1. Difference between Abstract class and interface
2. What is weak HashMap?

WeakHashMap is an implementation of the Map interface. WeakHashMap is almost same as HashMap except in case of WeakHashMap, if object is specified as key doesn’t contain any references- it is eligible for garbage collection even though it is associated with WeakHashMap. i.e Garbage Collector dominates over WeakHashMap.

1. What is Oracle?

Oracle Database is a proprietary multi-model database management system produced and marketed by Oracle Corporation. It is a database commonly used for running online transaction processing, data warehousing and mixed database workloads.

1. What is MongoDB?

MongoDB is a cross-platform, free, open-source document and oriented NoSQL database which is written in C++. MongoDB is very much useful for high volume data storage which provides high performance, high availability, and also automatic scaling

1. Difference between Oracle and MongoDB

* MongoDB is one of the most famous documents-oriented databases whereas Oracle Database is a multi-model database management system and it is highly used RDBMS to build enterprise applications.
* In MongoDB, data is stored in a collection in the form of document and Field. But in Oracle database, data is stored in the traditional way of RDBM i.e., in the form of tables in the form of rows and columns.
* Mongo database offers some API for user-defined Map/Reduce methods, whereas MapReduce is not supported in Oracle database.
* SQL is not supported in MongoDB, but SQL is supported in Oracle DB.
* MongoDB is a free and Open source whereas Oracle database is Commercial

1. What is AMI?

An Amazon Machine Image (AMI) is a special type of virtual appliance that is used to create a virtual machine within the Amazon Elastic Compute Cloud ("EC2"). It serves as the basic unit of deployment for services delivered using EC2.

1. What is Cloud Computing

cloud computing is the delivery of computing services—servers, storage, databases, networking, software, analytics, intelligence and more—over the Internet (“the cloud”) to offer faster innovation, flexible resources, and economies of scale.

1. Explain java8 features
2. Benefits of Cloud Computing

Cost: Cloud computing eliminates the capital expense of buying hardware and software and setting up and running on-site datacenters.

Speed: Most cloud computing services are provided self service and on demand, so even vast amounts of computing resources can be provisioned in minutes.

Security: Many cloud providers offer a broad set of policies, technologies, and controls that strengthen your security posture overall, helping protect your data, apps, and infrastructure from potential threats

1. Type of Cloud services

Infrastructure as a service (IaaS) : With IaaS, you rent IT infrastructure—servers and virtual machines (VMs), storage, networks, operating systems—from a cloud provider on a pay-as-you-go basis.

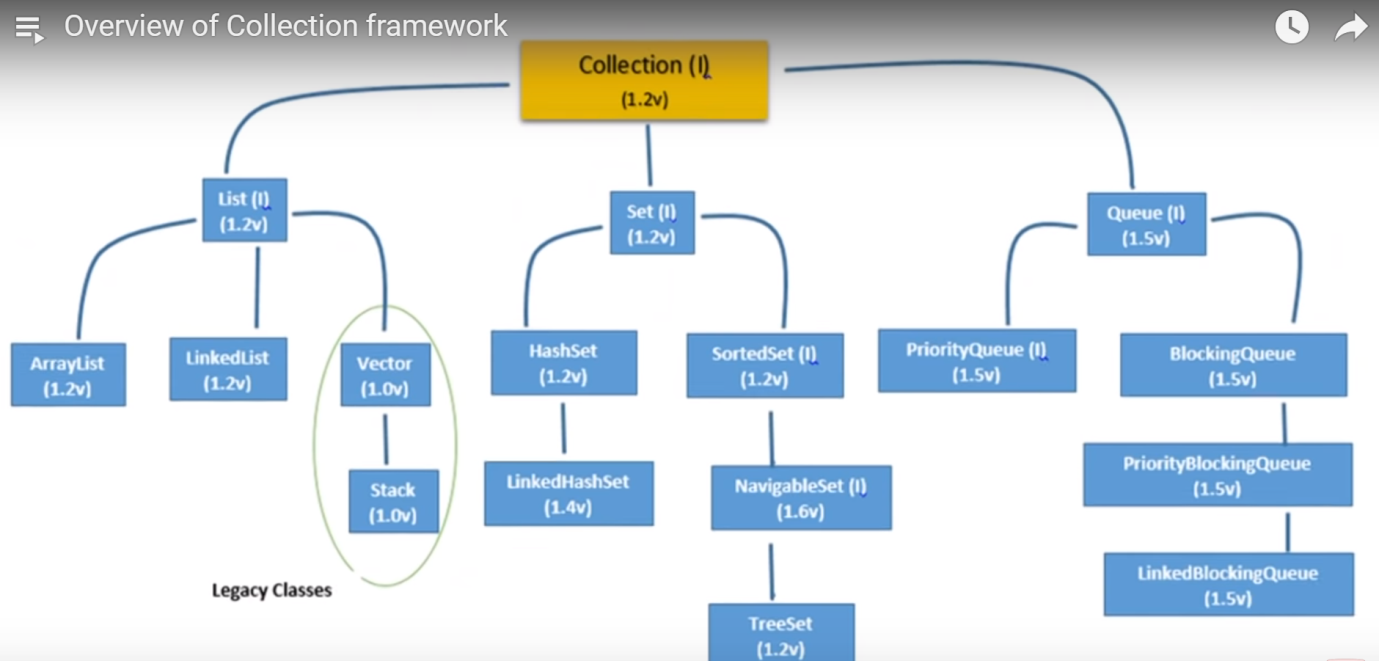
### Platform as a service (PaaS)

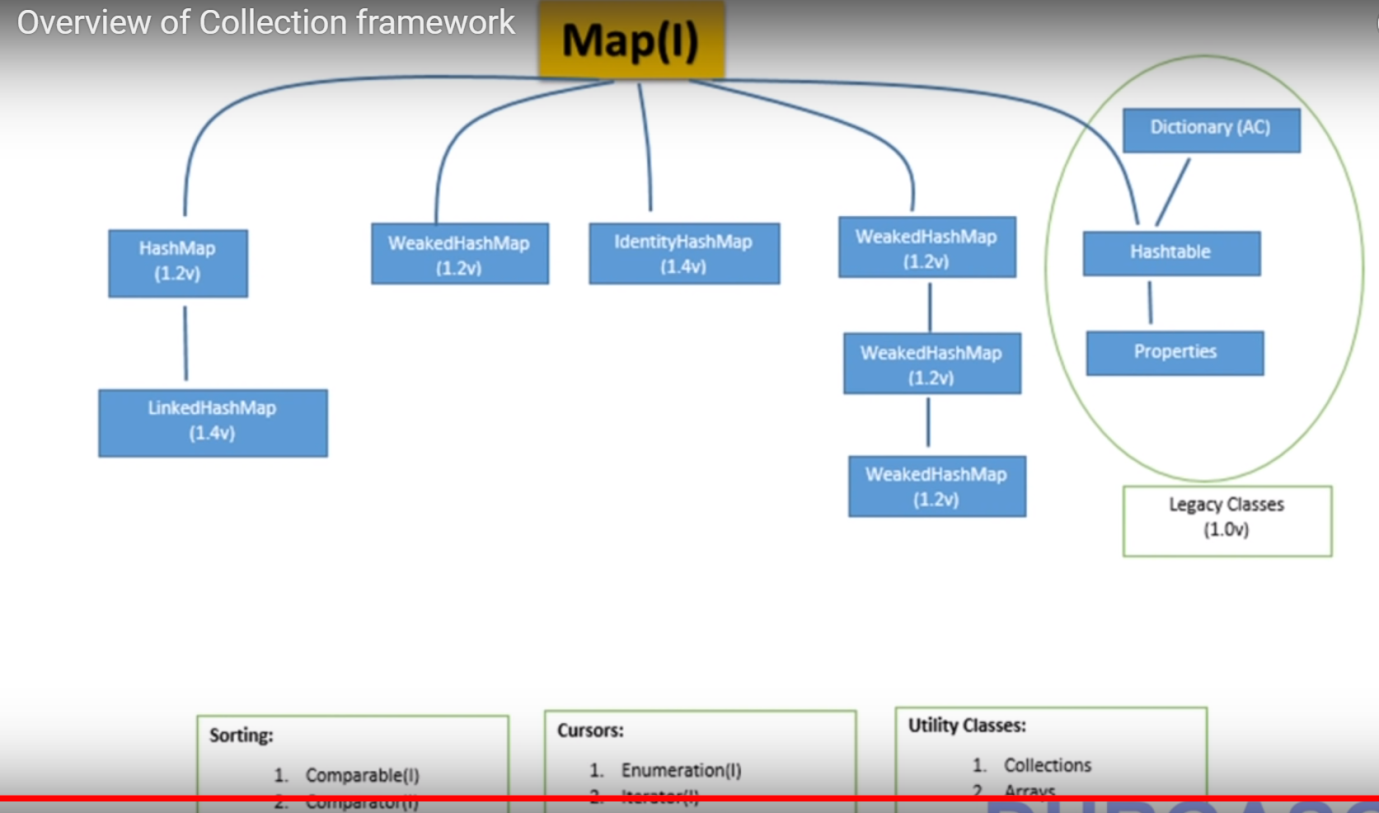
supply an on-demand environment for developing, testing, delivering, and managing software applications. PaaS is designed to make it easier for developers to quickly create web or mobile apps, without worrying about setting up or managing the underlying infrastructure of servers, storage, network, and databases needed for development.

### Software as a service (SaaS)

Software as a service is a method for delivering software applications over the Internet, on demand and typically on a subscription basis. With SaaS, cloud providers host and manage the software application and underlying infrastructure, and handle any maintenance, like software upgrades and security patching. Users connect to the application over the Internet, usually with a web browser on their phone, tablet, or PC.

**Collection Framework**



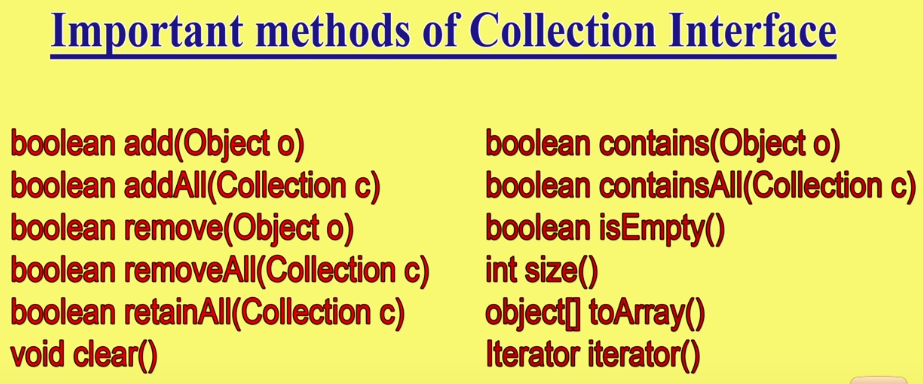


**Difference between Array vs ArrayList?**

* Array is a fixed length data structure while ArrayList is a variable length [Collection class](http://java67.blogspot.sg/2012/09/java-collection-interview-questions.html)
* You cannot use [Generics](http://javarevisited.blogspot.ca/2011/09/generics-java-example-tutorial.html) along with Array, but ArrayList allows you to use Generics to ensure type-safety.
* Array provides length variable which denotes length of Array while ArrayList provides size() method to calculate size of ArrayList.
* you cannot store primitives in ArrayList, it can only contain Objects. While Array can contain both primitives and Objects.
* Java provides add() method to insert element into ArrayList and you can simply use assignment operator to store element into Array.
* you can create instance of ArrayList without specifying size, Java will create Array List with default size but its mandatory to provide size of Array while creating either directly or indirectly by initializing Array while creating it.
* In terms of performance Array and ArrayList, Though automatic resize of ArrayList may slow down insertion a bit

**Collection interface**

* If you want to represent a group of individual objects as a single entity then we should go for collection interface.
* It defines the most common methods which are applicable for any collection objects.
* Any class in Collection interface implements Cloneable and Serializable interfaces.



**Difference between Collection and collections**

* Collection is an interface and Collections is a class.
* Collection is base interface for list set and queue. Collections is a class and it is called utility class.

**Iterators in JAVA**

Enumeration

* Enumeration is the first iterator present from JDK 1.0
* It is a interface used to get elements of legacy collections(Vector, Hashtable).
* Methods boolean hasMoreElements(); Object nextElement();

Limitations Enumeration

* Enumeration is for legacy classes(Vector, Hashtable) only. Hence it is not a universal iterator.
* Remove operations can’t be performed using Enumeration.
* Only forward direction iterating is possible.

## Iterable Interface

* The Iterable interface is the root interface for all the collection classes.
* The Collection interface extends the Iterable interface and therefore all the subclasses of Collection interface also implement the Iterable interface.
* By using Iterator, we can perform both read and remove operations
* It has 3 methods boolean hasNext(),Object next(), void remove()

Limitations of Iterator:

* Only forward direction iterating is possible.
* Replacement and addition of new element is not supported by Iterator.

[ListIterator](https://www.geeksforgeeks.org/iterators-in-java/#ListIterator)

* It is only applicable for List collection implemented classes like arraylist, linkedlist etc.
* It provides bi-directional iteration.

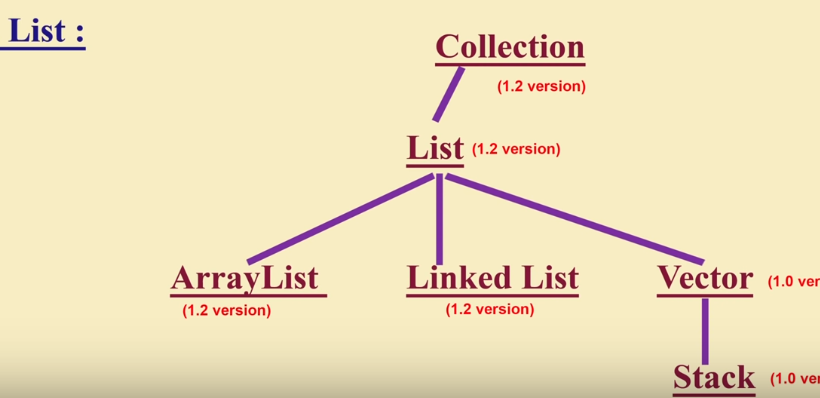
* ListIterator object can be created by calling *listIterator()* method present in List interface.
* It has 9 Methods: boolean hasNext() , Object next(), int nextIndex(), boolean hasPrevious(), Object previous() , int previousIndex(), void remove(), void set(Object obj), void add(Object obj)

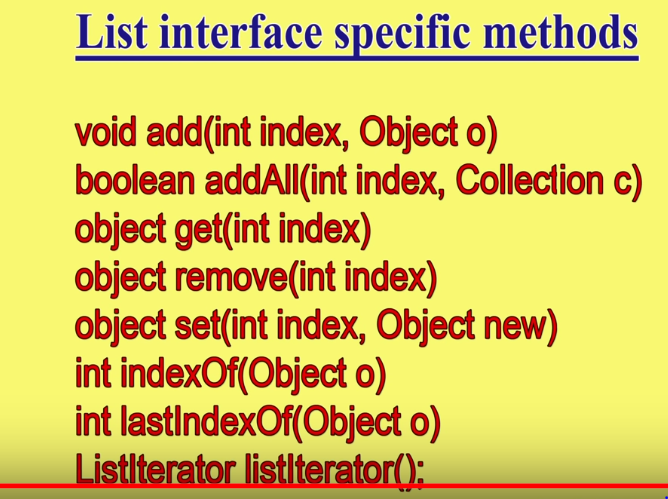
Limitations of ListIterator:

* It is the most powerful iterator but it is only applicable for List implemented classes, so it is not a universal iterator.

## **List Interface**

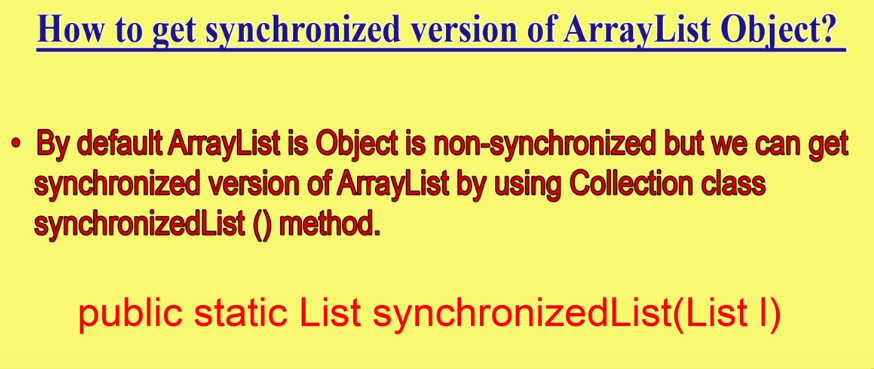
* The Java.util.List is a child interface of [Collection](https://www.geeksforgeeks.org/collections-in-java-2/).
* It is an ordered collection of objects in which duplicate values can be stored. List preserves the insertion order.
* Methods in List: void add(int index,Object O), boolean addAll(int index, Collection c), Object remove(int index), Object get(int index), Object set(int index, Object new), int indexOf(Object o), int lastIndexOf(Object o), List subList(int fromIndex,int toIndex)





## **ArrayList**

* ArrayList is a part of collection framework and is present in java.util package.
* It provides us dynamic arrays in Java.
* Insert order preserved
* Heterogenous objects allowed ( except in Treeset and TreeMap not allowed ).
* Null insertion is possible.
* **The Initial capacity of ArrayList is 10 (in Java 7 or before but in Java 8 onwards initial capacity is 0 ------🡪lazy initialization postpones this memory consumption till moment you will actually use the array list.)**
* ArrayList allows us to randomly access the list (It implements RandomAccess Interface).
* ArrayList is best choice if our frequent operation is retrieval operations, worst choice for insertion/removal of in middle.
* The ArrayList class maintains the insertion order and is non-synchronized.
* ArrayList cannot be used for primitive types, like int, char, etc

# **Marker interface in Java?**

* Marker Interface in java is an interface with no fields or methods within it.
* It is used to convey to the JVM that the class implementing an interface of this category will have some special behaviour.
* Marker interface is also called tag interface
* The following major marker interfaces
* Searilizable interface
* Cloneable interface
* Remote interface
* ThreadSafe interface

**LinkedList**

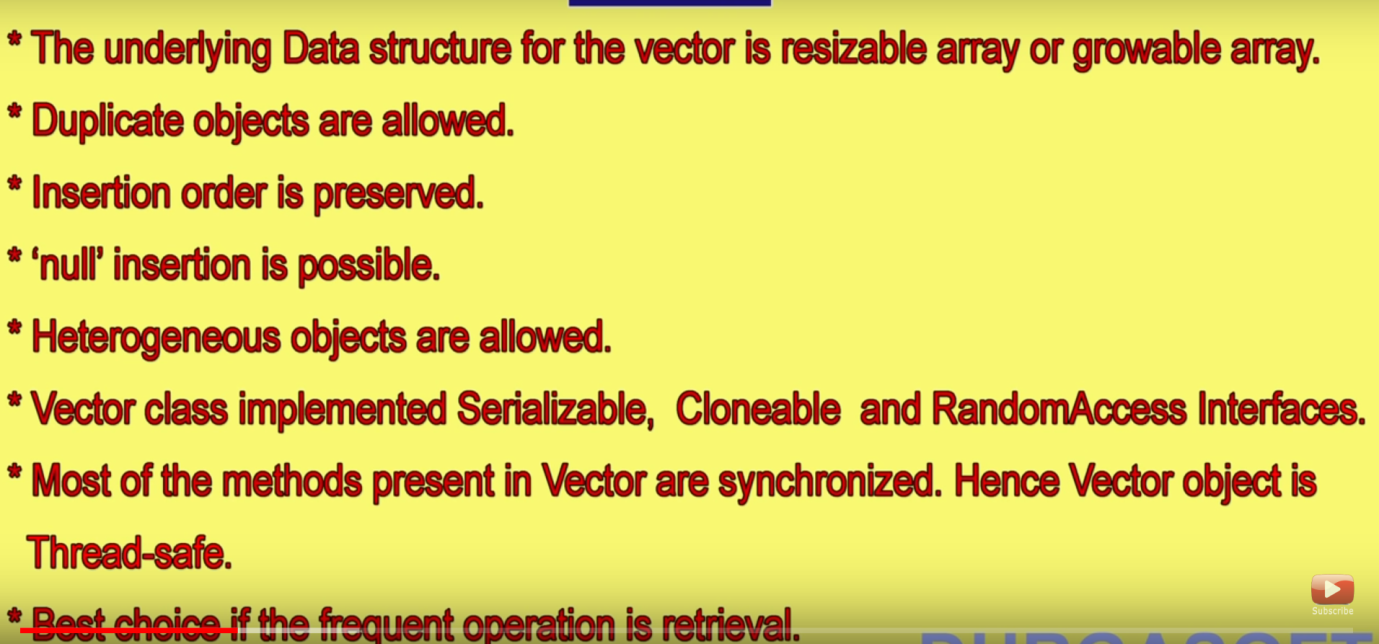
* LinkedList implements the Collection interface.
* Linked List are linear data structures(double linked list ) where the elements are not stored in contiguous locations and every element is a separate object with a data part and address part.
* It can store the duplicate elements.
* Heterogenous objects allowed (except in Treeset and TreeMap not allowed ).
* Null insertion is possible.
* LinkedList is best choice if our frequent operation is insertion/removal of in middle, worst choice is retrieval operations.

**Difference Between ArrayList and LinkedList**

|  |  |
| --- | --- |
| **ArrayList** | **LinkedList** |
| Implemented with the concept of dynamic array. | Implemented with the concept of doubly linked list. |
| Its elements can be accessed directly by using the get and set methods, since ArrayList is essentially an array. | Its performance on add and remove is better than Arraylist, but worse on get and set methods |
| Insertions are hard to compare to LinkedList | Insertions are easy and fast in LinkedList |
| Removals are hard to compare to LinkedList | Removal also better in LinkedList |
| Less memory overhead | LinkedList has more memory |
| In ArrayList elements will be stored in consecutive memory locations hence retrieval operation will become easy | In LinkedList the elements won't be stored in consecutive memory location and hence retrieval operation will be complex. |
| Its implement RandomAccess interface | It’s does not implement RandomAccess interface. |

# **Vector**

* The Vector class implements a growable array of objects.
* Vectors basically falls in legacy classes but now it is fully compatible with collections.
* They are very similar to ArrayList but Vector is synchronised.
* Vector allows us to randomly access the list (It implements RandomAccess Interface).



Difference Between Vector and ArrayList

|  |  |
| --- | --- |
| Vector | ArrayList |
| Vector is **synchronized**, which means only one thread at a time can access the code | arrayList is**not synchronized**, which means multiple threads can work on arrayList at the same time. |
| Vector is thread safe | ArrayList not thread safe |
| Vector operations give slower performance since they are synchronized | **ArrayList is faster**, since it is non-synchronized |
| Vector can use both [**Enumeration and Iterator**](https://www.geeksforgeeks.org/iterators-in-java/) for traversing over elements of vector | ArrayList can only use **Iterator,ListIterator** for traversing |

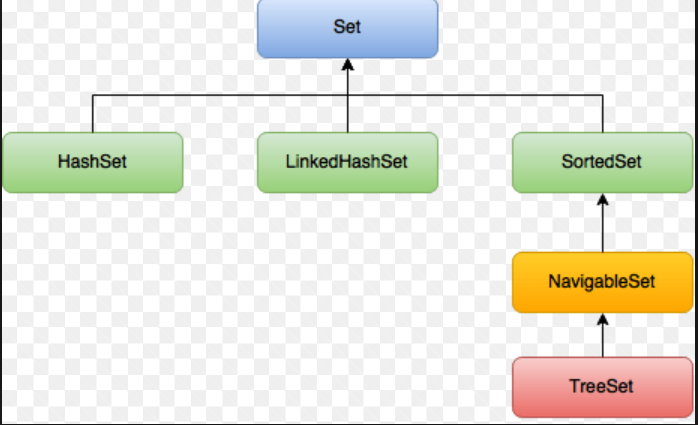
## **Stack**

* The stack is the subclass of Vector. It implements the last-in-first-out data structure
* The stack contains all of the methods of Vector class and also provides its methods like boolean push(), boolean peek(), boolean push(object o),empty(),search()

**Set Interface**

* Set is an interface which extends Collection. It is an unordered collection of objects in which duplicate values cannot be stored.
* We can store at most one null value in Set
* Set is implemented by HashSet, LinkedHashSet or TreeSet (sorted representation).
* Methods in Set: add (), clear (), contains (), isEmpty (), iterator (),

remove (), size ()



Difference between List and Set

|  |  |
| --- | --- |
| List | Set |
| List is an ordered collection it maintains the insertion order | Set is an unordered collection; it doesn’t maintain any order. There are few implementations of Set which maintains the order such as LinkedHashSet (It maintains the elements in insertion order) |
| List allows duplicates | Set doesn’t allow duplicate elements |
| List implementations: [ArrayList](https://beginnersbook.com/2013/12/java-arraylist/), [LinkedList](https://beginnersbook.com/2013/12/linkedlist-in-java-with-example/) etc | Set implementations: [HashSet](https://beginnersbook.com/2013/12/hashset-class-in-java-with-example/),  [LinkedHashSet](https://beginnersbook.com/2013/12/linkedhashset-class-in-java-with-example/), [TreeSet](https://beginnersbook.com/2013/12/treeset-class-in-java-with-example/) etc. |
| List allows any number of null values. | Set can have only a single null value at most. |
| [ListIterator](https://beginnersbook.com/2014/06/listiterator-in-java-with-examples/) can be used to traverse a List in both the directions | We can use [Iterator](https://beginnersbook.com/2014/06/java-iterator-with-examples/) (It works with List too) to traverse a Set. |
| List interface has one legacy class called [Vector](https://beginnersbook.com/2013/12/vector-in-java/) | Set interface does not have any legacy class. |

# **HashSet**

The HashSet class implements the Set interface

* Underlying data structure for HashSet is hashtable.
* As it implements the Set Interface, duplicate values are not allowed.
* Objects that you insert in HashSet are not guaranteed to be inserted in same order. Objects are inserted based on their hash code.
* NULL elements are allowed in HashSet.
* HashSet also implements Searlizable and Cloneable interfaces.
* HashSet uses HashMap for storing its object internally.

# **LinkedHashSet**

* A LinkedHashSet is an ordered version of [HashSet](http://quiz.geeksforgeeks.org/hashset-in-java/) that maintains a doubly-linked List across all elements.
* LinkedHashSet lets us iterate through the elements in the order in which they were inserted.

# **SortedSet Interface**

* SortedSet interface represents in Java Collection Framework.
* This interface extends [Set](http://quiz.geeksforgeeks.org/set-in-java/) and provides a total ordering of its elements.
* The elements in a SortedSet can be sorted in a natural order with Comparable interface or using a Comparator.
* The TreeSet class is an implementation for the SortedSet interface in the Collections Framework.
* Methods: comparator(),first(),headSet(E toElement),last(),subSet(E fromElement, E toElement),tailSet(E fromElement)

# **NavigableSet**

* NavigableSet represents a navigable set in Java Collection Framework.
* The NavigableSet interface inherits from the SortedSet interface.
* It behaves like a SortedSet with the exception that we have navigation methods available in addition to the sorting mechanisms of the SortedSet.
* The classes that implement this interface are, [TreeSet](http://quiz.geeksforgeeks.org/treeset-in-java/)

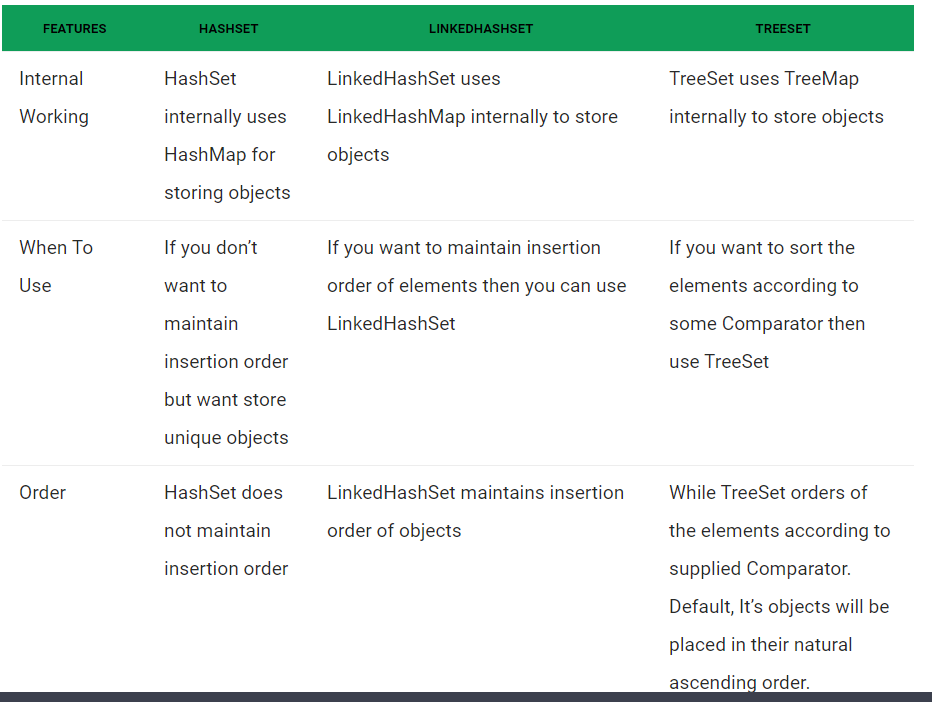
and ConcurrentSkipListSet

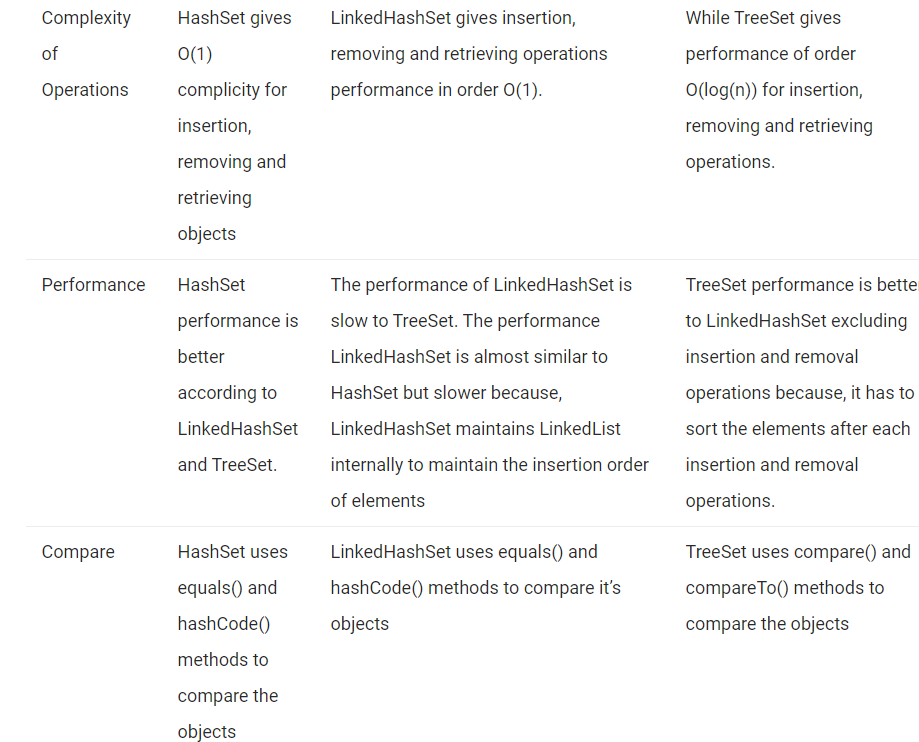
* Methods: Lower (E e) ,Floor(E e ),Ceiling(E e),Higher(E e),pollFirst(),pollLast()

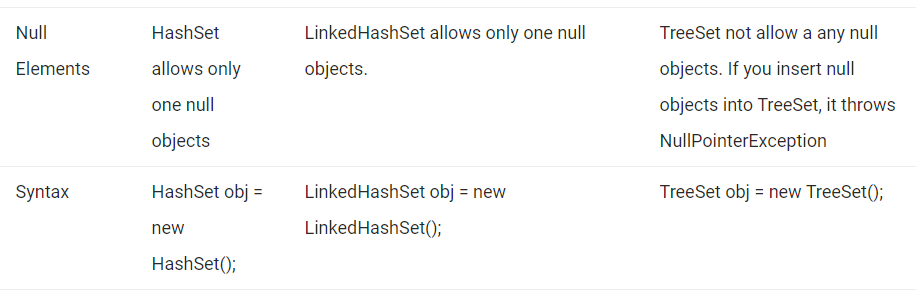
# **TreeSet**

* TreeSet is one of the most important implementations of the SortedSet interface in Java that uses a Tree for storage
* TreeSet implements the [SortedSet](https://www.geeksforgeeks.org/sortedset-java-examples/) interface so duplicate values are not allowed.
* Objects in a TreeSet are stored in a sorted and ascending order.
* TreeSet does not preserve the insertion order of elements but elements are sorted by keys.
* TreeSet does not allow to insert Heterogeneous objects. TreeSet serves as an excellent choice for storing large amounts of sorted information which are supposed to be accessed quickly because of its faster access and retrieval time.
* TreeSet is basically implementation of a self-balancing binary search tree like [Red-Black Tree](https://www.geeksforgeeks.org/red-black-tree-set-1-introduction-2/).

# **Differences Between HashSet, LinkedHashSet and TreeSet**





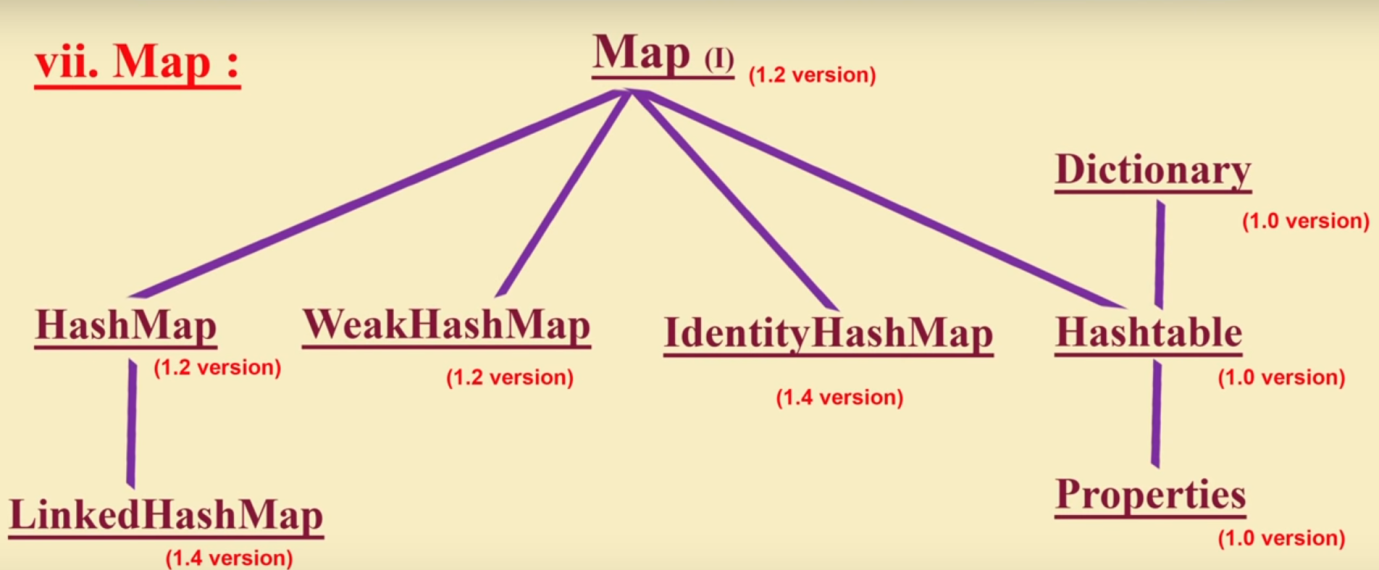


## **Queue Interface**

* Queue interface maintains the first-in-first-out order.
* If you want to represent group of individual objects prior to processing then we go for queue.
* It can be defined as an ordered list that is used to hold the elements which are about to be processed.
* [LinkedList](https://www.geeksforgeeks.org/linked-list-in-java/), ArrayBlockingQueue and [PriorityQueue](https://www.geeksforgeeks.org/priority-queue-class-in-java-2/) are the most frequently used implementations.
* If any null operation is performed on BlockingQueues, NullPointerException is thrown
* BlockingQueues have thread-safe implementations.
* All Queues except the Deques supports insertion and removal at the tail and head of the queue respectively. The Deques support element insertion and removal at both ends.

**Map Interface**

* Map is not a child interface of collection interface
* If you want to represent group of individual objects as key – value pair then we go for Map interface (Both Key and values are objects, duplicate keys are not allow, but duplicate values can allowed.
* The order of a map depends on specific implementations, e.g TreeMap and LinkedHashMap have predictable order, while HashMap does not.
* There are two interfaces for implementing Map in java: Map and SortedMap, and three classes: HashMap, TreeMap and LinkedHashMap.

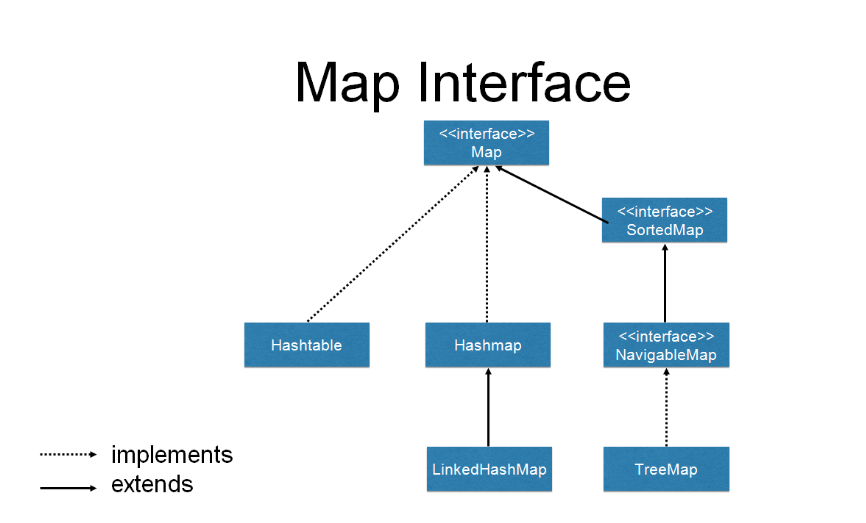


**Sorted Map Interface**

* It is child interface of Map Interface.
* If you want to represent group of key -value pairs according to some sorted order of keys.
* The main characteristic of a SortedMap is that, it orders the keys by their natural ordering, or by a specified comparator.
* null key or null value are not permitted.
* The keys are sorted either by natural ordering or by a specified comparator.

**Navigable Map Interface**

* **It is child interface of SortedMap interface, it provides several utility methods for navigation purpose.**



1. TreeSet implements the [SortedSet](https://www.geeksforgeeks.org/sortedset-java-examples/) interface so duplicate values are not allowed.

2. Objects in a TreeSet are stored in a sorted and ascending order.

3. TreeSet does not preserve the insertion order of elements but elements are sorted by keys.

4. TreeSet does not allow to insert Heterogeneous objects. It will throw classCastException at Runtime if trying to add hetrogeneous objects.

5. TreeSet serves as an excellent choice for storing large amounts of sorted information which are supposed to be accessed quickly because of its faster access and retrieval time.

6. TreeSet is basically implementation of a self-balancing binary search tree like [Red-Black Tree](https://www.geeksforgeeks.org/red-black-tree-set-1-introduction-2/). Therefore operations like add, remove and search take O(Log n) time. And operations like printing n elements in sorted order takes O(n) time.

7. Null insertion is allow (atmost one)

**Constructors of TreeSet class:**

1. **TreeSet t = new TreeSet();**

This will create empty TreeSet object in which elements will get stored in default natural sorting order.

2. **TreeSet t = new TreeSet(Comparator comp);**

This constructor is used when external specification of sorting order of elements is needed.

3. **TreeSet t = new TreeSet(Collection col);**

This constructor is used when any conversion is needed from any Collection object to TreeSet object.

4. **TreeSet t = new TreeSet(SortedSet s)**;

This constructor is used to convert SortedSet object to TreeSet Object.

**Servlet**

* **Servlet** technology is used to create a dynamic web application (resides at server side and generates a dynamic web page).
* **Servlet** technology is robust and scalable because of java language.
* The advantages of Servlet

1. **Better performance:** because it creates a thread for each request, not process.
2. **Portability:** because it uses Java language.
3. **Robust:** JVM manages Servlets, so we don't need to worry about the memory leak, garbage collection, etc.
4. **Secure:** because it uses java language.

**Servlets API’s:**  
 Servlets are build from two packages:

javax.servlet(Basic)` --- > Generic Servlet

javax.servlet.http(Advance) -🡪 http servlet

## Generic Servlet

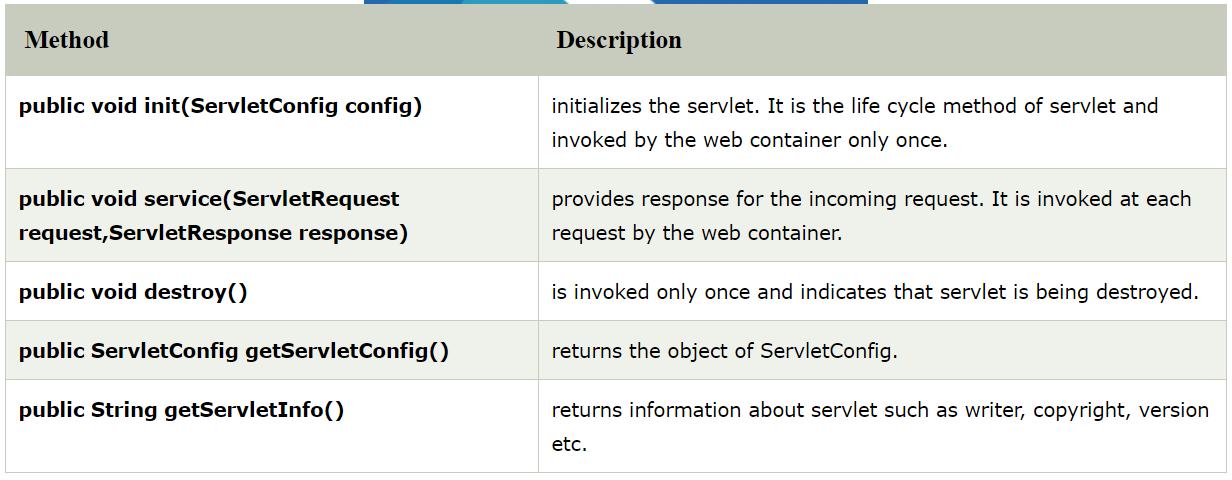
* if you are creating a Generic Servlet then you must extend javax.servlet.GenericServlet class.
* GenericServlet class has an abstract service() method. Which means the subclass of GenericServlet should always override the service() method.

## HTTP Servlet

* If you creating Http Servlet you must extend javax.servlet.http.HttpServlet class, which is an abstract class.
* Unlike Generic Servlet, the HTTP Servlet doesn’t override the service() method. Instead it overrides one or more of the following methods. It must override at least one method from the list below:

Servlet interface

There are 5 methods in Servlet interface. The init, service and destroy are the life cycle methods of servlet. These are invoked by the web container.



# ServletRequest Interface

## Methods of ServletRequest interface

**String getParameter(String name)**: It returns the value of the given parameter as String or null if the given parameter does not exist.

**Enumeration getParameterNames()**: It returns an Enumeration of Strings objects containing the names of parameters in the request.

**String[] getParameterValues(String name)**: It returns an array of Strings containing the all the values, the parameters has, returns null if parameter doesn’t have any value.

**String getCharacterEncoding()**: Returns the name of the character encoding used in the body of this request. This method returns null if the request does not specify a character encoding.

**void setCharacterEncoding(String env)**: Overrides the character encoding in the body of the request.

**int getContentLength()**: Returns the length of the request content in bytes.

**String getContentType()**: Returns the MIME type of the body of the request, or null if the type is not known.

# ServletResponse Interface

the servlet container creates HttpServletRequest and HttpServletResponse objects and passes them as an argument to the servlet service() method.

The response object allows you to format and send the response back to the client.

## Method of ServletResponse interface

1) String getCharacterEncoding(): It returns the name of the MIME charset used in body of the response sent to the client.  
2) String getContentType(): It returns the response content type. e.g. text, html etc.  
3) ServletOutputStream getOutputStream(): Returns a ServletOutputStream suitable for writing binary data in the response.  
4) java.io.PrintWriter getWriter(): Returns the PrintWriter object.  
5) void setCharacterEncoding(java.lang.String charset): Set the MIME charset (character encoding) of the response.  
6) void setContentLength(int len): It sets the length of the response body.  
7) void setContentType(java.lang.String type): Sets the type of the response data.  
8) void setBufferSize(int size): Sets the buffer size.  
9) int getBufferSize(): Returns the buffer size.  
10) void flushBuffer(): Forces any content in the buffer to be written to the client.  
11) boolean isCommitted(): Returns a boolean indicating if the response has been committed.  
12) void reset(): Clears the data of the buffer along with the headers and status code.

# RequestDispatcher  Interface

The **RequestDispatcher** interface defines an object that receives the request from client and dispatches it to the resource(such as servlet, JSP, HTML file). This interface has following two methods.

**public void forward(ServletRequest request, ServletResponse response)**: It forwards the request from one servlet to another resource (such as servlet, JSP, HTML file).

**public void include(ServletRequest request, ServletResponse response)**: It includes the content of the resource(such as servlet, JSP, HTML file) in the response.

# **SendRedirect**

The **sendRedirect()** method of **HttpServletResponse** interface can be used to redirect response to another resource, it may be servlet, jsp or html file.

# ServletResponse Interface

# **ServletConfig Interface**

An object of ServletConfig is created by the web container for each servlet. This object can be used to get configuration information from web.xml file.

## Methods of ServletConfig interface

**public String getInitParameter(String name)**: Returns the value of given parameter as String, or null if the given parameter doesn’t exist in web.xml.  
**public Enumeration getInitParameterNames()**: Returns an enumeration of all the parameter names.  
**public String getServletName()**: Returns the name of the servlet instance.  
**public ServletContext getServletContext()**: Returns an object of ServletContext.

**Note :**

The core advantage of ServletConfig is that you don't need to edit the servlet file if information is modified from the web.xml file.

# **ServletContext Interface**

An object of ServletContext is created by the web container at time of deploying the project. This object can be used to get configuration information from web.xml file. There is only one ServletContext object per web application.

If any information is shared to many servlet, it is better to provide it from the web.xml file using the **<context-param>** element.

1. The ServletContext object can be used to provide inter-application communication.

## Methods of ServletContext interface

Here is the list of frequently used methods of ServletContext interface.

**public String getInitParameter(String param)**: It returns the value of given parameter or null if the parameter doesn’t exist.

**public Enumeration getInitParameterNames()**: Returns an enumeration of context parameters names.

**public void setAttribute(String name,Object object)**: Sets the attribute value for the given attribute name.

**public Object getAttribute(String name)**:Returns the attribute value for the given name or null if the attribute doesn’t exist.

**public String getServerInfo()**: eturns the name and version of the servlet container on which the servlet is running.

**public String getContextPath()**: Returns the context path of the web application.

Advantage of ServletContext

**Easy to maintain** if any information is shared to all the servlet, it is better to make it available for all the servlet. We provide this information from the web.xml file, so if the information is changed, we don't need to modify the servlet. Thus it removes maintenance problem.

# **Attribute in Servlet**

An **attribute in servlet** is an object that can be set, get or removed from one of the following scopes:

1. request scope
2. session scope
3. application scope

# **Session Tracking in Servlets**

**Session** simply means a particular interval of time.

**Session Tracking** is a way to maintain state (data) of an user. It is also known as **session management** in servlet.

There are four techniques used in Session tracking:

1. **Cookies**
2. **Hidden Form Field**
3. **URL Rewriting**
4. **HttpSession**

# **Cookies in Servlet**

A **cookie** is a small piece of information that is persisted between the multiple client requests.

A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

There are 2 types of cookies in servlets.

1. Non-persistent cookie -- It is removed each time when user closes the browser.
2. Persistent cookie --- It is not removed each time when user closes the browser. It is removed only if user logout or signout.

Advantage of Cookies

1. Simplest technique of maintaining the state.
2. Cookies are maintained at client side.

Disadvantage of Cookies

1. It will not work if cookie is disabled from the browser.
2. Only textual information can be set in Cookie object.

# **JSP**

The JSP API consists of two packages:

1. javax.servlet.jsp
2. javax.servlet.jsp.tagext

## **javax.servlet.jsp package**

The javax.servlet.jsp package has two interfaces and classes.The two interfaces are as follows:

1. JspPage
2. HttpJspPage

The classes are as follows:

* JspWriter
* PageContext
* JspFactory
* JspEngineInfo
* JspException
* JspError

Methods of JspPage interface

1. **public void jspInit():** It is invoked only once during the life cycle of the JSP when JSP page is requested firstly. It is used to perform initialization. It is same as the init() method of Servlet interface.
2. **public void jspDestroy():** It is invoked only once during the life cycle of the JSP before the JSP page is destroyed. It can be used to perform some clean up operation.

## **The HttpJspPage interface**

The HttpJspPage interface provides the one life cycle method of JSP. It extends the JspPage interface.

### Method of HttpJspPage interface:

1. **public void \_jspService():** It is invoked each time when request for the JSP page comes to the container. It is used to process the request. The underscore \_ signifies that you cannot override this method.

# JSP Scriptlets

**Scriptlets are nothing but java code** enclosed within **<% and %> tags**. JSP container moves the statements enclosed in it to **\_jspService()** method while generating servlet from JSP.

each client’s request the \_jspService() method gets invoked, hence the code inside it executes for every request made by client.

**Note:** Semicolon at the end of scriptlet.

# **JSP expression tag**

The code placed within **JSP expression tag** is written to the output stream of the response. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

**<**%=  statement %**>**

**<**%= java.util.Calendar.getInstance().getTime() %**>**

# **JSP Declaration Tag**

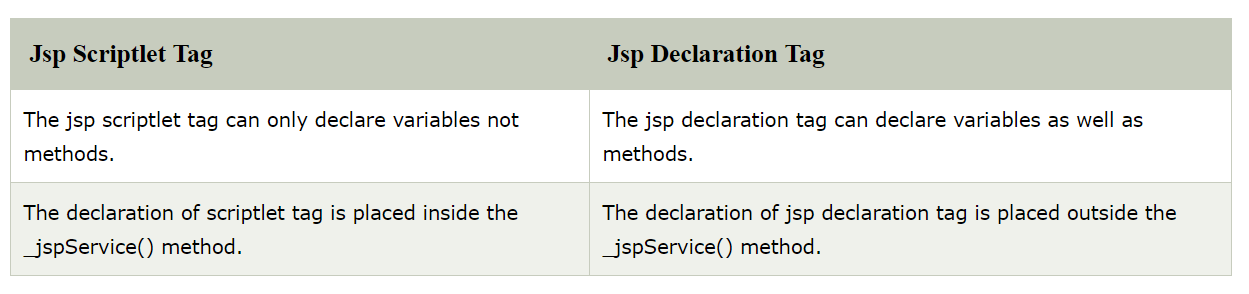
The **JSP declaration tag** is used to declare fields and methods.

The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.

So it doesn't get memory at each request.

**<**%!  field or method declaration %**>**

Difference between JSP Scriptlet tag and Declaration tag



**In which senerio to use arrayList or Map**

**How to update array element in MongoDB?**

# **what is use of Actutors in Spring Boot**

**Spring** Boot provides **actuator** to monitor and manage our **application**. **Actuator** is a tool which has HTTP endpoints. when **application** is pushed to production, you can choose to manage and monitor your **application** using HTTP endpoints.

**Monitoring our app, gathering metrics, understanding traffic or the state of our database becomes trivial with this dependency**

The main benefit of this library is that we can get production grade tools without having to actually implement these features ourselves.

Actuator is mainly used to **expose operational information about the running application** – health, metrics, info, dump, env, etc. It uses HTTP endpoints or JMX beans to enable us to interact with it.

**In Spring, you can use @PropertySource**annotation to externalize your configuration to a properties file.

@PropertySource(value="classpath:missing.properties", ignoreResourceNotFound=true)

**What is the use of @EnableWebMvc?**

When you're using Java code (as opposed to XML) to configure your Spring **application**, @**EnableWebMvc** is **used** to enable Spring MVC.

If you are using XML based configuration then use <mvc:annotation-driven/> as an alternative to @EnableWebMvc

# [**Spring: @Component versus @Bean**](https://stackoverflow.com/questions/10604298/spring-component-versus-bean)

**The limitations of the HTTP session**

Performance overhead due to **session** object being stored on server.

Overhead due to serialization and de-serialization of data