**JAVA THEORY**

WHAT IS JAVA?

Java is a High-level, Class based Object Oriented Programming Language. Java is developed by James Cosling in the Year of 1986.

ADVANTAGES OF JAVA ?

* Simple
* Object – Oriented
* Secured
* Multi – Threaded
* Robust – Meaning (able to handle run time errors)
* Platform Independent

We can divide java Software into two parts

* JDK
* JRE

JDK (JAVA DEVELOPMENT KIT)

* Jdk stands for java development kit.
* JDK is the combination of JRE AND JVM.
* It provide development kit and run time environment .
* Using jdk we can develop compile and execute new application and also modifying the existing application.

JRE (JAVA RUNTIME ENVIRONMENT)

* Jre stands for java runtime environment
* Using jre we cannot develop new application, we can execute already developed application only.

JVM (JAVA VIRTUAL MACHINE)

* Jvm stands for java virtual machine
* It is a java platform jvm is responsible for execute byte code faster.
* Jvm provide interpreter and jit.

JIT (JUST IN TIME COMPILER)

It is responsible to help interpreter for executing the java byte code faster. Basically jit improve the performance of java program.

JDK AND JVM ARE PLATFORM INDEPENDENT ?

No, jdk is not platform independent because separate installer are there for window and mac. JVM is also not platform independent. Java is platform independent that means we can execute java .class files in any os.

WHAT IS JAR FILES IN JAVA ?

Basically jar is a file format based on zip. It aggregate many java class files into one file.

What is package in java ?

* A java package is a group of similar types of classes, interfaces and sub-packages.
* Package in java can be categorized in two form, built-in package and user-defined package.
* There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

WHAT IS OOPS (OBJECT ORIENTED PROGRAMMING SYSTEM)

Object-Oriented Programming System (OOPs) is a programming concept that works on the principles of abstraction, encapsulation, inheritance, and polymorphism. It allows users to create objects they want and create methods to handle those objects.

* Object
* Class
* Inheritance
* Polymorphism
* Encapsulation
* Abstraction

WHAT IS OBJECT ?

* Object is a instance of class
* Object is physical entity
* Object is a real time entity
* Object is created many times as per Developers requirement
* Object is created using new keyword

WHAT IS CLASS ?

* Class is the blueprint of which object is created
* Class is a group of similar objects
* Class is logical entity
* Class does not allocate memory when it is created
* Class is created using class keyword

WHAT IS CLASS LOADER ?

Class-loader in java is subsystem of java virtual machine. Decided to loading class files when program executed.

WHAT IS WRAPPER CLASS AND WHY WE NEED IN JAVA :

Wrapper classes are fundamental in Java because **they help a Java program be completely object-oriented**. The primitive data types in java are not objects, by default. They need to be converted into objects using wrapper classes.

WHAT ARE THE MEMORY ALLOCATION ARE THERE IN JAVA ?

* Class Memory
* Heap Memory
* Stack Memory
* Program counter memory
* Native Method Stack Memory

INHERITANCE :

Inheritance is one of the main concept in oops, inheritance is the process of acquire the property form one class to another class like ‘is a relationship’ or parent child relationship.

TYPES OF INHERITANCE :

* Single – Possible
* Multiple – Not –Possible in java class’s but it is Possible in only interfaces
* Multi – Level - Possible
* Hierarchical - Possible
* Hybrid - Possible

POLYMORPHISM :

Polymorphism is one of the main concept in oops. In which we can do simple task in many different ways.

TYPES OF POLYMORPHISM :

* Compile - Time – Polymorphism (Static Polymorphism)
* Run – Time – Polymorphism (Dynamic Polymorphism)

COMPILE-TIME POLYMORPHISM:

The functions call is resolved during the compile time.

For Example : Method Overloading.

Method Overloading (Static Binding)

If a class have same multiple parameters with same name but difference in parameter called compile time polymorphism or Method Overloading. Method can be overloaded by the different no of parameters.

RUN-TIME POLYMORPHISM :

The function call is resolved during the runtime.

For example : Method overriding.

METHOD OVERRIDING(DYNAMIC BINDING)

If a subclass have a same method as declared in parent class is called run time polymorphism.

ENCAPSULATION :

Encapsulation is one of the main concept in oops. Encapsulation is the process of wrapping the data and code into a single unit.

In java we are achieving this encapsulation concept through access modifiers.

ACCESS MODIFIERS IN JAVA

* Public : Any where in the Project
* Private : Within the class
* Protected : within a package and outside the package through child class reference
* Default : within the package

ABSTRACTION :

Abstraction is the process of hiding the implementation details and showing only functionality to the user. It is show only essential details to the user not the internal things.

In java abstraction is achieved by interface and abstract class.

ABSTRACT CLASS :

Abstract class is a restricted class or unfulfilled classes. That cannot be used to create objects. Abstract Method can only be used in a abstract class, and it does not have a body. The body is provided by the subclass only.

We cannot create a object for abstract classes.

Abstract class can have a static variables method cannot be static.

INTERFACE :

Interface is a blueprint of class, it has a static constrains and abstract methods.

Interface in java is the set of rules and regulation

We are achieving the abstraction by using interface and abstract classes.

-------------------------------------------------------------------------------------------------------------

INTERFACE ABSTRACT CLASS

Support multiple implementation Does not support multiple implementation

Does not contain data members contain data members

Interfaces are slow Abstract class are fast

Keyword is interface Keyword is abstract

Static Static and Non Static

Update :

Since 1.8 relece in java interfaces are accept method with body

In interface’s we can create static method and default method.

WHAT IS CONSTRUCTOR :

Constructor is a block of code similar to methods, constructor is called when instance of object is created. Constructor does not have a return type, constructor name should be same name as class name.

USAGE OF CONSTRUCTOR :

* Constructor in java is used to initialize the state of an object
* Constructor is used to set the initial value of an object
* A constructor is invoked implicitly. The java class provide a default constructor if you don’t have any constructor.

--------------------------------------------KEY-WORDS--------------------------------------------

THIS KEYWORD :

The this keyword refers to the current object in a method or constructor. The most common use of the this keyword is to eliminate the confusion between class attributes and parameters with the same name.

SUPER KEYWORD :

The **super** keyword in Java is a reference variable which is used to refer immediate parent class object.

Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

We cannot use Super Keyword in static content.

Usage of Java super Keyword

* super can be used to refer immediate parent class instance variable.
* super can be used to invoke immediate parent class method.
* super() can be used to invoke immediate parent class constructor.

FINAL KEYWORD :

Final is one the not access specifier.

Final variable is a variable that cannot be changed after initialization.

Final method Its a method which cannot be overridden. Compiler throws an error

if we try to override a method which has been declared final in the parent class.

Final Class that cannot be sub classed.

STATIC KEYWORD :

Static is the keyword is used for memory management mainly, in OOPs concepts objects have state and behavior likewise class’s also have the state and behaviour. The static keyword is used to mention the class level in programming while using this in method signature we don’t need to create object for method to call the method.

FINALIZE METHOD() :

finalize() method in Java is used to release all the resources used by the object before it is deleted/destroyed by the Garbage collector. finalize is not a reserved keyword, it's a method. Once the clean-up activity is done by the finalize() method, garbage collector immediately destroys the Java object.

--------------------------------------------------STRING----------------------------------------------

WHAT IS STRING ?

String is a collection of characters or collection of sequence. In java programming language String is a class. Java provide a string class to store and manipulate the Strings.

String is a non-primitive data type.

In java String is Immutable

STRING POOL OR STRING CONSTANT POOL :

String Constant Pool is a memory space in heap memory. Specifically allocate to store String objects by created string literal. It is a special type of memory maintained by JVM.

WHAT ARE THE DIFFERENT WAY OF CREATING STRING IN JAVA :

Using String constructor : String st = new String(“Prasanth”); - this will store in heap memory.

String literal : String st = “prasanth”; - this will store in String constant pool.

Notes : In String class we can create a object without using new keyword.

IS STRING IS IMMUTABLE IN JAVA ?

Yes String is immutable in java. Immutable objects are lie constants values we cannot modify string values.

Once it is created immutable objects are like constants are final in nature.

Java String pool is possible because the String is immutable.

MUTABLE :

Mutable objects are concerned we can modify the values.

WHY STRING IS IMMUTABLE IN JAVA :

The String is Immutable in java because of the security, synchronization and concurrency, caching, and class loading.

If you assign the same String literal to many string variables JVM will save only one copy of the String object in the java string pool, and these variables will start referring to that String object.

Also another reason can be security. We know that almost every java program contains a string, and it is used to save important data like usernames and passwords.

So it should not be changed in-between. Otherwise, there will be a security problem.

------------------------------------------------------------------------------------------------------

**STRING STRING BUFFER**

--------------------------------------------------------------------------------------------------------

The length of the object is fixed The length of the StringBuffer Object

can be increased.

String Object is Immutable StringBuffer is Mutable

Consumes More Memory Consumes less memory

SCP is available Heap Memory is available

It is slower during concatenation it is faster

-------------------------------------------------------------------------------------------------------

**STRING STRING BUILDER**

---------------------------------------------------------------------------------------------------------

String is Immutable StringBuilder is Mutable

A String is Thread safe It is not Thread safe

Performance is slow Performance is fast

-------------------------------------------------------------------------------------------------------

**STRING BUFFER STRING BUILDER**

-------------------------------------------------------------------------------------------------------------

Thread safe Not Thread safe

Synchronized Not Synchronized

Slower Faster

--------------------------------------EXCEPTION HANDLING------------------------------

WHAT IS ERROR IN JAVA :

In java, both Errors and Exceptions are the subclasses of java.lang.Throwable class. Error refers to an illegal operation performed by the user which results in the abnormal working of the program. Programming errors often remain undetected until the program is compiled or executed. Some of the errors inhibit the program from getting compiled or executed. Thus errors should be removed before compiling and executing. It is of three types:

* Compile-time
* Run-time
* Logical

ERROR DEFINITON :

Errors in Java represent serious problems that typically cannot be recovered from. They are instances of subclasses of the **Error** class and usually indicate issues with the environment or JVM itself.

EXPECTION HANDLING :

Exception handling in java is one of the powerfull mechanism to handle runtime errors.

Exception is an unwanted or unexpected event which is occur during the execution of programs.

Exception is an abnormal condition which occurs during the execution of a program and disrupts normal flow of the program. If not handled properly it can cause the program to terminate abruptly.

CHECKED EXCEPTION :

A checked exception is caught at compile time.

EX :

FileNotFound Exception, SQLException, ClassNotFoundException, IOException.

UNCHECKED EXCEPTION :

An unchecked exception is the one which occurs at the time of execution.

EX :

Arithmatic Exception, ArrayIndexOutOfBound Exception, NullPointer Exception, Classcast Exception.

KeyWord in Exception Handling :

* try
* catch
* finally
* throw
* throws

TRY :

try block is used to enclose the code that might throw an exception. It must be used within

the method. If an exception occurs at the particular statement in the try block,

the rest of the block code will not execute.

Catch :

Java catch block is used to handle the Exception by declaring the type of exception within the parameter.

FINALLY :

The finally block in Exception is used to put important codes such as clean up code

e.x. closing the file or closing the connection. The finally block executes whether exception rise or not and

whether exception handled or not. A finally contains all the crucial statements regardless of the exception

occurs or not.

THROW :

Throw keyword is used to throw exception object to JVM explicitly.

Throw is followed by the instance

Throw is used within method

We cannot throw multiple exceptions.

THROWS :

Throws keyword is used to declare exception

Throws keyword is used in the method signature

Using Throws you can throws multiple exceptions

Some Extra Points :

We can use to delegate responsibility of exception handling to the called (JVM or Another Method)

It is required only for checked exception and for unchecked exception there in no use.

It is required only to convince compiler and its usage does not prevent abnormal termination of the program

----------------------------------------LAMBDA EXPRESSIONS------------------------------

LAMBDA EXPRESSION :

Lambda Expression is used to bring Functional programming features to the Object Oriented Programming

ADVANTAGES :

Bring the functional interface feature’s

Reduce the code complicity.

WHAT IS FUNCTIONAL INTERFACE:

If the Interface has only one abstract method then that interface is called functional interface.

Ex. Runnable, Callable,

the functional interface may or may not have multiple default function but only one abstract method. that types of interfaces called functional interfaces.

PRE-DEFINED FUNCTIONAL INTERFACES:

These Pre Defined functional interfaces used to write a lambda expression without using or creating functional interfaces.

* Predicate<T> calling method - test() - accept any type of values but return only Boolean values
* Function<T,R> calling method - apply() - accept and return any type of values
* Consumer<T> calling method - accept() - it did not return anything but accept all type of values
* Supplier<R> calling method - get() - it return any type of values but did not accept

---------------------------------------**MULTITHREATDING**-------------------------------

What is Thread in Java :

Thread is often referred to as a lightweight process. The process can be split down into so many threads.

https://www.geeksforgeeks.org/thread-in-operating-system/

WHAT IS MULTITHREADING IN JAVA ?

In Java, Multithreading refers to a process of executing two or more threads simultaneously for maximum utilization of the CPU.

A thread in Java is a lightweight process requiring fewer resources to create and share the process resources.

If you are using thread concept in your java program you can achieve in two different ways :

1> By Extending Thread class

2> By Implementing Runnable Interface

EX:

1> BY EXTENDING THREAD CLASS

class Multi\_Thread\_Demo extends Thread

{

public static void main(String args[])

{

}

}

2> BY IMPLEMENTING RUNNABLE INTERFACE

class Multi\_Thread\_Demo implements Runnable

{

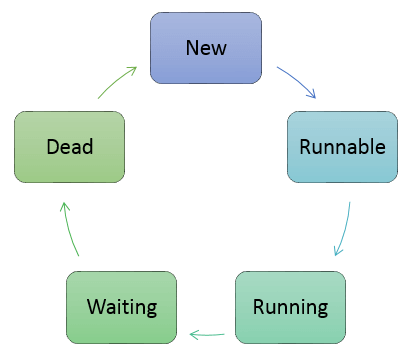
public static void main(String args[])

{

}

}

WHAT IS THREAD LIFE-CYCLE ? :



New -> Runnable -> Running -> Waiting -> Dead or Terminated.

New :

if you start the thread. the thread is in new state

Runnable :

After Starting a New Thread it will go Runnable State.

Running :

The 3rd state of Thread life cycle is Running

Waiting :

Here Thread will execute

Dead or Terminate :

After executing the thread. it will go to dead state.

WHAT IS THREAD PRIORITY ?

In Java, a thread's priority is an integer in the range 1 to 10. The larger the integer, the higher the priority.

The thread scheduler uses this integer from each thread to determine which one should be allowed to execute.

Thread default priority is 5.

HOW TO CHANGE THREAD PRIORITY ?

It can be changed using the method setPriority() of class Thread. There are three static variables for thread priority in Java i.e.

MIN\_PRIORITY, MAX\_PRIORITY and NORM\_PRIORITY. The values of these variables are 1, 10 and 5 respectively.

THREAD RELATED EXCEPTION ?

IllegalThreadStateException

IllegalArgumentException

IntereptedException

WHAT HAPPENED IF I START A THREAD TWICE ?

it will give IllegalThreadStateException.

WHAT IS THREAD-SCHEDULING ?

Thread Scheduler is the part of JVM(JAVA VIRTUAL MACHINE)

Thread Scheduler is the controller of thread execution. It has the overall control to execute all threads in our java programming.

It will decide which thread will execute first and which will execute last. we can modify thread priority by using setPriority() method.

IN MULTITHREADING THERED ARE TWO METHOD ARE AVAILABLE

1> Start NOTE : this start method will automatically calls run method. we don’t need to call run method externally. If we give the definition of run method in child classes like overriding then only the multithreading starting.

Note : Start(); this is the key word for representing multithreading in java programming.

2> Run

Without using those two above method we cannot use multi threading in our program.

START :

Start method will automatically call run method

In Object oriented programming non static method are calling with the use of objects. if we are using child class method first up all child method will call first then

only parent method will call. if child class's don’t have that method then parents method will call.

FOR EXAMPLE :

public Shanmugam

{

public static void main(String args [])

{

Shanmugam sh = new Shanmugam ();

sh.study();

public void study()

{

tamilStudy();

prasanthStudy();

baraniStudy();

}

public void tamilStudy()

{

System.out.println("You should become Doctor " );

}

public void prasanthStudy()

{

System.out.println("You should become Police " );

}

public void baraniStudy()

{

System.out.println("You should become Civil\_Servent " );}}}

public childs extends Shanmugam

{

public static void main(Stirng args[])

{

childs c = new childs();

c.study();

public void tamilStudy()

{

System.out.println("Tamil is going to become Engineer " );

}

public void prasanthStudy()

{

System.out.println("Prasanth is going to become Engineer" );

}

public void baraniStudy()

{

System.out.println("Barani is going to become Engineer " );

}}}

In this above program i am using method overriding concept. the result of this above program is child class result.

Parent class also have same method which child class have but because of method overriding and object calling principle child class methods will call first

then only parent class method will call.

MULTITHREADING METHODS:

yield() :

A yield() method is a static method of Thread class and it can stop the currently executing thread and will give a chance to other waiting threads of the same priority.

If in case there are no waiting threads or if all the waiting threads have low priority then the same thread will continue its execution.

that yield thread will go in runnable state

join() :

Join method in Java allows one thread to wait until another thread completes its execution. In simpler words, it means it waits for the other thread to die.

here all other threads are going in waiting state

sleep() :

Thread. sleep causes the current thread to suspend execution for a specified period.

This is an efficient means of making processor time available to the other threads of an application or other applications that might be running on a computer system.

here also sleeping thread will go in waiting state

interrupt() :

An interrupt is an indication to a thread that it should stop what it is doing and do something else.

It's up to the programmer to decide exactly how a thread responds to an interrupt, but it is very common for the thread to terminate.

this method will disturb sleep or waiting state threads

---------------------------------COLLECTION FRAMEWORK-----------------------------

WHAT IS COLLECTION FRAMEWORK :

In java collection is a interface that provide architecture to load and manipulate group of objects.

DIFFERENT BETWEEN COLLECTION AND COLLECTIONS :

Collection is a interface. Collections is a class which have all the method performed by child interfaces and class's.

COLLECTION FRAMWORK HIERARCHY:

ITERABLE -> COLLECTION -> LIST -> ABSTRACT LIST -> IMPLEMENTED CLASS'S LIKE ARRAYLIST AND LINKED LIST.

COLLECTION IMPLEMENTED INTERFACES :

List

Set

Queue

LIST :

List is the interface extended by collection framework.

* In list we can store duplicate values
* In list preserve insertion order.

The List interface is implemented by two class.

* Array List
* Linked List

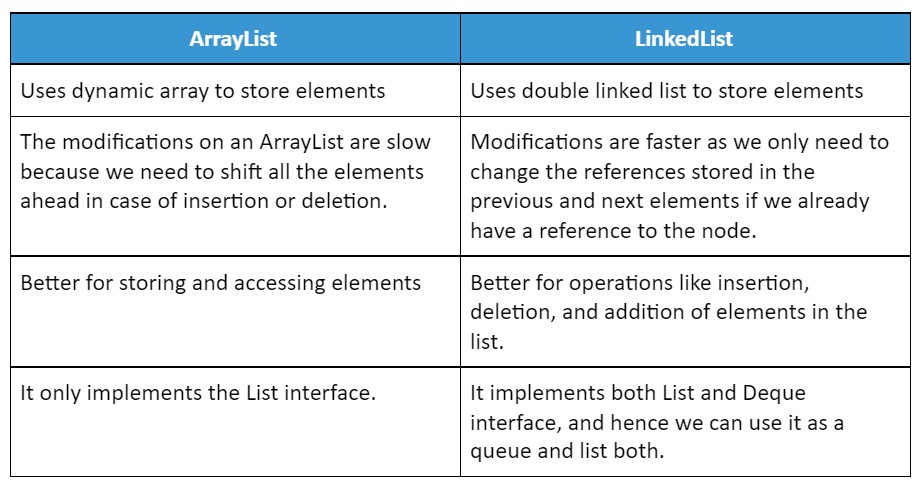
One of The Way of creating List:

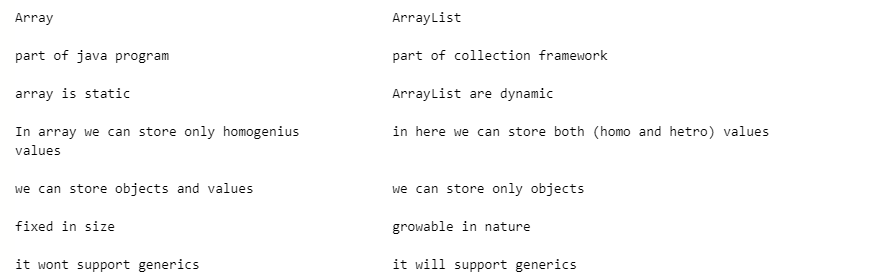
List<Object> lis = Arrays.asList("Tamil",23.84,'p',"Prasanth","Barani",73);

ARRAYLIST :

retrieving data very fast. default size is 10.

LINKED LIST:

useful for insertion and deletion.



SET :

SET PROPERTIES:

Does not allow duplicate values.

Insertion order is not maintained.

In set we don’t have a index concept. we have to specify the value in every methods like add, remove, contains.

HASHSET:

HashSet is implemented by HASHTABLE.

The default capacity is 16 and default load factor or filling ratio is 0.75. we can modify the default capacity and filling ration in HashSet.

HashSet hs = new HashSet<>(100); Here 100 is default capacity

HashSet hs = new HashSet<>(100, (float)0.90); Here 0.90 is load factor or filling ratio. the default load factor is 0.75. here we are override the default size and load factor.

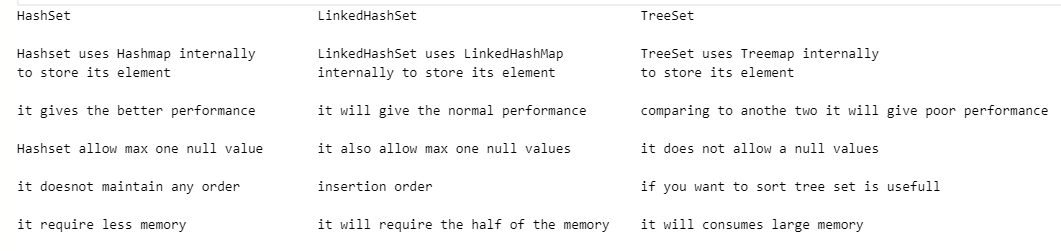
LINKED HASHSET :

LINKED HASHSET is implemented by LINKED LIST AND HASHTABLE. that's why it allow insertion order.

There is no difference between HASHSET AND LINKED HASHSET

But the only one diff is insertion order

LINKED HASHSET Maintain insertion order. Otherwise all are same.



QUEUE :

Queue has two implemented classes.

* PriorityQueue
* LinkedList

Default Methods for QUEUE Interface :

add(Object) - used to add the objects in queue classes. it this method will fail to add objects in the class then it will return Exception.

Offer(Object) - this method is also used to add the objects are values in the classes. If it is fail this will return false to the console.

element() - this method will return, get or give you the first element(Head Element). if the queue is empty then this will return Exception.

peek() - This method is also return, get or give you the first element(Head Element). if the queue is empty then this method will return Null.

remove()- this method will help us to remove the first element in the queue(Head Element). if the queue is empty then this will return Exception.

poll() - this method will help us to remove the first element in the queue(Head Element). if the queue is empty then this method will return Null.

----------------------------------------------------------------------------------------------------------

Priority Queue LinkedList

----------------------------------------------------------------------------------------------------------

Insertion Order Insersion Order

Duplicate values is accepted Duplicate values is accepted

Implemented Interface (Queue) Implemented Interface(Queue and List)

Accept only homogenous values Both homo and heterogeneous values.

-------------------------------------------------------------------------------------------------------------

Map :

Map is a separate interface. it is not a part of collection framework

Map is useful when you have search with key and value pair.

Map implemented many class for example HashMap and HashTable

HASHMAP INITIALIZATION:

HashMap hm = new HashMap<>();

HashMap<Integer, String> = new HashMap<Integer, String>()

Properties :

Map underlying data structure is HashTable

Insertion order is not maintained

Map is stored in the combination of key and value

No duplicate keys

duplicate values is allowed

Only one null key is allowed

Multiple null values is allowed

if you store duplicate key that duplicate key will override the original key for ex. 103 ram if you override that key like 103 ramesh. the value will change ram into ramesh.

MAP METHODS:

Put.(key,value); for adding data in map

putAll(map m1); for adding map collection in existing map

get(key); return value

remove(key);

containKey(key);

containvalue(value);

isEmpty();

size();

clear();

keyset(); - this method will return all the key values in the map – the return value is Set.

Values(); return type is collection

entrySet(); this method will return all entry set – return type Is SET

HASHTABLE:

HashTable is another implemented class for map interface.

The difference between HashMap and HashTable

------------------------------------------------------------------------------------------------------------

HashMap HashTable

-------------------------------------------------------------------------------------------------------------

From java 1.2 Since in the beginning

Not-Synchronized Synchronized

Multiple Threads allowed Only one Thread allowed at a time

Not Thread safe Thread safe

Performance is faster Performance is Poor

Comparing to HashTable

In key and values Null values are Null values and key not accepted

accepted as a key and values.

-------------------------------------------------------------------------------------------------------------

What is JDBC :

JDBC stands for java database connectivity.

JDBC is standard java API which help java programming language to communicate with database.

JDBC acts as mediator between java programming language and database.

TASKS DONE BY JDBC API ;

Making database connectivity.

Create SQL OR MySQK queryes.

Execute SQL/MySQL queries.

View and modify result.

COMMON COMPONENTS FO JDBC :

DriverManger (C) : Manges list of databases.

Driver (I) : Handles communication with Database

Connection (I) : Contains methods to establish connection to Database.

Statement (I) : Provides platform to run SQL queries.

ResultSet : Holds data retrieved from Database

SQLException (C) : Handles all errors occured during database operations.

STEPS FOR JDBC :

Load and register the driver

Establish the connection

Create platform to execute query

Execute the query

Process resultset

Close all the connections.

HIBERNATE – THEORY

What is Difference between JDBC and HIBERNATE :

JDBC :

JDBC is a technology.

Used of database connectivity.

JDBC user should taking care of creating and closing the connections.

Hibernate :

Hibernate is a framework.

ORM tool.

Framework will taking care creating and closing connections.

WHAT IS HIBERNATE :

Hibernate is a Java framework that simplifies the development of Java application to interact with the database. It is an open source, lightweight, ORM (Object Relational Mapping) tool. Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.

WHAT IS ORM : OBJECT RELATIONAL MAPPING :

An ORM tool simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database. ORM tool internally usages JDBC API for accessing database’s.

ADVANTAGES OF HIBERNATE :

* Open source
* Light weight
* Fast performance
* Automatic table creation
* An ORM tool simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database.

Interview questions :

What is concurrent hash map

What is deamon thread

What is the difference between http and https protocol

What is CORS

What is auto-configuration.

What is functional interface.

Java design patterns – singleton, prototyp, factory design pattern, mvc,

How to calculate performance of java application

Microservice architecture

API gateway.

Spring security module

**Spring Core Container**

**What is Spring :**

Spring is open-source, lightweight Java framework designed to simplify the development of enterprise Java applications.

**What is the Spring Core container?**

The Spring Core container, also known as the IoC (Inversion of Control) container, is the central component of the Spring Framework responsible for managing the lifecycle of Spring beans and their dependencies.

**What is Inversion of Control (IOC) :**

IOC stands for Inversion of Control. It is a design principle in software engineering where the control of object creation and lifecycle management is inverted or moved from the application code to a container or framework.

**What is Dependency Injection :**Dependency injection is the process of injecting one class bean into another class. In Spring we can achieve dependency injection in many ways like Constructor injection, Setter Injection and Field Injection.

**What are the primary components of the Spring Core container :**

The primary components of the Spring Core container include the BeanFactory interface and its implementations, such as the ApplicationContext interface and its implementations like ClassPathXmlApplicationContext and AnnotationConfigApplicationContext.

**What is Bean Factory. what are all the advantage and Disadvantages of using Bean Factory Interface and its implementation classes :**

* **Definition**:

The BeanFactory interface is the root interface for accessing the Spring IoC container. It defines a contract for implementing classes to manage bean instances, their configurations, and dependencies.

* **Responsibilities**:

The BeanFactory interface is responsible for bean instantiation, configuration, wiring (dependency injection), and lifecycle management.

* **Features**:

Provides methods to get beans by name, type, or through the use of qualifiers.

Supports lazy initialization of beans.

Enables hierarchical configuration of beans through parent contexts.

**What is ApplicationContext. what are all the advantage and Disadvantages of using ApplicationContext Interface and its implementation classes.**

* **Definition**:

ApplicationContext is an interface in the Spring Framework that extends the BeanFactory interface. It represents the Spring IoC container and provides advanced features for managing beans, handling events, internationalization, resource loading, and more.

* **Responsibilities**:

In addition to the responsibilities of BeanFactory, ApplicationContext provides support for internationalization, event propagation, resource loading, AOP integration, and other features essential for enterprise application development.

* **Features**:

Supports lazy and eager initialization of beans.

Provides advanced lifecycle management for beans.

Supports event propagation and handling through application events.

Offers integration with AOP (Aspect-Oriented Programming) for cross-cutting concerns.

Facilitates internationalization and localization of messages.

Enables resource loading from various sources, including files, URLs, and classpath locations.

**What are the different ways of configuring Spring beans :**

Spring beans can be configured using XML-based configuration, Java-based configuration using annotations like @Component, @Service, @Repository, etc., or a combination of both XML and annotations.

**What is the difference between BeanFactory and ApplicationContext in Spring :**

BeanFactory is the basic interface for accessing the Spring IoC container, providing basic functionality for managing beans. ApplicationContext is a sub-interface of BeanFactory that adds additional features like event propagation, internationalization, resource loading, etc.

**Explain the lazy initialization of beans in Spring :**

Lazy initialization in Spring delays the creation of a bean until it's first requested. This can improve application startup time and reduce memory consumption, especially for beans that are not used immediately.

**SPRING BEANS**

**Spring Bean Life Cycle** :

* **Spring Application Start**:

When the Spring application starts, the Spring IoC container initializes, and it begins to load and manage beans defined within the application context.

* **Loading of Beans**:

The container scans for bean definitions, which can be provided through XML configuration, Java configuration (using annotations like **@Component**, **@Service**, **@Repository**, etc.), or a combination of both.

* **Dependency Injection**:

Once the beans are loaded, the container identifies the dependencies between beans and injects them into each other. This is typically achieved through constructor injection, setter injection, or field injection, depending on the configuration.

* **Custom Initialization Method**:

After the dependencies are injected, any custom initialization methods defined by the developer are invoked. This initialization logic can be specified using annotations like **@PostConstruct** or by implementing the **InitializingBean** interface.

* **Bean Usage**:

Once the bean is fully initialized, it becomes available for use within the application context. Other beans or components can now use this bean as needed.

* **Bean Destruction**:

When the application context is shut down or the bean is no longer needed, the container manages the destruction of beans.

Developers can specify custom destruction logic using annotations like **@PreDestroy** or by implementing the **DisposableBean** interface. This allows for cleanup tasks such as releasing resources, closing connections, or any other necessary cleanup operations.

**Spring Bean Scopes :**

* **Singleton Beans**:

Singleton beans are the default scope in Spring. The Spring container creates and manages only one instance of a singleton bean per container.

Singleton beans are shared across the entire application context, and the same instance is returned whenever the bean is requested.

Example: Service classes that can be reused across multiple components in an application are often configured as singleton beans.

* **Prototype Beans**:

Prototype beans are created each time they are requested from the container.

Unlike singleton beans, prototype beans are not shared; a new instance is created for every request for the bean.

Prototype beans are useful when you want to maintain different states or configurations for each instance.

Example: DTO (Data Transfer Object) classes or objects that need to maintain state isolation.

* **Request Beans**:

Request beans are scoped to the lifecycle of an HTTP request.

A new instance of the request bean is created for each HTTP request, and it's available until the request processing completes.

Request scope is only applicable in a web-aware Spring ApplicationContext.

Example: Objects that need to maintain state during the processing of an HTTP request, such as form backing objects.

* **Session Beans**:

Session beans are scoped to the lifecycle of an HTTP session.

A single instance of the session bean is created for each HTTP session, and it remains available until the session expires or is invalidated.

Session scope is only applicable in a web-aware Spring ApplicationContext.

Example: Objects that need to maintain state throughout a user's session, such as shopping cart items.

* **Application Beans** (also known as "global session" beans):

Application beans are scoped to the lifecycle of a ServletContext.

There's only one instance of an application bean per ServletContext.

Application scope is only applicable in a web-aware Spring ApplicationContext.

* + Example: Objects that need to be available globally throughout the application, such as configuration settings or caches.
* **Custom Scope Beans**:
  + Spring allows you to define custom bean scopes to meet specific application requirements.
  + Developers can create custom scopes by implementing the **org.springframework.beans.factory.config.Scope** interface.
  + Example: Custom scopes tailored to specific use cases, such as conversation scope in a web application.

**WHAT IS REST ?**

REST stands for Representational State Transfer. It's an architectural style for designing networked applications. RESTful systems typically use HTTP (Hypertext Transfer Protocol) for communication and adhere to a set of constraints or principles, which include:

1. **Client-Server Architecture**: The system is divided into client and server components that are independent of each other and can evolve separately.
2. **Statelessness**: Each request from a client to the server must contain all the information necessary to understand and fulfill the request. In other words, the server should not rely on any context from previous requests.
3. **Cacheability**: Responses from the server can be explicitly marked as cacheable or non-cacheable, which helps improve performance and scalability.
4. **Uniform Interface**: The interface between the client and the server should be uniform across different parts of the system. This includes the use of standard HTTP methods (GET, POST, PUT, DELETE) for CRUD (Create, Read, Update, Delete) operations.
5. **Layered System**: The architecture should support a hierarchical system of components, such as proxies, gateways, and firewalls, allowing for scalability and encapsulation of concerns.
6. **Code-On-Demand (optional)**: Servers can optionally provide executable code (e.g., JavaScript) to clients, extending client functionality dynamically.

RESTful APIs (Application Programming Interfaces) are designed following these principles, allowing clients to interact with server resources using standard HTTP methods and representations (e.g., JSON or XML). This approach promotes simplicity, scalability, and interoperability.

**REST web services**: This term refers to the implementation of a web service or application that follows the REST architectural style. REST web services expose their functionality through RESTful APIs. In other words, REST web services are the implementations of the principles of REST architecture, and the RESTful API is the interface through which clients interact with these services.

SPRING SECURITY COMPLETE TURORIAL

By Default spring security framework protects all the paths present inside the web application. This behavior is due to the code present inside the method defaultSecurityFilterChaing(HttpSecurity http) of class SpringBootWebSecurityConfiguration.

@Bean

@Order(SecurityProperties.BASIC\_AUTH\_ORDER)

SecurityFilterChain defaultSecurityFilterChain(HttpSecurity http) throws Excpetion {

http.authorizeHttpRequest().anyRequest().authenticated();

http ttp.formLogin();

Http.httpBasic();

Return http.build();

}

Custom Security Filter for Own Logic :

We can secure the web applicaion APIs, paths as per our custom requirements using Spring Security framework like shown below.

@Configuration

Public class projectSecurityConfig{

@Bean

SecurityFilterChain defaultSecurityFilterChain(HttpSecurity http)throws Exception {

http.authorizeHttpRequests()

.requestMatchers(“/myAccount”,”/myBalance”, “/myLoan”, “/myCard”).authenticated()

.requestMatchers(“/notices”,”/contact”).permitAll()

.and().formLogin()

.and().httpBasic();

Return http.build();

}

We can deny all the request coming towrads our web application APIs, paths using Spring security framework like below.

@Configuration

Public class ProjectSecurityConfig{

@Bean

SecurityFilterChain defaultSecurityFilterChain(HttpSecurity http) throws Exception {

Http.authorizeHttpRequest()

.anyRequest().denyAll()

.and().formLogin()

.and().httpBasic();

return http.build();

}

}

LOADING USER DETIALS INSIDE APPLICATION :

In Spring security we can load users in three ways.

1. InMemoryUserDetailsManager.
2. JDBC
3. LDAP.

APPROACH 1. where we use withDefaultPasswordEncoder() method while creating the user details.

Public InMemoryUserDetailsManager userDetailsService(){

UserDetails admin = User.withDefaultPasswordEncoder()

.username(“admin”)

.password(“12345”)

.authoritires(“admin”)

.build();

UserDetails user= User.withDefaultPasswordEncoder()

.username(“user”)

.password(“12345”)

.authoritires(“read”)

.build();

Return new InMemoryUserDetailsManager(admin, user);

}

SECOND APPROACH :

Public InMemoryUserDetailsManager userDetailsService(){

InMemoryUserDetailsManager inMemoryUserDetialsManager = new InMemoryUserDetailsmanager();

UserDetials admin = User.withUsername(“admin”).password(“12345”).authorities(“admin”).build();

UserDetials user= User.withUsername(“user”).password(“12345”).authorities(“user”).build();

InMemoryUserDetialsManager.createuser(admin);

InMemoryUserDetialsManager.createUser(user);

Return inMemoryUserDetialsManager;

}

@Bean

Public PasswordEncoder passwordEncoder(){

return NoOpPasswordEncoder.getInstance();

}

What is Authentication :

Authentication is the process proving of who you are.

What is Authorization :

Authorization is about what you are allowed to do after provin who you are.