**TELLIPONEIC INTERVIEW**

**===============================================**

**1.**Tell me About your self?

2.What Is your current company?

3.Your are permanent or contract?

4.What is your payroll company?

5.We have a pf (provident fand)account?

6.What is your official notice period?

7.How soon you can join?

8.What is your ctc(cost to company)?

9.What expected ctc?

10.Is it negsable?

11.What is your total experience?

12.You have experience on build management tools?

13Where it is written code?

14.What tool use for brug track ?

15.How to assign your tasks?

16.Who assign the tasks?

17.What is methodology using your commonly?

18.Why your following agile or what is advantage of agile?

19.What is reason for leaving company?

20.Are you willing to relocate?

21.What is your domain?

22.What is company official email and company id?

23.How long working present company?

24.Present company team size?

25.Roles and responsibilities?

**DAILY ACTIVITIES**

**============================================**

1. Tell me about your self?
2. Tell me about your project?
3. what is methodology using your company?
4. What are the roles and Responsibilities Your Project?
5. Difference b/w spring and spring boot?
6. Difference b/w web services and micro services?
7. Spring boot advantages?
8. Micro service advantages?
9. Difference b/w soap and rest?
10. Restful advantages and architecture?
11. Micro service architecture advantages?

JAVA

============================================================

1. What is Java?
2. What is purpose of data types?
3. What are the data types in java explain each data type?
4. What is variable?
5. What are rules for writing variables?
6. How to declare variables?
7. How many types of data members in a class ? explain?
8. What is method? What rules of writing methods?
9. What is a class?
10. What is Object?
11. What are features of OOPs?
12. Class 2) Object 3) Data hiding 4) abstraction 5) Encapsulation 6) inheritance 7) polymorphism
13. Explain each feature of oops with real-time examples
14. Explain Is-A and has-A relationships?
15. Why java won’t provide support for multiple inheritance?
16. What is method signature?
17. What are rules for writing overriding?
18. What is constructer? What is purpose of constructer?
19. What is the difference b/w constructer and instance block?
20. What are the rules for writing constructer?
21. Explain super() and this() methods and super, this keywords?
22. Explain static and instance control flows?
23. What is abstract?
24. What is abstract method?
25. What is abstract class?
26. What is interface?
27. Explain detail extends vs. implements
28. What is marker interface? What is advantages of marker interface?
29. What is adapter class?
30. What is Difference b/w interface and abstract class?
31. What is package?
32. What is advantages of packages?
33. What are access modifiers of java?
34. Explain each and every access modifier? Explain difference places uses?
35. What is Exception?
36. What is purpose of exception handling?
37. Explain default exception handling?
38. What is checked exception?
39. What is un-checked exception?
40. Explain the exception handling using try and catch blocks?
41. Difference b/w final ,finally, finalize?
42. What is throws key word?
43. What is throw key word?
44. What are top10 exception of java?
45. Explain the exception hierarchy detail?
46. What is thread?
47. What is multithreading?
48. How many ways to create a thread?
49. What is thread scheduler?
50. Difference b/w t. start() and t.run() ?
51. Explain thread life cycle?
52. Explain detail thread priority?
53. Explain the yield() ,join() and sleep() methods?
54. What is synchronization what is advantage synchronization?
55. What is class leval lock?
56. Explain the synchronization block?
57. What is deadlock? Explain ?
58. What is daemon thread?
59. What is green thread?
60. What is array?
61. What is collection?
62. Difference b/w array and collection?
63. Difference b/w collection and collections?
64. What are the key interfaces of collection frame work?
65. Difference b/w list, set, map?
66. Explain list interface?
67. Difference b/w array list vs. linked list vs. vector vs. stack? sor
68. What is cursor?
69. Types of cursors?
70. Difference b/w each and every cursors?
71. Explain set interface?
72. Difference b/w hash set and linked hash set?
73. Explain sorted set interface?
74. Explain navigable set interface?
75. Difference b/w hash set vs. Linked hash set vs. tree set?
76. Difference b/w comparable and comparator?
77. Explain map interface?
78. What is entry?
79. **ConcurrentModificationException**
80. Explain total and detail map concept?
81. Explain queue interface?
82. Explain the java i/o package?
83. How to create a file?
84. What are writer &readers of i/o package?
85. Explain the java. Lang package?
86. What are the methods of object class? Explain?
87. What is String class what are methods of string class?
88. What is mutability and immutability?
89. Difference b/w string and string buffer and string builder?
90. What are methods of string buffer?
91. What is method chaining?
92. What wrapper class?
93. Explain auto boxing and un-boxing?
94. What is serialization ?
95. What is de-serialization?
96. How can achieve serialization and dieselization?
97. What is externalization?
98. Difference b/w serialization and externalization?
99. Explain serial version UID?

**HIBERNATE**

====================================

1. **What is framework?**
2. **What is hibernate?**
3. **What are the features of hibernate?**
4. **Explain each features of hibernate?**
5. **What are the supported collections of hibernate?**
6. **How to write hibernate mapping and configuration ?**
7. **How to create database tables automatically?**
8. **How to read the object from database using hibernate?**
9. **Difference b/w load() and get() ?**
10. **What are the life cycle states of hibernate?**
11. **Difference b/w update() and merge() ?**
12. **What are the generated values of hibernate?**
13. **Why should I make session factory object as singleton?**
14. **How to make session factory object as singleton?**
15. **How to use criteria api?**
16. **Difference b/w save() and persist() ?**
17. **Difference b/w save() and saveOrUpdate()**
18. **What are relationships of hibernate explain each?**
19. **What are annotations of hibernate ?**
20. **What are joins statements of hibernate?**
21. **Hibernate session factory is thread safe?**
22. **Hibernate session is thread safe?**
23. **What is difference between Hibernate save(), saveOrUpdate() and persist() methods?**

**SPRING**

**==================================**

1. **Why spring frame work?**
2. **what are the modules of spring framework?**
3. **What is tightly coupling?**
4. **What is loosely coupling?**
5. **Explain pojo , JavaBean, spring bean?**
6. **What is Dependency?**
7. **Types of Dependency Injections?**
8. **What are the types of dependcies?**
9. **How to create configuration in spring?**
10. **How to configuring setter injection (primitive and object types)?**
11. **What are the attributes of property tag?**
12. **Explain constructer injection?**
13. **What are the attributes of constructer-tag?**
14. **What is container?**
15. **What are the types of containers in spring?**
16. **How to start bean factory container?**
17. **What is spring ioc container?**
18. **Difference b/w class path resource and file system resource?**
19. **How to start Application context container?**
20. **What are the collection types dependcies?**
21. **What is the circular dependency?**
22. **Difference b/w setter and constructer injections?**
23. **Purpose of c-name and p-name util-name spaces?**
24. **How do you Instanceated a spring container with multiple configuration files?**
25. **What is purpose of bean autowireing?**
26. **Explain about byname,bytype,constructer?**
27. **Explain default autowireing?**
28. **What the bean scopes?**
29. **What are annotations of spring core?**
30. **What are the stereotype annotations in spring?**
31. **What are spring mvc annotations?**

**32.EXPLAN SPRING MVC FLOW**

**SPRINGBOOT**

**=============================================================**

1. **What is spring boot?**
2. **What is difference b/w spring and spring boot?**
3. **Advantages of Spring Boot Applications?**
4. **Draw back of Spring boot applications?**
5. **Why we need spring boot?**
6. **What are the spring boot components ?**
7. **What is the spring boot starter?**
8. **What is spring boot auto-configurator?**
9. **What is spring boot actuator?**
10. **What is spring boot CLI?**
11. **What is spring boot Initializer?**
12. **Why we need spring boot initializer?**
13. **What and why embedded servers?**
14. **How can you run SBA on custom port?**
15. **@SpringBootApplication**
16. **How can you logging in SBA?**
17. **What are embedded containers which are supported by Spring Boot?**
18. ***What Are Embedded Containers Which Are Supported By Spring Boot?***
19. ***What Is Dev. tools In Spring Boot?***
20. **How Can You Implement Spring Security In Spring Boot Application?**
21. **What Is Yaml?**
22. **How To Configure Datasource Using Spring Boot?**
23. **What Is The Configuration File Name Used By Spring Boot?**
24. **How To Reload My Changes On Spring Boot Without Having To Restart Server?**
25. **What Is The Difference Between An Embedded Container And A War?**
26. **Spring boot How Does It Work? How Does It Know What To Configure?**

**WEBSERVICES**

**=============================================================**

1. **What are the web services?**
2. **What are the types of webservices?**
3. **Diff b/w soap and rest webservices?**
4. **How many ways to communicate diffirent applications?**
5. **What is rest stands for?**
6. **What is rest?**
7. **What are the commonly used http methods in rest based architecture?**
8. **What are the restful webservices?**
9. **What is resource in Rest?**
10. **How to represent resource in Rest?**
11. **Which protocol used by restful webservices?**
12. **What is messaging in restful webservices?**
13. **What are the core components of http request?**
14. **What are the core components of http response?**
15. **What is addressing in restful webservices?**
16. **What is uri?**
17. **What is the format of uri in rest architecture?**
18. **What is purpose of http verb in rest based architecture?**
19. **How to develop standard uri?**
20. **What is statelessness in restful webservices?**
21. **What is the advantage of statelessness?**
22. **What do you mean idempotent?**
23. **Which type of webservices methods are to be idempotent?**
24. **Which type of web service methods are read only?**
25. **What is diff b/w put and post operations?**
26. **What is purpose of options method?**
27. **What is purpose of Head m*eth*od?**
28. **What is caching?**
29. **Which header of http response provides the date and time of the resource when it was created?**
30. **Which header of http response provides the date and time of the resource when it was last modified**
31. **Which hearder of http response provides control over caching?**
32. **What is the purpose of http status code?**
33. **What http status code 200 state?**
34. **What http status code 201 state?**
35. **What http status code 204 state?**
36. **What http status code 304 state?**
37. **What http status code 400 state?**
38. **What http status code 401 state?**
39. **What http status code 404 state?**
40. **What http status code 409 state?**
41. **What http status code 500 state?**
42. **Explan what is rest and restful?**
43. **Mention what tools are required to test your webservices?**
44. **What is the jax-ws and jax-rs?**
45. **Which markup language use for restful webservices?**

**MICROSERVICES**

**1. What is Micro Services?**

**2. What is Advantages or benefits of micro services?**

**3. What is draw backs of Micro services?**

**4. What is monolithic Architecture?**

**5. Diff b/w monolithic and micro services architecture?**

**6. Explain Monolithic Architecture?**

**7. Advantages and Disadvantages of Monolithic architecture?**

**8. How to Works micro Service architecture?**

**9. What is rest endpoint?**

**10. What is actuator?**

**11. What is cloud bus?**

**12. What is load balancer?**

**13. What is repository area?**

**14. What is a zuul proxy?**

**15. What is Eureka server?**

**16. What is Authorization server?**

**17. What is Config server?**

**18. What is Discovery server?**

**19. Purpose of circuit breaker?**

**20. What is load lancing?**

**21. Purpose of FeilgnClient?**

**22. What is spring cloud?**

# 23. **Differences between @RequestParam and @PathVariable in Spring MVC**

Both annotations @RequestParam and @PathVariable in [Spring MVC](https://www.dineshonjava.com/spring-web-mvc-framework-chapter-38/) are used for fetching the values of request parameters. These annotations have similar purpose but some differences in use. The key difference between @RequestParam and @PathVariable is that @RequestParam used for accessing the values of the query parameters where as @PathVariable used for accessing the values from the URI template

## [@RequestParam](https://www.dineshonjava.com/requestparam-annotation-in-spring-mvc-with-example/)

[***http://localhost:8080/tutorials/bookmark/?site=dineshonjava&id=200***](http://localhost:8080/tutorials/bookmark/?site=dineshonjava&id=200)

@RequestMapping(value = "/tutorials/bookmark")

public String bookmark(

@RequestParam(value="site", required=true) String site,

@RequestParam(value="id", required=false) String id){

...}

## @PathVariable

[***http://localhost:8080/tutorials/bookmark/100?site=dineshonjava&id=200***](http://localhost:8080/tutorials/bookmark/100?site=dineshonjava&id=200)

@RequestMapping(value = "/tutorials/bookmark/{siteId}")

public String bookmark(

@PathVariable(value="siteId") String siteId

@RequestParam(value="site", required=true) String site,

@RequestParam(value="id", required=false) String id){

...

Email**:** sreenivasarao.developer@gmail.com

**SREENIVASARAO K**  Phone No**: +91-9676919080**

**EXPERIENCE SUMMARY:**

* Having **4** **years** of experience in IT Industry Developing **Java/J2EE technologies.**
* Strong experience in working with **spring (IOC, ORM, MVC, BOOT).**
* Hands on experience in **Rest Full Web Services** and **MICROSERVICES**.
* Implemented **Object Oriented Programming concepts**.
* Have good experience in **Java Collection Framework.**
* Has comprehensive knowledge in **MVC** frameworks.
* Implemented **Design patterns Dao Design patterns.**
* Efficiency in using different kinds of servers like **Tomcat**.
* Experience in using **GIT** for code repository and as version control mechanism.
* Experience in using **LOG4J** for logging and debugging.
* Ability to work effectively both as an individual and as part of a team.

**TECHNICAL SKILLS:**

**Operating system** : Windows

**Languages** : Java and SQL

**J2EE Technologies** : Servlets, JSP.

**Frameworks** : Hibernate, Spring, Spring Boot

**Web Services** : Rest Full Web Services, MicroServices.

**Application/WebServers** : Tomcat

**IDE** : Eclipse, IntelliJ IDEA

**Databases** : My SQL

**Version controller** : GIT

**Project Management Tools** : Jira

**Build Tool**  : Maven, Gradle

**ACADEMICS:**

* **MCA** from **Osmania** University.

**PROFESSIONAL EXPERIENCE:**

* Working as **“Software Developer” at cognizant technology solutions,** from **March 2018 to till date**

**TECHNICAL EXPERIANCE:**

**PROJECT#1**

**Title** : Altares

**Client** : Dun & Bradstreet

**Role** : Developer

**Responsibility** : Coding&Development

**Environment** : Java, Spring Boot, Rest Full web services, MicroServices,

JPA, Tomcat, Maven, GIT, Log4J

**Period**  : Nov 2021 to Till Date

**Team Size** : 08

**Project Description:**

Altares provides real-time data from world's biggest commercial database, containing verified information on +330 million companies worldwide. Connect your systems or applications to our database and benefit from accurate and reliable insights about your B2B relationships. Our complete company profiles not only provide insights in your relationships, its structure and our technology also helps organisations improve efficiency in data management. It automates decision-making processes and creates a single view on your B2B relationships. specialties of Altares are Bedrijfsinformatie, Master data, Data quality, Data enrichment, identificatie, Data governance, Information Technology, Company data, Compliance, Risk Management, Market Intelligence, Marketing, Supply Management, Credit management, and Data.

**Responsibilities:**

* Implementing functionalities using design pattern oriented architecture includes **Controller**, **Service** and **DAO layers.**
* Implementing functionalities like **Pagination.**
* Implementing **Server side** validations.
* **Maven** used for Project Management Structure.
* Integrated **Spring Boot** with **JPA Repository.**
* Object Creation and injection was done using **Spring IOC** mechanisms.
* Coding and debugging by using **Debugging tool** and **Log4j**.
* Used **GIT** for code repository and as version control mechanism.
* Attending the Scrum meetings.

**PROJECT#2**

**Title** : **BLL Management System**

**Client** : BLL, London, UK

**Role** : Developer

**Responsibility** : Coding&Development

**Environment** : Java, Spring Boot, Rest Full web services, MicroServices,

JPA, Tomcat, Maven, GIT, Log4J

**Period**  : Dec 2018 to 2021 Nov

**Team Size** : 08

**Project Description:**

The main aim of the project is to provide utility to maintain day to day operations of apartments. This application helps them to store all transactions electronically in a system, which in turn saves lot time, money and energy.

This project is developed for a BLL construction company. This company built many apartments and they have a plan to construct many other apartments. Presently they maintain all records manually, to keep track of owner details, tenants details, apartment details, complaints, maintenance etc., now it is very difficult to manage all the data manually, also if some information is required urgently then to obtain this is very difficult.

To solve this problem now they are looking to develop one application. This application covers modules like admin module, complaints module, voting, and event management.

**Responsibilities:**

* Involved in developing Controller classes using **spring**.
* Implemented **Service** classes and **DAO** classes.
* Analyzing the requirements and prepare low level documents.
* Involved in handling the user defined Exceptions in DAO layer.
* Involved in **Server Side** Validations.
* Implemented **Pagination** using spring.
* Following service oriented architecture.
* Testing the codes against the bugs and errors and involving in Integration Testing.

Attending scrum meetings

**PROJECT#1**

**Title** : **E**citation

**Client** : Amgen, Thousand Oaks, USA

**Role** : Developer

**Responsibility** : Coding&Development

**Environment** : Java, Spring Mvc, Hibernate, Tomcat,

Jsp, Maven, GIT, Log4J

**Duration Date** : March2018 to Dec 2018

**Team Size** : 05

**Project Description:**

Ecitation Project is web based project the main aim of the project is who violates the traffic rules then maintain the violation history. This Project Covers Modules like Violation Entry, Hearing Details, Correspondence, PaymentDeatils, Penalty, Upload Download files, History.

**Responsibilities:**

* Involved into the group meetings with managers, team members for status updates, issue tracking and knowledge sharing.
* Working in as a developer in a team of 8 developers and one team lead.
* Involved in developing Controller Classes • Worked on IOC and Spring.
* Involving in developing Service and Service Implementation Classes.
* Involving in developing Repository classes and Config Classes

**Personal Details**

NAME K.SREENIVASARAO

D.O.B 04th MAY, 1986

Father’s Name Mr. K.RAMARAO

Nationality Indian

Graduation MCA

Place:

Date: **K.SREENIVASARAO**

**JVM ARCHITECTURE**

**Virtual Machine:**

* Which is not having psychical any exist**.**

**Types of** **Virtual Machines**

* There are two types

1. Hard-ware based
2. Application based

**Hard-ware based** **Virtual Machine**

* Same psychical machine several logical system we can able to create which are independent of each other.

**Advantages**

* Efficient utilization of resource

Ex: Could computing , vm-ware

* **Note:system admin relative terminology but not programmer.**

**Application based Virtual Machine**

* These are virtual machine act as Runtime Engine to run a particular programmer language.

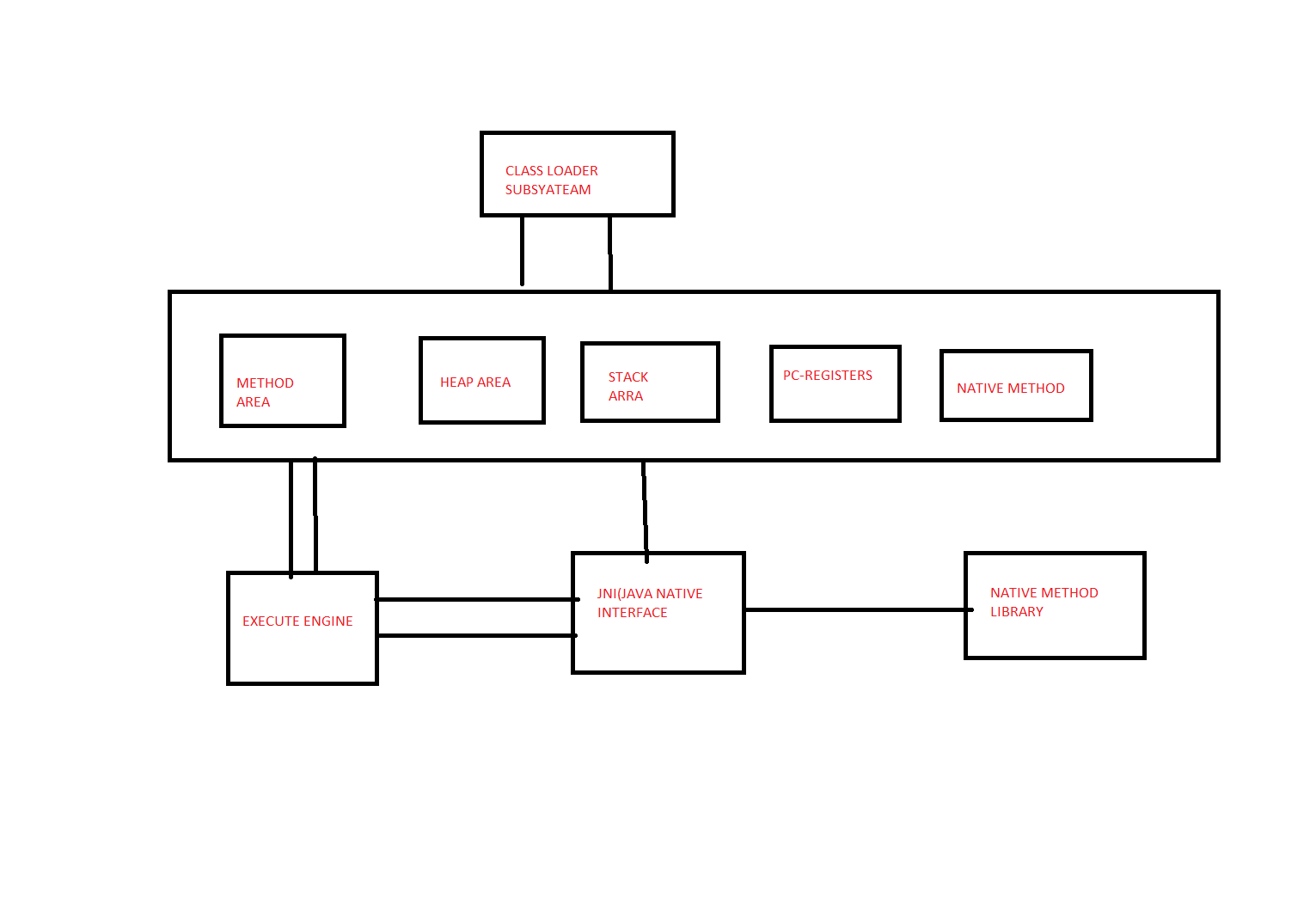
Ex: jvm(java virtual machine) act as a Runtime Engine to run java Application

**JVM**

* JVM is a Runtime Engine to run java based Application.
* JVM is a part of JRE which is part of JDK.
* JVM is Perform to Activities to load and Run Java Application

**Jvm Components**

1. Class loader Subsystem
2. Memory Area
3. Execute Engine



1. **Class loader Subsystem**

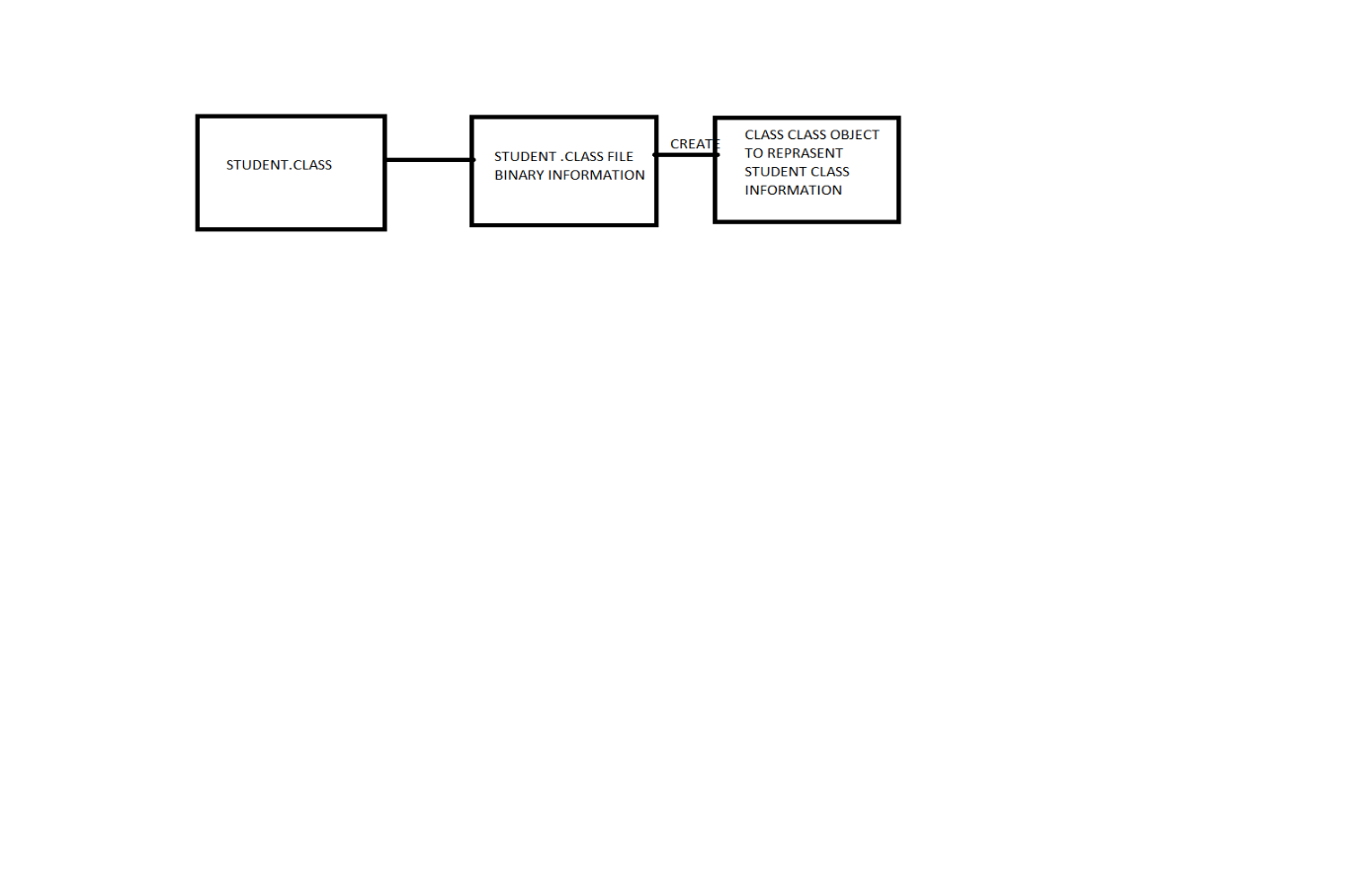
* **Class loaders mainly 3 activates**

1. Loading
2. Linking
3. Initialization

**Loading**:- Loading means Reading .class file data and store corresponding binary data in method area**.**

**Note**: for each .class file jvm will store

* Fully Qualified class name
* Methods/constructers/interfaces/class information
* Modifier information.



* Class loaders load .class file from harddisk(Stdent.class)
* After loading .class file to store method area which .class file contain all binary information of .classfile.
* After jvm will create class Class object to get class level information.

**2 Linking :-** linking perform 3 activities

1.verify 2.preparation 3 Resolution

-> verify structure of .class file for security reasons byte verification

->preparation if static variable allocate default values

**3.Intilization:-** for static variable original values assigned and static block execution top to bottom

**TYPES OF CLASS LOADERS**

**1.BootStrap Class loaders**

-> BootStrap class loaders responsible to load classes from Bootstrap classpath i.e jdk/jre/lib/rt.jar.

**2. Extension Class loaders**

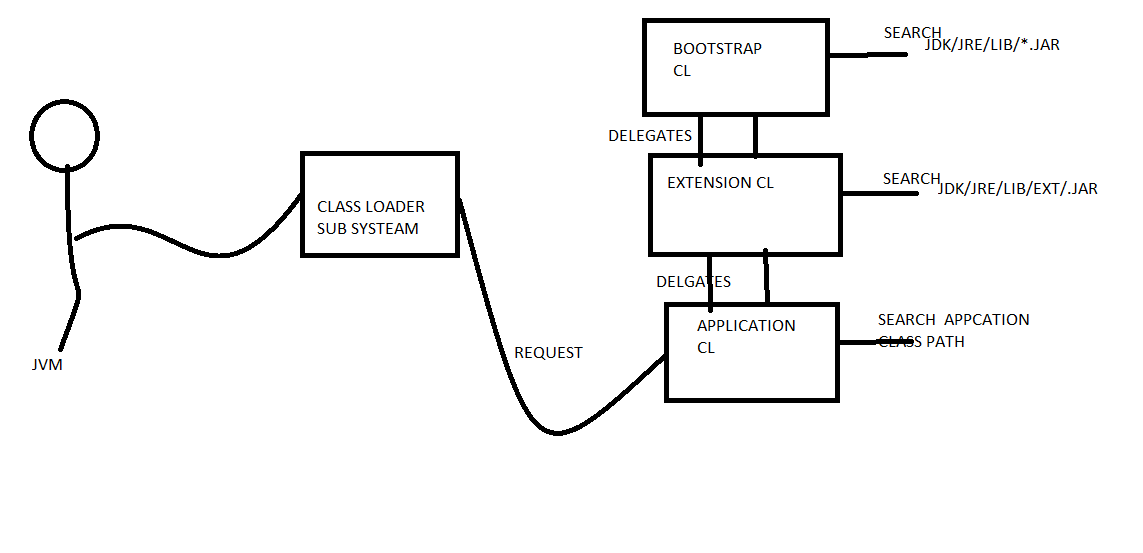
**->** Extension class loader to responsible to load extension class path.

Jdk/jre/lib/ext

**3.Applicatin Class loaders**

**->** to load class from Application class path.

**HOW CLASS LOADERS WORKS**

****

* Jvm will check .class loaded or not if not loaded jvm forward request to class loader subsystem
* Class loaders send request to Application class loader, won’t search anything Delegate’s request to Extension .Extension cl delegates to Request to BootStrap Cl
* Finally BootStrap class loader search in BootStrap class path. If available load the .class file.if not available delegates request back extension. That also not available delegates request back applicaton classloader if it is available the load the .class file if not available 3 class loaders thrown Exception ClassNotFOundException

**Core Java Technical Questions and Answers**

# **1. What is Java?**

* Java is one of the object oriented programming language.
* Platform independent programming language. Runs on any operating system like windows,linux,mac etc..
* Architectural neutral programming language. Means runs any processor like intel AMD.
* Java one of the secure programming language.

**2.What is purpose of data types?**

The main purpose of data types is to allocating sufficient memory space for to input of the program

**What are the data types in java explain each data type?**

Datatypes are divided into 3 types

1.Fundamental datatypes

2.derrived datatypes

3.customdefined Datatypes

**Fundamental datatypes:-**whose variable to store a single value but not multiple values

Int a;

a=10;//vaild

a=10,20,30,//invalid

* In java programming 8 types of Fundamental datatypes which are classified into 4 types.

1.Integer 2.Float 3.Character 4,Boolean

->  **Integer** type contain 4 types

byte,short,int,long

->**Float** type contain 2 types

Float, double

**derived datatypes:-** whose variables store multiple values of same type but not different type.

* int a[]={1,2,4,5,8}//vaild
* int a[]={1,’a’,4.5,5,8}//invalid

**custom defined :-**Datatypes: whose variable store multiple values either same or different types.

Ex.-

Student s=new Student();

s.add(101,”srinu”,’a’,4500.0);

1. **What is variable?**

A variable is an identifier whose values are changing during execution of the program.

**Rules for writing variables**

* First letter of variable must be an alphabet
* The length of variable can be upto 32 characters
* No special symbol are allowed except \_ underscore.
* No keywords to be used as variable names like while,if.else,final.

Variable declaration modifier datatype, variable,

Ex: private int a;

**5. How many types of data members in a class ? explain?**

There are two types of datamembers in a class

1. Instance/non-static datamembers
2. Static datamembers

|  |  |
| --- | --- |
| **Instance/non-static datamembers** | **Static datamembers** |
| 1.Instance datamembers memory space is created each and every time when ever object is created | 1.static datamembers memory space is created at time of class loading |
| 2.instance dm are always used for storing specific values | 2.static dm always storing common values |
| 3. instance dm declaration should not preceded by static kwd | 3.static dm declaration must be preceded by static kwd. |
| 4. instance dm access wrt to object name. | 3.static dm declaration access wrt to class name. |

* Each and every final variable must be static.

Ex: static final float PI=3.147f;

* But every static variable may or may not final
* All objects of java resides in heap memory
* All methods of java resides in stack memory
* All constant value resides in associative memory.

**6. What is method? What rules of writing methods**

-> in java writing any business logic and data access logic or persistence logic inside method only.

**Rules:-**

* No spaces allowed
* No keywords are used
* Follow the camel case.

**7. What is a class?**

**->** class is a collection of datamembers and methods.

**8. What is Object?**

-> instance of a class noting but object

**9. What are features of OOPs?**

Data hiding abstraction Encapsulation inheritance polymorphism

**OOPS CONCEPTS**

**DATA HIDING:**

* Data hiding is nothing but hiding the data
* To achieve data hiding every variable declare as private.
* Private data members can’t access outside.
* Advantage of data hiding is security.

Ex: banck balance, mail creditials

public class Account{

private double balance;

public double getBalancy{

//validation then get the balance

//advantage is security} }

**ABSTRACTION:**

->Hiding internal implantation to highlight set Service we are offering is called Abstraction.

Ex:- Atm gui screen, like withdraw,balance,mini statement.

To achieve the abstraction by using interface concept.

->advantage is security,

->with out effecting end-user we can able to change any technology.

-> maintainability, modularity. Internal any changes

**ENCAPSULATION**

->process of binding data members and corresponding methods into a single unit.

-> every java class example of Encapsulation.,

**ENCAPSULATION= DATA HIDING+ABSTARCTION**

* Here declare private variable is data hiding and abstraction below end user point of you don’t know method implementation that why abstraction.
* Advantages Encapsulation is all advantage of data hiding, abstraction
* Dis advantages perfamence, length of code(like setter and getter methods) and time consuming for verification

**public** **class** Employee {

**private** **int** empId;

**private** String empName;

**private** **double** salary;

**private** String designation;

**public** **void** setEmpId(**int** empId){

**this**.empId=empId;

}

**public** **int** getEmpId(){

**return** empId;

}

**public** String getEmpName() {

**return** empName;

}

**public** **void** setEmpName(String empName) {

**this**.empName = empName;

}

**public** **double** getSalary() {

**return** salary;

}

**public** **void** setSalary(**double** salary) {

**this**.salary = salary;

}

**public** String getDesignation() {

**return** designation;

} }

}

* **Tightly** **ENCAPSULATION**

->every variable present in a class is private.thight class is tightly encapsulation

Class A{ Class B{

Private int ids; } //tightly encapsulation Private int ids;}//not TE

**INHERITANCE**

* Obtaining features of one class to Inherited to the another class by using extend keyword.
* Is-a relationship of example of inherence.
* Advantages of inheritance is code reusability.
* Reduce the time.
* Most commonly use methods to define parent. By default available all child classes.

Note:WRITE ONCE IN PARENT BY DEFAULT AVAILBE IN CHIlD WITH DEFINE ANY METHOD.

Points to Remember:

class Parent

{

public void m1(){}

}

class Child extends Parent

{

public void m2(){}

}

public class Test {

public static void main(String[] args) {

Parent p=new Parent();

p.m1();// parent class reference we can access parent method only.

p.m2();//CE parent class reference we can't access child class method.

Child c = Child();

c.m1();//vaild

c.m2();// vaild

Parent p=new Child();

p.m1();//vaild

p.m2();// CE parent reference can hold child object but cant call child class object other wise we will get compile time error.

Child c=new Parent();//child class reference can’t hold parent object get CE.

}

**class** Parent {

**public** **void** m1() {

System.***out***.println("m1 method");

}

**public** **void** m2() {

System.***out***.println("m2 method");

}

**public** **void** m3() {

System.***out***.println("m3 method");

}

**public** **void** m4() {

System.***out***.println("m4 method");

}

**public** **void** m5() {

System.***out***.println("m5 method");

}}

**class** Child1 **extends** Parent {

**public** **void** m5() {

System.***out***.println("m5 method");

}

**public** **void** m6() {

System.***out***.println("m6 method");

}}

**class** Child2 **extends** Parent {

**public** **void** m7() {

System.***out***.println("m7 method");

}

**public** **void** m8() {

System.***out***.println("m8 method");

}}

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Parent p = **new** Parent();

p.m1();

p.m2();

p.m3();

p.m4();

p.m5();

Child1 c = **new** Child1();

// all method can acess

c.m1();

c.m6();

Child2 c2=**new** Child2();

// all methods can access

c2.m2(); c2.m8(); } }

**TYPES OF INHERITANCE**

**1.SINGLE INHERITENCE:**

Single parent and single child **.**

Ex:- Class Child extends Parent

**Note :java supported**

**2.MULTIPLE INHERITENCE:**

Multiple inheritance not supported by java wrt classes.but it supported through interfaces.

* Here our class does not extends any other class then only our class super class object.

Single child class multiple parent classes extends at a time**.**

**Note :java not supported**

**3 MULTILEVAL INHERITENCE**

Suppose A,B,C clases are there. A CLASS FEATURES AVILABLE B. B FEATURES AVAILABLE IN C.

**Note :java supported**

CLASS A

{

}

CLASS B EXTENDS B

{

}

CLASS C EXTENDS C

{

}

**4.HIRACHECAL INHERITENCE**

Single parent class multiple child classes.

**Note :java supported**

**Ex Class A**

**{**

**}**

**Class B extends A**

**{**

**}**

**Class C extends A**

**{**

**}**

**5.HIBRID INHERTENCE**

If we are using deferent types of inheritance simultaneously at time

**Note :java Not supported, because Multiple Inheritance.**

**METHOD SIGNATURE**

->Public int m1(int a,float f)

-> method name followed by type of parameters.

**OVERLOADING**

**->**In a class both methods having same Name different type of arguments.

-> any return type any modifier no problem.

->overloading is also known as compile type or static polymorphism or early banding.

class Test

{

public void m1(){

system.out.println("defalt argment method");

}

public void m1(){

system.out.println(int a);

}

public void m1(){

system.out.println("int arg method");

}

public void m1(float f){

system.out.println("float argment method");

}

public class Test {

public static void main(String[] args) {

Test t=new Test();

t.m1();

t.m1(10);

t.m1(10.25);

note: Corseponding methods will call.

}

**Imp Note**: In Overloading method resolution(process of execution) all ways takes care by complier only based on reference type but not runtime object.

Case study1:

When calling t.m1() method corresponding constructer not available immediate won,t get any compile time error. It will check promote to next leval.finally not matched any thing then only will get CE.

This is called automatic promotion in overloading.

Byte->shor->int->long->float->double

|

Char

Case2: in object version t.m1(null) -> we will get chance child.

**OVERRIDING**

**->**Two different classes method name is same signature also same i.e called overriding. return also same. But here return type is same or co- variant type accept**.**

**->**child class not satisfy with parent implantation to redefine method using inheritance.

Object->String/stringBuffer/stringBudller

String->String

StringBuffer->StringBuffer

Int->Int

**package** com.oops.examples;

**class** Parent{

**public** **void** m1()

{

System.***out***.println("parent m1 method");

}

**public** **void** m2()

{

System.***out***.println("parent m2 method");

}

}

**class** Child **extends** Parent{

**public** **void** m1(){

System.***out***.println("child m1 method");

}

**public** **void** m3(){

# System.***out***.println("child m3 method");

}

}

**public** **class** OppsExamples {

**public** **static** **void** main(String[] args) {

Parent p=**new** Parent();

p.m1();//parent m1 method

p.m2();//parent m2 method

Child c=**new** Child();

c.m1();//child m1 method

c.m2();//parent m2 method

c.m3();//child m3 method

Parent p1=**new** Child();

p1.m1();//child m1 method

p1.m2();//parent m2 method

}

}

**IMP NOTE:-**in overriding method resolution always takes by jvm based on runtime object**.**

* Overriding also know as runtime polymorphism or dynamic or late binding.

**Rules of Overriding**

**Rule 1->** method name and method signature must be matched.

**Rule 2**-> return type must be same or co-variant type accepted.

Object->String/stringBuffer/stringBudller

String->String

StringBuffer->StringBuffer

int->int

Number->Byte->Short->Integer->Long->Float->Double

* Primitive type parent and child must be same.

this role applicable only object type wrapper class only.

**Rule 3**->private method not visible to the child classes. Overriding concept not applicable for private methods.

**Rule 4->** parent class method declare as final. That method can’t be override to child**.** Otherwise CE**.**But not Non final to final acceptable**.**

**Rule 5->** abstract to non-abstract, non-abstract to abstract possible.

**Rule 6**-> sync to non sync and non sync is possible. Native to non native is possible,strctfp to non strictfp both side accept.

**Rule 7->** wrt to Modifier we can increase scope but not decrese the scope otherwise we will get CE**.** Same type or increase scope no problem.

Default<- protected<- public

Public=Public

Protected= protected->public

Default= default-> protected->public

**Rule-8->**static to non-static ,non-static to static is not possible.

Here parent and child both are static won’t get any compile time error, but not overriding concept.it is a method hiding.

Note: here class level method is static method. Not override with object leval method.

* In method hiding method resolution always takes care by compiler based on reference type.

**Difference b/w overriding and method hiding**

**Method hiding:-**

* Both parent and child class methods should be static.
* method resolution always takes care by compiler based on reference type.
* Method hiding is compiler or static polymorphism or early binding.

**Overriding:-**

* Both parent and child methods not static.
* In overriding Method resolution always takes by jvm based on Runtime object.
* Overriding is Runtime Polymorphism or dynamic polymorphism or late binding

**Rule-9:-**parent contain var-arg method child contain normal method both methods are not overriding.it is a overloading.method resolution always takes care by compiler based on reference**.**

**Rule-10:-**overriding concept applicable only methods but not variable.

* Variable resolution always takes care by compiler based on reference type**.**

**PLOYMORPHISAM**

**->**Process of represent one form in multiple forms is called Polymorphism**.**

Polymorphism example of overloading and overriding**.**

* Polymorphism divided into types

1.static Polymorphism or compile time Polymorphism or early binding

2.Dynamic Polymorphism or Runtime Polymorphism or late binding

* Static polymorphism examples of overloading and method hiding
* Dynamic polymorphism examples of overriding

**OBJECT TYPE CASTING**

* Here compiler will check 2 rules.

1.types casting we can take same type or Object type.

Object o=new String(“srinu”);

StringBuffer s=( StringBuffer)o;//here check parent and child relation

* Here o is object type StringBuffer is child type.
* Here o is Object type and StringBuffer is child type.

2.after type casting assignments must be same type or Parent type.

Object o=new String(“srinu”);

Object s=( StringBuffer)o;

Or

StringBuffer s=( StringBuffer)o;

* Here StringBuffer is child type Object is Parent type
* Above two rules followed by compiler other wise wil get CE.
* Jvm wil check at Runtime Internally Runtime object of type. Here must be same or derived type we can use. Otherwise we will get Runtime Exception.

Object o=new String(“srinu”);

Object s=( StringBuffer)o;//ClassCastException

Object o=new String(“srinu”);

Object s=( String)o;//vaild

* Below code 3 rules are statisfied.

Object o=new String(“srinu”);

String s=(String)o;

**18.What is constructer? What is purpose of constructer?**

-> class name and constructer name both are same that is called constructer.

-> the purpose of constructer is to instillation of object.

**Rules for writing Constructer.**

* Class name and constructer name must be matched,
* Return type concept not applicable in constructer even void also.other wise we will get CE.
* Only allowed modifiers in constructer is public,private.default,protected.

**IMPs in Constructers**

**Default Constructer**-> If we are not writing any constructer then only compiler generate default constructer, if we are writing at least one constructer compiler won’t generate default constructer**.**

* Overloading concept is applicable for constructers. We can write multiple overloaded constructers.
* Inheritance and overriding concept is not possible in constructers.

**19. What is the difference b/w constructer and instance block?**

|  |  |
| --- | --- |
| **constructer** | **instance block** |
| 1.the purpose of constructer is to instillation of object. | 1.we want perform any object creation then we should go for instance block |
| both will executed every object creation but instance block followed by construct | Like update one entry in db increment count value every object creation |

**20.Explain super() and this() methods and super, this keywords?**

|  |  |
| --- | --- |
| Super(),this() | Super, this |
| 1.these are constructer calls to call super class and current class constructers. | These are kwds to refer super class and current class instance methods |
| 2.we can use only in constructers as first line | We can use any where except static area. |
| 3.we can use only once in constructer | We can use any number of times. |

**21.Explan static control flow?**

-> whenever executing a java class first identification of static members from top to bottom

-> execution of static variable assignments and static block from top to bottom

-> execution of main method.

-> static control flow is one time activity which will perform at the time class loading.

Class Test

{

static int i=10;

staic

{

m1();

system.out.println("first static block");

}

public static void main(String[]args)

{

m1();

system.out.println("main method");

}

public staic void m1(){

system.out.println(j);

}

staic

{

system.out.println("second static block");

}

static int j=20;

**}**

**Note:** in a class we can declaere any no.of static blocks but all the static block executed top to bottom**.**

**22.Explan Instance control flow?**

**->** when ever we are executing a java class first static control flow will be executed.in the static control flow if we are creating an object,

1. Identification of instance members from top to bottom.

2.execution of instance variable assignments and instance blocks from top to bottom

3.execution of constructer.

-> instance control not one time activity it will be performed every object creation.

**What are Various Modifiers Available in java?**

1 **PUBLIC** :- CLASS AND METHOD BOTH ARE PUBLIC THEN ONLY WE CAN ACESS OUT SIDE PACKAGE. GLOBAL LEVAL MODIFIR

**2 DEFAULT**  -> IF VARIABLE OR METHOD DELCARE DEFAULT WE CAN’T ACESS OUT SIDE PACKAGE

PACKAGE LEVAL MODIFIER

**3 PRIVATE->** CLASS LEVAL MODIFIER WE CAN ACESS WITH IN CLASS ONLY

**4 PROCTECTED**: if variable or method declare as protected with the current package any where can access .but outside package child class we can access.

5.**ABSTRACT**:- abstract is a modifier applicable for classes and methods but not variables.

**1.Abstact() method:-**

abstract() only declaration but not implementation such type of method is called

abstract().

Ex:1 Ex2

Public class Vechile public class Loan

{ {

public absract void noofWheels(); public abstract intreast();

} }

* For example Loan is class we having no of loan intrests like gold loan intrest, personal loan intreast.if don,t know intrest to declare method.

Note:if class contain at least one abstract method that class must declare abstract other wise we will get compile time error.

**2.Abstract Class**

**->**if class declare as abstract that class is abstract class.abstarct class object creation is not possible because partial implantation class. Child class is responsible for implantation.

**Abstract class vs Abstract Method**

* A class contain at least one abstract method that class compulsory as abstract.other wise we will get compile time.
* A class does not contain any abstract method still I can declare abstract class.in a class 0 number of abstract methods also.(dummy implantation may be adapter class)

**6.Final Modifier :-**final is a Modifier Applicable for classes and variables and Methods.

**1.Diffirence B/w Final,finally,Finalize?**

**Final:-** Final is a Modifier Applicable classes, methods and variables,

* If Final Declare as Class that class can’t participate in Inheritance.
* If Final Declare as Variable that Variable Can’t be Modified.
* If Final Declare as Method that method can’t override.

**Finally:-**

Finally is a Block always associated with try and catch block for maintain cleanup activities.

**Finalize**:- Finalize a method present in object class. Finalize a method Garbage Collector call Just before Destroying Object for cleanup activities**.** like connection, session close, like that.

**7. Synchronization Modifier:-**

synchronized is a modifier applicable only methods and blocks but not class and variables**.**

**synchronization method?**

**->** two threads are executing same java object there may be chance of data inconsistency problem is occur .to overcome these problem method declare as synchronized keyword, then allowed only at time.

**synchronization block?**

-> if very few lines of code required synchronization then it’s not recommended to declare entire method as synchronized. We have Enclosure those few lines of the code by using synchronized block.

-> the main advantage of synchronized block over synchronized method it’s reduce waiting time of thread and improves performance of application.

**8.StrictFP Modifier:** if a class or method declare as StrictFP then every floating point calculation present in every concureate method as follow IEEE754 standard so that we will get platform independent Results**.**

**9.Transient Modifier:**

-> transient is a modifier Applicable only for variables.

-> At the time of serialization if we don’t save the value of a particular variable to meet security constraints then we should go for transient keyword.

**10.Native Modifier:**

-> Native is modifier Applicable only for methods and we can’t apply any where else.

-> the methods which are implemented in non-java (Mostly c, c++ ) are called native methods are foreign methods.

11.**Volatile Modifier**:

-> volatile is a Modifier applicable only variable , if declare variable as volatile every thread jvm will create local copy.

**12.Static Modifier:**

**->** The **static keyword** in [Java](https://www.javatpoint.com/java-tutorial)

is used for memory management mainly. We can apply static keyword with [variables](https://www.javatpoint.com/java-variables)

, methods, blocks and [nested classes](https://www.javatpoint.com/java-inner-class)

. The static keyword belongs to the class than an instance of the class.

The static can be:

1. Variable (also known as a class variable)
2. Method (also known as a class method)
3. Block
4. Nested class



## **1) Java static variable**

If you declare any variable as static, it is known as a static variable

* The static variable gets memory only once in the class area at the time of class loading.

### Advantages of static variable

It makes your program **memory efficient** (i.e., it saves memory

## **2) Java static method**

If you apply static keyword with any method, it is known as static method.

* A static method belongs to the class rather than the object of a class.
* A static method can be invoked without the need for creating an instance of a class.

## **3) Java static block**

* Is used to initialize the static data member.
* It is executed before the main method at the time of class loading.
* It executes logic only at the time of class loading by jvm
* Static block always executed before the main method.

**INTERFACE**

* Interface is a just service requirement specification. Never talk about implementation.
* Class is responsible for implantation interface methods, our class implements that interface.
* Every method present in inside interface public and abstract methods ,we declaring or not.

Ex:

Interface Interft class Test implements Interft

{ {

Public void m1(); public void m1{}

Public abstract void m2(); } public void m2{} }

* **DIFFIRENCE B/W INTERFACE AND ABSTRACT CLASS?**

|  |  |
| --- | --- |
| **Interface** | **Abstract class** |
| 1. If we don’t anything about implementation just we have service requirement specification, then we should go for interface | 1.if we are talking about implementation but not completely(partial) then we should go for abstract class |
| 2.inside interface every method public and abstract methods we are declaring or not. 100% abstract methods | 2.every method present in abstract class need not be public and abstract, we can take concreate method also. |
| 3 we can’t declare interface method with following modifiers  Public->private, protected  Abstract,final,static,syncronized,native,strictfp | 3 . there is no restriction abstract class method modifiers |
| 4. every variable inside interface is always public st final variable we are declaring or not | 4. need not be public static final |
| 5. we can’t declare interface variables  Private, protected, transient, volatile | 5. there is no restriction |
| 6.interface variables at the time of declaration compulsory initialization only. Otherwise we will get CE. | 6.not required to perform initialization at the time declaration |
| 7.inside interface we can’t declare instance and static blocks. Otherwise we will get CE.  8.inside interface we can’t declare constructers | 7.inside interface we can declare instance and static blocks.  8.inside abstract class we can declare constructers |

**What is marker interface? What is advantages of marker interface?**

* Which interface doesn’t contain any method such type of interface is called marker interface.
* But by implanting that interface providing some ability is provide that ability to make java language simple.

Ex: Serialization ,Cloneble,RandomAcess, interfaces.

**What is adapter class?**

* Adapter class is simple java class that implements an interface with empty Implantation.

Class Test implements Interft interface Interft {

**{ m1();**

**} m2(); }**

Adapter Class is a simple java class that implements an interface with **only an empty implementation**. Let’s suppose, there is an interface **Intref** which has various methods as follows:

interface Intref{

public void m1();

public void m2();

public void m3();

:

public void m1000();

}

As, in this interface, there are **1000** methods present and if we want to implement this interface in any particular class then we will have to override all of these 1000 methods of Intref in the implementation class. We can do this as follows:

class Test implements Intref{

public void m1(){

// some statements

}

public void m2(){

// some statements

}

public void m3(){

// some statements

}

==

public void m1000(){

// some statements } }

The problem with this approach is that it increases the length of the code and reduces readability. We can solve this problem by using the Adapter class.

abstract class Adapter implements Intref{

public void m1(){};

public void m2(){};

public void m3(){};

public void m1000(){}; }

Now we have to extends this Adapter class in our Test class and just override those required methods which we will require in the Test class.

class Test extends Adapter{

public void m1(){

System.out.println("This is m1() method.");

}

public void m80{

System.out.println("This is m80() method."); }}

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| MODIFIERS | OUTER  CLASSES | INNER CLASSES | METHODS | VARIABLES | BLOCKS | OUTER  INTERFACE | INNEER INTERFACE | CONSTRUCTERS |
| PUBLIC |  |  |  |  |  |  |  |  |
| PRIVATE |  |  |  |  |  |  |  |  |
| DEFAULT |  |  |  |  |  |  |  |  |
| PROTECTED |  |  |  |  |  |  |  |  |
| FINAL |  |  |  |  |  |  |  |  |
| ABSTARCT |  |  |  |  |  |  |  |  |
| STATIC |  |  |  |  |  |  |  |  |
| SYNCHONIZED |  |  |  |  |  |  |  |  |
| NATIVE |  |  |  |  |  |  |  |  |
| TRASIENT |  |  |  |  |  |  |  |  |
| VOLATILE |  |  |  |  |  |  |  |  |

**31.What is package?**

**->** package is a collection of classes, interface sub paclages,and sub packages contains classes and interfaces and sub sub packages.it means group of classes and interfaces into single unit

**32. What is advantages of packages?**

-> Main advantages of packages are improve the modularity.

-> Resolve naming complex(unique identification our components)

->it improve the maintainability of the application.

-> it provide security for the our components.

**35. What is Exception?**

**->** un-wanted un-excepted event to disturb normal flow of control is called Exception.

**36. What is purpose of exception handling?**

-> the main purpose of the Exception handling is highly recommended handle the Exceptions.

-> two ways handle the Exception in java 1.throws keyword and 2.try and catch blocks.

**37.Explain default exception handling?**

-> inside a method if any Exception occurs the method when which is raised is responsible to create Exception object by including the following information.

1. Name Of Exception

2.disicription of Exception

3 location at which Exceptions occurs [stack stace]

* After creating Exception object method hand ours that object to the jvm.
* Jvm will check weather the method contains any Exception handling code or not.
* If method doesn’t contain Exception Handling code then jvm terminates that method ub-normally and removes corresponding entry from stack.
* Then jvm identifies caller method and checks weather caller method contains any handballing code or not. If doesn’t contain handling code then jvm terminates that caller method also ub-normally and removes from stack.
* This process will be continued until main method and if the maim method doesn’t contain handling code then jvm terminates main method also ub-normally and removes corresponding entry from the stack then jvm handler default exception handling start.

**public** **class** Test {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

*m1*();}

**public** **static** **void** m1(){

*m2*();

}

**public** **static** **void** m2()

{

System.***out***.println("hii...");

System.***out***.println(10/0);// Exception in thread "main" java.lang.ArithmeticException: / by zero}}

# **Difference between ClassNotFoundException and NoClassDefFoundError**

**ClassNotFoundException:-**

Object o= Class.forName(args[0].newInstance(); if don’t know class names at beginning

**->**dynamical provide class names, at runtime corresponding .class file is available then get ClassNotFoundException

**NoClassDefFoundError:**

Test t=new Test();//if know the class name at begging

Hardcoded class names , at runtime corresponding .class file is available then get NoClassDefFoundError

**Note**:- in a program if at list one method termination ub-normally then the program termination is ub-normall termination

* If all method termination is normally then only program termination is normal termination.

**Difference b/w Exception and Errors?**

* Throwble class contain two child classes 1. Exception 2,Error

|  |  |
| --- | --- |
| **Exception** | **Error** |
| 1. Most of time Exception caused by our program   Ex: FNF,CNF | 1.Most of the time error caused by our program  Due to lack of system resources.(our system sufficient heap area not available) |
| 2 Exceptions are coverable | 2.Errors are not coverable |

**.14 control flow in try catch finally?.**

* try
* {statement1;
* statement2;
* statement3;
* }
* catch(X e)
* {
* statement4:
* }
* finally
* {
* statement5;
* }
* statement6;

**case 1-> no Exception**

->1,2,3,5,6 normal termination

case 2-> if Exception raise statement 2 and corresponding catch block matched

-> 1,4,5,6 normal termination

case3:- if Exception raise statement 2 and corresponding catch block not matched

->1,5,-> abnormal termination

Case 4:- if Exception raised stmt4

5 abnormal termination

**15.Diffirence b/w checked Exception and un-checked Exception?**

|  |  |
| --- | --- |
| **Checked Exception** | **Un-checked Exception** |
| 1. which Exception are checked by compiler that Exception are Exception   Ex:FileNotFoundException,ClassNotFound | 1.which Exception are not-checked by compiler that Exception are Exception.  Ex.ArthmeticException ,NPE |
| 2.All IO Exceptions are checked Exceptions | 2 All Errors and Runtime Exceptions are Un checked Exceptions |

**16. difference b/w fully checked and Partial Checked Exceptions**

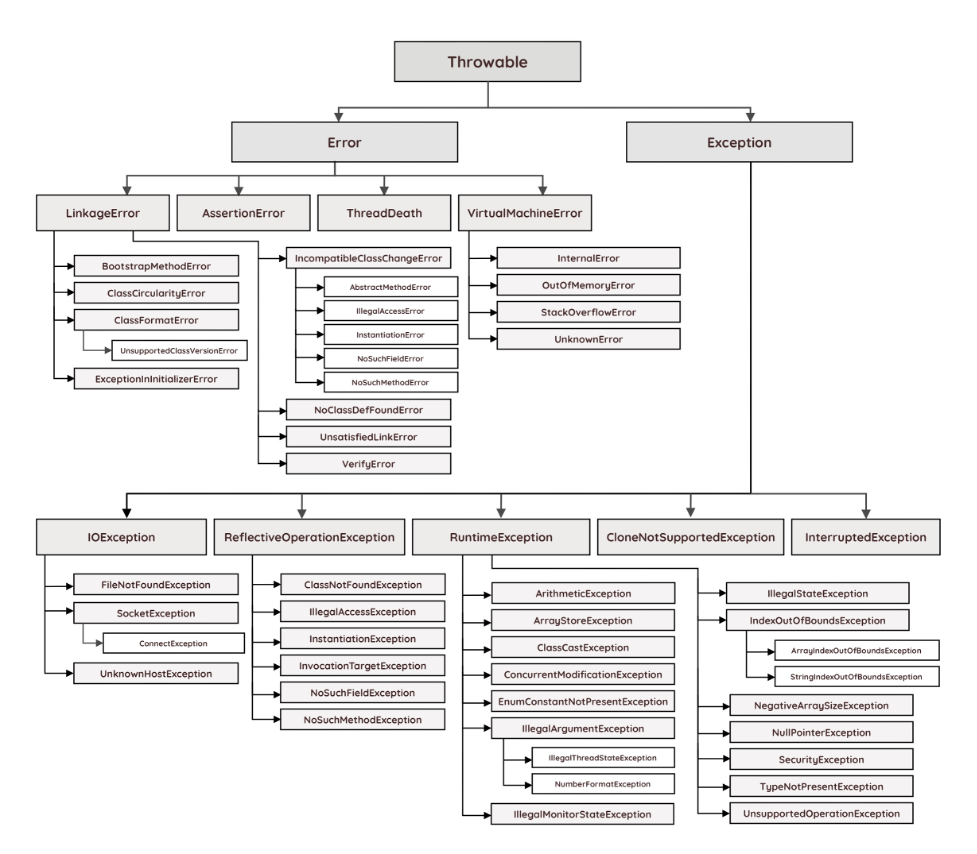
**->** a checked Exception is said to be fully checked exception if and only if all it is child classes also checked.

Ex:IoException

InterrptedEception

-> a checked Exception is said to be partial checked exception if and only if some of child classes are unchecked

Ex:Exception,Throwable.



**42.What is throws key word?**

-> the purpose of throws keyward is when ever using throws keyward handle the exceptions jvm is responsible handle the exceptions.

**43. What is throw key word?**

-> to hand-over our created Exception object to the jvm manual.

**46.What is thread?**

-> every java program at least one thread that is a main thread it is created by jvm

**47.What is multithreading?**

-> two threads are executing same java object simultaneously, that called multithreading.

Or

Process of executing multiple threads simultaneously, that called multithreading.

**48. How many ways to create a thread?**

There are two types created the thread.

1. Extend the Thread class
2. Implements Runnable Interface.

**49. What is thread scheduler?**

**->** it is a part of jvm it is responsible to schedule threads. If multiple threads are waiting to get the chanse of execution. Which order threads are executed designed by the thread scheduler only**.**

**50. Difference b/w t. start() and t.run() ?**

-> in the case of t.start() new thread will be created which is the responsible for execution is run method.

-> in the case of run() method new thread won’t be created and run() will executing normal method calling.

**51. Explain thread life cycle?**

New/born----🡪Ready-------🡪Running--------------🡪Dead

**52. Explain detail thread priority?**

-> every thread in java as some priority it may be default priority generated by jvm or customized priority provided by the programmer.

-> the valid range of thread priority is 1 to 10. 1 is min priority 10 is max priority 5 is Norm priority .

**53. Explain the yield() ,join() and sleep() methods?**

**Yield**():- when ever calling this method to pause the current executing thread to give the chanse for waiting thread of same priority.

**Join()-**if thread want to wait until completing some other thread then we should go for join().

**Sleep():**if a thread don’t want perform any operation for a particular amount of time.

**54. What is synchronization what is advantage synchronization?**

**->** synchronized is a modifier applicable only methods and blocks but not class and variables**.**

**->** two threads are executing same java object there may be chanse of data inconsistency problem is ocuur .to overcome these problem method declare as synchronized keyword, then allowed only at time.

**55. What is class level lock?**

Every class in java has a unique lock which is noting but class leval lock

**56.** **Explain the synchronization block?**

-> if very few lines of code required synchronization then it’s not recommended to declare entire method as synchronized. We have Enclosure those few lines of the code by using synchronized block.

-> the main advantage of synchronized block over synchronized method it’s reduce waiting time of thread and improves performance of application.

57. **What is deadlock? Explain ?**

-> if two threads are waiting for each other forever such infinite situation is called deadlock.

Synchronized keyword is only reason for deadlock situation.

**58. What is daemon thread?**

**->** the threads which are executing in background are called Deamon thread.

Ex: Garbage Collector.

**59. What is green thread?**

**->** the thread which managed completely by jvm without taking undelined os support is called green thread.

**60. What is array?**

**->** an array is an indexed collection of fixed number of homogenies data elements

-> the main advantage of array is we can represent multiple values by using single variable so that readability of code is improved.

**61. What is collection?**

-> if we want represent group of individual object as a single entity we should for the collection.

**62. Difference b/w array and collection?**

|  |  |
| --- | --- |
| **Array’s** | **Collection** |
| **1**.Arrays are fixed in size i.e is once we creates an array we can’t increase and decrease the size based on our requirement | **1.**collection are growable in nature i.e based on our requirement we can increase and the size |
| **2.**wrt to memory wise arrays are not recommended to use. | **2.**wrt memory collection are recommend to use |
| **3**.w.r t performance arrays are recommended to use. | **3**.w.r t performance collection are not recommended to use. |
| **4.**Array’s can hold homogenies datatype elements | **4.**collection canhold homogenies datatype and heterogeneous elements |
| **5.**Array’s can hold both primitives and objects | **5.**collection canhold only object types but not primitives |

**63.** **Difference b/w collection and collections?**

|  |  |
| --- | --- |
| **Collection** | **Collections** |
| 1. if we want represent group of individual object as a single entity we should for the collection. | 1. collections is an utility class present in java.util package to define serval utility methods for collection objects[like sorting, searching etc…] |

**64. What are the key interfaces of collection frame work?**

**->** 9 key interfaces of collection framework.

1.Collection 2.List 3.Set 4.SortedSet 5.NavigbleSet 6.Map 7.SoretedMap 7.NavigableMap 9.Queue

**65. Difference b/w list, set, map?**

|  |  |  |
| --- | --- | --- |
| **List(I)** | **Set(I)** | **Map(I)** |
| 1. It is a child interface of the collection   if we want represent group of individual objects as a where duplicates are allowed and insertion order must be preserved then we should go for List. | 1. It is a child interface of the collection   if we want represent group of individual objects as a where duplicates are not allowed and insertion order not preserved then we should go for Set. | 1. It is not a child interface of the collection   if we want represent group of individual objects as a where key and pair format required and duplicate keys are not allowed and value can duplicate. Then we should go for Map. |
| Implementation classes    ArrayList  LinkedList  Vector  Stack | Implementation classes  HashSet  LikedHashSet  SortedSet(i)  NavigableSet(i)  Treeset | Implementation classes  HashMap  LinkedHashMap  Hashtable  SortedMap(i)  NavigableMap(i)  Treemap |

**66.** **Difference b/w ArrayList vs. LinkedList**

|  |  |
| --- | --- |
| **ArrayList** | **LinkedList** |
| 1. It is best choose for frequent operation retrieval.   Because it implements RandomAcess Interface | 1. It is best choose for frequent operation is insertion in the middle.   Because it’s follow double linked list data Structure it creates node in the middle  Doesn’t implements RandomAcess Interface |
| 1. Underline data structure is resizable or growble Array | 2.Underline data structure is double linkedList |
| 1. Worst choose for insertion because lot of shifting operations required**.** |  |

**67**  **Difference b/w ArrayList vs Vector**

|  |  |
| --- | --- |
| **ArrayList** | **Vector** |
| 1 every method present in ArrayList is non-synchronized methods | 1 every method present in Vector synchronized methods |
| 2 At a time multiple threads are allowed to operate ArrayList object so it is not thread safe. | 2,At a time only one thread allowed to operate Vector Object so it is thread safe. |
| 3 performance is high ,because no waiting time | 3 performance is low because of waiting time. |

**68 What is cursor?**

* If we want to get objects one by one from the collection then we should go for cursor.

**69. Types of cursors?**

**->** there are 3 types of cursors available in java

1. Enumeration 2.Iterator 3.ListIterator

**70. Difference b/w each and every cursors?**

|  |  |  |
| --- | --- | --- |
| **Enumeration** | **Iterator** | **ListIterator** |
| 1 applicable only legacy class | 1 Applicable for any collection object | 1 applicable for only list objects |
| 2 single direction forward | 2 single direction forward | 2 bi directional cursor |
| 3 only read operation | 3 only read and remove operations get elements | read/remove/replace/add  operations |
| 4 methods  1.hasmoreElements()  2.nextElement() | Methods  1.hasNext()  2.next()  3.remove() | Methods  9 methods Available |

**71. Difference b/w Set and SortedSet?**

|  |  |
| --- | --- |
| **Set** | **SortedSet** |
| 1 Set is a child interface of Collection | 1.SortedSet is a child interface of Set. |
| 2.dublicates are not allowed, insertion order not preserved. | 2.dublicates are not allowed and sorting order required we should go for sortedSet. |

**72. Difference b/w HashSet and LinkedHashSet?**

|  |  |
| --- | --- |
| **HashSet** | **LinkedHashSet** |
| 1 underline data structure is Hashtable | 1. underline data structure is combination of linked list and Hashtable |
| 2.insertion order is not preserved | 2 insertion order is preserved. |

**73. Difference b/w HashSet and TreeSet?**

|  |  |
| --- | --- |
| **HashSet** | **TreeSet** |
| 1 underline data structure is Hashtable | 1. underline data structure is balanced tree |
| 2.insertion order is not preserved | 2 insertion order not applicable but all objects will be inserting sorting order |
| 3 same and different type of objects are allowed | 3 same type of objects allowed. |
| 4.null insertion is possible | 4.null insertion is not possible |

**Note:** Except TreeSet, TreeMap, Hashtable anywhere null acceptance.

* Except TreeSet, TreeMap same and different types of objects allowed**.**

**74. Difference b/w comparable and comparator?**

|  |  |
| --- | --- |
| **comparable** | **comparator** |
| 1 comparable meant for default sorting order | 1 comparator meant for customized sorting order |
| 2 comparable present in lang package | 2 comparator prasernt util package |
| 3 it contains only one method   1. compareTo() | 3 it contains two methods   1. compare() 2. equals() |

**75.What is Hashing?**

-> Hashing is the process of mapping the data to some represented integer value using hashing algorithm

-> in java a hashcode is an integer value that is linked with each object.

**76. What is entry?**

-> Each key value pair is called Entry.

**77.Diffirence b/w HashMap and LinkedHashMap**

|  |  |
| --- | --- |
| **HashMap** | **LinkedHashMap** |
| 1. Underline data structure is Hashtable | 1.underline data structure is combination of linked list+Hashtable |
| 1. Insertion order not preserved and it is based on hashcode of keys | 2 insertion order is preserved |
|  |  |

**78.Diffirence b/w HashMap and ConcurentHashMap**

|  |  |
| --- | --- |
| **HashMap** | **ConcurentHashMap** |
| HashMap is non-Synchronized in nature | HashMap is Synchronized in nature |
| HashMap is not Thread-safe | ConcurrentHashMap is Thread-safe in nature |
| Permanence is high | Performance is low |
| While one thread is Iterating the HashMap object, if other thread try to add/modify the contents of Object then we will get Run-time exception saying **ConcurrentModificationException** | Whereas In ConcurrentHashMap we wont get any exception while performing any modification at the time of Iteration |
| In HashMap, null values are allowed for key and values | ConcurrentHashMap null value is not allowed for key and value, otherwise we will get Run-time exception saying**NullPointerException.** |

**79.Diffirence b/w HashMap and Hashtable**

|  |  |
| --- | --- |
| **HashMap** | **Hashtable** |
| Every method present in HashMap is not synchronized. | Every method present in Hashtable is synchronized. |
| Not thread safe | Thread safe |
| Performance is high | Performance is low. |

**80. Difference b/w TreeSet and TreeMap**

|  |  |
| --- | --- |
| **TreeSet** | **TreeMap** |
| 1 it is child class of SortedSet interface | 1 it is child class of SortedMap interface |
| Collection required we should go for TreeSet | Where is key/pair format required we should go for TreeMap |

**81.Diffirence b/w Hashtable and ConcurentHashMap**

|  |  |
| --- | --- |
| **Hashtable** | **ConcurentHashMap** |
| Thread safe | Thread safe |
| It’s maintain total Hashtable object locked | It’s maintain bucket level lock   * Read operation lock not required |

**82. Difference b/w HashMap and IdentityHashMap**

|  |  |
| --- | --- |
| **HashMap** | **IdentityHashMap** |
| 1 child class of map interface | 1 child interface of Map |
| 2 HashMap jvm will use .equals() to identify the dublicate keys | 2 IdentityHashMap ==operator to identify the dublicate keys |

**83.How to get Synchronized version normal collection methods?**

Ex: ArrayList list=new ArrayList();// by default list is non-synchronized

List list2=Collctions.synchronizedList(list);//now list2 is synchronized

* Same call
* Collctions.synchronizedList(list)
* Collctions.synchronizedSet(set)
* Collctions.synchronizedMap(map)

**84. Detail about total internally HashMap works**

-> All objects are saved based on keys .

-> HashMap<k v> map=new HashMap<k v>()

-> when ever creating HashMap objects internal create Array of bucket size will be 16.

->here each bucket is one noded linkedList

Map.put(101,”srinu”);

* When ever calling put() internally calculate hashcode of key,
* Hashcode is a integer value that is a bigger value.
* Then calculate index value to store appariate index bucket like

|  |  |  |  |
| --- | --- | --- | --- |
| Hashcode | key | value | null |

* Key is null to store the 0th bucket (hashcode of null)
* When ever calling map.get(i); method to calculate key of hashcode.
* That hashcode matched with index hashcode then get the data.

**85.why override equals() and hashcode()?**

-> .equals() method by default reference comparison except String and all Wrapper classes overridden content comparison.

-> if we want content comparision we can override .equlas() method.

-> in hashing related Data Structure two equality object store same bucket,it return same hash code ,it return true. That’s why hashcode overridden**.**

**Note: hashing related data Structures they will use bucket concept. All objects are saved based on hashcode**

**86 What is ConcurrentModificationException**

While one thread is Iterating the HashMap object, if other thread try to add/modify the contents of Object then we will get Run-time exception saying **ConcurrentModificationException**

**87. Explain the java i/o package?**

**java I/O** (Input and Output) is used *to process the input* and *produce the output*.

Java uses the concept of a stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations.

We can perform **file handling in Java** by Java I/O API.

1. File
2. FileWriter
3. FileReader
4. BufferWritter
5. BufferReader
6. PrintWriter

**88. Explain the java. Lang package?**

**->** default package of java is lang package.it contains mostly commonly used class interfaces are defined in separate package which is nothing but lang package. 1. Object 2.String 3.StringBuffer 4.StringBudiler 5.Wrapper classes.

**OBJECTS CLASS METHODS**

* Object class contain mostly commonly required method for every java class
* Every class in java child classs of object either directly or indirectly so that object class method by defaulit available every java class.
* Java not support for multiple inheritance with respected classes becoz java only can extend one class.

**1.public String toString()**

🡪 we can use toString() method to get String representation an object

🡪String s=(Object).toString();

🡪when ever we are trying print object reference internally Object class toString() method will be called.

Student s=new Student();

Sop(s.toString());

🡪if our class doen,t contain toString() method object class toString() method will be executed

//classname@hascode in hexadecimal form like Test@254899

🡪over come above pblm to override own toString() method

Public String toString(){

retun Student[“rollno”+rollno + “name”+name]

}**IMPNOTE:- all wrapper classes toString() by default override like int,flot,String.**

**2.public native int hascode()**

🡪every object a unique number generated by jvm which is ntng but hascode.

🡪hascode won,t represent address of object

🡪jvm will use hascode while saveing object into hashing related data stracture like hashtable,hashMap,hashSet etc….

🡪the main advantage of saveing objects based on hashcode is serch operation will become easy(most powerful search algirtm upto today is hashing)

🡪if we are giving the chance to object class hascode methodit will generated hashcode based on address of the object

🡪based on our requirement we can override hashcode method in our class to generated our own hashcode .

🡪overriding hashcode method is said to be proper if and only if every object we can generate a unique number has hashcode

Class Student

{

---------------//this is inproper way of overriding hashcode() method becoz for all student objets we are generating same number has hascode

Public int hascode(){

return 100;

}Class Student

{

Public int hascode(){

Return rollno;}//it proper way overriding becoz we are generating a diffirent hascode for every object

toSring vs Hashode

🡪if we are giving the chance to object class toString() method it will internally call hascode() method

🡪if we are overriding toString then our toString() method may not call hascode method

**3.public Boolean equals(Object o)**

**🡪**we can use equls()method to check equlity of objects

Obj.equls(obj)

🡪if our class doesn,t contain equls() method then object class equls () method will executed

🡪if object class equls() method got executed which is meant for reference comparision(address comparision) i.e if two references pointing to the same object then only .equls() method returns true.

🡪based on our requirement we can override equls() method for content comparision

**How we can override .equls() method for content comparision?**

Public Boolean equls(Object obj){

We can override from ide}

🡪while overriding equls() method for content comparision we have to take care about following

🡪what is meaning of equlity (i.e wheather we have to check only name or rollno or both)

🡪if we passing diffirent type object our equls method should not rise classcastexception i.e we have handle callsscastexception returns false

🡪if we are passing null arg then our equls() method should npe:we have handle npe to return false.

**4.public final Class getClass()**

🡪can use getClass() method we if we want know the fullyqulified classnames,mehod information

🡪by using this class class object we can access class leval properties like fullyqulified name of the class method information constructer information etc

Ex:Object o=new String(“srinu”);

Class c=o.getClass();

Method[] m=c.getDeclareMethods();

**5.protected native Object clone()**

🡪the process of creating exact dublicate object is called cloneing.

🡪purpose of cloneing is to maintain backup copy and preserve state of object

🡪by ysing clone() method to achive the cloneing

**6.protected void finalize()**

-🡪just before destroying the object garbagecollecter call finlize ()method to perform cleanup activities once finalize() method complete automatically gc destroy that object

**7.wait()**

**8.wait(long ms)**

**9.wait(long ms,int ns)**

**10.notify()**

**11.notifyAll()**

**🡪**we can use these method for interthread communication

🡪 the thread which is exepating updation, it responsible to call wait() method then immediately the thread will enter into waiting state.

* The thread which is responsible for updation, after performeing updation the thread can call notify()
* The waiting thread will get that notification continue its execution with those updates

**89.What is String class what are methods of string class?**

The String class **represents character strings**. All string literals in Java programs, such as "abc" , are implemented as instances of this class. Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because String objects are immutable they can be shared.

**What is mutability and immutability**?

|  |  |
| --- | --- |
| **String** | **String Buffer** |
| 1.String Objects are Immutable | 1.StringBuffer Objects Mutable |
| 1. Immutable means Once create a String Object we can’t perform any changes in the existing Object. If we are Trying to perform any Changes, New Object will be created. This non changeable behaver noting but Immutability. | Once We Create a StringBuffer Object we can perform any type of changes in the existing object. This changeable noting but mutability |

**Difference B/w StrinBuffer and StringBuilder?**

|  |  |
| --- | --- |
| **StrinBuffer** | **StringBuilder** |
| 1.every method present in StringBuffer is synchronized | 1.every method present in StringBuilder is non-synchronized |
| 2.thread safe  At a time Only one thread to allow to operate. | 2. not thread-safe  At a time, multiple threads to allow to operate. |
| 3.increase waiting time. performance is low. | 3.threads not required to operate.no waiting time. performance is high. |

**String vs StringBuffer ns StringBuilder?**

->if content is fixed and won’t change frequently we should go for String.

->If content is not fixed and keep on changing but thread safe is required we should for StringBuffer.

->>If content is not fixed and keep on changing but thread safe is not required we should for

StringBuidder**.**

**Sting Methods**

**public** **class** StringExamples {

**public** **static** **void** main(String[] args) {

String orginalString = "sreenivasarao";

CharSequence str1 = orginalString.subSequence(0, 4);

String str2 = orginalString.substring(3);

String str3 = orginalString.substring(2, 6);

String str4 = orginalString.concat("vkp");

String str5 = orginalString.intern();

String str6 = orginalString.replace('s', '$');

String str7 = orginalString.replace("arao", "$");

String str8 = orginalString.replaceAll("sreeni", "srinivasaraok");

String str9 = orginalString.replaceFirst("o", "f");

String str10 = orginalString.toLowerCase();

String str11 = orginalString.toString();

String str12 = orginalString.toUpperCase();

String str13 = orginalString.trim();

**boolean** b = **true**;

Character ch = 'a';

**int** i = 55;

**double** d = 55;

**float** f = 25;

**char**[] ch2 = { 'A', 'B' };

String str14 = orginalString.*valueOf*(ch2);

**char** str15 = orginalString.charAt(1);

**boolean** str16 = orginalString.contains("sreenivasarao");

**boolean** str17 = orginalString.equalsIgnoreCase("SREENIVASaRAO");

**byte**[] get = orginalString.getBytes();

**int** length = orginalString.length();

String[] split = orginalString.split(",");

**for** (String string : split) {

System.***out***.println(string);// sr,ee,ni,va,sa,rao to print one by one

}

String[] split2 = orginalString.split("#");

**for** (String str21 : split2) {

System.***out***.println(str21);

}

**boolean** new1 = orginalString.startsWith("sree");// true

**char**[] ch1 = orginalString.toCharArray();// to convert string to char[]

**boolean** match = orginalString.matches("sreenivasarao");}

**Wrapper classes**

The wrapper classes in Java are used to convert primitive types ( int , char , float , etc) into corresponding objects. Each of the 8 primitive types has corresponding wrapper classes.

## **Use of Wrapper classes in Java**

Java is an object-oriented programming language, so we need to deal with objects many times like in Collections, Serialization, Synchronization, etc. Let us see the different scenarios, where we need to use the wrapper classes.

* **Change the value in Method:** Java supports only call by value. So, if we pass a primitive value, it will not change the original value. But, if we convert the primitive value in an object, it will change the original value.
* **Serialization:** We need to convert the objects into streams to perform the serialization. If we have a primitive value, we can convert it in objects through the wrapper classes.
* **Synchronization:** Java synchronization works with objects in Multithreading.
* **java.util package:** The java.util package provides the utility classes to deal with objects.
* **Collection Framework:** Java collection framework works with objects only. All classes of the collection framework (ArrayList, LinkedList, Vector, HashSet, LinkedHashSet, TreeSet, PriorityQueue, ArrayDeque, etc.) deal with objects only.

The eight classes of the java.lang package are known as wrapper classes in Java. The list of eight wrapper classes are given below

**Primitive Data types and their Corresponding Wrapper class**



**91. What is method chaining?**

Method Chaining is **the practice of calling different methods in a single line instead of calling other methods with the same object reference separately**. Under this procedure, we have to write the object reference once and then call the methods by separating them with a (dot.).

**92. Explain auto boxing and un-boxing?**

## **Autoboxing**

The automatic conversion of primitive data type into its corresponding wrapper class is known as autoboxing, for example, byte to Byte, char to Character, int to Integer, long to Long, float to Float, boolean to Boolean, double to Double, and short to Short.

Since Java 5, we do not need to use the valueOf() method of wrapper classes to convert the primitive into objects.

## **Unboxing**

The automatic conversion of wrapper type into its corresponding primitive type is known as unboxing. It is the reverse process of autoboxing. Since Java 5, we do not need to use the intValue() method of wrapper classes to convert the wrapper type into primitives

**public** **class** WrapperExample{

**public** **static** **void** main(String args[]){

**byte** b=10;

**short** s=20;

**int** i=30;

**long** l=40;

**float** f=50.0F;

**double** d=60.0D;

**char** c='a';

**boolean** b2=**true**;

//Autoboxing: Converting primitives into objects

Byte byteobj=b;

Short shortobj=s;

Integer intobj=i;

Long longobj=l;

Float floatobj=f;

Double doubleobj=d;

Character charobj=c;

Boolean boolobj=b2;

//Printing objects

System.out.println("---Printing object values---");

System.out.println("Byte object: "+byteobj);

System.out.println("Short object: "+shortobj);

System.out.println("Integer object: "+intobj);

System.out.println("Long object: "+longobj);

System.out.println("Float object: "+floatobj);

System.out.println("Double object: "+doubleobj);

System.out.println("Character object: "+charobj);

System.out.println("Boolean object: "+boolobj);

//Unboxing: Converting Objects to Primitives

**byte** bytevalue=byteobj;

**short** shortvalue=shortobj;

**int** intvalue=intobj;

**long** longvalue=longobj;

**float** floatvalue=floatobj;

**double** doublevalue=doubleobj;

**char** charvalue=charobj;

**boolean** boolvalue=boolobj;

//Printing primitives

System.out.println("---Printing primitive values---");

System.out.println("byte value: "+bytevalue);

System.out.println("short value: "+shortvalue);

System.out.println("int value: "+intvalue);

System.out.println("long value: "+longvalue);

System.out.println("float value: "+floatvalue);

System.out.println("double value: "+doublevalue);

System.out.println("char value: "+charvalue);

System.out.println("boolean value: "+boolvalue);

}}

1. **What is serialization ?**

* The process of converting object supported form file supported form or network supported form that is called serialization.

1. **What is de-serialization**?

The process of converting file supported form or network supported form

to object supported form that is called de-serialization

1. **What is externalization**

 Externalization in Java is **used whenever you need to customize the serialization mechanism**. If a class implements an Externalizable interface, then serialization of the object will be done using the method writeExternal().

**96. Difference b/w serialization and externalization?**

|  |  |  |
| --- | --- | --- |
| **Key** | **Serialization** | **Externalization** |
| Interface | It is a marker interface | Whereas, it contains two methods readExternal() and writeExternal() |
| Process | Default serialization | Custom serialization |
| UID | serialVersionUID | No UID |
| Storage | Stores the data that have objects | Directly stores objects |
| Access | No access | Complete control to customize the serialization |
| Performance | It provides relatively slow performance. | It provides full control over the implementation approach |

**99. Explain serial version UID?**

**->** At time of serialization jvm will save unique id of every object based on .class file. Then at time de-serialization receiver side jvm compare the local .class unique id with file .class, if both are matched the de-serialized this serial version UID**.**

**100. Main() method**

-> main() method is the first point of the program execution.

-> it should be active at the time of class loading**.**

**101. can we execute main() method at the time of class Loading?**

**->** yes, we can executed by calling it from static block

**102.why spring is immutable?**

**Because of the scp concept String is Immutable.**

* Reason, same objects can be reused multiple times by using one reference, if we are changing the content reaming reference are effected. That’s why String objects immutable.

**103. Advantages of scp(String constant pool)**

* Same objects can be reused multiple times instead of creating new object.
* Memory will be saved
* Performance will improved.

**104.why scp concept is available only for String objects but not for StringBuffer?**

* String is Most commonly used java objects that’s why scp concept is available in String
* StringBuffer may or May Not be use that’s why scp concept is not available in StringBuffer.

**104.why spring objects are immutable where as SpringBoot objects are Mutable?**

**->** String objects scp concept is available reusing same object .

-> multiple reference are same object if we are trying to change reaming reference are effected that’s why String is Immutabilty.

-> StringBuffer there is no scp concept every time new object will be created that’s why SB is mutable.

-> All wrapper objects are immutable by default

**Access Specifiers and Access Modifiers?**

Ex: private class Test {

{

}//modifier private not allowed to here compile error.

Only applicable modifiers top-level classes public, default,final,abstract,stctfp.

**9.Explain about Sytem.out.println();**

Class System{

Static printStream out;

}

**System.out.println(“hello”);**

* System is a class present in java.lang package.
* Out is static variable present in System in class of type PrintStream.
* Println() is a method present in PrintStream class.

**Note:- the main purpose of System.out.println(“hello”) print the data in console.**

**Explain about public static main (String[]args)**

Class Test

{

}//compile the code.

* If run the above program to get NoSuchMethodError. Because there is no main method.
* Then jvm will check main method
* Jvm will check prototype main method should be public static void main(String[] args).

**public static void main(String[] args)**

**why? Main method is above?**

**public:-** to call jvm from any ware

**static:-** without existing object also jvm has to call this method and main no way related any object.

**void:-** main method won’t return anything to jvm.

**main:-**this the name which is configure inside a jvm

**String[] args:-**This is command line arguments

**Note:**main method syntax is very strict if we perform any change won’t get any compile time error but to get Runtime exception NoSuchMethodError

**Acceptable changes in Method**

* the order of modifiers is not important that is instead of public static we can take static public also**.**

**->** we can declare String[] in any acceptable form.

Main(String[]args)

Main(String []args)

Main(String args[])

* instead of “args” we can take any java idetityfier.

->instead String[] we can take var arg parameter.

Main(String… args)//valid

->we can declare main method with following modifiers

Final,Synchrnized,strictfp.

->inheritance concept is applicable main method.

->overloading concept applicable main method.

**->**overridingconcept not applicable main method.i.e method hiding because of static kwd.

**12. Difference b/w C c=new C() and P p=new C()**

|  |  |
| --- | --- |
| **C c=new C()** | **P p=new C()** |
| 1. if we are know exact runtime object then we should go for this Approach | 1. if we are don’t know exact runtime object then we should go for this Approach |

# **Generics**

The main purpose of Generics is Type Safety and Type casting.

-> type safety mains adding the if not generic add the any type of data.in generics add any data.

# **Differences between Normal ArrayList and Generic ArrayList**

|  |  |
| --- | --- |
| **ArrayList a=new ArrayList()** | **ArrayList<String> a=new ArrayList<String>()** |
| 1.it is not generic type | 1.it is generic type |
| 2.Type safety not there add any type of data | 2.Type safety there add only String of data |
| 3.there is maybe chance of chance of typecasting problems comes | 3.no type casting problems occurs. |

# **18.What is difference between new operator and newInstance() method**

Both are using create object only but below difference.

|  |  |
| --- | --- |
| **new operator** | **newInstance()** |
| 1.if we know the classname at beginning create object by using new operator | 1. if we don’t know the classname at beginning create object by using newInstance() |
| 2 Test t=new Test(); | Object o= Class.forName(args[0].newInstance(); |

# **Jar vs war vs ear**

**Jar:(java Achieve)**  jar is contain a group of .class files.

**War:(Web Achieve):** war file represents one web application.

**Ear(Enterprise** **Achieve):** ear file represents enterprise Application but not web application.

# **20. What is the difference between JDK,JRE and JVM**

**JDK:(JAVA DEVLOPMENT KIST):-** JDK PROVIDE ENVIRAMENT TO DEVELOP AND RUN APPLICATION

**JRE:(JAVA RUNTIME ENIVIRALMENT)-** JRE PROVIDE ENVIRAMENT JUST RUN THE JAVA APPLICATION

**JVM:(JAVA VIRTUAL MACHINE)-** JVM IS RESPONSEBLE RUN JAVA PROGRAM LINE BY LINE

**NOTE:-**jvm is a part of jre, jre is a part of jdk.

# **20. Difference between path and class path**

**Path**: set location of the java path, for example not set the java path unable to work javac command**.**

**Classpath:** class path describes the location of where required .class files are available. If we are not setting class path may not compile and run the java application.

# **21. Web Server vs Application Server**

**Web Server:** web server provide environment to run web application.

**Application Server:** Application Serverprovide environment to run enterprise application. Like j2ee.application server can provide support for any technology from java j2ee.

* Application server internal contain webserver.

**HIBERNATE**

1. **What is framework?**

Framework is a set of jar files. each jar file contain set of predefined classes, its provide some functionality**.**

1. **What is hibernate?**

Hibernate is a one of the open source orm tool.used for the Exchanging object b/w java Application and Database**.**

**3.What are the features of hibernate?**

The main features of Hibernate is

1.Hql 2.Caching 3.Lazyloading 4.locking 5.Filters 6.Criteria

**4.Explain each features of hibernate?**

**HQL:-** Hql queries database independent queries.no need to change database to database.

Hql quires tuned queries means object oriented quires.

**Caching:-**Hibernate provide two leval chace. First level cache is session cache.first level cache alive until session is closed**.** The second level cache exists as long as the session factory is alive.

**Lazy loading:-** The main purpose of lazy loading is to fetch the needed objects from the database.to achieve the lazy loading using load() method its create proxy object then load the data.

**5.What are the supported collections of hibernate?**

1.List 2.Set 3.Map

**6**.**How to write hibernate mapping and configuration ?**

Student.hbm.xml Mapping file

<hibernate-mapping>

<class name= “Student” Table= “Student”>

<id name= “studentId” column= ”id”/>

<id name= “studentname” column= ”sname”/>

</class>

</ hibernate-mapping>

Configiration File

<hibernate-configration>

<session-factory>

<property-name= “connection.driverclass>

OracleJdbcDriver.OracleDriver</property>

url=””

username=””

password=”

</hibernate-configuration>

**7. How to create database tables automatically?**

Add the application.propertyfile hbm2ddl.auto=create,update

**8. How to read the object from database using hibernate?**

Java application can call load() and get() methods.

**9. Difference b/w load() and get() ?**

**Load():-** load() method is lazy loading because it create proxy object then load the data. If the given id does,t exist in a database then load() returns ONFException**.**

**Get():-**get method is early loading If the given id does,t exist in a database it return null.

**10. What are the life cycle states of hibernate?**

There are 3 life cycle states in hibernate

1.Trasient State 2.Persistent State 3.Destached State

**11.Difference b/w update() and merge() ?**

Using these method Destached State object to persistence state object

**Update():-** same object already available in cache then update() method is fail.

**Merge():-** but merge() method works only it copy the change.

**12.What are the generated values of hibernate?**

1.assigned 2.increment 3.sequence 4.identity hillo

**13.Why should I make session factory object as singleton?**

session factory object is a heavy weight object because it holds connection setting mapping settings and all mapping files.every time object creation is performance wise not required so to make the session factory is singleton.

**14.How to make session factory object as singleton?**

Take private constructer**.**

**15.** **Difference b/w save() and persist() ?**

**Save():-**the return type of save method is serializable. Save() method can be use across the transaction. Save() method takes more time to execute.

**Persist():-**the return type of persist() method is void. persist() method can be use within the transaction and it takes less time to execute.

**16.Difference b/w save() and saveOrUpdate()**

**Save():-** save() method store the object in to the database

**SaveOrUpdate():-** this method based on our requirement do the object also.

**17.What are relationships of hibernate explain each?**

1.One-to-One

2.One-to-Many

3.Many-to-One

4.Many-to-Many

**18.What are joins statements of hibernate?**

1.inner join

2.left outer join

3.right outer join

4.fully join

19.**Hibernate session factory is thread safe?**

**Yes**

**20.Hibernate session is thread safe?**

**No**

**21. diffirence b/w Sql and Hql?**

|  |  |
| --- | --- |
| **SQL** | **HQL** |
| 1 Stracture Query language | 1 hibernate Qurey language |
| 2 it is related to database | 2 it is related to hibernate |
| 3 database dependent | 3 database independent |
| 4 table quires  Select sname from student07 | Hql quires object oriented quireies  Ex select name fro student |

**22.What are annotations of hibernate ?**

* **@Entity :**specifies that the class is an entity.
* To be able to store Point objects in the database using JPA we need to define *an entity class*
* **@Table :** It is used to provide the name of the table.
* **@Column :**it is used to specify a mapped column for the field.
* **@Id :**defines the primary key field of the entity.

**@GeneratedValue :**allows the auto generation of the field’s .no need to set the id

**@Inheretance**

**@Emadable**

**@Embedded**

**@Trasient**

**@DiscriminatorColumn**

**@OneToOne**

**@OneT0Many**

**@ManyToOne**

**@ManyToMany**

**@JoinColumn**

**@JoinTable**

**SPRING**

1. **Why spring frame work?** 
   * Spring is modular framework.
   * Spring is a light-weight framework.it means we use the pojo classes to develop the application
   * Spring is non-invasive framework it means a program no need to extend class or interface.
   * Spring is provide loose coupling b/w objects through dependency injection.
   * Spring provide own container called Ioc.

**2.what are the modules of spring framework?**

**->** Spring core, spring aop, spring mvc, spring jee, spring Dao, spring test

**3.What is tightly coupling?**

If the changes done in dependency class are forcing do the changes in dependent class both classes are tightly coupled.

Public class Travel public class Car

{ {

Private Car c=new Car(); public void go()//move()

Public void Journey() {

{ ===

Car.go();//move(); } }

} }

**4**.**What is loosely coupling?**

If the changes done in dependency class don,t changing in dependent class.

If poji/pojo model is follow then they method names become fixed in pojo classes.

1. **Explain pojo , JavaBean, spring bean?**

**Pojo:-**pojo class is a ordinary java class which does,t exceed the boundary of java api**.**

**JavaBean:-**javabean also a java class with some policies

1.class must be public

2.which private property must contain either setter or getter or both methods**.**

**SpringBean:-**every class that developed as a part of a spring application is called springbean**.**

1. **What is Dependency?**

In spring framework every variable created in a class is called dependency.

1. **Types of Dependency Injections?**

1.setter Injection

2.constructer injection

3.lookup method injection

4.interface injection

1. **What are the types of dependcies?**

1.premitive type or value type

2.Referencetype or object type

3.Collection type

1. **How to create configuration in spring?**

Spring configuration

<beans>

<bean id=”id1” class=”fully qualified class name”>

-- -- -------

</bean>

<bean id=”id2” class=”fully qualified class name”>

* -- -- -------

</bean>

</beans>

1. **How to configuring setter injection (primitive and object types)?**

**Primitive or value type**

**================**

Public class A

{

Private int number;

Public void setNumber(int number)

{

this.number=number;}

=== == }

Xml

<beans>

<bean id=”id1” class=”A”>

<property name=”number” value=”10”/>

</bean>

</beans>

**Setter injection Object Type**

Public class A{

Private Test test;

Public void setTest(Test test){

this.test=test;

}}

**Xml**

<beans>

<bean id=”id1” class=”A”>

<property name=”test” ref=”id2”>

</bean>

<bean id=”id2” class=”B”>

<property name=”test2” ref=”id3”>

</bean>

</beans>

**11.What are the attributes of property tag?**

1.Name

2.value

3 ref

Name is mandatory value or ref only one can use at time.

**12.Explain constructer injection?**

Public class A{ public class Test{

Private int x; private int id;

Public void intX(int x){ private String name

this.x=x;}} public Test(int id, String name){

**xml** this.id=id;

<beans> this.name=name;

<bean id=”id1” class=”A> } }

<constructer-arg value=”100”/>

</beans>

**13.What are the attributes of constructer-tag?**

1. value

2.ref

3.type

4.index

**Type** is a option attribute

**Index** specify the order.

**14.What is container?**

In java we call a class which provides runtime support for other classes as a container.

**15. What are the types of containers in spring?**

1.BeanFactory Container

2.ApplicationContext Container

**16.How to start BeanFactory container?**

Beanfactory is an interface these implementation class is XmlBeanFactory**.**

* + To create XmlBeanFactory we need to pass Resource object as a parameter.
  + Resource is an interface it has 2 implantation classes.

1.ClasspathResource

2.FileSystemResource

Ex. Resource res=classpathResource(“spring.xml”);

BeanFactory factory=new XmlBeanFactory(res);

1. **What is spring ioc container?**

Spring IoC Container is **the core of Spring Framework**. It creates the objects, configures and assembles their dependencies, manages their entire life cycle.

**18.What is dependency Injection?**

The fundamental functionality provide by the spring container is called dependency injection. this container inject the one object to another object**.**

**19.Difference b/w class path resource and file system resource?**

Both Are implementation classes of Resources interface.

* + classpath Resource object is created when spring xml file at the classpath location.
  + Spring xml file is not at classpath then filesystem object is created.

**20. How to start Application context container?**

**ApplicationCotext interface has 3 implenation classes.**

1.ClasspathXmlApplicationContext

If java class in which we are starting container spring configuration file both are at one location then we created.

2.FileSystemXmlApplicationContext

If there exist diff location then we create FileSystemXmlApplicationContext object.

3.XmlWebApplicationContext

Using mvc module

**21.What are the collection types dependcies?**

Spring support 4 types of collection dependencies

1.List 2.Set 3.Map 4.Properties

**22.What is the circular dependency?**

Suppose A and B are 2 beans where A depends on B , B depends on A and in both constructer injection is defined then we will get circular dependency problem**.**

**23.Difference b/w setter and constructer injections?**

**Setter:-**setter injection makes bean class object as mutable (we can change)

**Slove the circular dependency problem**

**Constructer**:- constructer injection makes bean class object as immutable(we can’t change)

Don’t slove the circular dependency problem**.**

**24.Purpose of c-name and p-name util-name spaces?**

To avoid property tag and reduce the configuration.

<p-namespace avoid <property>tag

<c-name space avoid >construction-org> tag.

**25.How do you Intenerated a spring container with multiple configuration files?**

**There is two ways to intenerated with multiple configuration files**

**1.ImportResource**

**Applicationcontest3.xml**

<beans>

<import resource=”applicationcontest1,xml”/>

<import resource=”applicationcontest2,xml”/>

</beans>

ApplicationContext ctx=new ApplicationContext(“applicationcontest3.xml”);

**2.String[]**

String[] configfile={“app1,xml,app2,xml”};

ApplicationContest ctx=new ClasspathXmlApplicationContest(configfile);

**26.What is purpose of bean autowireing?**

The purpose is no need write setter injection explicitly. It inject automatically.

**27.Explain about byname,bytype,constructer?**

These are attributes of autowireing..

**byName**:- spring container looks bean name same as property name.then inject.

**byType**:- spring container looks bean name same as class name then injected.

**Constructer: -** In Spring, “Auto wiring by Constructor” is actually **auto wiring by Type in constructor argument**. It means, if data type of a bean is same as the data type of other bean constructor argument, auto wire it

**28**.**Explain default autowireing?**

If multiple beans in spring configuration are following a common auto wiring strategy then instated of adding auto wire attribute each <bean> tag we can apply <default-auto wire> in <beans> tag.

<beans default-autowire=”byname”>

<bean id=”id1” class=”A”/>

<bean id=”id2” class=”B”/>

<bean id=”id3” class=”C”/>

</beans>

Here A,B,C beans follow same auto wire strategy.

**29.What the bean scopes?**

**1.Singleton**:- By default is singleton it allows to create only one instance.

**2.prototype:-**new instance every time is created.

**3.Request:-**single bean instance per Http request

**4.session:-** single bean instance per Http Session

**5.global session:-** single bean instance per global http session.

**SPRING ANNOTATIONS**

🡪 Spring contain 3 types of annotations

1.Streotype Annotations

2.Autowired Annotations

3.Misslenious Annotations

**1.Streotype Annotations**

**🡪**Streotype Annotations are 4 types

[1.@Controller](mailto:1.@Controller) :- it indicates the this controller class.

[2.@Service:-](mailto:2.@Service:-) it indicates the this service class

[**3.@Repository**](mailto:3.@Repository) :- it indicates the this dto clas

4.**@Component**:- purpose is this Annotation is spring automatically detect our components.it means no need write any component scan code [explicitly @controller](mailto:explicitly.@controller) , @service , @repository also same becoz all are inherited from @component only

**2.Autowired Annotations**

[1.@Autowired](mailto:1.@Autowired)

🡪if we are adding @Autowired annotation on filed no need to write setter injection at runtime.

When we write @Autowired on top the filed the container will search Bean id matching with property name ,if found injects.

[2.@Qulifier](mailto:2.@Qulifier)

🡪Some times ambiguity occurs when same depency bean class is configured

More than once

**3.Misslenious Annotations**

[**1.@Value**](mailto:1.@Value)

🡪this annaotation used for injecting value of value type dependency.

[**2.@Scope**](mailto:2.@Scope)

🡪in spring framework by default scope is singleton then any changes done through this annotation

[**3.@postConstruct&&@preDestroy**](mailto:3.@postConstruct&&@preDestroy)

🡪it is the custominit and custom destroy methods

[**4.@Configuration**](mailto:4.@Configuration)

🡪using these annaotation to make the class as configuration class

[**5.@ComponentScan**](mailto:5.@ComponentScan)

🡪using these annotation to scan the components

[**6.@Bean**](mailto:6.@Bean)

🡪it is method leval annotation in configuration class we define methods.

to create object expecitlly.

[**7.@ImportResorce**](mailto:7.@ImportResorce)

🡪this annotation used for import configuration file

[**8.@PropertySorce**](mailto:8.@PropertySorce)

🡪these annotation load the properties file(application.property)

[**9.@Primary:-**](mailto:9.@Primary:-) purpose of @primary annotation is to give higher preference to a bean. When

there are multiple beans of same type

[**10.@Required:**-](mailto:10.@Required:-) this annotation Aplies to bean property setter methods.it indicates that affected

bean property must be populated in xml configuration at configuration time.other wise we get

Exception.

**Diffirence b/w @qulifier and @primary Annotations**

**@Qulifier** Annotation only avoid the ambiguity problem only.

**@primary** Annotation give the higher preference of two beans type is same**.**

These annotations we can declare class level and dependency level and method level also.

If class level @component is required.

If method level @Bean is Required**.**

**SPRING MVC ANNOTATIONS**

**1.@RequestMapping**

**🡪**This annotation maps HTTP requests to handler methods of MVC and REST controllers.

**2. @PathVariable**:

🡪to pass the data in the uri.

@RequestMapping(value=”add/test/{id},method=post)

Public void add(@PathVariable(“id”) int id){

}

3. **@RequestParam**

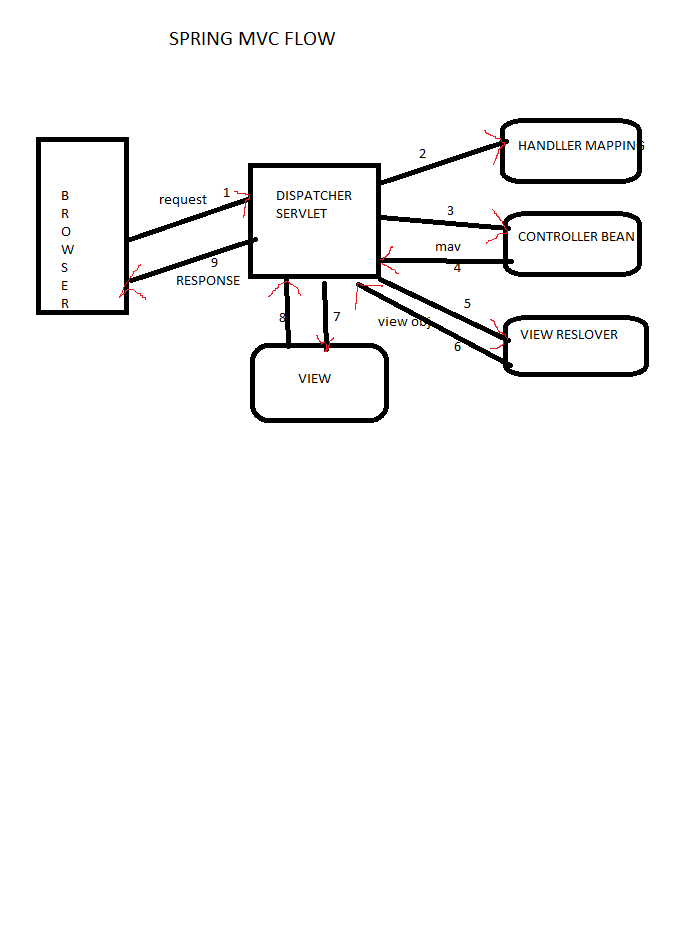
🡪@**RequestParam annotation** is used to bind the values of a query string.

-🡪**Diifrence b/w @PathVariable and @RequestParm**

🡪both are the bind the values only but small deffirence is url request.

Pathvarible---http://localhost:8080/myfoos/foos/abc

RequestParam---http://localhost:8080/foos?id=abfffggfgdffvdggg

****

* + First Request send to Browser to Dispatcher Servlet
  + Dispatcher servlet is the front controller of mvc Application.
  + Dispatcher servlet take the help of Handler Mapping.
  + It identifies Appropriate controller bean for given request.
  + Dispatcher Servlet call Controller Bean it returns model and view object it contains view name and model data.
  + Dispatcher servlet call View Resolver, It returns appropriate view Object.
  + Dispatcher Servlet call render() method of View object to response to Dispatcher Servlet.
  + Finally Dispatcher Servlet will send the response to client.

**SPRINGBOOT**

**1.What is spring boot?**

Spring boot makes it easy to create standalone, production-grade spring based Applications. that you can “just run”.

**2.What is difference b/w spring and spring boot?**

**Spring:**

* + More boilerplate code.
  + Lot of xml configuration
  + More decencies
  + Developing time is more
  + It doesn’t provide any embedded server.

**Spring Boot:**

* + Reduce boilerplate code
  + Only annotation
  + Less dependencies.
  + It’s provide embedded server

**3.Advantages of Spring Boot Applications?**

-> it is easy to develop spring based applications with java or groovy.

-> it reduces lot of development time.

-> Avoid lot of boilerplate code.

-> It is easy to integrate with spring jdbc, spring orm,spring security.

-> It provide Embedded servers like Tomcat, jetty etc..

-> it easy used Build tools like maven and gradle.

-> to Avoid writing lot of import statements.

-> to avoid xml configuration completely.

-> the main of spring boot framework is to reduce development, unit testing and integration**.**

**4.Draw back of Spring boot applications?**

-> Spring boot may unnecessary increase the deployment binary size with unused decencies.

->if we convert your old spring application to Spring boot application and can be time consuming.

**5.Why we need spring boot?**

Spring boot aims to simply the spring application development.

**6.What are the spring boot components ?**

Spring boot components

1.Spring Stater

2.spring AutoConfigurator

3.CLI

4.Actuator

**6.What is the spring boot starter?**

Stater is one dependency only that dependency is internally pointed to Bundle of decencies.

->whenever declare Stater in pom.xml file what are available decencies to download automatically at the time build application.

Ex:

Spring-boot-Stater-Web

Spring-boot-Stater-Jdbc

Spring-boot-Stater-Security

Spring-boot-Stater-JPA

Here Spring-boot-Stater is preemption name like web is module name

Note: if we want custom starter is required like this

Module Name-Spring-boot-Stater

**Pom.xml**

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.5.2</version> <!-- spring boot version number -->

<relativePath/> <!-- lookup parent from repository -->

</parent>

**7. What is spring boot auto-configurator?**

Auto configuration means it is a process of configuring setup for our application.

->auto configuration every thing take care all configuration things no need write explicitly code.

Ex: How to connect with database

How to connect with servers

How to configure your web modules.

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-autoconfigure</artifactId>

<version>2.6.6</version>

</dependency>

**8.What is spring boot CLI?**

This component requires for the Groovy developers.

* + Spring boot Application develop 3 different languages
    1. Java
    2. Catlin
    3. Groovy

**9.What is spring boot actuator?**

Actuator noting but to provide redmate production support for our Application.

* + Production support means once Application go to live Monitoring every thing through Actuator only
* Like health monitoring , load monitoring And All production Activities. Memories

endpoints etc**.**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

**10.What is spring boot Initializer?**

Spring boot initializer is a spring boot tool used for easy to create Spring Boot Projects.

 With the help of Spring Initializer, we can easily **generate the structure of the Spring Boot Project**. It offers extensible API for creating JVM-based projects.

**11.What and why embedded servers?**

No need to install server.

No need to deploy the war file**.**

**12.How can you run SBA on custom port?**

Write aplication.propeties file

Server. Port=8080

13.**How does Spring Boot works?**

Spring Boot automatically configures your application based on the dependencies you have added to the project by using annotation. The entry point of the spring boot application is the class that contains @SpringBootApplication annotation and the main method.

Spring Boot automatically scans all the components included in the project by using @ComponentScan annotation.

**14.What is Spring Boot dependency management?**

Spring Boot dependency management is used to manage dependencies and configuration automatically without you specifying the version for any of that dependencies

**15. What does the @SpringBootApplication annotation do internally?**

The @SpringBootApplication annotation is equivalent to using @Configuration, @EnableAutoConfiguration, and @ComponentScan with their default attributes. Spring Boot enables the developer to use a single annotation instead of using multiple. But, as we know, Spring provided loosely coupled features that we can use for each annotation as per our project needs.

**16.Can we create a non-web application in Spring Boot?**

Yes, we can create a non-web application by removing the web dependencies from the classpath. along with changing the way Spring Boot creates the application context.

**17.Is it possible to change the port of the embedded Tomcat server in Spring Boot?**

Yes, it is possible. By using the **server.port** in the **application.properties**

### 18. Can we override or replace the Embedded tomcat server in Spring Boot?

Yes, we can replace the Embedded Tomcat server with any server by using the Starter dependency in the **pom.xml** file. Like you can use spring-boot-starter-jetty as a dependency for using a jetty server in your project.

### 19. Can we disable the default web server in the Spring boot application?

**Yes, we can use application.properties to configure the web application type i.e spring.main.web-application-type=none.**

### 20. How to disable a specific auto-configuration class?

You can use exclude attribute of @EnableAutoConfiguration if you want auto-configuration not to apply to any specific class.

//use of exclude

@EnableAutoConfiguration(exclude={className})

### 21. How can we create a custom endpoint in Spring Boot Actuator?

To create a custom endpoint in Spring Boot 2.x, you can use the @Endpoint annotation. Spring Boot also exposes endpoints using @WebEndpointor, @WebEndpointExtension over HTTP with the help of [Spring MVC](https://www.edureka.co/blog/spring-mvc-tutorial/), [Jersey](https://www.edureka.co/blog/java-web-services-tutorial/), etc.

**SPRINGBOOT ANNOTATIONS**

[**1.@SpringBootApplication**](mailto:1.@SpringBootApplication)

**🡪@**@SpringBootApplication annotation to enble three feutures componentscan,configuration,autoconfiguration

**🡪@**@SpringBootApplication  same as **@Configuration, @EnableAutoConfiguration and @ComponentScan**

**🡪**@SpringBootApplication encapsulates **@Configuration, @EnableAutoConfiguration, and @ComponentScan** annotations with their default attributes.

[**2.@ComponentScan**](mailto:2.@ComponentScan)

This annotation enables component-scanning to scan the components like controller,service,dao etc..

[**3.@Configuration**](mailto:3.@Configuration)

using these annaotation to make the class as configuration class.it is a java based configuration**.**

[**4.@AutoConfiguration**](mailto:4.@AutoConfiguration)

This annotation enables the magical auto-configuration feature of Spring Boot, which can automatically configure.

🡪 add dependencies in springboot like spring-boot-starter-parent, spring-boot-starter-web

**WHAT HAPPEN INTERNALLY SPRINGBOOT APPLICATION**

@SpringBootApplication

Public class Application{

Public static void main(String[]args){s

SpringApplication.run(Application.class,args);

}

}

🡪above main class internally to enble componentscan and autoconfigured.

🡪when ever call run() method internally will happen like

1.set up all default configuration

2.start the applicationContext

3.perform the scanning

4.start the tomcat server

**WEBSERVICES**

**1.What are the web services?**

Web service is collection of open protocols used for the exchanging data between applications or system.or webservice is a shareing the information of two application.

**2.What are the types of web services?**

There are two types of web services…

1.soap web services

2.Rest web services

**3.Diff b/w soap and restful webservices?**

**Soap:-**

* to transfer the data xml only.
* Heavy weight because it using xma data.
* Javascript can’t call web services.

**Restful:-**

* To transfer xml and json data.
* Light weight
* Java script can call webservices using restful webservices.

**4. How many ways to communicate different applications?**

1.DB 2.RMI/EJB 3.JMS 4.WebServices

**5.What is rest stands for?**

Rest stands for representational state transfer**.**

**6.What is rest?**

Rest is a web stranded based architecture uses http protocols for data communication

**7.What are the commonly used http methods in rest based architecture?**

**1 GET**: provide read only access to a rsource**.**

2.**PUT**: used to create a new resource

3.**DELETE**:used to remove the resource.

4.**POST** : used to create a new resource

5.**OPTIONS**:used to get the supported operations on a resource

**8.What are the restful webservices?**

Web service based rest architecture is known as restful webservices.

**9.What is resource in Rest?**

Rest architecture treats every content as a resource .these resource can be text file html file pages images or videos

**10.How to represent resource in Rest?**

Text, json,xml, xml and json are most popular representations of resource.

**11.Which protocol used by restful webservices?**

Using http protocol for communication b/w client and server**.**

**12.What is messaging in restful webservices?**

A client sends messages in form of a http request and server respond in form of a Http Response**.**

**13.What are the core components of http request?**

**1. Verb:** Indicate Http methods such as GET,POST,PUT,DELETE ETC..

**2.url:**Uri to identify the resource on server.

**3.HttpVersion:**Indicate Http version.

**4.Request Header:**contains metadata for the Http request message as key-value pairs.

**5.RequestBody:**message content.

**14**.**What are the core components of http response?**

**1.status:** Indicate server status for the request resource ex:404 means resource not found.

**2.HttpVersion**: Indicate version.

**3.response Header**: contains metadata for the http response message as key-value pair.

**4.Response Body**: Response message content.

**15.What is addressing in restful webservices?**

Addressing refers to locating a resource.

**16.What is uri?**

Uri stands for uniform resource Identifier each resource in Rest architecture is identify by its uri.

**17.What is the format of uri in rest architecture?**

<protocal>://<service-name>/<Resourcetype>/<ResourceId>

**18.What is purpose of http verb in rest based architecture?**

Verb identifies the operation to be performed on the resource.

**19.How to develop standard uri?**

1.Avoid using spaces

2.use lowercase letters

3.use http verb

**20.What is statelessness in restful webservices?**

As per Rest Architecture a Restful web services should not keep a client state on server. This restriction is called restlessness.

**21.What is the advantage of statelessness?**

Webservice can treat each method request independently.

Webservice need not to be maintain clients previews interations.it simply application design.

**22.What do you mean idempotent?**

Idempotent means their result will always same no matter how many times these operations are invoked.

**23.Which type of webservices methods are to be idempotent?**

Put and delete operations are idempotent**.**

**24.Which type of web service methods are read only?**

Get operations are read only.

**25.What is diff b/w put and post operations?**

Put and post operations are nearly same with the difference put operation is idempotent as post operations.

**26.What is purpose of options method?**

It should returns list down the supported operation in a webservice and should be read only.

**27.What is purpose of Head method?**

It should returns only http header, no body and should be read only.

**28.What is caching?**

Caching refers to storing the server response in the client itself.

**29.What is the purpose of http status code?**

Http status code are standard codes refers predefined status of task done at server.

**30.What http status code 200 state?**

It means Ok show success.

**31.What http status code 201 state?**

It means CREATED when a resource is successful created.

**32.What http status code 204 state?**

It means No-Content when response body is empty for example delete request.

**33.What http status code 304 state?**

It means Not ModiFied.

**34.What http status code 400 state?**

It means Bad Request.

**35.What http status code 401 state?**

It means FORBIDDEN states that used rights user is not having access to method being for example delete acess without admin.

**36.What http status code 404 state?**

.it means NotFound state that method is not Available.

**37.What http status code 409 state?**

It means CONFLECT states conflict situation while executing the methods for example adding duplicate entry.

**38.What http status code 500 state?**

INTERNAL SERVER ERROR

**39.Explan what is rest and restful?**

**Rest** is a web stranded based architecture uses http protocols for data communication.

**restful** :Web service based rest architecture is known as restful web services

**40.Mention what tools are required to test your webservices?**

SOAPUI tool for SOAP WS

**41.What is the jax-ws and jax-rs?**

Both JAX-WS and JAX-RS are libries for doing communication is various ways in java.

**42.what is RestTemplate?**

It is simpilifies the interation with http server and restful systems.

Resttemplate used to consume the restful webservices.

Created RestTemplate Object using Resttemplate method like getForObject() etc..

RestTemplate rt=new RestTemplate ();

Student s=rt.getForObject(url, student.class);

**43.what is ResponseEnity?**

ResponseEnitity Represents Http response including headers,body and status.

**RESTFUL WEBSERVICES ANNOTATIONS**

[**1.@post**](mailto:1.@post)

[**2.@get**](mailto:2.@get)

[**3.@delete**](mailto:3.@delete)

[**4.@update**](mailto:4.@update)

[**5.@path**](mailto:5.@path)

[**6.@param**](mailto:6.@param)

### [7.@QueryParam](mailto:7.@QueryParam)

### [8.@RestController](mailto:8.@RestController)

### 🡪put annotation on top class make the class is rest class

### [9.@RequestBody](mailto:9.@RequestBody)

### Simply put, the @RequestBody annotation Spring will convert  HttpRequest body to a transfer or domain object

### [10.@ResponseBody](mailto:10.@ResponseBody)

* if you put @ResponseBody annotation in the method level, Spring will convert the return object in to the http response body.

@produce:

@Consume

Above 2 annotations used for add the media type like “application/json/xml”,”text/plan”

ALL REST CALLS

=============================================

1.

2. @RequestMapping(value = "/ex/foos", method = RequestMethod.GET)

@ResponseBody

public String getFoosBySimplePath() {

return "Get some Foos";

}

2.@RequestMapping(value = "/ex/foos", method = POST)

@ResponseBody

public String postFoos() {

return "Post some Foos";

}

3.RequestMapping and HTTP Headers

@RequestMapping(value = "/ex/foos", headers = "key=val", method = GET)

@ResponseBody

public String getFoosWithHeader() {

return "Get some Foos with Header";

}

4)

@RequestMapping( value = "/ex/foos", headers = { "key1=val1", "key2=val2" }, method = GET)

@ResponseBody

public String getFoosWithHeaders() {

return "Get some Foos with Header";

}

5. @RequestMapping Consumes and Produces

@RequestMapping( value = "/ex/foos", method = GET, headers = "Accept=application/json")

@ResponseBody

public String getFoosAsJsonFromBrowser() {

return "Get some Foos with Header Old";

}

6)

@RequestMapping( value = "/ex/foos", method = GET, produces = { "application/json","application/xml" })

RequestMapping with Path Variable

7)Single @PathVariable

@RequestMapping(value = "/ex/foos/{id}", method = GET)

@ResponseBody

public String getFoosBySimplePathWithPathVariable( @PathVariable("id") long id) {

return "Get a specific Foo with id=" + id;

}

8) Multiple @PathVariable

@RequestMapping(value = "/ex/foos/{fooid}/bar/{barid}", method = GET)

@ResponseBody

public String getFoosBySimplePathWithPathVariables

(@PathVariable long fooid, @PathVariable long barid) {

return "Get a specific Bar with id=" + barid +

" from a Foo with id=" + fooid;

}

9) @PathVariable with RegEx

@RequestMapping(value = "/ex/bars/{numericId:[\\d]+}", method = GET)

@ResponseBody

public String getBarsBySimplePathWithPathVariable(

@PathVariable long numericId) {

return "Get a specific Bar with id=" + numericId;

}

============================

10) RequestMapping with Request Parameters

http://localhost:8080/spring-rest/ex/bars?id=100

@RequestMapping(value = "/ex/bars", method = GET)

@ResponseBody

public String getBarBySimplePathWithRequestParam(

@RequestParam("id") long id) {

return "Get a specific Bar with id=" + id;

}

11) http://localhost:8080/spring-rest/ex/bars?id=100&second=something

@RequestMapping(

value = "/ex/bars",

params = { "id", "second" },

method = GET)

@ResponseBody

public String getBarBySimplePathWithExplicitRequestParams(

@RequestParam("id") long id) {

return "Narrow Get a specific Bar with id=" + id;

}

12) @RequestMapping – multiple paths mapped to the same controller method:

@RequestMapping(

value = { "/ex/advanced/bars", "/ex/advanced/foos" },

method = GET)

@ResponseBody

public String getFoosOrBarsByPath() {

return "Advanced - Get some Foos or Bars";

}

13) @RequestMapping – multiple HTTP request methods to the same controller method"

@RequestMapping( value = "/ex/foos/multiple", method = { RequestMethod.PUT, RequestMethod.POST }

)@ResponseBody

public String putAndPostFoos() {

return "Advanced - PUT and POST within single method";

}

14) @RequestMapping – a fallback for all requests:

@RequestMapping(value = "\*", method = RequestMethod.GET)

@ResponseBody

public String getFallback() {

return "Fallback for GET Requests";

}

@RequestMapping( value = "\*", method = { RequestMethod.GET, RequestMethod.POST ... })

@ResponseBody

public String allFallback() {

return "Fallback for All Requests";

}

**What is Json?**

It is a javascript object notation

Json is alight weight data-interchange format

**Why do you use json?**

A json is alight-weight and portable compare to other languages.

**What are the json datatypes**

* + 1. **String**
    2. **Number**
    3. **Boolean**
    4. **Null/empty**
    5. **Object**
    6. **array**

**MICROSERVICES**

1. **What is Micro Services?**

Micro services is a an architecture style that stricture an application as a collection of loosely

coupled services.

In which large software application are composed of one or more services.

1. **What is Advantages or benefits of micro services?**

-> Micro services are quicker to build and deploy.

-> if one Micro Service is fails then others services will continuous to work.

->code for different services can be written in different languages.

->the micro services architecture enables continues delivery.

-> Easy to understand a small piece of functionality.

->Easy to Modify for developers.

->Easy to scale and integrate with third party services

-> Improves devlopment time and load time for IDE.

-> less dependencies.

**3. What is draw backs of Micro services?**

->developing distributed systems can be complex because everything is now an independent.

Service you have to carefully handle request.

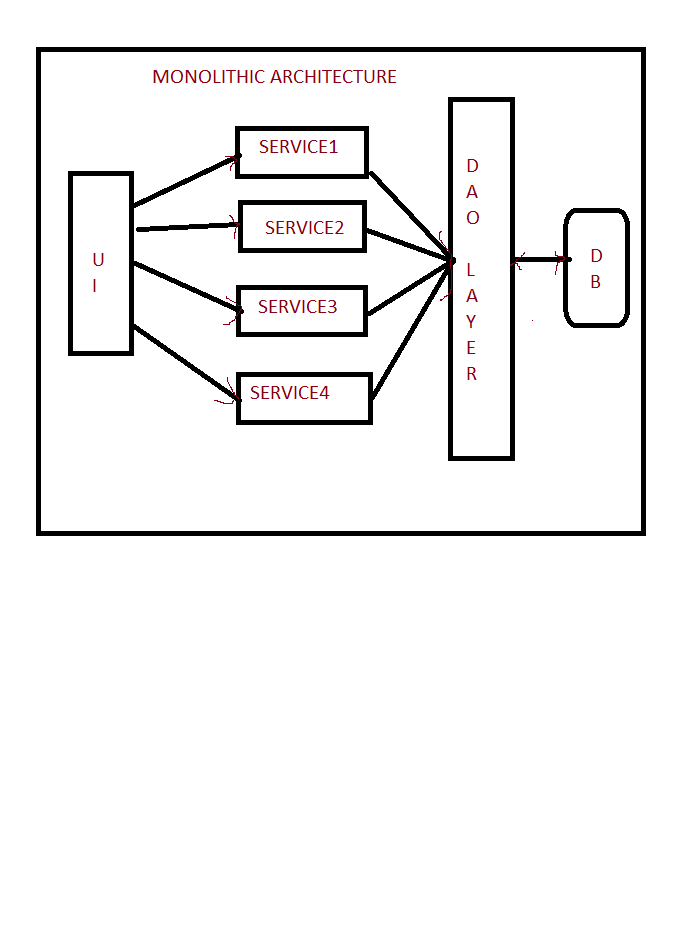
-> testing micro services based applications can be difficult compared to a monolithic

application.

**4. What is monolithic Architecture?**

A monolithic Architecture describes a single-tired software application in which

the user interface and data access code are combined in a single program**.**



**5. Diff b/w monolithic and micro services architecture?**

**Monolithic Arch:-**

* + A single code base for the entire application.
  + Configureing and hard to maintain.
  + Entire application developed in one programing language.

**Micro service Arch :-**

* + Multiple code bases each Micro Service has It own code base.
  + Much readability and much easier to maintain.
  + Micro service can be developed in a different programing language.

**6. Explain Monolithic Architecture?**

**7. Advantages and Disadvantages of Monolithic architecture?**

**Advantages**:

* Single deployment unit.
* Source code for entire application at a single location.
* We can directly create a deployment unit such as war, ear file ide itself.

**Drawbacks**:

* Even small code changes entire application rebuild and redeploy the Application.
* Complexity
* Overloaded ide
* Continues deployment is difficult.
* You must redeploy the entire application on each update.

**9. What is rest endpoint?**

Every micro service needs to communicate with the configure server. The configure server

publish a rest endpoint through which micro services communicated or we can the properties in a browser**.**

**10. What is actuator?**

If any properties for a micro service have been changed that means refreshed through actuator

refresh rest endpoints by doing this our micro service got updated without rebuilding the application.

**11. What is cloud bus?**

This is optional but very useful.

* If we need to refresh the actuator endpoints manually for each service it is a complex.in this
* scenario the cloud bus helps to push the updated properties.

**12. What is load balancer?**

If we maintain only one configure server one point of time it’s failure it should load balanced so

we can run multiple instances of Configserver and load balance pool should have public

address where every micro service communicated.

**13. What is repository area?**

The config server stores all micro services property files in a version control system can be git.

* The config server stores each micro services property based on Service ID.
* Ex:- spring.application.name=Employee payroll.

**14. What is a zuul proxy?**

Zull acts as an Api gateway or edge service.

* It receives all the request coming from the ui and then delegates the request to internal
* Micro service

**15. What is Eureka server?**

The eureka server is a noting but a service discovery pattern implementation where every

micro service is registered.

The purpose of micro service is what are the instances currently registered with eureka and to

know the server up/down status.

**16. What is Authorization server?**

Manages the allocation of the keys tokens and other temporary access codes.

**17. What is Config server?**

Using config server we are storing all properties for all the micro service in a central git

repository. Each micro service is able to pull his own properties file based on application name

and profiles.

**18. What is Discovery server?**

Discover server mainly used to eureka discovery receives heartbeat messages from each

instance belonging to service.

**19. Purpose of circuit breaker?**

If we have any problem in rest application like exception timeout, if any other problems call the

fallback method.

Ex:-

**20. What is load balancing?**

If any problem occur in one instance or server using loadbalanceing to move the another

instance this achievable with Ribbon concept.

Ex:

**21. Purpose of FeilgnClient?**

If we have problem occur in two different services like Exception at load

balancing time using feign client.

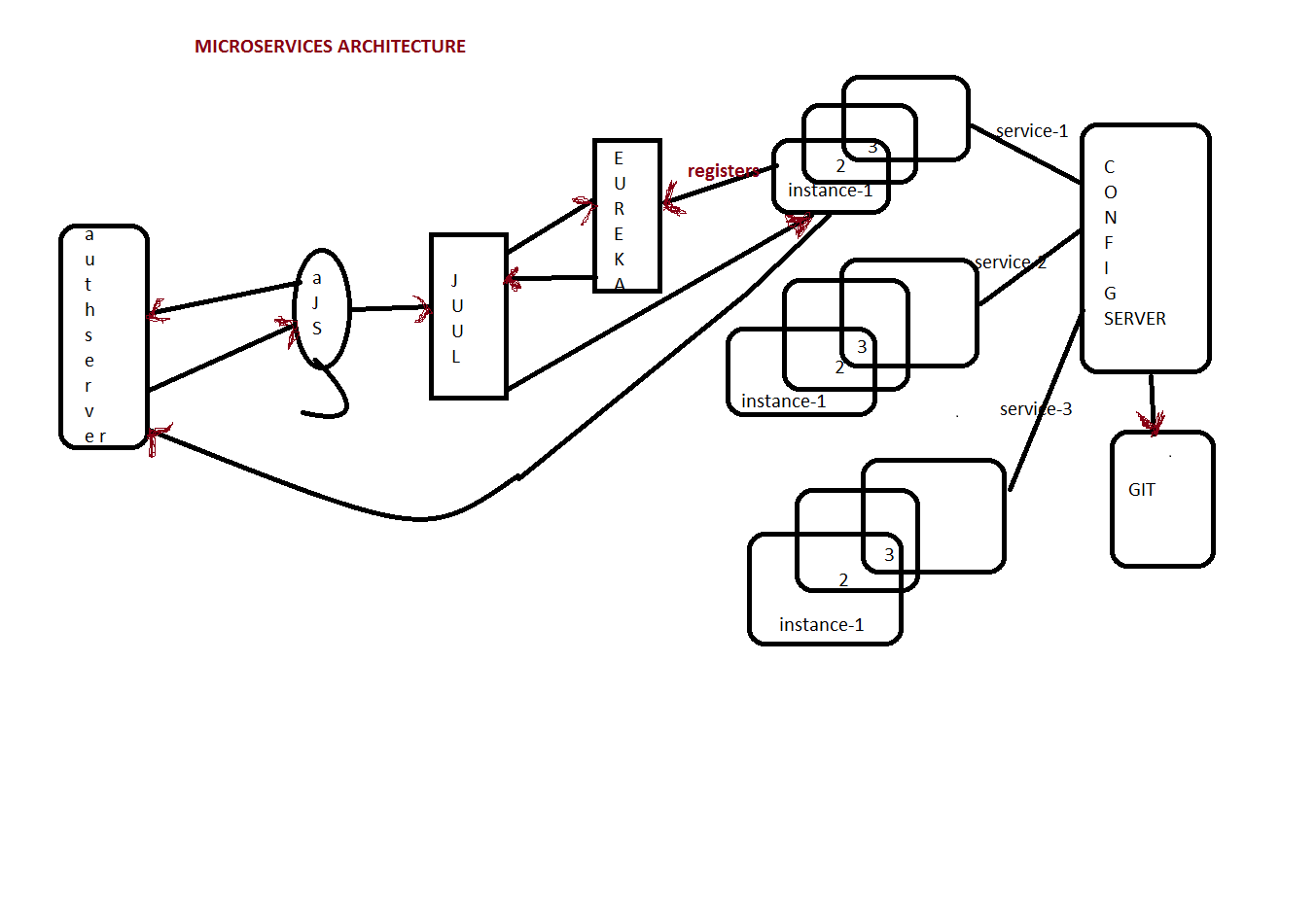
Ex:

**22. What is spring cloud?**

Spring cloud provides tool for developers to quickly build some of the common patterns in distributed systems.

* Service discovery
* Zull or proxy or router
* Config server
* Authorization server
* Resource server
* Hytrix
* Flienclient
* Circutebreaker
* Ribbon

Above all things one cloud only.

 **BASIC MICROSERVICE FLOW**

**===============================================**

1. **Every module is developed as a Rest application .**
2. **2.configuration is provided in config server for every rest application as property file or YML file.**
3. **Every application is clustered using ribbon , so we do have multiple instance for every rest application .**
4. **T Access any rest application first clint has to get token by providing user/pwd , client secret .**
5. **next using token it has call any rest application through Zull or gateway server proxy, router).**
6. **6.Before processing request rest application contact with authorization server to verify token authorization and authentication.**
7. **7.so every request first goes to ZULL, ZULL take the help of eureka server and it Access ribbon to process the request in appropriate rest application .**
8. **8.Every rest application is divided into controller , service , dao and utility layers.**
9. **9.Every layer is communicated with entity classes.**
10. **10.Controller is responsible to receive data from Clint , calling request service , and sending response to client . Response include data, response code .**
11. **11.Service layer is responsible business logic, server side validations and transaction s.**
12. **12.Dao layer is responsible for data access form database.**
13. **13.Util layer contains common code used by all layers.**
14. **14.Communication b/w client and our application happen with json .**
15. **15.With in the project communication b/w rest application to another happens using rest templet, feign**
16. **16.Exception handling is done using controller advice in rest application . We are using fall back along To provide alternate logic in case of called of rest application failed.**
17. **17.For circuit breaker we are using hystrixe.**

**ANNOTATION OF SPRING CLOUD**

### ****1. @EnableConfigServer****

### ****2. @EnableEurekaServer****

### ****3. @EnableDiscoveryClient****

### ****4. @EnableCircuitBreaker****

### ****5. @HystrixCommand(fallbackMethod = “fallbackMethodName”)****

### ****6.**** @EnableEurekaClient

### 7. @RibbonClient

### 8. @RibbonClient

### [9.@EnableZuulProxy](mailto:9.@EnableZuulProxy)

### 10.@EnableResourceServer

**OBJECTS CLASS METHODS**

* Object class contain mostly commonly required method for every java class
* Every class in java child classs of object either directly or indirectly so that object class method by defaulit available every java class.
* Java not support for multiple inheritance with respected classes becoz java only can extend one class.

**1.public String toString()**

🡪 we can use toString() method to get String representation an object

🡪String s=(Object).toString();

🡪when ever we are trying print object reference internally Object class toString() method will be called.

Student s=new Student();

Sop(s.toString());

🡪if our class doen,t contain toString() method object class toString() method will be executed

//classname@hascode in hexadecimal form like Test@254899

🡪over come above pblm to override own toString() method

Public String toString(){

retun Student[“rollno”+rollno + “name”+name]

}**IMPNOTE:- all wrapper classes toString() by default override like int,flot,String.**

**2.public native int hascode()**

🡪every object a unique number generated by jvm which is ntng but hascode.

🡪hascode won,t represent address of object

🡪jvm will use hascode while saveing object into hashing related data stracture like hashtable,hashMap,hashSet etc….

🡪the main advantage of saveing objects based on hashcode is serch operation will become easy(most powerful search algirtm upto today is hashing)

🡪if we are giving the chance to object class hascode methodit will generated hashcode based on address of the object

🡪based on our requirement we can override hashcode method in our class to generated our own hashcode .

🡪overriding hashcode method is said to be proper if and only if every object we can generate a unique number has hashcode

Class Student

{

---------------//this is inproper way of overriding hashcode() method becoz for all student objets we are generating same number has hascode

Public int hascode(){

return 100;

}Class Student

{

Public int hascode(){

Return rollno;}//it proper way overriding becoz we are generating a diffirent hascode for every object

toSring vs Hashode

🡪if we are giving the chance to object class toString() method it will internally call hascode() method

🡪if we are overriding toString then our toString() method may not call hascode method

**3.public Boolean equals(Object o)**

**🡪**we can use equls()method to check equlity of objects

Obj.equls(obj)

🡪if our class doesn,t contain equls() method then object class equls () method will executed

🡪if object class equls() method got executed which is meant for reference comparision(address comparision) i.e if two references pointing to the same object then only .equls() method returns true.

🡪based on our requirement we can override equls() method for content comparision

**How we can override .equls() method for content comparision?**

Public Boolean equls(Object obj){

We can override from ide}

🡪while overriding equls() method for content comparision we have to take care about following

🡪what is meaning of equlity (i.e wheather we have to check only name or rollno or both)

🡪if we passing diffirent type object our equls method should not rise classcastexception i.e we have handle callsscastexception returns false

🡪if we are passing null arg then our equls() method should npe:we have handle npe to return false.

**4.public final Class getClass()**

🡪can use getClass() method we if we want know the fullyqulified classnames,mehod information

🡪by using this class class object we can access class leval properties like fullyqulified name of the class method information constructer information etc

Ex:Object o=new String(“srinu”);

Class c=o.getClass();

Method[] m=c.getDeclareMethods();

**5.protected native Object clone()**

🡪the process of creating exact dublicate object is called cloneing.

🡪purpose of cloneing is to maintain backup copy and preserve state of object

🡪by ysing clone() method to achive the cloneing

**6.protected void finalize()**

-🡪just before destroying the object garbagecollecter call finlize ()method to perform cleanup activities once finalize() method complete automatically gc destroy that object

**7.wait()**

**8.wait(long ms)**

**9.wait(long ms,int ns)**

**10.notify()**

**11.notifyAll()**

**🡪**we can use these method for interthread communication

🡪 the thread which is exepating updation, it responsible to call wait() method then immediately the thread will enter into waiting state.

* The thread which is responsible for updation, after performeing updation the thread can call notify()
* The waiting thread will get that notification continue its execution with those updates

**JAVA8 FEATURES**

Java8 is concise code because of enabling functional programming languages**.**

**Features:**

**1.Lamda Expression.**

**2.Functional Interface.**

**3.Predefined Functional interface**

**4.Default and Static Methods.**

**5.Streams**

**6.Date And Time Api**

**7.Double colon operator(: :)**

**8.Optinal Class**

**9.Narshron JavaScript Engine**

**1.Lamda Expression:-**

=> It is an anonymous function

Anonymous means Nameless (no name) , without return type, without modifiers function.

**2.Functional Interface:-**

**=> which interface contains only one abstract method such type of interfaces are called functional interfaces.**

**Ex:-**

**1.Runnable -> run()**

**2.Comparable ->compareTo()**

Above 2 methods contains only one abstract method.

Use of functional interfaces are if we want invoke lambda expression compulsory function interface is required**.**

* Inside functional interface to take default and static methods also but it contain at least one abstract method is required.

**3.Predefined Functional interface:**

There are four types of functional interfaces.

1.Predicate => test()

2.Function => apply()

3.Consumer =>accept()

4.Suppiler => get()

**Predicate:-**

Predicate is a predefined functional interface it contains one abstract method i.e

Public Boolean test(type parameter**)**

* Purpose of these interface where ever condition check we can write the predicate function.

**Function:-**

It is a predefined functional interface it contains only one method i.e apply()

Public R apply(T t)

Purpose of function is any return type int,String,student,but not boolean we should go for function**.**

**Ex:**

**Consumer:**

It is also predefined functional interface it contain only abstract method i.e accept()

* Purpose of consumer interface is won’t return any thing just consume we should go for consumer.

Ex:

**Suppiler:**

It is also predefined fictional interface it contains one method i.e get()

* Just supply my required object and it won’t take any input.

Ex:-

**4.Default and Static Methods:-**

The purpose of default methods are without effecting implementation classes if we want to add new method to the interface.

* Only defaults methods only write inside interface but not class.
* Object class methods we can’t declare default methods like

default int hascode()

* Interface contains static methods also
* Main method also take inside interface
* If every thing is static unnecessary go the class because of the reason static method inside interface.

**5.Stream Api**

The main objective or purpose of stream is if we want to process object from the collection**.**

**Difference b/w collection and stream**

If we want represent a group of objects as a single entity we should go for collection.

If we want process object from collection we should go for stream.

**6.Date And Time API**

 advantage is that **all time representations in Java 8 Date Time API are immutable and thus thread-safe**. All mutating methods return a new copy instead of modifying state of the original object.

**JAVA TOOLS**

**1.Application Development And Testing all Environments**

**2.JIRA**

**3.MAVEN**

**4.GIT**

**5.AGILE**

**6.SERVERS**

**7.DATABASES**

**8.IDE’S**

**9.LOG4J**

**10.DESSIGN PATTERANS**

**1.Application Development And Testing all Environments.**

=> when ever develop the one feature or one service as a developer to test application in local machine that is called developer testing.

=> then when ever completed dev testing to push the code git repository.

=> when ever push the code that piece of code to test QA team if any find issue to raise the Bug.

=>that Bug solve the developer then dev test same process.

=> as a developer when ever completed dev testing to test the application stage environment.

=>to deploy war file stage environment.

=>if not issue stage environment it should go the production environment noting but live.

Note: if any issue in production environment to maintain the separate team as a developer to check the log files if find the any issue to fix the issue.

**2.JIRA**

=> my project management tool is JIRA.

=>project management tool is nothing but we can manage the defects we can manage the projects and we can manage the sprints.

=> in our company all task assign by throw Jira only.

=> when ever joining time my project gave credential for Jira.

=>my project manager assign to task throw Jira to check what are task assigned to me and do the work based on priority.

=> when we start the task click the more action to change the state To-do to InProgress and when ever completed the task to change state Inprogress to done state.

Note: before start the task to write comment I Am working on this task when ever complted the issue add comment for completed.

After completed the sprint if and pending to move tom next sprint.

**3.MAVEN**

**What is Maven?**

Maven is a build management tool it is used for simply the build process and it’s build the project.

* **Why maven is required.**
* Maven project eliminate the adding set of jars in each project.
* It creates own write project structure,
* Building and deploying the project is simple.

**What is Build Tool**

* A build tool takes care of everything for building process like..
* Generate the source code and compile the source code.

Note:

* When ever adding dependences in pom.xml first time to download the all jar file from central repository to store all jars in local repository noting bit .m2

**BUILD PROCESS**

* + 1. Create Project
    2. Write source code
    3. Add dependences
    4. Prepare test cases and executed test cases
    5. Deploy server
    6. Run the server
    7. Send request to the browser
  + Using the maven to simply the build process**.**

**4.GIT**

**What is git?**

Git is a version control systems and source code management. It’s handle small and large projects.

**What is purpose of git?**

* The main purpose of git is to manage the project.
* It is designed for coordinating work among programmers.
* It can be used to track the changes.
* It allows multiple developers to work together**.**

**Benefits of version control systems**

* + 1. Compare files
    2. Identity differences
    3. Merge the changes
    4. Identity the current development version.
  + **Own git account creation.**

We need to create git account -> to login git account based on creditials -> to create a new Repository -> based on repository to generate cloned url -> to create our own branch**.**

* + **How to push the code from IDE’S**
    1. Write click on Project
    2. Go to Team
    3. Select commit
    4. Enter commit message
    5. Click push and commit
    6. Ok

How to push the code from command prampt

Open gitbash -> git add [File Name] ->git commit -> commit message -> git push orgin Master.

**Git Commands**

**git add[srinu.txt] :-** add a file to the staging area.

* **Git status :-** check the status it means what are the files are

adding or no add.

**Git commit -m[commit messages] :** commit the changes

**Git rm -r [srinu.txt] :** Remove a file or folder

**Git pull origin[BN]** : pull the changes remote repository.

**Git push origin :** push a branch to your remote repository**.**

**Git log :** view the changes.

**Git clone** [**url:[UN][pwd**](url:[UN][pwd)**]** create local copy of a remote repository.

**5.AGILE**

=> Before Agile Process we can use waterfall model. waterfall model is a time taken process because of waiting for the each phase.

=> it first need to complete analysis then enter design then implementation then testing then maintains to waiting for each phase.

=> this process more time taken process,

=> the Approach is highly risky and costly so it is not recommended to use.

=> using agile model we project is divided into multiple tasks.

=> each task we need to provide life cycle model.

=>each task we can call one scrum.

=>advantages of agile model is parallel work is possible.

=>after completion of one life cycle again we have start one more task.it is completely iteration model.

=>after completion of the each task they can send or delivery to the client.

=> each life cycle is one scrum.

=> while designing each scrum they will conduct scrum/sprint meetings.

-> In scrum meeting to discuss what did today which task on work.

This is scrum process.

**How to create task jira.**

Create Issue -> select Issue type -> add summary -> enter prority- > assign the task -> create.

**6.SERVERS**

**Server**

Every web application definitely run on any web, application servers

**There are two types of servers**

1. **Webserver**
2. **Application Server**

**Webserver:**

webservers are like tomcat, jetty.

**Application Server:**

application servers are like jboss,weblogic,webspare,glashfish.

**7.DATABASES**

**What is Database?**

A database is an organized collection of data generally stored and accessed electronically from a computer systems.

**What is DDL and DML commands?**

* DML commands are Insert, update, delete.
* DDL commands are created, alter, drop.

**QUIREYS**

1. **Insert:-** insert into student(id,name.address) values (1,”srinu”,”Khammam”);
2. **Update:** update student set name=naresh, address=”hyd” where id=1;
3. **Delete:** delete from student where id=1;
4. **Select:** select \* from student;
5. **Joins:** select d.district, s.state, c.country from Person p join district d join state s join country c where p.addernum=”62266356747”;
6. **Subquery**: select \* from student id IN(select Id from student where address=”Khammam”);

1. create table Employee (empid int NOT NULL, empname varchar (50) NOT NULL, salary int NOT NULL, primary key(empid));

2. Insert into Employee (empid, empname, salary) values(101,’srinu’,50000);

3. Update Employee set empname=’Naresh’, salary=45000, where empid=101;

4. Delete from Employee where empid=101;

5. Alter table Employee add age int not null;

6. Alter table Employee drop age;

7. Select \* from Employee;

8. select empname, salary from Employee;//only particular records print

9. select \*from employee where salary= (select max(salary) from employee);

10. select \*from employee where salary= (select min(salary) from employee);

11. select \* from employee limit 1, 3; //first three records will print.

12. select \* from employee group by salary order by salary asc limit 0, 4//first 4 salaries asc order print.

13. select \* from employee group by salary order by salary desc limit 0, 4 //desc print

14. select \* from employee group by empname order by empname asc limit 0, 4//ascending names

15. select \* from employee group by empname order by empname asc//all records print

16. SELECT salary from employee order by salary desc limit 1,1//second highest salary

17. SELECT empname, salary, COUNT(salary) AS NumberOfTimes

FROM employee GROUP BY salary HAVING COUNT(salary) > 1//duplicate salary

18. SELECT empname, salary, COUNT(empname) AS NumberOfTimes

FROM employee GROUP BY empname HAVING COUNT(empname) > 1//duplicate empname

**8.IDE’S**

I have worked ide’s are eclipse and Intelij. the main advantages of ide’s are project development time is very less because of ide’s gives suggestions.

* Increase the efficiency faster coding with less effort.

**9.LOG4J**

It is used for bug tracking.to write and configure code like the following.

Private static final LOGGER Logger=Logger.getLogger(Student. Class);

Logging levals

Logger.info();

Logger,error();

Logger.debug();

Logger.fatal();

Logger.warn();

public class LogClass {

private static org.apache.log4j.Logger log = Logger.getLogger(LogClass.class);

public static void main(String[] args) {

log.trace("Trace Message!");

log.debug("Debug Message!");

log.info("Info Message!");

log.warn("Warn Message!");

log.error("Error Message!");

log.fatal("Fatal Message!");

}

}

**10.DESSIGN PATTERANS**

By using the design patterns you can **make your code more flexible, reusable and maintainable.**

**Singleton design pattern**

Singleton Pattern says that just**"define a class that has only one instance and provides a global point of access to it".**

#### **Advantage of Singleton design pattern**

* Saves memory because object is not created at each request. Only single instance is reused again and again.

Ex:

1. **class** JDBCSingleton {
2. //Step 1
3. // create a JDBCSingleton class.
4. //static member holds only one instance of the JDBCSingleton class.
6. **private** **static** JDBCSingleton jdbc;
8. //JDBCSingleton prevents the instantiation from any other class.
9. **private** JDBCSingleton() {  }
11. //Now we are providing gloabal point of access.
12. **public** **static** JDBCSingleton getInstance() {
13. **if** (jdbc==**null**)
14. {
15. jdbc=**new**  JDBCSingleton();
16. }
17. **return** jdbc;
18. }

**\* How HashMap internally works**

**What is hashing?**

Hashing is the process of mapping the data to some reprasentive integer value using the concept of hashing algorthims .

* + In java a hash code is an integer value that is linked with each object.

**How HashMap internally works**

Map<String, Integer> map=new HashMap<String, Integer>();

1. First when ever creating HashMap object. Internally create array of buckets.
2. The size wil be **16**
3. Each bucket nothing but a node. Node means a linked list.
4. When ever call map. put(“srinu”, 1); put method hashcode will be calculated and store the appropriate index bucket.
5. Both keys are returns same hashcode another linked list will be created .
6. Based on key hashcode will be generated. If key is null it will store 0th bucket.
7. When ever calling map.get(“abc”) method based on hashcode to retrieve the data.
8. Hashcode will be same to check key , based on key we get the data.

**JENKINS**

* + 1. **What is Jenkins?**
  + Jenkins is a contionus integration tool that allows continuous development ,test and deployement of newly created codes

**2 what is contiouns integration?**

* + The contiouns integration is a process of merging all developers works in a shared enviralment (nothing but git).

**3 what is continuous Delivery?**

* + To generate product in short cycle which is work done by developers in their shared enviralment.

**Process**

* + When ever push the code to git , Jenkins jobs are trigger.
  + Before push the code needed some configuration in Jenkins.

**Step-1**

* + Create the newjob in (new item)-> click freestyle project.

**Step-2**

* + In source code management add the repository url and git creditial like username and password

**Step-3**

* + Add the build Triggers like build periodically or poll scm after adding will check every minit or every update will check.[\* \* \* \* \*]

**Step-4**

* + Add the batch commands like mvn package.

**Step-5**

* + Add add to contionue plugin
  + In post build action select the deploy war/ear to container. In that add \*\*/\*.war filename will search package with war name placed to tomcat server.

**Step-6**

* + Add the container means which server is using select that one
  + Add the tomcat url and creditionals
  + Finally apply the all the changes.

**Note:-**when ever push the code to git then jobs are trigger automatically if the build success to deploy war file in server automatically**.**

**Jenikins Advantages**

* + It is a open source and easy to install.
  + It is platform independent.
  + It integrades various devops stages in ci/cd process.
  + Easy to configure widely varitery of plug-ins
  + Easy to automatic ci and cd process
  + It reduces the time to find out build errors in automations

**SIGLETON CLASS**

* + for any java class if any allowed to create only one object such type of class is called singleton class.

Ex; Runtime, BusinessDelete, ServiceLocator

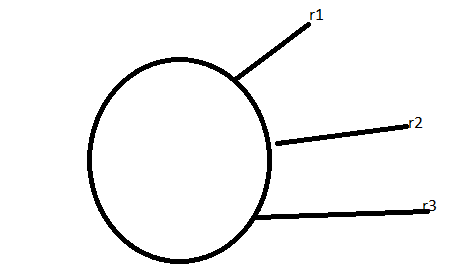
**Advantages of Singleton class**

* + **If several people have same requirement to create separate for every requirement**
  + **We have to create only one object we can reuse same object every simller requirements . so that perfamence and memory utilization improved.**
  + **Ex.**

Rutime r1=Runtime.getRuntime();

Rutime r2=Runtime.getRuntime();

Rutime r3=Runtime.getRuntime();

****

***Return the same object every time***

**How to create our own singleton class**

* + We create our own singleton class for these we have to use private constructer and private static variable and public factory method.

**Class Test {**

**private static Test t=null;**

**private Test() {**

**} public static Test getTest()**

**{**

**if(t==null)**

**{**

**t=new Test();**

**}**

**return t;**

**}**

**}**

* + **When ever creating first time object is created (because firsttime is null)**
  + **Second time onwards existing object is returning.**

**Note:**

* + Class is not final but we have not allowed create child class.

Class Parent

{

Private parent()

{

}

}

* + Private take private constructer impossible to create child class.

**Factory Design Patteran**

* + Which is class it’s give the object it’s manufacturing object.
  + If we have extra class in feature it will not changing client application.
  + The factory design pattern is used when we have a superclass with multiple sub classes and based on input we need to return on of the sub-class.

**Advantages**

* + Factory design patteran removes the instanceation of actual implementation class from client code .
  + Factory Method Pattern **allows the sub-classes to choose the type of objects to create**. It promotes the loose-coupling by eliminating the need to bind application-specific classes into the code.

**What is Api in Java**

* + Api is a collection of prewritten packages, classes, interfaces with their respective methods, fileds,constructers.
  + Api is a Intermediate software b/w two applications or same Application.

**Ex:**

**RestApi**

**Postman**

**RestClient**

**Soap UI**

**OPTIONAL CLASS IN JAVA**

Java introduced a new class Optional in jdk8.

* + It is a final class, it is not partisepate inhertence
  + It is used to avoiding Java.lang.NullPointerException
  + It provides methods which are used to check the presence of value for particular variable.
  + To avoid the abnormal termination, we use Optional class.

**empty():-** It returns an empty Optional object. No value is present for this Optional.

**public boolean isPresent():-** It returns true if there is a value present, otherwise false.

**Global Exception Handling**

* + Global Exception is a highly recommended handled the Exceptions then proper response to the client.
  + It’s main handaller common type Exceptions then reduce the code.
  + There are two Annotations are mainly used global-exception handling.
    1. @ControllerAdvice
    2. @ExceptionHandaller
  + Proper response to the client means meaing full Exception message.if code is break client is not understanding, that’s why highly recommended handler Exception.
  + Mainly global exception handling use common type exceptions means
  + For ex: NullpointerException once handle the NullpointerException no need to handle again and agin then reduce the code.
  + We can handle custom exception also.

**How to write globalExceptions**

**Step-1**

* + The make class is ContaollerAdivese.
  + @ ContaollerAdivese is a Annotation to handle Exception Globally.
  + In controller Advise class to write the Annotation on top of method is @ExceptionHandller this annotation used to handle the specific exception and sending the custom response to the client.

@ControllerAdvice

public class ControllerAdvice

@ExceptionHandler(EmptyInputException.class)

Public ResponseEntity<String> handllerEmptyInput(EmptyInputException

emptyInputException){

return new ResponseEntity<>("Input filed is Empty please

look",HttpStatus.BAD\_REQUEST);

}

// in service layer write this for custom exections only

if(employee. getName(). isEmpty()||employee.getName().length()==0)

throw new EmptyInputException ();

@Component

public class EmptyInputException extends RuntimeException {

private static final long serialVersionUID = 1L;

private String errorCode;

private String errorMessage;

after setter getters

public EmptyInputException (){

} }

Predefined Exception

@ExceptionHandler(NoSuchElementException.class)

public ResponseEntity<String> handllerNoSuchElementException(NoSuchElementException

noSuchElementException){

return new ResponseEntity<>("Input id is not avilable in db please change

id",HttpStatus.NOT\_FOUND);}

Public class ControllerAdvice extends ResponseEntityExceptionHandler

When ever extending in contains no .of free defined methods we can override.

@Override

protected ResponseEntity<Object>

handleHttpRequestMethodNotSupported(HttpRequestMethodNotSupportedException ex,

HttpHeaders headers, HttpStatus status, WebRequest request) {

return new ResponseEntity<Object>("please change httpmethod type type as soon as

possiable",HttpStatus.NOT\_FOUND);}

**SWAGGER**

* + Swagger is nothing but Documentation from your API Design

Or

* + Swagger2 is an open source project used to generate the rest API documents for Restful webservices.
  + Swagger is generation and matining the api documentation up to date.

**How to Enable to swagger**

* + **To Enable swagger in springboot application we need to add following tifactdependencies.**

**Step-1**

**In pom.xml**

< dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.7.0</version>

</dependency>

**Step-2**

* + Write the swagger configuration class
  + To class add @EnableSwagger2 annotation
  + To write Docket relating configuration

**Step-3**

* + To run application to hit url on browser

http://localhost:8080/swagger-ut.html#/

**SPLUNK**

* + Splunk is an extranal powerful platform it is used to analyze data and logs produced by application

Or

* + Splunk allows you to monitor search and analyze data and logs through a web interface.
  + Splunk is segregate all logs create each services based on the index.
  + Note: based on the index each microservice to segregate logs

**Step-1**

* + In spring boot framework any using log4j.sl4j logging framework. then forward to the splunk.
  + Splunk and springboot running two diffirent servers
  + To connect splunk server we need to write log4j-spring.xml file to add the all the details. Like url, host, token, index, source.

**Step-1**

* Download and install local machine default port 8080.

**Step-2**

* Run on splunk server we need to configure and generate url, host,token, index source.
* Settings->datainputs-> select httpEvent Collector http port 8088.
* New token-> add the index name alerts email all the detail configure all the things.

**JUNIT TESTING**

* + Junit is the testing framework it is used for unit testing of java code
  + Junit=javacode+unittesting.
  + I am using annotations for junit testing.
    1. @SpringBootTest
    2. @Test
    3. @Mock
    4. @InjectsMock
    5. @Before

**@SpringBootTest**

* + These annotation create an application context and loads the full applicationcontext.which mean we can @Autowire any bean component scanning into our test.

**@Test**

* + These annotation is method level we can write logic inside method.

**@Mock**

* + This is used when we want to execute the preconditions or any intilazation based on statements before running every test class.

**@InjectsMock**

* + (to injects the required dependencies)
  + Used for this annotations to injecting which class testing

Ex: I am going to test EmployeeService then I am injecting that service

Spring will give bean of the class.

**ServerSide Validations**

**public** **class** PagenationController {

//PAGENATION

/\*

\* public void pageNation(Employee employee,Session session) {

\* List<Employee> list=servce.getAllEmployees();

List list2=list.subList(0, 3);

\* session.setAttribute("list2",list);

\*

\* }

\* FILE UPLOAD

\* @Autowired

\* private MultipartFile multipartFile;

\*

\* public ModelAndView upload(MultipartFile file)

\* {

\* File f=new File("E://");

\* f.transeferTo(file);

\* FILE DOWNLOADING

\*

\* public void downlaodFile(HttpServletResponse response)

\* {

\* FileInputStream fis=new FileInputStream("test.xml")

\* BufferOutputStream bos=new BufferOutputStream(response.getOutputstream();

\* int data=fis.read();

\* while(data=-1)

\* {

\* bof.write(data)

\* data=fis.read();

\* }}

\*

\* Server side Validations

\* =====================

\* public class User

\* {

\* @NotNull

\* private String name;

\*

\* @Asserttrue

\* private boolean working;

\*

\* @size(min=10 max=20 message="")

\* private String aboutme;

\*

\* @min(value=10,message="")

\* @max(value=150,message="")

\* private int age;

\* @Email

\* private String email;

\*

\*/}

**AREA OF SQUARE**

**package** com.ibm.AreaofSquare;

**public** **class** AreaofSquare {

**public** **static** **void** main(String[] args) {

**double** side=5.5;

**double** areaofsquare=0;

areaofsquare=side\*side;

System.***out***.println("Area of square is::"+areaofsquare);

}

}

**AREA OF TRANGLE**

**package** com.ibm.AreaofTrangile;

**public** **class** AreaofTrangile {

**public** **static** **void** main(String[] args) {

**int** length=25;

**int** width=45;

**double** area=0;

area=length\*width;

System.***out***.println("Area of trangle is :"+area);

}

}

**ARRAY EXAMPLE**

**package** com.ibm.Array;

**public** **class** ArrayExamples {

**int**[] it = { 35,2 ,5,2,2};

**boolean** [] b={**true**,**false**};

**short**[]s=**new** **short**[3];{

s[0]=1;

s[1]=2;

s[2]=3;

//s[3]=4;

**for**(**short** st:s)

System.***out***.println(st);

}

String [] str={"srinu","naresh","suresh","ramu","nagesh","suri"};

**public** **void** m1() {

**int** length2 = it.length;

System.***out***.println("length is::"+length2);

**int** length=0;

**for** (**int** i = 0; i < it.length; i++) {

length=i;

}

System.***out***.println("Integer array length is::"+length);

}

**public** **void** m2() {

**int** length=0;

**for** (String st:str) {

length = str.length;

}

System.***out***.println("String array length is::"+length);

}

**public** **void** sum() {

**int** sum = 0;

**for** (**int** i=0;i<it.length;i++) {

sum += it[i];

}

System.***out***.println("The sum is::" + sum);

}

**public** **static** **void** main(String[] args) {

ArrayExamples array = **new** ArrayExamples();

array.m1();

array.sum();

array.m2();

}

}

**ARRAY LARGEST AND SMALLEST NUMBERS**

**package** com.ibm.ArrayLargeAndSmallNumber;

**public** **class** ArrayLargestAndSmallestNumber {

**public** **static** **void** main(String[] args) {

**int** [] number={5,2,44,57,5,1,2,8,9,44,5,74,85,92,21,47,4215,5,66,12};

**int** smallest=number[0];

**int** largest=number[1];

**for**(**int** i=0;i<number.length;i++){

**if**(number[i]>largest){

largest=number[i];

}

**else** **if**(number[i]<smallest){

smallest=number[i];

}

}

System.***out***.println("higest number in array is::"+largest);

System.***out***.println("smallest number in array is::"+smallest);

}

}

**AUTOBOXING AND UNBOXING**

**package** com.ibm.AutoboxingUnboxing;

**public** **class** BoxingUnBoxingExample {

**public** **static** **void** main(String[] args) {

**int** i=100;

Integer it=**new** Integer(i);

System.***out***.println(it);//auto boxing

Integer ii=**new** Integer(50);

**int** data=ii;

System.***out***.println(data);//auto -unboxing

}

}

**CONSTRUCTERS**

**package** com.ibm.Constructer;

**public** **class** ConstructerExample {

**private** **int** id;

**private** String name;

**private** String address;

**public** ConstructerExample(**int** id,String name){

**this**.id=id;

**this**.name=name;

System.***out***.println(id+name);

}

**private** ConstructerExample(**int** id,String name,String address){

**this**.id=id;

**this**.name=name;

**this**.address=address;

System.***out***.println(id+name+address);

}

**public** **static** **void** main(String[] args) {

ConstructerExample construct=**new** ConstructerExample(1,"srinu");

ConstructerExample construct2=**new** ConstructerExample(1,"srinu","khammaa");

ConstructerExample construct3=**new** ConstructerExample(2,"naresh","wargal");

}

}

**DATATYPE CONVERSION**

**package** com.ibm.Conversions;

**public** **class** ConversionsExamples {

**public** **static** **void** main(String[] args) {

**int** i = 100;

//automatic type conversion

**long** l = i;

//automatic type conversion

**float** f = l;

String s="250";

**int** in=Integer.*parseInt*(s+100);

System.***out***.println(in);

**int** test=Integer.*valueOf*(s);

System.***out***.println(test);

System.***out***.println("Int value "+i);

System.***out***.println("Long value "+l);

System.***out***.println("Float value "+f);

**int** i1 = 100;

String st=String.*valueOf*(i1+"hiii");

System.***out***.println(st);

}

}

**CURSORS**

**package com.ibm.Cursors;**

**import java.util.ArrayList;**

**import java.util.Iterator;**

**import java.util.List;**

**public class CursorsExample {**

**public static void main(String[] args) {**

**List<PlanetSoft>list=new ArrayList<PlanetSoft>();**

**list.add(new PlanetSoft(1, "A", "MOTINAGAR"));**

**list.add(new PlanetSoft(1, "A", "MOTINAGAR"));**

**list.add(new PlanetSoft(1, "A", "MOTINAGAR"));**

**list.add(new PlanetSoft(1, "A", "MOTINAGAR"));**

**list.add(new PlanetSoft(1, "A", "MOTINAGAR"));**

**list.add(new PlanetSoft(1, "A", "MOTINAGAR"));**

**Iterator<PlanetSoft> it=list.listIterator();**

**while (it.hasNext()) {**

**PlanetSoft planetSoft = (PlanetSoft) it.next();**

**System.out.println(planetSoft);**

**}**

**}**

**}**

**STATIC DATA MEMBERS**

**package** com.ibm.Datatypes;

**public** **class** StaticDatamembersExamples {

**private** **static** **byte** *b* = 127;

**private** **static** **short** *s* = 200;

**private** **static** **int** *i* = 500;

**private** **static** **long** *l* = 100;

**private** **static** **float** *f* = 25.0f;

**private** **static** **double** *d* = 50.0d;

**private** **static** **char** *ch* = 'A';

**private** **static** **boolean** *bo* = Boolean.***FALSE***;

**private** **static** String *str* = "srinuvasrao";

**public** **static** **void** main(String[] args) {

System.***out***.println(StaticDatamembersExamples.*bo*);

System.***out***.println(StaticDatamembersExamples.*s*);

System.***out***.println(StaticDatamembersExamples.*i*);

System.***out***.println(StaticDatamembersExamples.*l*);

System.***out***.println(StaticDatamembersExamples.*f*);

System.***out***.println(StaticDatamembersExamples.*d*);

System.***out***.println(StaticDatamembersExamples.*ch*);

System.***out***.println(StaticDatamembersExamples.*bo*);

System.***out***.println(StaticDatamembersExamples.*str*);

}

}

**EVEN NUMBER**

**package** com.ibm.EvenNumber;

**public** **class** EvenNumber {

**public** **static** **void** main(String[] args) {

**int** i=100;

**for**(**int** x=0;x<i;x++){

**if**(x%2==0){

System.***out***.println(x);

}

}

}

}

ODD NUMBER

**package** com.ibm.OddNumber;

**public** **class** OddNumber {

**public** **static** **void** main(String[] args) {

**int** i=100;

**for**(**int** x=0;x<i;x++){

**if**(x%2!=0){

System.***out***.println(x);

}

}

}

}

**EXCEPTION EXAMPLES**

**package** com.ibm.Exception;

**public** **class** AllExceptions {

**public** **void** intdata(){

**int** a[]=**new** **int**[5];//declaration and instantiation

a[0]=10;//initialization

a[1]=20;

a[2]=70;

a[3]=40;

a[4]=50;

a[5]=60;

//traversing array

**for**(**int** i=0;i<a.length;i++)//length is the property of array

System.***out***.println(a[i]);

}

**public** **static** **void** main(String[] args) {

AllExceptions all=**new** AllExceptions();

all.intdata();

}

}

**package com.ibm.Exception;**

**import java.io.File;**

**import java.io.IOException;**

**public class CheckedUncheked {**

**public void createFile(){**

**File files=new File("D://Srinu.txt");**

**try {**

**boolean createNewFile = files.createNewFile();**

**System.out.println("sucessfully file is created::"+createNewFile);**

**} catch (IOException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**}**

**public void unchecked() throws MyException{**

**try{**

**int x=10;**

**int y=0;**

**System.out.println(x/y);**

**}**

**catch(ArithmeticException ae){**

**throw new MyException("not divide by zero");**

**}**

**}**

**public static void main(String[] args) throws MyException {**

**CheckedUncheked unchecked=new CheckedUncheked();**

**//unchecked.createFile();**

**unchecked.unchecked();**

**}**

**}**

**package** com.ibm.Exception;

**public** **class** ExceptionExample {

**private** **int** age;

**public** **void** eligibility(**int** age) **throws** MyException{

**if**(age>18){

System.***out***.println("YOU ARE eligibility FOR VOTE");

}

**else** **if**(age>=18)

{

System.***out***.println("YOU ARE eligibility FOR VOTE becoz18");

}

**else**{

**throw** **new** MyException("YOU ARE NOT ELIGIBLITY FOR VOTE BECOZ YOUR AGE LESSTHEN 18");

}

}

**public** **static** **void** main(String[] args) **throws** MyException {

ExceptionExample ex=**new** ExceptionExample();

ex.eligibility(18);

}

}

**MYEXCEPTION**

**package** com.ibm.Exception;

**public** **class** MyException **extends** Exception{

**public** MyException(String massage){

**super**(massage);

}

}

**package** com.ibm.Exception;

**public** **class** TryCatchFinallyExample {

//String str=null;

**public** **void** stringData(String str){

**try**{

**if**(!str.contains("null")){

System.***out***.println("hi it is contain data");

}

}

**catch**(NullPointerException ne){

System.***out***.println("given value is null");

}

}

**public** **void** division(){

**try**{

**int** x=10;

**int** y=5;

System.***out***.println("the value is::"+x/y);

}

**catch**(ArithmeticException ae){

System.***out***.println("plese change by zero");

}

}

**public** **static** **void** main(String[] args) {

TryCatchFinallyExample tcf=**new** TryCatchFinallyExample();

//tcf.stringData("srinu");

tcf.division();

}

}

**package com.ibm.Exception;**

**import java.io.File;**

**import java.io.FileNotFoundException;**

**import java.io.IOException;**

**public class TryCatchFinallyExample2 {**

**public void createFile() throws FileNotFoundException {**

**File file=new File("MyTxt.txt");**

**try {**

**boolean fileexist = file.exists();**

**System.out.println(fileexist);**

**}**

**catch (NullPointerException ne) {**

**// TODO Auto-generated catch block**

**ne.printStackTrace();**

**}**

**catch (ArithmeticException ae) {**

**// TODO Auto-generated catch block**

**ae.printStackTrace();**

**}**

**catch (Exception e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace(); }**

**}**

**public static void main(String[] args) {**

**TryCatchFinallyExample2 tcf=new TryCatchFinallyExample2();**

**try {**

**tcf.createFile();**

**} catch (FileNotFoundException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();}}**

**}**

**FEBONICSERIES EXAMPLE**

**package** com.ibm.FibonacciExample1;

**public** **class** FB2 {

**public** **static** **void** main(String[] args) {

**int** n1=0;

**int** n2=1;

**int** n3;

**int** count=10;

System.***out***.print(n1+""+n2);

**for**(**int** i=2;i<count;++i){

n3=n1+n2;

n1=n2;

n2=n3;

System.***out***.print(n3);

}

}

}

**package** com.ibm.FibonacciExample1;

**public** **class** FibonacciExample1 {

**public** **static** **void** main(String[] args) {

**int** n1=0,n2=1,n3,i,count=10;

System.***out***.print(n1+" "+n2);//printing 0 and 1

**for**(i=2;i<count;++i)//loop starts from 2 because 0 and 1 are already printed

{

n3=n1+n2;

System.***out***.print(" "+n3);

n1=n2;

n2=n3;

}

}

}

//0 1 1 2 3 5 8 13 21 34

**FILES WRITERS AND READER**

**package com.ibm.File;**

**import java.io.BufferedWriter;**

**import java.io.File;**

**import java.io.FileWriter;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**public class FileExample {**

**public static void main(String[] args) {**

**File file=new File("D://FILE/SRINU.TXT");**

**boolean exists = file.exists();**

**System.out.println(exists);**

**try {**

**boolean createNewFile = file.createNewFile();**

**if(createNewFile){**

**System.out.println("File sucessfully created::"+createNewFile);**

**}**

**else{**

**System.out.println("File not created::"+createNewFile);**

**}**

**} catch (IOException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**try {**

**FileWriter fw=new FileWriter("D://FILE/SRINU.TXT");**

**String str="HI THIS IS SRINIVASRAO i live in hyderabad";**

**fw.write(str);**

**fw.flush();**

**System.out.println("SUCESSFULLY WRITTEN");**

**} catch (IOException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**FileWriter fw;**

**try {**

**fw = new FileWriter("D://FILE/SRINU.TXT");**

**BufferedWriter bw=new BufferedWriter(fw);**

**String str="HI THIS IS SRINIVASRAO i live in hyderabad.i have done mca in 2015";**

**bw.write(str);**

**bw.flush();**

**} catch (IOException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**FileWriter fw2;**

**try {**

**fw2 = new FileWriter("D://FILE/SRINU.TXT");**

**PrintWriter pw=new PrintWriter(fw2);**

**pw.println("srinu");**

**pw.println("naresh");**

**pw.println("naven");**

**pw.println("viru");**

**pw.println("rakesh");**

**pw.println("naredra");**

**pw.flush();**

**} catch (IOException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**}**

**}**

**package com.ibm.File;**

**import java.io.File;**

**import java.io.FileOutputStream;**

**import java.io.FileWriter;**

**import java.io.IOException;**

**public class FileAndBufferdOutputStreams {**

**public static void main(String[] args) throws IOException {**

**File file=new File("D://FILE/SRINU2.TXT");**

**FileOutputStream fos=new FileOutputStream(file);**

**String str="gghjdfshjhhshshjjshhb dajbsf mbdbsb nbdsabs";**

**byte[] bytes = str.getBytes();**

**fos.write(bytes);**

**fos.flush();**

**System.out.println(str);**

**}**

**}**

**package com.ibm.File;**

**import java.io.BufferedReader;**

**import java.io.FileNotFoundException;**

**import java.io.FileReader;**

**import java.io.IOException;**

**public class FileReaderExample {**

**public static void main(String[] args) throws IOException {**

**try {**

**FileReader fr=new FileReader("D://FILE/SRINU.TXT");**

**int i;**

**while((i=fr.read())!=-1)**

**System.out.print((char)i);**

**fr.close();**

**} catch (FileNotFoundException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();}**

**System.out.println();**

**FileReader fr=new FileReader("D://FILE/SRINU.TXT");**

**BufferedReader br=new BufferedReader(fr);**

**int read = br.read();**

**while((read=br.read())!=-1){**

**System.out.print((char)read);}**

**}**

**}**

**HAS-A RELATION**

**package** com.ibm.HasARelation;

**class** Test{

Student s=**new** Student();

**public** **void** m1(){

System.***out***.println("parent test");

}

}

**class** Test2 **extends** Test{

**public** **void** m2(){

System.***out***.println("it is child information ");

**int** id = s.getId();

s.getAddress();

}

}

**public** **class** HasARelationExample {

**public** **static** **void** main(String[] args) {

Test2 t=**new** Test2();

t.m1();

t.m2();

}

}

**INHERITANCE**

**package** com.ibm.Inheritance;

**class** Parent{

**synchronized** **void** m1(){

System.***out***.println("parent class iformation");

}

**static** **void** m2(){

System.***out***.println("parent class iformation");

}

**protected** **void** m3(){

System.***out***.println("parent class iformation");

}

**public** **void** m4(){

System.***out***.println("parent class iformation");

}

**private** **void** m5(){

System.***out***.println("parent class iformation");

}

**void** m6(){

System.***out***.println("parent class iformation");

}

**final** **void** m7(){

System.***out***.println("parent class iformation");

}

**strictfp** **void** m8(){

System.***out***.println("parent class iformation");

}

**native** **void** m9();

}

**class** Child **extends** Parent{

**public** **void** m10(){

System.***out***.println("child data");

}

}

**public** **class** InheritanceExample {

**public** **static** **void** main(String[] args) {

/\*Parent p=new Child();

p.m1();\*/

Child c=**new** Child();

c.m1();

c.*m2*();

c.m3();

c.m4();

//c.m5();

c.m6();

c.m7();

c.m8();

c.m9();

c.m10();

}

}

**INSTANCE BLOCK**

**package** com.ibm.InstanceBlock;

**public** **class** InstanceBlockExample {

{

System.***out***.println("Instance Block1");

}

{

System.***out***.println("Instance Block2");

}

{

System.***out***.println("Instance Block3");

}

{

System.***out***.println("Instance Block4");

}

InstanceBlockExample(){

System.***out***.println("this is Constructer");

}

**public** **static** **void** main(String[] args) {

InstanceBlockExample iblock=**new** InstanceBlockExample();}

}

**INTEGER DUBLICATE**

package com.ibm.IntDublicate;

import java.util.HashMap;

import java.util.Map;

import java.util.Set;

public class IntDublicateExample {

public void intDublicate(int[] itr) {

Map<Integer, Integer> map = new HashMap<Integer, Integer>();

for (Integer it : itr) {

if (map.containsKey(it)) {

map.put(it, map.get(it) + 1);

} else {

map.put(it, 1);

}

}

Set<Integer> keySet = map.keySet();

for (Integer id : keySet) {

if(map.get(id)>1){

System.out.println("key:::"+id+"value:::"+map.get(id));

}

}

}

public static void main(String[] args) {

IntDublicateExample id=new IntDublicateExample();

int[] inte={1,2,1,5,4,1,2,5,8,4,5,6,9,5,7,2};

id.intDublicate(inte);

}

}

**INTEGER REVERSE**

**package** com.ibm.IntegerReverse;

**public** **class** IntegerReverse {

**public** **static** **void** main(String[] args) {

**int** reverse = 0;

**int** it = 558898436;

**while** (it != 0) {

reverse = reverse \* 10;

reverse = reverse + it % 10;

it = it / 10;

}

System.***out***.println("Reverse number is::" + reverse);

}

}

**INTERFACE AND ABSTRACT**

**package** com.ibm.InterfaceAndAbstract;

**interface** Interf{

**public** **int** m1();

**int** m2(**int** a, **int** b);

**void** m3();

**public** **abstract** **void** m4();

**default** **void** m5(){}

**public** **static** **void** m6(){}

}

**class** Test **implements** Interf{

@Override

**public** **int** m1() {

System.***out***.println("it is a child m1 method");

**return** 0;

}

@Override

**public** **int** m2(**int** a, **int** b) {

// **TODO** Auto-generated method stub

System.***out***.println(a+b);

**return** a+b;

}

@Override

**public** **void** m3() {

System.***out***.println("this m3 methos for child");

}

@Override

**public** **void** m4() {

System.***out***.println("this is child m4 method");

}

**public** **void** m5(){

System.***out***.println("it is default method");

}

**public** **static** **void** m6(){

System.***out***.println("it is static method");

}

}

**public** **class** InterfAbsractExample {

**public** **static** **void** main(String[] args) {

Test t=**new** Test();

t.m1();

t.m2(1, 25);

t.m3();

t.m4();

t.m5();

}

}

**RANDOM NUMBER**

**package** com.ibm.Randaom;

**import** java.util.Random;

**public** **class** RandomExample {

**public** **static** **void** main(String[] args) {

Random r=**new** Random();

**int** nextInt = r.nextInt(10000);

System.***out***.println(nextInt);

}

}

**LIST EXAMPLE**

**package com.ibm.List;**

**import java.util.ArrayList;**

**import java.util.LinkedList;**

**import java.util.List;**

**import java.util.Stack;**

**import java.util.Vector;**

**public class ListExample {**

**public static void main(String[] args) {**

**List<PlanetSoft> list=new ArrayList<>();**

**list.add(new PlanetSoft(101, "mahaveertowers", "Motinagar"));**

**list.add(new PlanetSoft(102, "aparamtowers", "khammam"));**

**list.add(new PlanetSoft(103, "yellareTowers", "srnagar"));**

**list.add(new PlanetSoft(104, "vkptowers", "ameerpet"));**

**System.out.println(list);**

**List<PlanetSoft> list2=new LinkedList<>();**

**list2.add(new PlanetSoft(101, "mahaveertowers", "Motinagar"));**

**list2.add(new PlanetSoft(102, "aparamtowers", "khammam"));**

**list2.add(new PlanetSoft(103, "yellareTowers", "srnagar"));**

**list2.add(new PlanetSoft(104, "vkptowers", "ameerpet"));**

**System.out.println(list2);**

**List<PlanetSoft> list3=new Vector<PlanetSoft>();**

**list3.add(new PlanetSoft(101, "mahaveertowers", "Motinagar"));**

**list3.add(new PlanetSoft(102, "aparamtowers", "khammam"));**

**list3.add(new PlanetSoft(103, "yellareTowers", "srnagar"));**

**list3.add(new PlanetSoft(104, "vkptowers", "ameerpet"));**

**System.out.println(list3);**

**Vector<PlanetSoft> list4=new Stack<PlanetSoft>();**

**list4.add(new PlanetSoft(101, "mahaveertowers", "Motinagar"));**

**list4.add(new PlanetSoft(102, "aparamtowers", "khammam"));**

**list4.add(new PlanetSoft(103, "yellareTowers", "srnagar"));**

**list4.add(new PlanetSoft(104, "vkptowers", "ameerpet"));**

**System.out.println(list4);**

**}**

**}**

**SET EXAMPLE**

**package com.ibm.Set;**

**import java.util.Comparator;**

**import java.util.HashSet;**

**import java.util.LinkedHashSet;**

**import java.util.NavigableMap;**

**import java.util.NavigableSet;**

**import java.util.Set;**

**import java.util.SortedSet;**

**import java.util.TreeSet;**

**public class SetExample {**

**public static void main(String[] args) {**

**Set<PlanetSoft> set=new HashSet<>();**

**set.add(new PlanetSoft(101, "rajivbulding", "rajivNagar"));**

**set.add(new PlanetSoft(102, "begumpetBuliding", "begumpet"));**

**set.add(new PlanetSoft(103, "mahaveertowers", "motinagar"));**

**set.add(new PlanetSoft(104, "gatedcomunity", "hyderabad"));**

**set.add(new PlanetSoft(105, "venkatagiribuliding", "krishnanagar"));**

**System.out.println(set);**

**Set<PlanetSoft> set2=new LinkedHashSet<>();**

**set2.add(new PlanetSoft(101, "rajivbulding", "rajivNagar"));**

**set2.add(new PlanetSoft(102, "begumpetBuliding", "begumpet"));**

**set2.add(new PlanetSoft(103, "mahaveertowers", "motinagar"));**

**set2.add(new PlanetSoft(104, "gatedcomunity", "hyderabad"));**

**set2.add(new PlanetSoft(105, "venkatagiribuliding", "krishnanagar"));**

**System.out.println(set2);**

**Set<Integer> set3=new TreeSet<Integer>();**

**set3.add(10);**

**set3.add(1);**

**set3.add(8);**

**set3.add(7);**

**set3.add(6);**

**System.out.println(set3);**

**Set<String> set4=new TreeSet<String>();**

**set4.add("B");**

**set4.add("Z");**

**set4.add("K");**

**set4.add("P");**

**set4.add("L");**

**System.out.println(set4);**

**Set<PlanetSoft> set5=new LinkedHashSet<>();**

**set5.add(new PlanetSoft(101, "rajivbulding", "rajivNagar"));**

**set5.add(new PlanetSoft(101, "rajivbulding", "rajivNagar"));**

**set5.add(new PlanetSoft(101, "rajivbulding", "rajivNagar"));**

**set5.add(new PlanetSoft(101, "rajivbulding", "rajivNagar"));**

**set5.add(new PlanetSoft(101, "rajivbulding", "rajivNagar"));**

**set5.add(new PlanetSoft(101, "rajivbulding", "rajivNagar"));**

**System.out.println(set5);}}**

**MAP EXAMPLE**

**package com.ibm.Map;**

**import java.util.HashMap;**

**import java.util.Hashtable;**

**import java.util.LinkedHashMap;**

**import java.util.Map;**

**import java.util.TreeMap;**

**public class MapExample {**

**public static void main(String[] args) {**

**Map<String,Integer>map=new HashMap<>();**

**map.put("srinu", 10);**

**map.put("naresh", 8);**

**map.put("bharat", 4);**

**map.put("ramu", 3);**

**map.put("zkiran", 6);**

**System.out.println(map);**

**Map<String,Integer>map2=new LinkedHashMap<>();**

**map2.put("srinu", 10);**

**map2.put("naresh", 8);**

**map2.put("bharat", 4);**

**map2.put("ramu", 3);**

**map2.put("zkiran", 6);**

**System.out.println(map2);**

**Map<String,Integer>map3=new Hashtable<>();**

**map3.put("srinu", 10);**

**map3.put("naresh", 8);**

**map3.put("bharat", 4);**

**map3.put("ramu", 3);**

**map3.put("zkiran", 6);**

**System.out.println(map3);**

**Map<String,Integer>map4=new TreeMap<>();**

**map4.put("srinu", 10);**

**map4.put("naresh", 8);**

**map4.put("bharat", 4);**

**map4.put("ramu", 3);**

**map4.put("zkiran", 6);**

**System.out.println(map4);**

**Map<PlanetSoft,Integer>map5=new LinkedHashMap<>();**

**map5.put(new PlanetSoft(101, "B", "KHAMMAM"), 5);**

**map5.put(new PlanetSoft(101, "B", "KHAMMAM"), 2);**

**map5.put(new PlanetSoft(101, "B", "KHAMMAM"), 1);**

**map5.put(new PlanetSoft(101, "B", "KHAMMAM"), 8);**

**map5.put(new PlanetSoft(101, "B", "KHAMMAM"), 7);**

**System.out.println(map5);**

**} }**

**MAP SORTING**

**package com.ibm.MapSorted;**

**import java.util.ArrayList;**

**import java.util.Collections;**

**import java.util.Comparator;**

**import java.util.HashMap;**

**import java.util.List;**

**import java.util.Map;**

**import java.util.Map.Entry;**

**import java.util.Set;**

**import java.util.TreeSet;**

**import com.ibm.Map.PlanetSoft;**

**public class MapSortExample {**

**@SuppressWarnings("unchecked")**

**public static void main(String[] args) {**

**Map<PlanetSoft,Integer> map=new HashMap();**

**map.put(new PlanetSoft(15, "B", "MOTINAGAR"), 5);**

**map.put(new PlanetSoft(1, "A", "SRNAGAR"), 15);**

**map.put(new PlanetSoft(5, "D", "AMEERPET"), 7);**

**map.put(new PlanetSoft(12, "Z", "DELHI"), 2);**

**map.put(new PlanetSoft(8, "F", "KPHB"), 1);**

**Set<Entry<PlanetSoft, Integer>> entrySet = map.entrySet();**

**List list = new ArrayList<>(entrySet);**

**Collections.sort(list, new Comparator<Entry<PlanetSoft, Integer>>() {**

**@Override**

**public int compare(Entry<PlanetSoft, Integer> o1, Entry<PlanetSoft, Integer> o2) {**

**// TODO Auto-generated method stub**

**return o1.getKey().getBulidingId()<o2.getKey().getBulidingId() ?-1:o1.getKey().getBulidingId()>o1.getKey().getBulidingId() ? 1:0;**

**}**

**});**

**System.out.println(list);**

**}**

**}**

**OBJECT EXAMPLE**

**package** com.ibm.Object;

**public** **class** ObjectExamples {

**public** **static** **void** main(String[] args) **throws** InterruptedException {

String str="srinu";

String str2=**new** String("srinu");

System.***out***.println(str.toString());

System.***out***.println(str.hashCode());

System.***out***.println(str.equals(str2));

System.***out***.println(str.getClass().getName());

}

}

**OVERLOADING EXAMPLE**

**package** com.ibm.Overloading;

**public** **class** OverloadingExample {

**public** **final** **void** search(Integer itr) {

System.***out***.println("get integer data");

}

**private** String search(String str) {

System.***out***.println(str);

**return** "it is string data" + str;

}

**public** **static** **void** main(String[] args) {

OverloadingExample ole = **new** OverloadingExample();

ole.search(1);

ole.search("srinu");

}

}

OVERRIDING EXAMPLE

**package** com.ibm.Overriding;

**abstract** **class** Parent{

**public** **abstract** **void** test();

**int** m1(){

System.***out***.println("parent synchronized iformation");

**return** 0;

}

**static** **void** m2(){

System.***out***.println("parent static iformation");

}

**protected** **void** m3(){

System.***out***.println("parent protected iformation");

}

**public** **void** m4(){

System.***out***.println("parent public iformation");

}

**private** **void** m5(){

System.***out***.println("parent private iformation");

}

**void** m6(){

System.***out***.println("parent default iformation");

}

**final** **void** m7(){

System.***out***.println("parent final iformation");

}

**strictfp** **void** m8(){

System.***out***.println("parent strictfp iformation");

}

}

**class** Child **extends** Parent{

**strictfp** **int** m1(){

System.***out***.println("child synchronized iformation");

**return** 0;

}

**public** **static** **void** m2(){

System.***out***.println("child static iformation");

}

**protected** **void** m3(){

System.***out***.println("child protected iformation");

}

**public** **void** m4(){

System.***out***.println("child public iformation");

}

**private** **void** m5(){

System.***out***.println("child private iformation");

}

**void** m6(){

System.***out***.println("child default iformation");

}

/\* final void m7(){

System.out.println("parent class iformation");

}\*/

**void** m8(){

System.***out***.println("child strictfp iformation");

}

@Override

**public** **void** test() {

// **TODO** Auto-generated method stub

}

}

**public** **class** OverridingExamples {

**public** **static** **void** main(String[] args) {

Child c=**new** Child();

c.m1();

c.*m2*();

c.m3();

c.m4();

//c.m5();

c.m6();

c.m7();

c.m8();

}

}

**PLINDROME EXAMPLE**

**package** com.ibm.PalindromeExample;

**public** **class** PalindromeExample {

**int** r,sum=0,temp;

**int** n=454; //It is the number variable to be checked for palindrome

{

temp=n;

**while**(n>0){

r=n%10; //getting remainder

sum=(sum\*10)+r;

n=n/10;

}

**if**(temp==sum)

System.***out***.println("palindrome number ");

**else**

System.***out***.println("not palindrome");

}

}

PRE AND POST INCREMENT

**package** com.ibm.PreAndPostIncrement;

**public** **class** PreandPostIncrement {

**public** **static** **void** main(String[] args) {

**int** i=1;

System.***out***.println(i++);//uses then increment

System.***out***.println(++i);//increment then uses

}

}

**PRIME NUMBER EXAMPLE**

**package** com.ibm.Prime;

**public** **class** PrimeNumberExample {

**public** **static** **void** main(String[] args) {

**int** temp;

**int** number=103;

**boolean** isPrime=**true**;

**for**(**int** i=2;i<=number/2;i++){

temp=number%i;

**if**(temp==0){

isPrime=**false**;

**break**;

}

}

**if**(isPrime)

System.***out***.println("is primenumber:::"+ number);

**else**{

System.***out***.println("is not a primenumber:::"+ number);

}

}

}

QUEUE EXAMPLE

package com.ibm.Queue;

import java.util.LinkedList;

import java.util.Queue;

public class QueueExample {

public static void main(String[] args) {

Queue<String> q=new LinkedList<>();

q.add("B");

q.add("C");

q.add("M");

q.add("Z");

q.add("L");

System.out.println(q);

}

}

**SCANNER EXAMPLE**

**package** com.ibm.Scanner;

**import** java.util.Scanner;

**public** **class** ScannerExample {

**public** **static** **void** main(String[] args) {

String s = "Hello, This is JavaTpoint.";

//Create scanner Object and pass string in it

Scanner scan = **new** Scanner(s);

//Check if the scanner has a token

System.***out***.println("Boolean Result: " + scan.hasNext());

//Print the string

System.***out***.println("String: " +scan.nextLine());

scan.close();

System.***out***.println("--------Enter Your Details-------- ");

Scanner in = **new** Scanner(System.***in***);

System.***out***.print("Enter your name: ");

String name = in.next();

System.***out***.println("Name: " + name);

System.***out***.print("Enter your age: ");

**int** i = in.nextInt();

System.***out***.println("Age: " + i);

System.***out***.print("Enter your salary: ");

**double** d = in.nextDouble();

System.***out***.println("Salary: " + d);

in.close();

}

}

**SECOND LARGEST NUMBER**

**package** com.ibm.SecondLargest;

**public** **class** SecondLargest {

**public** **static** **void** main(String[] args) {

**int** arr[] = { 14, 46, 47, 86, 92, 52, 48, 36, 66, 85 };

**int** largest = arr[0];

**int** secondLargest = arr[0];

System.***out***.println("The given array is:" );

**for** (**int** i = 0; i < arr.length; i++) {

System.***out***.print(arr[i]+"\t");

}

**for** (**int** i = 0; i < arr.length; i++) {

**if** (arr[i] > largest) {

secondLargest = largest;

largest = arr[i];

} **else** **if** (arr[i] > secondLargest) {

secondLargest = arr[i];

}

}

System.***out***.println("\nSecond largest number is:" + secondLargest);

}

}

**SERILIZATION EXAMPLE**

**package com.ibm.Serilization;**

**import java.io.FileInputStream;**

**import java.io.FileNotFoundException;**

**import java.io.FileOutputStream;**

**import java.io.IOException;**

**import java.io.ObjectInputStream;**

**import java.io.ObjectOutputStream;**

**import java.io.Serializable;**

**public class SerilizationAndDeserilzation implements Serializable{**

**public static final long serialVersionUID = 1L;**

**transient static int i = 10;**

**transient int j = 20;**

**public void serialize() throws IOException {**

**try {**

**FileOutputStream fos=new FileOutputStream("D://FILE/SERILIZE.txt");**

**ObjectOutputStream oos=new ObjectOutputStream(fos);**

**SerilizationAndDeserilzation sad=new SerilizationAndDeserilzation();**

**oos.writeObject(sad);**

**System.out.println("sucessfully serilized");**

**} catch (FileNotFoundException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**}**

**public void deserilize() throws IOException, ClassNotFoundException{**

**try {**

**FileInputStream fis=new FileInputStream("D://FILE/SERILIZE.txt");**

**ObjectInputStream ois=new ObjectInputStream(fis);**

**SerilizationAndDeserilzation si=(SerilizationAndDeserilzation)ois.readObject();**

**System.out.println(si.i);**

**System.out.println(si.j);**

**} catch (FileNotFoundException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**}**

**public static void main(String[] args) throws IOException, ClassNotFoundException {**

**SerilizationAndDeserilzation sad=new SerilizationAndDeserilzation();**

**sad.serialize();**

**sad.deserilize();}**

**}**

**SINGLETON EXAMPLE**

**package** com.ibm.Singleton;

**public** **class** SingletonExample {

**private** **static** SingletonExample *singletonExample*;

**private** SingletonExample(){}

**public** **static** SingletonExample getInstance(){

**if**(*singletonExample*==**null**){

*singletonExample*=**new** SingletonExample();

}

**return** *singletonExample*;

}

**public** **static** **void** main(String[] args) {

}

}

**SORTEDLIST EXAMPLE**

**package com.ibm.SortedList;**

**import java.util.ArrayList;**

**import java.util.Collections;**

**import java.util.Comparator;**

**import java.util.List;**

**public class SortedListExample {**

**public static void main(String[] args) {**

**List<PlanetSoft> list=new ArrayList<>();**

**list.add(new PlanetSoft(100, "Z", "KHAMMAM"));**

**list.add(new PlanetSoft(50, "C", "WYRA"));**

**list.add(new PlanetSoft(100, "B", "XEROX"));**

**list.add(new PlanetSoft(100, "K", "DELHI"));**

**list.add(new PlanetSoft(100, "W", "SIGAPUR"));**

**list.add(new PlanetSoft(100, "D", "KHAMMAM"));**

**Collections.sort(list, new Comparator<PlanetSoft>() {**

**@Override**

**public int compare(PlanetSoft o1, PlanetSoft o2) {**

**return o2.getBulidingName().compareTo(o1.getBulidingName());**

**}**

**});**

**System.out.println(list);**

**} }**

**SORTED SET EXAMPLE**

**package com.ibm.SortedSet;**

**import java.util.ArrayList;**

**import java.util.Collections;**

**import java.util.Comparator;**

**import java.util.HashSet;**

**import java.util.List;**

**import java.util.Set;**

**public class SortedSetExample {**

**public static void main(String[] args) {**

**Set<PlanetSoft> set = new HashSet<>();**

**set.add(new PlanetSoft(200, "B", "WARANGAL"));**

**set.add(new PlanetSoft(300, "A", "KHAMMAM"));**

**set.add(new PlanetSoft(100, "D", "HYDERAB"));**

**set.add(new PlanetSoft(145, "Z", "DELHI"));**

**set.add(new PlanetSoft(750, "K", "SIGPURA"));**

**List<PlanetSoft> list=new ArrayList<>(set);**

**Collections.sort(list, new Comparator<PlanetSoft>() {**

**@Override**

**public int compare(PlanetSoft o1, PlanetSoft o2) {**

**// TODO Auto-generated method stub**

**//return o1.getBulidingId()<o2.getBulidingId() ? -1:o1.getBulidingId()>o2.getBulidingId() ? 1:0;**

**return o1.getBulidingAddress().compareTo(o2.getBulidingAddress());**

**}});**

**System.out.println(list);**

**}**

**STRING EXAMPLE**

**package** com.ibm.String;

**public** **class** StringExample {

**public** **static** **void** main(String[] args) {

String str1="srinu";

String str2="srinu";

String str3="SRINU";

String str4=**new** String("srinu");

str4="srinu";

String str5=**new** String("srinu");

System.***out***.println(str4==str5);

System.***out***.println(str4.equals(str5));

String str6="srinuvasrao";

String str7="srinuvas";

str7=str7.concat("rao");

System.***out***.println(str1.equals(str2));//true

System.***out***.println(str1==str2);//true

System.***out***.println(str1.equals(str3));//false

System.***out***.println(str1.equalsIgnoreCase(str3));//true

System.***out***.println(str1==str3);//false

System.***out***.println(str1.equals(str4));//true

System.***out***.println(str1==str4);//false

System.***out***.println(str4.equals(str5));//true

System.***out***.println(str4==str5);//false

System.***out***.println(str6.equals(str7));//true

System.***out***.println(str6==str7);//false

}

}

**STRING REVERSE**

**package** com.ibm.String;

**public** **class** StringReverse {

**public** **static** **void** main(String[] args) {

String str="sreenivasarao";

**int** length = str.length();

**for**(**int** reverse=length-1;reverse>=0;reverse--){

//System.out.print(str.charAt(reverse));

}

**int** reverse=0;

**int** it=5588984;

**while**(it !=0){

reverse = reverse \* 10;

reverse = reverse + it%10;

it = it/10;

}

System.***out***.println(reverse);

}

}

STRING BUFFER

**package** com.ibm.StringBuffer;

**public** **class** StringBufferdExample {

**public** **static** **void** main(String[] args) {

StringBuffer sb1=**new** StringBuffer("srinu");

StringBuffer sb2=**new** StringBuffer("srinu");

System.***out***.println(sb1.equals(sb2));//false

System.***out***.println(sb1==sb2);//false

System.***out***.println(sb1.reverse());

System.***out***.println(sb2.append("vasrao"));

System.***out***.println(sb2.hashCode());

}

}

STRING BULIDER

**package** com.ibm.StringBuilder;

**public** **class** StringBuilderExample {

**public** **static** **void** main(String[] args) {

StringBuilder sb1=**new** StringBuilder("srinu");

StringBuilder sb2=**new** StringBuilder("srinu");

StringBuffer sb3=**new** StringBuffer("srinu");

System.***out***.println(sb1.equals(sb2));

System.***out***.println(sb1==sb2);

System.***out***.println(sb1.equals(sb3));

//System.out.println(sb1==sb3);

System.***out***.println(sb1.reverse());

}

}

STRING DUBLICATE

package com.ibm.StringDublicate;

import java.util.HashMap;

import java.util.Map;

import java.util.Map.Entry;

import java.util.Set;

public class StringDublicate {

public void dublicateString(String str) {

char[] charArray = str.toCharArray();

Map<Character,Integer> map=new HashMap<Character,Integer>();

for(Character ch:charArray){

if(map.containsKey(ch)){

map.put(ch, map.get(ch) +1);

}

else{

map.put(ch, 1);

}

}

Set<Character> keySet = map.keySet();

for (Character character : keySet) {

if(map.get(character)>1){

System.out.println("key====="+character +"value=====" +map.get(character));

}

}

}

public static void main(String[] args) {

StringDublicate sd=new StringDublicate();

sd.dublicateString("sreenivasarao");

}

}

**SUM OF ARRAY**

**package** com.ibm.SumofArray;

**public** **class** SumofArray {

**public** **static** **void** main(String[] args) {

**int** sum=0;

**int**[] it={2,55,2,85,45,2,6,7,3,9,1,0,8,5};

**for**(**int** i=0;i<it.length;i++){

sum+=it[i];

//or

sum=sum+i;

}

System.***out***.println(sum);

System.***out***.println("sum of array is::"+sum);

}

}

**SUPER AND THIS KEYWORDS**

**package** com.ibm.SuperAndThis;

**class** Test{

**public** **int** id;

Test(**int** id){

**this**.id=id;

System.***out***.println("this is parent id");

}

}

**class** Test2 **extends** Test{

Test2(**int** id) {

**super**(id);

**int** i = **super**.id;

System.***out***.println(i);

System.***out***.println("this is child");

}

}

**public** **class** SuperAndThisExample {

**public** **static** **void** main(String[] args) {

Test2 t=**new** Test2(1);

}

}

**DEADLOCK EXAMPLE**

**package** com.ibm.Thread;

**public** **class** DeadLockExample {

String str1 = "srinu";

String str2 = "rajee";

Thread t1=**new** Thread("it is thread 1"){

**public** **void** run(){

**while**(**true**){

**synchronized** (str1) {

**synchronized** (str2) {

System.***out***.println(str1+str2);

}

}

}

}

};

Thread t2 = **new** Thread("it is thread2"){

**public** **void** run() {

**while** (**true**) {

**synchronized** (str2) {

**synchronized** (str1) {

System.***out***.println(str2 + str1);

}

}

}

}

};

**public** **static** **void** main(String[] args) {

DeadLockExample deadlock=**new** DeadLockExample();

deadlock.t1.start();

deadlock.t2.start();

}

}

**RUNNABLE EXAMPLE**

**package** com.ibm.Thread;

**public** **class** RunableExample **implements** Runnable {

@Override

**public** **void** run() {

**for**(**int** i=0;i<10;i++){

System.***out***.println("child thread");}

}

**public** **static** **void** main(String[] args) {

RunableExample r=**new** RunableExample();

Thread t=**new** Thread(r);

t.start();

**for**(**int** i=0;i<10;i++){

System.***out***.println("main thread");

}

}

}

**SYNCHRONIZATION EXAMPLE**

**package** com.ibm.Thread;

**class** Test{

**public** **synchronized** **void** synchronize(**int** x){

**for**(**int** i=0;i<10;i++){

System.***out***.println("it is synchronize thread::"+(x+i));

}

**try** {

Thread.*sleep*(400);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}}}

**class** MyThread1 **extends** Thread{

**private** Test test;

MyThread1(Test test){

**this**.test=test;

}

**public** **void** run(){

test.synchronize(10);

}

}

**class** MyThread2 **extends** Thread{

**private** Test test;

**public** MyThread2(Test test) {

**this**.test=test;

}

**public** **void** run(){

test.synchronize(20);

}}

**public** **class** SynchronizationExample {

**public** **static** **void** main(String[] args) {

Test t1=**new** Test();

MyThread1 mythread1=**new** MyThread1(t1);

MyThread2 mythread2=**new** MyThread2(t1);

mythread1.start();

mythread2.start();;

}}

SYNCHROZATION BLOCK EXAMPLE

**package** com.ibm.Thread;

**class** TestBlock{

**public** **void** sblock(**int** x){

**synchronized** (**this**) {

**for**(**int** i=0;i<10;i++){

System.***out***.println("sblock is::"+(x+i));

**try** {

Thread.*sleep*(300);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}}}}

**class** MyThreadBlock1 **extends** Thread{

**private** TestBlock testblock;

MyThreadBlock1(TestBlock testblock){

**this**.testblock=testblock;

}

**public** **void** run(){

testblock.sblock(10);

}

}

**class** MyThreadBlock2 **extends** Thread{

**private** TestBlock testBlock;

MyThreadBlock2( TestBlock testBlock){

**this**.testBlock=testBlock;

}

**public** **void** run(){

testBlock.sblock(20);

}

}

**public** **class** SynchronizedBlockExample {

**public** **static** **void** main(String[] args) {

TestBlock block=**new** TestBlock();

MyThreadBlock1 thread1=**new** MyThreadBlock1(block);

MyThreadBlock2 thread2=**new** MyThreadBlock2(block);

thread1.start();

thread2.start();

}}

THREAD EXAMPLE

**package** com.ibm.Thread;

**public** **class** ThreadExample **extends** Thread{

**public** **void** run(){

**for**(**int** i=0;i<10;i++){

System.***out***.println("child thread");

}

}

**public** **static** **void** main(String[] args) **throws** InterruptedException {

ThreadExample t1=**new** ThreadExample();

t1.start();

t1.*yield*();

t1.*sleep*(1000);

**for**(**int** i=0;i<10;i++){

System.***out***.println("main thread");

}

}

}

**STAR AND NUMBER PRINT EXAMPLES**

**public** **class** StarPrints {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**for**(**int** i=0;i<=5;i++){

**for**(**int** j=0;j<i;j++){

System.***out***.print("\*");

}

System.***out***.println("");

}

}

}

**public** **class** Starprint2 {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**for**(**int** i=0;i<=5;i++){

**for**(**int** j=0;j<=5;j++){

System.***out***.print("\*");

}

System.***out***.println("");

}

}

}

**public** **class** StarPrint3 {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**for**(**int** i=5;i>0;i--){

**for**(**int** j=0;j<i;j++){

System.***out***.print("\*");

}

System.***out***.println("");

}

}

}

**public** **class** NumberPrint {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**for**(**int** i=0;i<=5;i++){

**for**(**int** j=0;j<i;j++){

System.***out***.print(j+1);

}

System.***out***.println("");

}

}

}

**public** **class** NumberPrint {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**for**(**int** i=0;i<=5;i++){

**for**(**int** j=0;j<i;j++){

System.***out***.print(j+1);

}

System.***out***.println("");

}

}

}

**JAVA8 LOGICAL PROGRAMS**

**MULTITHREADING EXAMPLE**

**package** com.java8.logicals;

**public** **class** MulitiThreadingExample1 {

**public** **static** **void** main(String[] args) {

Runnable r=() -> { **for**(**int** i=0;i<10;i++){

System.***out***.println("child thread");

}

};

Thread t=**new** Thread(r);

t.start();

**for**(**int** i=0;i<10;i++){

System.***out***.println("main Thread");

}

}

}

USER DEFINED FUNCTIONAL INTERFACE

**package** com.java8.logicals;

@FunctionalInterface

**interface** Test {

**public** **void** sum(**int** a, **int** b);

**public** **static** **void** mymethod(){

System.***out***.println("my static method");

}

**public** **static** **void** mymethod2(){

System.***out***.println("my static method2");

}

**default** **void** m1(){

System.***out***.println("my method as default");

}

}

**public** **class** FunctionalInterfaceUserdefinedExample {

**public** **static** **void** main(String[] args) {

Test t=(a,b)->System.***out***.println(a+b);

t.sum(20, 30);

Test.*mymethod*();

t.m1();

}

}

**package** com.java8.logicals;

**interface** Interf {

// public int square(int a);

**public** **void** m1(String s);

}

**public** **class** UserdefinedFunctionalIterfaceExample2 {

**public** **static** **void** main(String[] args) {

// Interf i=a->a\*a;

// System.out.println(i.square(5));

Interf i1 = s -> System.***out***.println(s.length());

i1.m1("sreenivasaro");

//System.out.println(s.);

}

}

COMPARATOR EXAMPLE

package com.java8.logicals;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

public class ComparatorExample {

public static void main(String[] args) {

ArrayList<Integer> list=new ArrayList<Integer>();

list.add(58);

list.add(3);

list.add(8);

list.add(98);

list.add(1);

list.add(12);

list.add(58);

list.add(528);

list.add(518);

System.out.println(list);

Collections.sort(list);

System.out.println(list);

Collections.reverse(list);

System.out.println(list);

Comparator<Integer> c=(o1,o2) ->(o1<o2)? -1:(o1>o2)? 1:0;

Collections.sort(list, c);

System.out.println(list)

}

}

**COMPARATOR STRING EXAMPLE**

package com.java8.logicals;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

public class ComparatoStringExample {

public static void main(String[] args) {

ArrayList<String> list=new ArrayList<String>();

list.add("srinu");

list.add("amar");

list.add("ajay");

list.add("bsrinu");

list.add("nsrinu");

list.add("ksrinu");

list.add("dsrinu");

list.add("rajee");

Collections.sort(list);

System.out.println(list);

Comparator<String> c=(o1,o2)->o2.compareTo(o1);

Collections.sort(list, c);

System.out.println(list);

}

}

**COMPARATOR OBJECT EXAMPLES**

package com.java8.logicals;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

public class ComparatorObjectExample {

public static void main(String[] args) {

ArrayList<Student> list = new ArrayList<Student>();

list.add(new Student(11, "suresh", 15, "amarvathi", 8));

list.add(new Student(1, "abhi", 25, "wgrl", 12));

list.add(new Student(10, "bstin", 18, "ngda", 9));

list.add(new Student(101, "nsrinu", 19, "medak", 11));

list.add(new Student(141, "rajee", 30, "hyd", 15));

list.add(new Student(111, "yadav", 9, "vzd", 5));

System.out.println(list);

/\*Comparator<Student> c = (o1, o2) -> o1.getStudentId() < o2.getStudentId() ? -1

: o1.getStudentId() > o2.getStudentId() ? 1 : 0;

Collections.sort(list, c);

System.out.println(list);\*/

/\*Comparator<Student> c1=(o1,o2)->o2.getStudentName().compareTo(o1.getStudentName());

Collections.sort(list, c1);

System.out.println(list);\*/

/\*Comparator<Student> c2=(o1,o2)->o1.getAddress().compareTo(o2.getAddress());

Collections.sort(list, c2);

System.out.println(list);\*/

Comparator<Student> c3=(o1,o2)->o1.getAge()>o2.getAge()?-1:o1.getAge()<o2.getAge()? 1:0;

Collections.sort(list, c3);

System.out.println(list);

}

}

**PREDICATE EXAMPLE1**

package com.java8.logicals;

import java.util.ArrayList;

import java.util.List;

import java.util.function.Predicate;

public class PredicateExamples {

public static void main(String[] args) {

List<Student> list=new ArrayList<Student>();

list.add(new Student(11, "suresh", 15, "amarvathi", 8));

list.add(new Student(1, "abhi", 25, "wgrl", 12));

list.add(new Student(10, "abhi", 18, "ngda", 9));

list.add(new Student(101, "suresh", 19, "wgrl", 11));

list.add(new Student(141, "sabhi", 30, "hyd", 15));

list.add(new Student(111, "srinuchowdary", 9, "vzd", 4));

Predicate<Student> p1= k->k.getAge()>18 && k.getAddress().equals("wgrl");

Predicate<Student> p2=o1->o1.getStudentName().equals(o1.getStudentName());

for (Student student : list) {

if(p2.test(student)){

System.out.println(student);

}

}

//System.out.println(p1.test(k));

/\*Predicate<Integer> p=i->i%2==0;

System.out.println(p.test(10));//true

System.out.println(p.test(25));//false

}

}

**PREDICATE EXAMPLE2**

**package** com.java8.logicals;

**import** java.util.function.Predicate;

**public** **class** PredicateExample2 {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int**[] x={0,5,10,15,20,25,30,35};

Predicate<Integer> p1=i->i%2==0;

Predicate<Integer> p2=i->i>10;

**for**(**int** i:x)

{

/\*if(p1.and(p2).test(i)){

System.out.println(i);//20,30 after applying both conditions

}\*/

**if**(p1.or(p2).test(i)){

System.***out***.println(i);//0,10,15,20,25,30,35 even either greterthen 10 here 5 missing

}

/\*if(p1.negate().test(i)){

System.out.println(i);//5,15,25,35

}\*/

}

}}

**FUNCTION EXAMPLES**

**CONSUMER EXAMPLE**

package com.java8.logicals;

import java.util.ArrayList;

import java.util.List;

import java.util.function.Consumer;

public class ConsumerExample {

public static void main(String[] args) {

List<Student> list=new ArrayList<Student>();

list.add(new Student(11, "suresh", 15, "amarvathi", 8));

list.add(new Student(1, "abhi", 25, "wgrl", 12));

list.add(new Student(10, "abhi", 18, "ngda", 9));

list.add(new Student(101, "suresh", 19, "wgrl", 11));

list.add(new Student(141, "sabhi", 30, "hyd", 15));

list.add(new Student(111, "srinuchowdary", 9, "vzd", 4));

Consumer<Student> c=s1 ->System.out.println(s1);

for (Student student : list) {

c.accept(student);

}

}

}

**SUPPILER EXAMPLES**

package com.java8.logicals;

import java.util.Date;

import java.util.function.Supplier;

public class SuppilerExample {

public static void main(String[] args) {

// TODO Auto-generated method stub

Supplier<Date> s=()-> new Date();

System.out.println(s.get());

}

}

**package** com.java8.logicals;

**import** java.util.function.Supplier;

**public** **class** SupplerExample2 {

**public** **static** **void** main(String[] args) {

Supplier<String> s=()->{

String otp="";

**for**(**int** i=0;i<6;i++){

otp=otp+(**int**)(Math.*random*()\*10);

}

**return** otp;

};

System.***out***.println(s.get());

}

}

**STREAMS EXAMPLE**

package com.java8.logicals;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

import java.util.Optional;

import java.util.Set;

import java.util.stream.Collectors;

public class StreamsExample1 {

public static void main(String[] args) {

List<Student> list=new ArrayList<Student>();

list.add(new Student(11, "suresh", 15, "amarvathi", 8));

list.add(new Student(1, "abhi", 25, "wgrl", 12));

list.add(new Student(10, "bstin", 18, "ngda", 9));

list.add(new Student(101, "nsrinu", 19, "medak", 11));

list.add(new Student(141, "sabhi", 30, "hyd", 15));

list.add(new Student(111, "srinuchowdary", 9, "vzd", 4));

// List<Student> l1=list.stream().filter(i->i.getAge()<10).collect(Collectors.toList());

List<Student> l2=list.stream().filter(i->i.getStudentName().startsWith("s")).collect(Collectors.toList());

List<Student> l3=list.stream().filter(i->i.getStudentName().equals("abhi")).collect(Collectors.toList());

List<Student> l4=list.stream().filter(i->i.getStudentName().length()>10).collect(Collectors.toList());

List<Student> l5=list.stream().filter(i->i.getAge()%2!=0).collect(Collectors.toList());

List<Integer> l6=list.stream().map(i->i.getAge()\*2).collect(Collectors.toList());

List<Integer> l7=list.stream().map(i->i.getAge() +5).collect(Collectors.toList());

List<Student> lnew=list.stream().sorted((s1,s2)->s1.getStudentId()<s2.getStudentId() ?-1:s1.getStudentId()>s2.getStudentId() ? 1:0).collect(Collectors.toList());

List<Student> l8=list.stream().sorted((s1,s2)->-s1.getAddress().compareTo(s2.getAddress())).collect(Collectors.toList());

List<Student> l9=list.stream().sorted((s1,s2)->-s1.getAddress().compareTo(s2.getAddress())).collect(Collectors.toList());

Student l10=list.stream().max((k1,k2)->k1.getAge()<k2.getAge()?-1:k1.getAge()>k2.getAge()? 1:0).get();

Student l11=list.stream().min((k1,k2)->k1.getAge()<k2.getAge()?-1:k1.getAge()>k2.getAge()? 1:0).get();

Student lnew1=list.stream().max((s1,s2)->s1.getAddress().compareTo(s2.getAddress())).get();

Student lnew2=list.stream().min((s1,s2)->s1.getAddress().compareTo(s2.getAddress())).get();

//List<Student> l12=list.stream().sorted(Collections.reverseOrder()).collect(Collectors.toList()).get(-1);

//List<Student>=list.stream().forEach(System.out::println);

System.out.println(lnew2);

}}

**BI-PREDICATE**

**package** com.java8.logicals;

**import** java.util.function.BiPredicate;

**public** **class** BipredicateExample {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

BiPredicate<Integer,Integer> bi=(a,b)->(a+b)%2==0;

System.***out***.println(bi.test(25, 51));//true

}

}

**BI-FUNCTION**

package com.java8.logicals;

import java.util.ArrayList;

import java.util.List;

import java.util.function.BiFunction;

public class BiFunctionExample {

public static void main(String[] args) {

// TODO Auto-generated method stub

List<Student1> list=new ArrayList<Student1>();

BiFunction<Integer, String, Student1> bf=(studentId,studentName)->new Student1(studentId,studentName);

list.add(bf.apply(250, "suresh"));

list.add(bf.apply(350, "naveen"));

list.add(bf.apply(850, "rajee"));

list.add(bf.apply(550, "mahesh"));

list.add(bf.apply(50, "suresh"));

for(Student1 s:list){

System.out.println(s.studentId);

System.out.println(s.studentName);

}

}

}

JAVA8 EXAMPLES(COLLECTIONS)

package com.ibm.Java8Features;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

public class CollectionsExample {

public static void main(String[] args) {

List<Student> list=new ArrayList<>();

list.add(new Student(10, "srinu", "kmm"));

list.add(new Student(1, "naresh", "wargal"));

list.add(new Student(12, "kumar", "hyd"));

//Collections.sort(list, (o1,o2)->o1.getId()<o2.getId() ? -1:o1.getId()>o2.getId() ? 1:0);

Collections.sort(list,(x1,x2)->x1.getAddress().compareTo(x2.getAddress()));

System.out.println(list);

}

}

**package** com.ibm.Java8Features;

**interface** Interf{

**public** **void** add(**int** a,**int** b);

}

**interface** Interf2{

**public** **void** Stringlength(String str);

}

**public** **class** Java8Features {

**public** **static** **void** main(String[] args) {

Interf it=(a,b)->System.***out***.println(a+b);

it.add(10, 35);

Interf2 it2=str->System.***out***.println(str.length());

it2.Stringlength("sreenivasaraok");

}}

**package** com.ibm.Java8Features;

**interface** MyNewInterf{

**public** Sample get(String str);

}

**class** Sample{

**public** Sample(String str){

System.***out***.println("this sample program");

}

}

**public** **class** ConstructerReference {

**public** **static** **void** main(String[] args) {

MyNewInterf f=Sample::**new**;

Sample sample = f.get("srinu");

Sample sample2 = f.get("srinu");

Sample sample3 = f.get("srinu");

Sample sample4 = f.get("srinu");

}

}

**package** com.ibm.Java8Features;

**interface** MyNewInterf{

**public** Sample get(String str);

}

**class** Sample{

**public** Sample(String str){

System.***out***.println("this sample program");

}

}

**public** **class** ConstructerReference {

**public** **static** **void** main(String[] args) {

MyNewInterf f=Sample::**new**;

Sample sample = f.get("srinu");

Sample sample2 = f.get("srinu");

Sample sample3 = f.get("srinu");

Sample sample4 = f.get("srinu"); }}

**package** com.ibm.Java8Features;

**interface** MyInterface{

**void** m1();

**public** **void** m2();

**public** **abstract** **void** m3();

**public** **default** **void** m4(){}

**public** **static** **void** m5(){

System.***out***.println("m5 method");

}

}

**class** Test **implements** MyInterface{

@Override

**public** **void** m1() {

System.***out***.println("m1 method");

}

@Override

**public** **void** m4() {

System.***out***.println("m4 method");

}

@Override

**public** **void** m2() {

System.***out***.println("m2 method");

}

@Override

**public** **void** m3() {

System.***out***.println("m3 method");

}

}

**public** **class** DefaultAndStaticMethods {

**public** **static** **void** main(String[] args) {

Test t=**new** Test();

t.m1();

t.m2();

t.m3();

t.m4();

MyInterface.*m5*();

}

}

**package** com.ibm.Java8Features;

**import** java.util.function.Consumer;

**class** Movie {

String name;

Movie(String name) {

**this**.name = name;

}

}

**public** **class** ConsumerExample {

**public** **static** **void** main(String[] args) {

Consumer<Movie> c1=m->System.***out***.println(m.name+"ready to relaese");

Consumer<Movie> c2=m->System.***out***.println(m.name+"it is aflop show");

Consumer<Movie> c3=m->System.***out***.println(m.name+"storing database");

Consumer<Movie> cc=c1.andThen(c2).andThen(c3);

Movie movie=**new** Movie("ysrbiopic");

/\*c1.accept(movie);

c2.accept(movie);

c3.accept(movie);\*/

cc.accept(movie);//consumer chaining possible

}

}

.

**package** com.ibm.Java8Features;

**public** **class** MethodReference1 {

**public** **void** m1(){

**for**(**int** i=0;i<10;i++){

System.***out***.println("child thread");

}

}

**public** **static** **void** main(String[] args) {

MethodReference1 m=**new** MethodReference1();

Runnable r=m::m1;

Thread t=**new** Thread(r);

t.start();

**for**(**int** i=0;i<10;i++){

System.***out***.println("it is a main thread");

}

}}

**package** com.ibm.Java8Features;

**interface** MyInterf{

**public** **void** add(**int** x,**int** y);

}

**class** Child **implements** MyInterf{

@Override

**public** **void** add(**int** x, **int** y) {

System.***out***.println("the sum is::"+(x+y));

}

}

**public** **class** MethodReference2 {

**public** **static** **void** main(String[] args) {

/\*MyInterf ifs=(x,y)->System.out.println(x+y);

ifs.add(25, 26);\*///this is using lambda expression

Child c=**new** Child();

MyInterf ifs2=c::add;//method reference

ifs2.add(55, 255);

}

}

**package** com.ibm.Java8Features;

**import** java.util.function.Predicate;

**public** **class** PredefinedFunctional {

**public** **static** **void** main(String[] args) {

String str="srinu";

Predicate<String> p=x->x.length()>8;

**boolean** test = p.test(str);

System.***out***.println(test);

}

}

package com.ibm.Java8Features;

import java.util.Date;

import java.util.function.Supplier;

public class SuppilerExample {

public static void main(String[] args) {

Supplier<Date> s1=()->new Date();

System.out.println(s1.get());

// TODO Auto-generated method stub

Supplier<String> s=()->{

String otp="";

for(int i=0; i<6;i++){

otp=otp+(int)(Math.random()\*10);

}

return otp;

};

System.out.println(s.get());

}

}

package com.ibm.Java8Features;

import java.time.DayOfWeek;

import java.time.LocalDate;

import java.time.LocalDateTime;

import java.time.LocalTime;

import java.time.Month;

import java.time.Period;

import java.time.ZoneId;

import java.time.ZonedDateTime;

public class TimeAndDateApiExamples {

public static void main(String[] args) {

//LocalTime ltime=LocalTime.now();

//System.out.println(ltime);//22:08:48.157

//LocalDate date=LocalDate.now();

//System.out.println(date);//2019-02-14

LocalDate date2=LocalDate.now();

System.out.println(date2);

int dayOfMonth = date2.getDayOfMonth();

int month = date2.getMonthValue();

int year = date2.getYear();

System.out.printf("%d-%d-%d",dayOfMonth,month,year);

LocalTime ltime2=LocalTime.now();

System.out.println(ltime2);

int hour = ltime2.getHour();

int minute = ltime2.getMinute();

int second = ltime2.getSecond();

int nano = ltime2.getNano();

System.out.printf("%d-%d-%d-%d",hour,minute,second,nano);

System.out.println();

LocalDateTime ldt=LocalDateTime.now();

System.out.println(ldt);

int dayOfMonth2 = ldt.getDayOfMonth();

int monthValue = ldt.getMonthValue();

int dayOfYear = ldt.getYear();

System.out.printf("%d-%d-%d",dayOfMonth2,monthValue,dayOfYear);

System.out.println();

int hour2 = ldt.getHour();

int minute2 = ldt.getMinute();

int second2 = ldt.getSecond();

int nano2 = ldt.getNano();

System.out.printf("%d-%d-%d-%d",hour2,minute2,second2,nano2);

System.out.println();

LocalDateTime ldt1=LocalDateTime.of(1988, Month.MAY, 04, 02, 01);

System.out.println(ldt1);

LocalDate birthday=LocalDate.of(1987, 12, 23);

LocalDate today=LocalDate.now();

Period p=Period.between(birthday, today);

System.out.println("year"+p.getYears()+"month"+p.getMonths()+""+p.getDays());

ZoneId zone=ZoneId.systemDefault();

System.out.println(zone);

ZonedDateTime zdt=ZonedDateTime.now(zone);

System.out.println(zdt);

}

}

package com.ibm.Java8Features;

import java.util.ArrayList;

import java.util.List;

import java.util.stream.Collectors;

import java.util.stream.Stream;

public class StreamApi {

public static void main(String[] args) {

List<Integer> list = new ArrayList<Integer>();

list.add(2);

list.add(21);

list.add(18);

list.add(17);

list.add(14);

list.add(29);

// list.stream().filter(i->i%2==0).forEach(System.out::println);//even

// list.stream().filter(i->i%2!=0).forEach(System.out::println);//odd

//list.stream().sorted((obj1, obj2) -> (obj1 < obj2) ? 1 : (obj1 > obj2) ? -1 : 0).collect(Collectors.toList())

// .forEach(System.out::println);

Integer min=list.stream().min((i1,i2)->i1.compareTo(i2)).get();

// System.out.println("MIN VALUE IS::"+min);//2

Integer max=list.stream().max((i1,i2)->i1.compareTo(i2)).get();

// System.out.println("MAX VALUE IS::"+max);//29

Integer[] array = list.stream().toArray(Integer[]::new);

Stream.of(array).forEach(System.out::println);//Interger array to stream

for(Integer i1:array){

System.out.println(i1);//convert stream object int array.

}

List<Integer> marks = new ArrayList<Integer>();

marks.add(2);

marks.add(21);

marks.add(18);

marks.add(17);

marks.add(14);

marks.add(29);

long count = marks.stream().filter(f -> f < 10).count();

// System.out.println(count);

// List<Integer> updateMarks =

// marks.stream().map(i->i+5).collect(Collectors.toList());

// System.out.println(updateMarks);//add 5 grace marks every student [7,

// 26, 23, 22, 19, 34]

List<String> list2 = new ArrayList<String>();

list2.add("srinu");

list2.add("naveen");

list2.add("sundar");

list2.add("kamal");

list2.add("ravi");

//list2.stream().sorted((s1, s2) -> -s1.compareTo(s2)).collect(Collectors.toList()).forEach(System.out::println);

// list2.stream().filter(s->s.startsWith("s")).forEach(System.out::println);

// list2.stream().filter(s->s.con

// ains("k")).forEach(System.out::println);

// list2.stream().sorted().collect(Collectors.toList()).forEach(System.out::println);;

// list.stream().sorted((obj1,obj2)-> (obj1<obj2)?1:(obj1>obj2) ?

// 1:0).collect(Collectors.toList()).forEach(System.out::println);;

// list2.stream().sorted((obj1,obj2)->obj2.compareTo(obj1)).forEach(System.out::println);;

}

}

package com.ibm.Java8Features;

import java.util.ArrayList;

import java.util.List;

import java.util.function.Predicate;

public class PredicateExample {

public static void main(String[] args) {

List<Student> list=new ArrayList<Student>();

list.add(new Student(12, "B", "KMM", 39));

list.add(new Student(10, "A", "WAL", 44));

list.add(new Student(1, "M", "DELHI", 72));

list.add(new Student(8, "K", "VZG", 59));

Predicate<Student> p=m->m.getMarks()>35;

for(Student student:list){

if(p.test(student)){

System.out.println(student);

}

}

String []str={"srinuvas","tendul","mahed","viratkohli","klrahul"};

Predicate<String> predicate=s->s.length()%2==0;

for(String even:str){

if(predicate.test(even)){

System.out.println("even is:::"+even);//srinuva,tendul,viratkohli

}

}

Predicate<String> predicate2=k->k.length()>8;

for(String lth:str){

if(predicate2.test(lth)){

System.out.println("length is::"+lth);//mahed,srinu

}

}

Predicate<String> predicate3=s->s.startsWith("s");

for(String startwith:str){

if(predicate3.test(startwith)){

System.out.println("Start with is::"+startwith);//srinu

}

}

for (String string : str) {

if(predicate.and(predicate3).test(string)){

System.out.println("And condition is::"+string);

}

}

/\*//if(predicate3.test(string)){

if(predicate.and(predicate3).test(string)){

//if(predicate.or(predicate2).test(string))//or condition use fst condition or secnd condition satisfy

System.out.println(string);//and condition means both conditions are satisfied

\*/

}

}

**package** com.ibm.Java8Features;

**public** **class** Student {

**private** **int** id;

**private** String name;

**private** String address;

**private** **int** marks;

**public** **int** getMarks() {

**return** marks;

}

**public** **void** setMarks(**int** marks) {

**this**.marks = marks;

}

**public** **int** getId() {

**return** id;

}

**public** Student(**int** id,String name,String address){

**this**.id=id;

**this**.name=name;

**this**.address=address;

}

**public** Student(**int** id,String name,String address,**int** marks){

**this**.id=id;

**this**.name=name;

**this**.address=address;

**this**.marks=marks;

}

@Override

**public** String toString() {

**return** "Student [id=" + id + ", name=" + name + ", address=" + address + ", marks="+marks+"]";

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;}

**public** String getAddress() {

**return** address;}

**public** **void** setAddress(String address) {

**this**.address = address;}}

package com.ibm.Java8Features;

import java.util.function.Function;

import java.util.function.Predicate;

public class FunctionExample {

public static void main(String[] args) {

Function<Integer, Integer> f=i->i\*i;

System.out.println(f.apply(5));

Function<String,String>fi=s->s.concat("k");

System.out.println(fi.apply("sreenivasarao"));

/\*Function<Student,String> f= s->{

double marks=s.marks;

String grade="";

if(marks>80) grade="GRADE A[it is a distination student]";

else if(marks>60) grade="GRADE B[it is first class student]";

else if(marks>50) grade="GRADE C[it is second class student]";

else if(marks>40) grade="GRADE D[it is third class student]";

else grade="GRADE E[this student is failed]";

return grade;

};

Predicate<Student> p=s->s.marks>20;

Student []s={ new Student(1, "rajee", 90),

new Student(2, "srinu", 80),

new Student(3, "kumar", 75),

new Student(4, "naveen", 60),

new Student(5, "suri", 45),

new Student(5, "sruthi", 25)

};

for(Student student:s){

if(p.test(student)){

System.out.println("STUDENT NAME IS::"+student.studentname);

System.out.println("STUDENT MARKS::"+student.marks);

System.out.println(f.apply(student));

System.out.println();

}}}}