*Java 9 Features*

1. *Factory methods for collections - collection factory methods have been added.*

*A collection factory method in Java is a static method that provides a simple way of initializing an immutable Collection<E>.*

*Being immutable, no elements can be added to, removed from, or modified inside the Collection<E> after it is initialized.*

*List and Set interfaces have “of()” methods to create an empty or no-empty Immutable List or Set objects*

*Ex –*

*Empty List example: - List immutableEmptyList = List.of();*

*Non-Empty List example: - List immutableList = List.of("Abc", "Def", "Ghi");*

*Empty Map Example: - Map emptyImmutableMap = Map.of()*

*emptyImmutableMap ==> {}*

*Non-Empty Map Example: - Map nonemptyImmutableMap = Map.of(1, "one", 2, "two", 3, "three")*

*nonemptyImmutableMap ==> {2=two, 3=three, 1=one}*

1. *From Java 9, we can create private methods inside an interface that help to share common code between non-abstract methods.*

*Example -*

*public interface Person{*

*private Long createID(){*

*// Implementation goes here.*

*}*

*private static void displayDetails(){*

*// Implementation goes here.*

*}*

*}*

1. *Java 9 REPL (JShell) - Oracle Corp has introduced a new tool called “jshell”. It stands for Java Shell and known as REPL (Read Evaluate Print Loop). It is used to execute and test any Java Constructs like class, interface, enum, object, statements etc. very easily.*
2. *Module*

*the problems of Current Java System. Java SE 8 or earlier systems have following problems in developing or delivering Java Based applications.*

* *As JDK is too big, it is a bit tough to scale down to small devices. Java SE 8 has introduced 3 types of compact profiles to solve this problem: compact1, compact2, and compact3. But it does not solve this problem.*
* *JAR files like rt.jar etc are too big to use in small devices and applications.*
* *]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]As JDK is too big, our applications or devices are not able to support better Performance.*
* *There is no Strong Encapsulation in the current Java System because “public” access modifier is too open. Everyone can access it.*
* *As JDK, JRE is too big, it is hard to Test and Maintain applications.*
* *As the public is too open, They are not to avoid the accessing of some Internal Non-Critical APIs like sun.\*, \*.internal.\* etc.*
* *As User can access Internal APIs too, Security is also big issue.*
* *Application is too big.*

*Its a bit tough to support Less Coupling between components.*

*To solve all these problems, Oracle Corp is going to release Java Module system in Java SE 9 Release.*

*Advantages of Java SE 9 Module System*

*Java SE 9 Module System is going to provide the following benefits*

* *As Java SE 9 is going to divide JDK, JRE, JARs etc, into smaller modules, we can use whatever modules we want. So it is very easy to scale down the Java Application to Small devices.*
* *Ease of Testing and Maintainability.*
* *Supports better Performance.*
* *As public is not just public, it supports very Strong Encapsulation. (Don’t worry its a big concept. we will explore it with some useful examples soon).*
* *We cannot access Internal Non-Critical APIs anymore.*
* *Modules can hide unwanted and internal details very safely, we can get better Security.*
* *Application is too small because we can use only what ever modules we want.*
* *Its easy to support Less Coupling between components.*
* *Its easy to support Single Responsibility Principle (SRP).*

1. ***Enhanced @Deprecated annotation*** *-Initially, it was just a Marker Interface, but now it provides more information about deprecated API and also provide a Tool to analyse an application’s static usage of deprecated APIs. They have added two methods to this Deprecated interface:* ***forRemoval()*** *and* ***since()*** *to serve this information.*
2. ***Stream API Improvements***

*In Java SE 9, Oracle Corp has added four useful new methods to java.util.Stream interface. As Stream is an interface, all those new implemented methods are default methods. Two of them are very important: dropWhile() and takeWhile() methods*

***6.1 Stream takeWhile*** *method takes each element that matches its predicate. It stops when it get unmatched element.*

***6.2 dropWhile() Method***

*Stream dropWhile method returns result on the basis of order of stream elements.*

*Ordered stream: It returns a stream that contains elements after dropping the elements that match the given predicate.*

*Unordered stream: It returns a stream that contains remaining elements of this stream after dropping a subset of elements that match the given predicate.*

***6.3 ofNullable Method***

*Stream ofNullable method returns a sequential stream that contains a single element, if non-null. Otherwise, it returns an empty stream.*

*It helps to handle null stream and NullPointerException.*