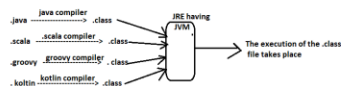
=> The structure of the code is called semantics  
=> The rules of the coding is called syntax

=> Programming languages are the base for creating technologies, frameworks, tools, Operating systems, Database softwares and etc...

=> Window OS is created using vc++ language  
=> Linux OS is created using c language.

eg: C++, Java, C#, Scala and etc...

=> Java, Scala, Groovy, Kotlin, Go and etc... called JVM based languages becoz all these languages give compiled code as .class file which can be executed by using the common JVM.



=> There is no need learning other JVM based languages becoz JAVA is continuously having its own new features and other features inspired from JVM based languages...

3 domains in Java

=> JAVA language (core java)  
=> JAVA Technologies (adv.java)  
=> JAVA Frameworks (spring, spring boot, hibernate, struts, JSF and etc...)

### Software Technologies

=> Software technology is a software specification giving set of rules and guidelines in the form of technology APIs (java packages having classes, interfaces, enums and annotations) to create technology based implementation softwares

=> Working with Technology based implementation softwares is nothing but working with technology

eg: JDBC, Servlet, JSP, EJB, JMS and etc...

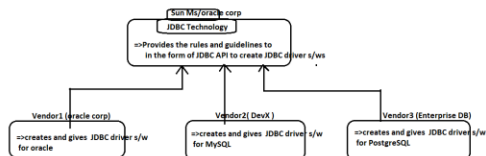
JSP: Java Server Pages  
EJB: Enterprise Java Beans  
JMS: Java MessagingService

=> JDBC Technology implementation software comes in the form JDBC driver s/w... To use JDBC Technology in our Apps, we need to arrange JDBC driver s/w

=> To Work with Servlet technology in our Apps, we need to arrange the Servlet Technology based implementation software that is servlet container

=> To Work with JSP technology in our Apps, we need to arrange the JSP Technology based implementation software that is JSP container

=> To Work with EJB technology in our Apps, we need to arrange the EJB Technology based implementation softwares that is EJB container



### Two types of Technologies

#### 1. Open Technologies

=> Here the rules and guidelines of technology are open to the all vendors in the market to create and release the technology based implementation softwares

eg: All Java Technologies are the open Technologies  
JDBC, Servlet, JMS, JSP and etc...

#### 2. Proprietary Technologies

=> Here the rules and guidelines of proprietary technologies are specific to one vendor company and only that vendor company is allowed to develop the implementation softwares...

eg: All Microsoft technologies like asp.net, vb.net and etc...

note: Technologies are called semi-finished products they give 30% to 40% readymade facilities and makes the developers to develop the remaining 60% to 70% code..

## Frameworks

=====

Framework is a special Installable software that is built on the top of one or more technologies having ability to generate the common logics of the Apps dynamically based on the App specific logics given by the Programmers.

Framework is a special Installable software that provides abstraction on one or more technologies to simplify the Application's development process.

note:: Every Framework Internally uses one or more technology to generate the common logics dynamically but it does not make the programmer to know about it more over It gives the feeling to the programmers that they are developing everything of the app using the framework itself.

### examples of JAVA frameworks

struts ----> from apache  
JSF ----> from sun MS/ oracle corp  
spring ----> from Interface21 (pivotal team)  
spring boot ----> from Interface21 (pivotal team)  
hibernate ----> from softTree (RedHat)  
ibatis ----> from apache  
toplink ----> from oracle corp  
Eclipse Link ----> from Eclipse  
and etc..

For developing  
complex and large scale Apps

### .net frameworks

asp.net mvc , asp.net core and etc...

### php frameworks

Drupal , wordpress and etc..

For Developing  
faster and smaller  
web applications

### Java script frameworks

jquery, angular, reactjs , vuex , Express Js and etc...

For small,medium scale  
websites

### python frameworks

Django , flask and etc..

### Plain JDBC App ( Technology based Application Development)

=====

- 1) Load jdbc driver class to activate JDBC driver s/w  
[JDBC driver s/w is the bridge b/w Java App and DB s/w]
- 2) Establish the Connection with DB s/w from the Java app  
[ acts as the road b/w Java app and DB s/w]
- 3) Create the JDBC Statement obj that acts as the vehicle  
between Java App and Db s/w  
[ This vehicle carries inputs (SQL Queries) from the Java App  
to DB s/w and also carries outputs (SQL Queries execution results) from  
DB s/w to Java App ]
- 4) Send and execute the SQL Queries in DB s/w using the Statement object
- 5) Gather SQL Queries results from the Db s/w using the Statement object
- 6) Handle the exceptions
- 7) Close the connection with Db s/w

Common logics

Application Specific  
Logics

Common Logics

=>In Technologies based Application development , the Programmer needs to take care of both Common logics and Application specific logics i.e the programmer is overburdened here.

=> The technology based application development is having boilerplate code problem ..

note:: The code that repeats across the multiple parts of the Project either with no changes or with minor changes is called boilerplate code..  
(Common logics)

### Spring JDBC App ( spring framework App)

=====

- 1) create jdbcTemplate class obj having one DB details  
[JdbcTemplate takes care of the common logics of application  
like 1,2,3 and 6,7 of the above code by internally using plain JDBC code]
- 2) Send and execute SQL Queries to the DB s/w
- 3) Gather results and process the results.

(application specific logics)

note:: while developing the framework style Apps we just need to take care of only Application specific logics becoz the framework will take care of the common logics the application ..

### Advantages of working with frameworks

=====

- a) provides abstraction on multiple technologies and simplifies the application development
- b) Makes the Programmers to write the lesser code becoz the majority code will be generated by the Applications Dynamically
- c) Gives productivity in Programming (Doing more work in less time with Good Quality)
- d) Framework APIs (packages ) designed based on realtime use-cases /scenarios .. which can be used in real Projects directly for completing various requirements.
- e) Easy to learn and easy to use



Different types of Java Frameworks (Based on the kind of apps we develop)

- a) Web application frameworks /MVC frameworks
- b) ORM frameworks
- c) JEE Frameworks/ Application Works
- d) Webservice Frameworks / Distributed App Frameworks
- e) BigData Frameworks

#### d) Webservice Frameworks / Distributed App Frameworks

These frameworks are based on web services methodology of programming or application development to develop Distributed Apps as the web based interoperable Applications.

Interoperability means the client and server apps of Distributed App can be developed either in same domain/language/technology or different domain/language/technologies

=> webservice is the best methodology or process to develop the distributed Apps (data exchange format is xml)

Two types of webservices

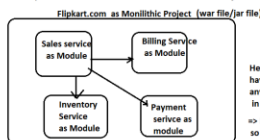
- a) SOAP based webservices (Uses SOAP over https protocol for the communication b/w client and server apps) (old and legacy)
- b) Restful webservices (Uses http with JSON as protocol for the communication b/w client and server apps) (Modern and best)

Java Frameworks to develop SOAP based webservices :: jax-ws , axis , apache cxf and etc..

Java Frameworks to develop Restful webservices :: jax-rs , jersey , Rest easy , Spring Rest , spring boot Rest (Best)

Microservices is an architecture to develop the Projects which says do not develop all the services of the Project as different module bound in a Project... It say develop different services as different apps/projects and integrate them.

old style Project development is called Monolithic Architecture based Project development

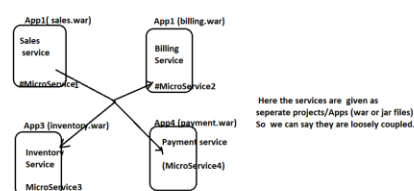


Here all services are bind to a single Project having Inter communication i.e. we can not any service outside of project to use in another Project.  
=> Here all services/modules are interlinked and tightly coupled so they can't be taken out and can not be used independently.

MVC Architecture based Projects Monolithic architecture Projects

In Microservices Architecture every service will be developed separate project.. So that each service/module can be used independently and can be used along with other services.. More importantly we can take out each service/module and can be used in another project easily

Flipkart.com App having Microservices Architecture



Here the services are given as separate projects/Apps (war or jar files) So we can say they are loosely coupled.

In Microservices architecture each Micro Service /App will be developed Restful webservices and lots of Microservices related tools , design patterns, best practices will be applied to make them executing more effectively..

Microservices = Restful webServices + + +

Microservices = Restful webService Apps + tools + design Patterns + Best practices + ....

#### e) BigData Frameworks

=> The data that is beyond storing and processing capacity is called BigData..  
i.e. this data can not be stored in regular HDD and DB s/w's  
eg: youtube data, FB data, google data , amazon data and etc...

=> BigData frameworks are given to store and process this bigdata by using the regular or ordinary computers connected in a network.  
(Commodity Hardware)

eg: Hadoop -> from apache  
spark -> from apache (100 times faster than hadoop)

BigData is processing in part DataScience operations..

1024 bytes -> 1kb  
1024 kbs -> 1MB  
1024 MBs -> 1GB  
1024 GBs -> 1TB  
1024 TBs -> 1PB  
1024 PBs -> 1ZB  
..  
..

Two types of Java frameworks (Based on the kind of Programming we do)

- 1. Invasive Frameworks
- 2. Non-Invasive Frameworks

Invasive Frameworks

=> These frameworks Apps are tightly coupled with frameworks apis i.e. the classes of App development either should implement framework api interfaces or should extend from framework api classes.

=> The classes of these frameworks based App development can not be taken out and can not be used in other frameworks

=> These frameworks statements is " Come to me , use to me , develop Apps with me and stay for ever with me"  
eg: struts framework

real life examples :: joining in the company with bond  
entering with marriage relationship

Non-Invasive Frameworks

=> These Frameworks based Apps are loosely coupled with framework apis i.e. the classes of App development need not to implement or extend from framework api interfaces or classes.

=> The App classes will be developed as ordinary java classes, so they can be moved to other frameworks..

=> Non-Invasive frameworks statement is " come to me ,use me to develop the Apps , if do not like me then take ur code or classes and execute in other frameworks"

eg: spring ,spring boot , hibernate , jsf and etc..

real life examples :: joining in the company with out bond  
making friendship with others

The Application whose services can be accessed either locally or remotely from different types of client apps developed in different technologies is called Distributed App..

eg: Phone App (any UPI Payment app)

=> The services of PhonePe App (distributed App) can be accessed from different types of client Apps like web applications, mobile apps, desktop apps, IOT apps , swiping machines (embedded system Apps) and etc..  
[ This indicates the server App (Distributed App) can be developed in any language/ technology and the Client Apps who want to consume the services of server app can be developed in any language/technology.. this is nothing but interoperability]

=> To Develop Distributed Apps in Java use RMI/EJB and etc.. technologies

=> To develop Distributed Apps in Java use webservices based frameworks like jax-ws , jax-rpc, axis , apache cxf, resteasy , spring rest, spring boot rest and etc..

=> Distributed App deals with App to App interaction

=> web application details with browser to App interaction



- ⇒ POJO class (Plain Old Java Object class)
- ⇒ POI (Plain Old Java Interface)
- ⇒ Java bean class
- ⇒ Bean class / component class
- ⇒ spring Bean class

- The ordinary java class with out any
- It is the class that designed as a
- in advanced technologies and from
- This class does not implement tech
- extend from technology/framework a
- non-void programming takes the
- It is the class that can be compiled
- POJO class is not aggre of extend
- extension from technology/framework

—image, exception handling, multi-threading, collections, `assert`, `using`, `streamline`, reflection api, networking and etc... part of java language

—`@file`, `@url`, `@serializable`, `@SuppressWarnings` and etc... java technologies

—`struts`, `spring`, `spring boot`, `hibernate` and etc... are java frameworks

- ⇒ APTs are also called as libraries and they come in the form of jar files.

↳ P000 classes can have 1 or more b, methods having b, flag:

```

class Test extends Demo {
    ...
}
class Demo extends HttpServlet {
    ...
}

```

```
public class Test {
    // Internal coding
    // Hibernate libraries [jar files] are required
    // in class path [Exceptional case]
}
```

note: Most of the times, the ordinary java class that is used in the Projects developed with the support of java technologies and frameworks is called POJO class.

- The Interface that is not extending from technology/framework/api. Interface is called POI
- we can create POIs by just using java interfaces
- while developing advanced applications using technologies or frameworks... if a r developing the interfaces as single interfaces of core java style then they are called POIs
- POI classes and POIs model Programming supports Non-Invasive Programming  
(Invasive couplings)

part of java language api	"Deread" is not a POI beco java rtd. Deread is part of RM Technology
Interface Deread extends Deread	Interface Deread

```
interface Demand extends Demand {
    ...
}

interface Demand extends java.util.Serviceable {
    ...
    (part of Serviceable
    technology)
}
```

**CLASSPATH (Exceptional case)**

- setter methods are given to set data, to modify data of properties (fields/member variables)
- getter methods are given to get data of properties (fields/member variables)
- Java bean classes are used as helper classes. In projects to combine multiple values to single object and to send that object data from 1 class to another class of same Project or two different Projects

order and filling details

instead of keeping multiple different values in arrays or collections, prefer using java bean classes because java bean classes contain properties and

- c) All Member variables (Java bean properties) must be taken as private and non-static [ to protect the state of the Java bean class obj from direct access through subclasses]
- d) For Every bean property (member variable) there must be separate setter method and getter method

```
//sample has been class
//Sample has been class
public class Credentials implements java.io.Serializable {
    //has been properties
```

```

    // the data of the bean property
    }

    public long getCardNo() {
        return cardNo;
    }
}

```

To read data from the bean methods

4 setters and 4 getters methods

note: Java bean classes do not contain any b. methods with b. logic.

[5] : since no constructor is placed, the Java compiler generates  
 a default constructor as the default constructor

- ↳ **Primitives** must **not** contain getter and setter methods.
- ↳ **Java Bean** class must contain **getter** and **setter** methods.
  - ↳ But **POJO** class may or may not contain **getter** and **setter** methods.





## Spring core module

- ⇒ It is base module for other spring modules (spring having 20+ modules)
- ⇒ If this module is used alone, we can develop only standalone apps
- ⇒ This module gives two built-in spring containers as light weight containers

They are

- BeanFactory Spring Container (Basic Container)
- ApplicationContext spring container (Advanced Container)

note: Spring/spring boot supplied two containers are no way alternate to Servlet container (jsp container) becoz the servlet container/jsp container can be used only for web application development where as the spring/spring boot containers can be used for standalone apps, web applications, distributed apps and etc..

- ⇒ The spring Containers are useful for two operations

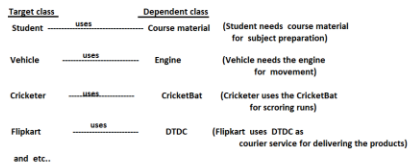
- For Spring Bean life cycle management
- For Dependency Management

### Spring Bean life cycle management

- ⇒ The Java class whose object is created and managed by spring container is called spring bean.
- ⇒ Spring Container takes care of spring bean life cycle management i.e spring container loads spring bean class, creates the object, manages the object and destroys the object.
- ( Spring container takes care of all the activities of spring bean from birth to death i.e right from object creation to object destruction it takes care of all the activities)

### Dependency Management

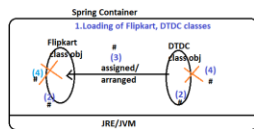
- ⇒ arranging dependent class object to target class object is called Dependency Management
- ⇒ The class that uses the other class services is called target class (or) main class
- ⇒ The class that acts as helper class to target class is called dependent class (or) helper class



and etc..

of  
⇒ As Part of Dependency Management, It is the work Spring container to assign/arrange Dependent class object for target class object.

- ⇒ In spring bean life cycle management, The spring container creates both target and dependent spring bean class/obj, manages those objects and destroys the objects if needed.
- ⇒ In Dependency Management, the spring container assigns/arranges the Dependent class object to target class object ready to use for target class object.



(1),(2),(4) → are spring Bean life cycle management Operations  
(3) → is Dependency Management operation

- creating objects for Flipkart, DTDC classes
  - Assigns/Arranges DTDC class obj to Flipkart class object, (Dependency Management)
  - Destruction Flipkart, DTDC spring bean class obj
- (Both these operations are taken care by spring container)

⇒ 1, 2, 4 are the Spring bean life cycle management

### Core Java App performing Dependency Management

- Programmer loads and creates the target and dependent class obj (Flipkart, DTDC class obj)
- Programmer assigns Dependent class obj to target class object (DTDC class object to Flipkart class obj)
- Programmer manually calls the b.methods on Flipkart class obj which internally uses DTDC class obj methods
- Programmer nullifies the target and Dependent class obj to keep them ready for Garbage Collection

All operations are manual operations  
(More burden to programmer)

### Spring App performing Dependency Management

- create spring container
- spring Container loads both target and dependent classes (Flipkart, DTDC classes)
- spring container creates the objects for both target and dependent classes (objects for Flipkart,DTDC classes)
- spring container assigns/arranges dependent class object to target class obj (assigning DTDC class obj to Flipkart class obj)
- Programmer get Flipkart class obj(target class obj) from Spring container to call b.methods on it
- spring Container destroys both target and dependent class obj

Here most of the operations will be taken care by Spring container .. So burden on the programmer will be reduced.. So the spring based Dependency Management is more recommended.

note: Spring containers are also called IOC containers

**IOC : Inversion Of Controller**

Why the Spring containers are called IOC containers?

Ans) In traditional Java apps all activities of life cycle mgmt and dependency management are under the control of Programmer i.e programmer does those operations manually like loading classes, creating objects, assigning dependent class obj to target class obj and etc... this indicates the entire control is in the hands of programmer.  
( washing clothes manually, growing chicken birds manually)

In spring Apps, all the above said life cycle management and dependency management operations are taken care by Spring Container by taking the control from the Programmer and performing the operations on behalf of programmer ..i.e It is reverse or inversion of control .. Since Spring containers are taking control from programmer and performing all activities life cycle management and dependency management.. So the spring containers are called IOC containers.

( washing clothes using washing machine, poultryfarm with automation)

note: Most of the times, the industry prefers calling Spring containers as the IOC containers.

Spring Container=IOC containers

## Different Approaches of Implementing Dependency Management

1. Dependency Lookup (DL)
2. Dependency Injection (DI)

The process of assigning or arranging dependent class obj to target class obj is called "Dependency Management"

=> It is like  
Arranging DTDC class obj for Flipkart class obj  
Arranging CourseMaterial obj for Student class obj  
Arranging CricketBat for Cricketer class obj

### Dependency Lookup (DL)

=> Here The target Java class writes some logics and spends some time to search and get Dependent class obj from various resources.

Ex1:: Student(target class) getting CourseMaterial(Dependent class) from the Faculty/Institute by request for it

Ex2:: Java App getting Useful <sup>(dependent class obj)</sup> ~~Random object~~ <sup>(like Database/PayTM...)</sup> ~~at~~ <sup>from</sup> ~~from~~ <sup>Jndi registry</sup>

JNDI :: Java Naming and Directory Interface

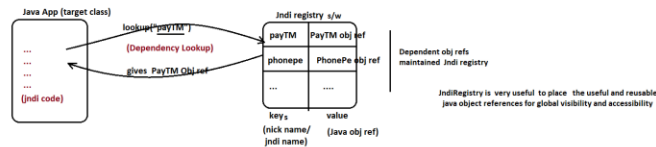
=> To provide Global visibility and accessibility to Java object.. we need to place its reference in Jndi registry

need of Jndi registry

- => If object is local to method then it is specific to that method
- => If object is taken as the instance variable ... in the class it is specific to object of the class
- => If object is taken as the static variable in the class then it is specific to that class
- => So to provide global visibility and accessibility to any Java Object then place its reference in Jndi registry (Special Registry where the references of Java objects can be placed)

Examples of Jndi registries

RMI Registry, COS Registry, Weblogic Registry, GlassFish Registry, Wildfly Registry and etc..



note: Here the Java app acting as Target class is having Jndi code/logic to search and get dependent PayTM obj ref from Jndi registry using Jndi lookup which is also called as Dependency Lookup

note:: In Dependency Lookup, The target class pulls Dependent class obj from different resources.

pros (advantages) of Dependency Lookup

=> The target class can search and get only the required Dependent class obj

cons (Disadvantages)

=> The target should write some logics and should spend some time to search and get Dependent class obj.

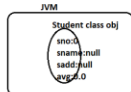
### 2. Dependency Injection (DI)

=> Here the underlying server/container/framework/runtime env/JVM dynamically assigns dependent class object to target class object.

eg1: Faculty/Institute assigning course material to Student the moment student joins the course  
eg2: ServletContainer assigns ServletConfig object to our ServletComp class object during the Initialization of Servletcomp class obj



eg3:: JVM assigns default values to the Java object properties/fields/attributes during the Initialization of Object



Student class obj is target obj here and the default value assigned to properties as the dependent values.

note:: In Dependency Injection, the underlying container /server/framework/JVM/Runtime env.. dynamically pushes the dependent values to target class object.

pros (advantages) of Dependency Injection

=> Developer need not to do any work becoz the underlying server/container/framework/JVM/Runtime env.. dynamically takes care of assigning dependent values to target class object.  
=> Target class can use the dependent obj/values directly with out wasting time to get them

cons (Disadvantages)

=> The underlying server/container/Jvm/framework/... can inject both necessary or unnecessary dependent obj/values to target class obj

Spring supports both dependency Lookup and Dependency Injection.. But we use Dependency Injection for most of the times becoz It gives performance advantage by reducing burden on the Programmer

### Spring's Dependency Management

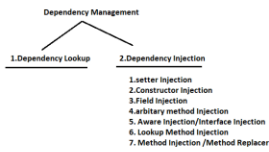
1. Dependency Lookup
2. Dependency Injection
  - a) setter Injection
  - b) constructor Injection
  - c) Field Injection
  - d) arbitrary method Injection
  - e) aware Injection/Interface Injection
  - f) Lookup Method Injection (LMI)
  - g) method Injection/Method Replacer



[illegible]

What is dependency Management?

Ans) The process of arranging dependent class obj to target class obj is called Dependency Management



=> making the underlying spring container assigning the dependent spring class obj to target spring bean class object is called Dependency Injection / Bean Wiring.

=> Spring bean instantiation means making the spring container creating the spring bean class obj  
=> Spring Bean wiring means making the spring container assigning the dependent spring bean class object target spring bean class obj

#### setter injection

=> if the spring container (IOC container) is using the setter method of target spring bean class to assign dependent spring class obj to target spring class object then it is called setter injection



=> To perform setter injection cpts in xml driven cpts use <property> tag.  
=> To perform setter injection cpts in annotation driven cpts use @Autowired

IOC container can create spring bean class obj (spring bean instantiation) in 3 ways

- (a) using 0-param constructor
- (b) using parameterized constructor
- (c) using factory method

#### (a) using 0-param constructor

=> if the spring bean is cfg with out any injections  
=> if the spring bean is cfg having only setter injections

if the spring bean is cfg with out any injections (example)

```
<!-- applicationContext.xml -->
<bean id="dt1" class="java.util.Date"/>
```

Internal code (assumption) :: java.util.Date dt1=new java.util.Date();

=> if the spring bean is cfg having only setter injections (example)

```
<!-- applicationContext.xml -->
<bean id="dt1" class="java.util.Date">
  <property name="year" value="1900"/>
  <property name="month" value="11"/>
  <property name="date" value="10"/>
</bean>
```

3 setter methods like setYear(), setMonth() and setDate() will be called to assign 3 values to java.util.Date class object. (setter injection)

=> Based on the above code the spring container/IOC container creates java.util.Date class obj having name (dt1) as the spring bean using 0-param constructor and calls 3 setter methods on that object to inject 3 simple values 3 properties (here 3 simple values are the dependent values)

Internal code (assumption)

```
java.util.Date dt1=new java.util.Date(); //bean instantiation
dt1.setDate(10); dt1.setMonth(11); dt1.setYear(1900+100); //setter injection
```

#### (b) using parameterized constructor

=> if we configure the spring bean for the constructor injection  
then the IOC container uses parameterized constructor to create spring bean class obj and to inject the dependent values to the spring bean class object.

=> To perform constructor injection in xml driven cpts use <constructor-arg> tag

=> To perform constructor injection in annotation driven cpts use @Autowired on top of constructor  
note: @Autowired placed on the setter method performs setter injection  
@Autowired placed on the parameterized constructor performs constructor injection

note: under <bean> we place <constructor-arg> for "n" times then it uses "n" param constructor to create spring bean class obj and to inject dependent values/objects to spring bean class obj

applicationContext.xml

```
<!-- applicationContext.xml -->
<bean id="dt2" class="java.util.Date">
  <constructor-arg name="year" value="110"/> //assigns 1900+110 to the "year" property
  <constructor-arg name="month" value="11"/>
  <constructor-arg name="date" value="20"/>
</bean>
```

The above code uses 3 param constructor to create java.util.Date class obj having name dt2 and to assign 3 values to it.

Internal code (assumption)

```
java.util.Date dt2=new java.util.Date(110,11,20);
```

What is factory method ? (core java recap)

The java method that is capable of creating and returning its own class or related class or any other class object is called factory method

Two types of factory methods

- a) static factory method
- b) non-static factory method /Instance factory method

#### static factory method

=> useful to create object with out using the object, directly by using the class name

eg1: Thread t=new Thread.currentThread();  
(static factory method returning its own class obj)

eg2: Calendar cal=Calendar.getInstance();  
(static factory method returning relevant class obj) (java.util.Calendar is an abstract class)  
The above getInstance() returns GregorianCalendar class obj which is the sub class obj Calendar class. So we can refer that object using Calendar class ref variable.

eg3: Properties props=System.getProperties();  
(static factory method returning unrelated class obj) (java.util.Properties is the sub class java.util.Hashtable)

#### b) non-static factory method /Instance factory method

=> To create new object by using existing object and its data

eg1: String s1=new String("hello");  
String s2=s1.concat("123"); //gives hello123  
(instance factory method returning its own class obj)

eg2: StringBuffer sb=new StringBuffer("hello how are u");  
String s3=sb.substring(0,5); //gives hello  
(instance factory method returning unrelated class object)