What is difference between JDK,JRE and JVM?

**JVM**

JVM is an acronym for Java Virtual Machine, it is an abstract machine which provides the runtime environment in which java bytecode can be executed. It is a specification.

JVMs are available for many hardware and software platforms (so JVM is platform dependent).

**JRE**

JRE stands for Java Runtime Environment. It is the implementation of JVM.

**JDK**

JDK is an acronym for Java Development Kit. It physically exists. It contains JRE + development tools.

### Heap

It is the runtime data area in which objects are allocated.

Stack

|  |
| --- |
| Java Stack stores frames.It holds local variables and partial results, and plays a part in method invocation and return. |
| Each thread has a private JVM stack, created at the same time as thread. |
| A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes. |

**Just-In-Time(JIT) compiler:** It is used to improve the performance.JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation.Here the term ?compiler? refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

**Object based programming** languages follow all the features of OOPs except Inheritance and Polymorphism. Examples of object based programming languages are JavaScript, VBScript etc.

**Object** means a real word entity such as pen, chair, table etc. **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

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

**Constructor** is just like a method that is used to initialize the state of an object. It is invoked at the time of object creation.

**Static variable** is used to refer the common property of all objects.

**A static method** belongs to the class rather than object of a class.

**Static Block** used to initialize the static data member. It is excuted before main method at the time of classloading.

**What is Inheritance?**

Inheritance is a mechanism in which one object acquires all the properties and behaviour of another object of another class. It represents IS-A relationship. It is used for Code Resusability and Method Overriding.

## **Why multiple inheritance is not supported in java?**

To reduce the complexity and simplify the language, multiple inheritance is not supported in java.

Consider a scenario where A, B and C are three classes. The C class inherits A and B classes. If A and B classes have same method and you call it from child class object, there will be ambiguity to call method of A or B class.

Since compile time errors are better than runtime errors, java renders compile time error if you inherit 2 classes. So whether you have same method or different, there will be compile time error now.

**Association** is a relationship between two separate classes which can be of any type say one to one, one to may etc. It joins two entirely separate entities.

Aggregation is a special form of association which is a unidirectional one way relationship between classes (or entities), for e.g. Wallet and Money classes. Wallet has Money but money doesn’t need to have Wallet necessarily so its a one directional relationship. In this relationship both the entries can survive if other one ends. In our example if Wallet class is not present, it does not mean that the Money class cannot exist.

Composition is a restricted form of Aggregation in which two entities (or you can say classes) are highly dependent on each other. For e.g. Human and Heart. A human needs heart to live and a heart needs a Human body to survive. In other words when the classes (entities) are dependent on each other and their life span are same (if one dies then another one too) then it’s a composition. Heart class has no sense if Human class is not present.

### What is method overloading?

If a class have multiple methods by same name but different parameters, it is known as Method Overloading. It increases the readability of the program. method overloading is not possible by changing the return type in java.

### What is method overriding?

If a subclass provides a specific implementation of a method that is already provided by its parent class, it is known as Method Overriding. It is used for runtime polymorphism and to provide the specific implementation of the method.

### What is covariant return type?

Now, since java5, it is possible to override any method by changing the return type if the return type of the subclass overriding method is subclass type. It is known as covariant return type.

### What is final variable?

If you make any variable as final, you cannot change the value of final variable(It will be constant).

### What is final method?

Final methods can't be overriden

### What is final class?

Final class can't be inherited.

### What is Runtime Polymorphism?

Runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time.

### Abstraction in Java

**Abstraction** is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only important things to the user and hides the internal details for example sending sms, you just type the text and send the message. You don't know the internal processing about the message delivery.

Abstraction lets you focus on what the object does instead of how it does it.

### Ways to achieve Abstaction

There are two ways to achieve abstraction in java

1. Abstract class (0 to 100%)
2. Interface (100%)

**An interface in** java is a blueprint of a class. It has static constants and abstract methods only.

The interface in java is a mechanism to achieve fully abstraction. There can be only abstract methods in the java interface not method body. It is used to achieve fully abstraction and multiple inheritance in Java.

Java Interface also represents IS-A relationship.

It cannot be instantiated just like abstract class.

An interface that have no member is known as marker or tagged interface. For example: Serializable, Cloneable, Remote etc. They are used to provide some essential information to the JVM so that JVM may perform some useful operation.

# Encapsulation in Java

**Encapsulation in java** is a process of wrapping code and data together into a single unit, for example capsule i.e. mixed of several medicines.We can create a fully encapsulated class in java by making all the data members of the class private. Now we can use setter and getter methods to set and get the data in it. The **Java Bean** class is the example of fully encapsulated class.

### What is the difference between abstraction and encapsulation?

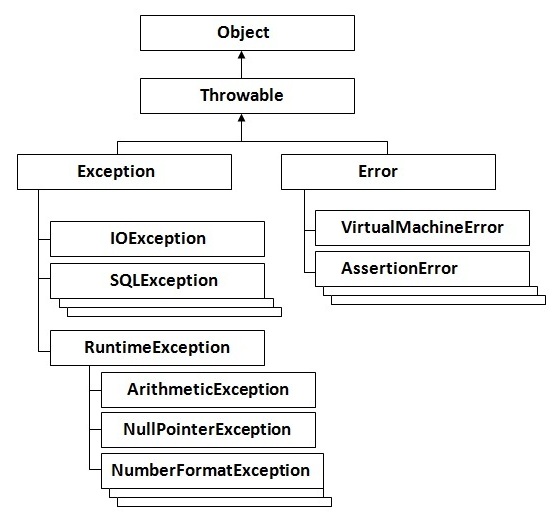
Abstraction hides the implementation details whereas encapsulation wraps code and data into a single unit.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Access Modifier** | **within class** | **within package** | **outside package by subclass only** | **outside package** |
| **Private** | Y | N | N | N |
| **Default** | Y | Y | N | N |
| **Protected** | Y | Y | Y | N |
| **Public** | Y | Y | Y | Y |

### What is exception

**Dictionary Meaning:** Exception is an abnormal condition. In java, exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

Hierarchy of Java Exception classes



### Types of Exception

There are mainly two types of exceptions: checked and unchecked where error is considered as unchecked exception. The sun microsystem says there are three types of exceptions:

1. Checked Exception
2. Unchecked Exception
3. Error

### 1) Checked Exception

The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions e.g.IOException, SQLException etc. Checked exceptions are checked at compile-time.

### 2) Unchecked Exception

The classes that extend RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time rather they are checked at runtime.

### 3) Error

Error is irrecoverable e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

**Java inner class** or nested class is a class i.e. declared inside the class or interface.

We use inner classes to logically group classes and interfaces in one place so that it can be more readable and maintainable.

### Difference between nested class and inner class in Java

Inner class is a part of nested class. Non-static nested classes are known as inner classes.

### Types of Nested classes

There are two types of nested classes non-static and static nested classes.The non-static nested classes are also known as inner classes.

1. Non-static nested class(inner class)
   * a)Member inner class
   * b)Annomynous inner class
   * c)Local inner class
2. Static nested class

|  |  |
| --- | --- |
| **Type** | **Description** |
| [Member Inner Class](http://www.javatpoint.com/member-inner-class) | A class created within class and outside method. |
| [Anonymous Inner Class](http://www.javatpoint.com/anonymous-inner-class) | A class created for implementing interface or extending class. Its name is decided by the java compiler. |
| [Local Inner Class](http://www.javatpoint.com/local-inner-class) | A class created within method. |
| [Static Nested Class](http://www.javatpoint.com/static-nested-class) | A static class created within class. |
| [Nested Interface](http://www.javatpoint.com/nested-interface) | An interface created within class or interface. |

### What is Garbage Collection?

Garbage collection is a process of reclaiming the runtime unused objects.It is performed for memory management.

What is difference between final, finally and finalize?

|  |
| --- |
| **final:** final is a keyword, final can be variable, method or class.You, can't change the value of final variable, can't override final method, can't inherit final class. |
| **finally:** finally block is used in exception handling. finally block is always executed. |
| **finalize():**finalize() method is used in garbage collection.finalize() method is invoked just before the object is garbage collected.The finalize() method can be used to perform any cleanup processing. |

### What is serialization?

Serialization is a process of writing the state of an object into a byte stream.It is mainly used to travel object's state on the network.

### What is Deserialization?

Deserialization is the process of reconstructing the object from the serialized state.It is the reverse operation of serialization.

### What is transient keyword?

If you define any data member as transient,it will not be serialized

### What is Externalizable?

Externalizable interface is used to write the state of an object into a byte stream in compressed format.It is not a marker interface.

**Java Reflection** makes it possible to inspect classes, interfaces, fields and methods at runtime, without knowing the names of the classes, methods etc. at compile time. It is also possible to instantiate new objects, invoke methods and get/set field values using **reflection**.

### What is multithreading?

Multithreading is a process of executing multiple threads simultaneously. Thread is basically a lightweight sub-process, a smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.But we use multithreading than multiprocessing because threads share a common memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.Java Multithreading is mostly used in games, animation etc.

# **Life cycle of a Thread (Thread States)**

1. New
2. Runnable
3. Running
4. Non-Runnable (Blocked)
5. Terminated

### What is synchronization?

Synchronization is the capabilility of control the access of multiple threads to any shared resource.It is used:

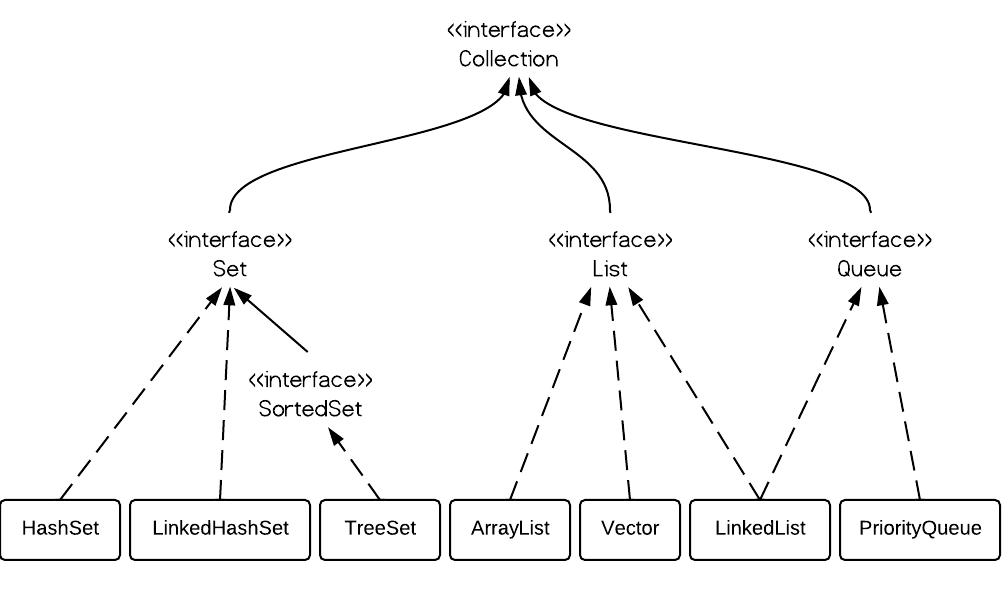
1. To prevent thread interference.
2. To prevent consistency problem

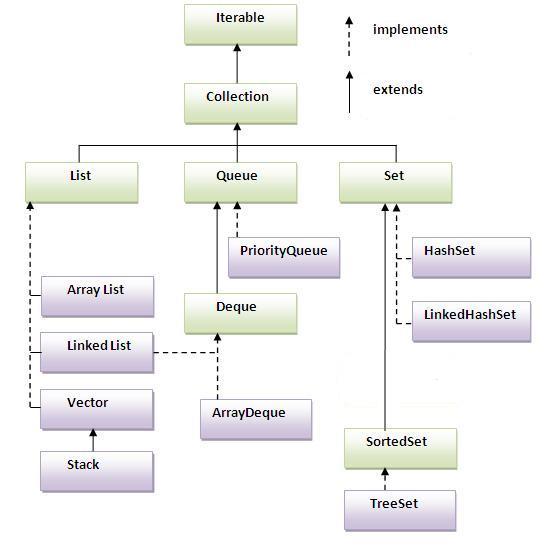
There are two types of thread synchronization mutual exclusive and inter-thread communication.

1. Mutual Exclusive
   1. Synchronized method.
   2. Synchronized block.
   3. Static synchronization.
2. Cooperation (Inter-thread communication in java)

# Collections in Java

# **Collections in java** is a framework that provides an architecture to store and manipulate the group of objects.





### Java Non-generic Vs Generic Collection

Java collection framework was non-generic before JDK 1.5. Since 1.5, it is generic.

Java new generic collection allows you to have only one type of object in collection. Now it is type safe so typecasting is not required at run time.

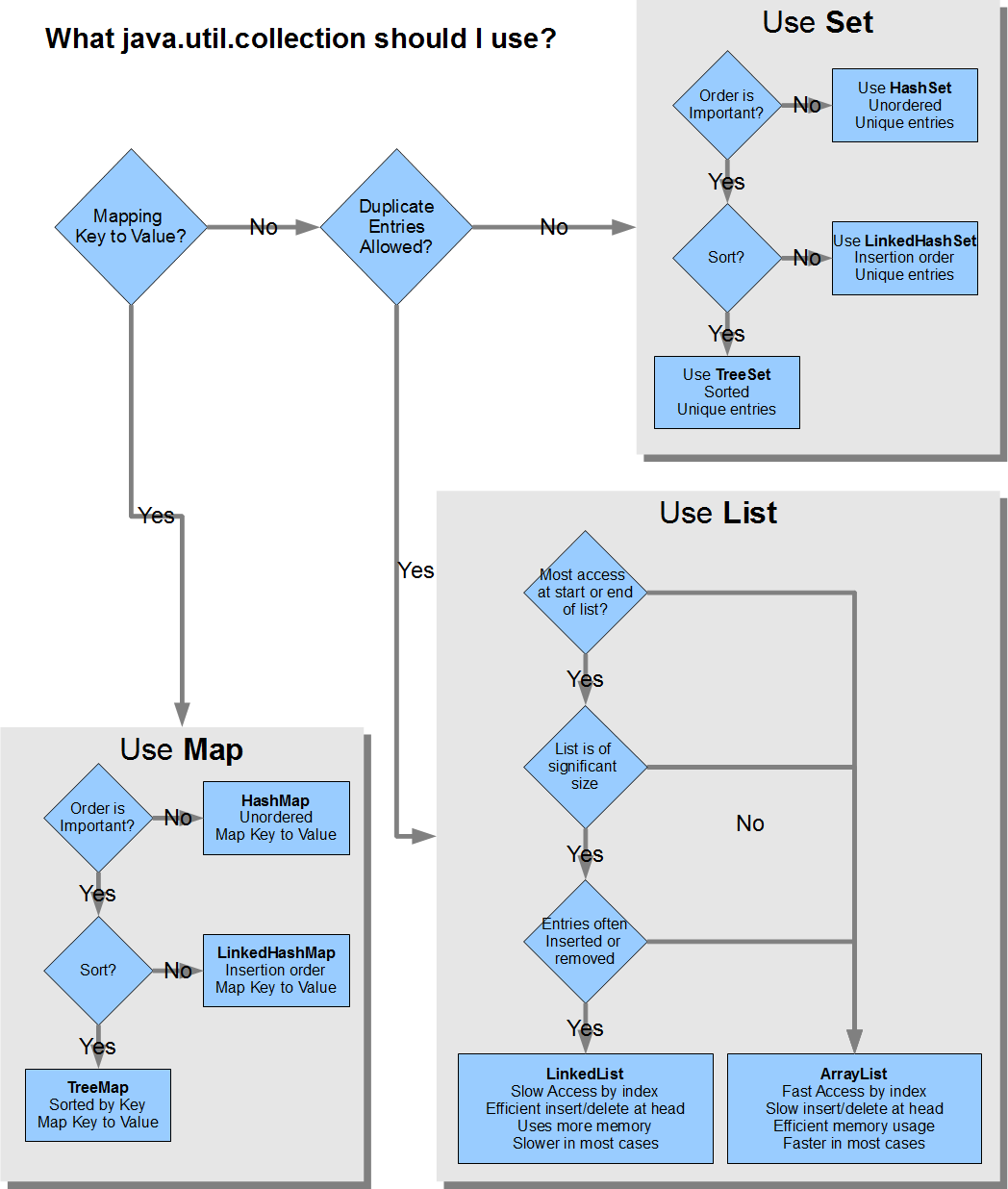
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Principal collection class | Duplicate? | Ordered? | Sorted? | Thread-safe? |
| ArrayList<E> | Yes | Yes | No | No |
| LinkedList<E> | Yes | Yes | No | No |
| Vector<E> | Yes | Yes | No | Yes |
| HashSet<E> | No | No | No | No |
| LinkedHashSet<E> | No | Yes | No | No |
| TreeSet<E> | No | Yes | Yes | No |
| HashMap<K, V> | No | No | No | No |
| LinkedHashMap<K, V> | No | Yes | No | No |
| Hashtable<K, V> | No | No | No | Yes |
| TreeMap<K, V> | No | Yes | Yes | No |

# Java Comparable interface

Java Comparable interface is used to order the objects of user-defined class.This interface is found in java.lang package and contains only one method named compareTo(Object). It provide single sorting sequence only i.e. you can sort the elements on based on single data member only. For example it may be rollno, name, age or anything else.

**Collections** class provides static methods for sorting the elements of collections. If collection elements are of Set or Map, we can use TreeSet or TreeMap. But We cannot sort the elements of List. Collections class provides methods for sorting the elements of List type elements.

**Java Comparator interface** is used to order the objects of user-defined class. This interface is found in java.util package and contains 2 methods compare(Object obj1,Object obj2) and equals(Object element). It provides multiple sorting sequence i.e. you can sort the elements on the basis of any data member, for example rollno, name, age or anything else.



HashMap works on the principal of hashing.

HashMap uses the hashCode() method to calculate a hash value. Hash value is calculated using the key object. This hash value is used to find the correct bucket where Entry object will be stored.

HashMap uses the equals() method to find the correct key whose value is to be retrieved in case of get() and to find if that key already exists or not in case of put().

Hashing collision means more than one key having the same hash value, in that case Entry objects are stored as a linked-list with in a same bucket.

With in a bucket values are stored as Entry objects which contain both key and value.

In Java 8 hash elements use balanced trees instead of linked lists after a certain threshold is reached while storing values. This improves worst case performance from O(n) to O(log n).

**Volatile: -** The volatile modifier guarantees that any thread that reads a field will see the most recently written value.

When is volatile enough?

If two threads are both reading and writing to a shared variable, then using the volatile keyword for that is not enough. You need to use synchronization in that case to guarantee that the reading and writing of the variable is atomic.

But in case one thread reads and writes the value of a volatile variable, and other threads only read the variable, then the reading threads are guaranteed to see the latest value written to the volatile variable. Without making the variable volatile, this would not be guaranteed.

**What is Busy Spinning? Why Should You Use It in Java?**

One of the interesting multithreading question to senior Java programmers, busy spinning is a waiting strategy, in which a thread just wait in a loop, without releasing the CPU for going to sleep. This is a very advanced and specialized waiting strategy used in the high-frequency trading application when wait time between two messages is very minimal.

**Difference between Heap and Stack Memory in Java JVM**

Even though both are part of JVM and both consumers memory allocated to the Java process, there are many differences between them e.g. Heap memory is shared by all threads of Java application but Stack memory is local to each thread. Objects are created in heap memory but method frames are stored in Stack memory, and size of heap space is much bigger than the small size of Stack in Java.

**Difference between Iterator and ListIterator in Java?**

Iterator only allows you to traverse in one direction i.e. forward, you have just got a next() method to get the next element, there is no previous() method to get the previous element. On the other hand, ListIterator allows you to traverse the list in both directions i.e. forward and backward.

**Data Structure and algorithm**

**Binary Search** using Recursion in Java, we calculate middle position by taking start and end position and check if the target element is equal to the middle element or not. If target, the number of element you are searching in an array is equal then our search is complete, but if the target is greater than middle we look on second half of array and if the target is less than middle element then we look into the first half of array.

# **Difference between Web Server vs Application vs Servlet Containers in Java JEE**

In the Java EE, or J2EE or JEE world, there are a lot of confusion between terminology, which is quite evident that now we have three words (J2EE, Java EE, and JEE) to describe the same technology platform. You will see Java developers using the terms like Web Server, Application Server, and Web containers interchangeably but they are not exactly same. The main difference between Web server and application server is that web server is meant to serve static pages e.g. HTML and CSS, while Application Server is responsible for generating dynamic content by executing server side code e.g. JSP, Servlet or EJB. One of the most popular web servers is Apache HTTPD which is often used in conjunction with Tomcat to host many Java web application. Now, tomcat is not exactly an application server, it's more of a servlet engine or web container or also known as servlet containers because it provides the runtime environment for [Servlet](http://java67.blogspot.com/2016/03/6-difference-between-forward-and-sendredirect-in-Servlet-JSP.html) and [JSP](http://java67.blogspot.com/2012/10/servlet-jsp-interview-questions-answer-faq-experience.html) but doesn't provide the services like EJB and distributed transaction which are a key feature of the application server in Java JEE world.

In order to better understand the difference, you must know some popular examples of the web server, application server, and web containers. Apache and IIS are two popular web servers. Apache is used everywhere including Java world but IIS is more popular in Microsoft ASP .NET world. From Java EE perspective couple of popular application servers are IBM WebSphere, Oracle WebLogic, Glassfish and Redhat's JBoss. Coming back to web containers or servlet engines, [Apache Tomcat](http://javarevisited.blogspot.com/2013/07/how-to-configure-https-ssl-in-tomcat-6-7-web-server-java.html) and [Jetty](http://javarevisited.blogspot.com/2012/05/5-difference-between-application-server.html) are two of the most popular Servlet engine in Java web world.

Remember, each one of the have different purpose and many times used together in Java Web world. For example, one of the popular setups is apache fronting Tomcat. You can use this setup if your Web application is only using JSP and Servlet and doesn't need EJB or distributed transaction feature. More basic services like Database connection pooling is also provided by Tomcat. On the other hand, if you have full fledged Java EE application including EJB then you need a proper application server like WebSphere, WebLogic or JBoss to host your application.

**Types of Session Tracking in Servlet**

Since Session management needs to work with all web browsers and also considers user's security preference, often an identifier i.e. a SessionId is used to keep track of request coming from the same client during a time duration. There are four main ways to manage Session in Java Web application written using Servlet and JSP.

1) URL rewriting

2) Cookies

3) Hidden Form fields

4) HTTPS and SSL

**Difference between forward() and sendRedirect() and include() )**

Servlet in JEE platform provides two methods forward() and sendRedirect() to route an HTTP request to another Servlet for processing. Though, both are used for forwarding HTTP requests for further processing there are many differences between forward() and sendRedirect() method e.g. forward is performed internally by Servlet, but a redirection is a two-step process, where Servlet instruct the web browser (client) to go and fetch another URL, which is different from the original. That's why forward() is also known as a server-side redirect and sendRedirect() is known as the client-side redirect.

Just remember that, even though you can get the RequestDispatcher reference from ServletContext.getRequestDispatcher() method or ServletRequest.getRequestDispatcher() method but there is a slight difference, the pathname of Servlet must begin with a / and is interpreted as relative to the current context root, while in case of ServletRequest, path can be relative.

**Enums** are a collection of a finite number of well-known objects, often we need to iterate over them. Enums are also final in Java and has a private constructor, which means you can not create enum instances once declared.

# **Spring**

Spring framework is an open source Java platform that provides comprehensive infrastructure support for developing robust Java applications very easily and very rapidly.

Spring is the most popular application development framework for enterprise Java. Millions of developers around the world use Spring Framework to create high performing, easily testable, reusable code.

## Dependency Injection (DI):

The technology that Spring is most identified with is the Dependency Injection (DI) flavor of Inversion of Control. The Inversion of Control (IoC) is a general concept, and it can be expressed in many different ways and Dependency Injection is merely one concrete example of Inversion of Control.

## Aspect Oriented Programming (AOP):

One of the key components of Spring is the Aspect oriented programming (AOP) framework. The functions that span multiple points of an application are called cross-cutting concerns and these cross-cutting concerns are conceptually separate from the application's business logic. There are various common good examples of aspects including logging, declarative transactions, security, and caching etc.

The Spring container is at the core of the Spring Framework. The container will create the objects, wire them together, configure them, and manage their complete lifecycle from creation till destruction. The Spring container uses dependency injection (DI) to manage the components that make up an application.

## Spring Beans Scope

When defining a <bean> in Spring, you have the option of declaring a scope for that bean. For example, To force Spring to produce a new bean instance each time one is needed, you should declare the bean's scope attribute to be**prototype**. Similar way if you want Spring to return the same bean instance each time one is needed, you should declare the bean's scope attribute to be**singleton**.

|  |  |
| --- | --- |
| **Scope** | **Description** |
| singleton | This scopes the bean definition to a single instance per Spring IoC container (default). |
| prototype | This scopes a single bean definition to have any number of object instances. |
| request | This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext. |
| session | This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext. |
| global-session | This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext. |

**BeanPostProcessor** Interface

**Bean Definition Inheritance**