**Java 9 features**

1. Private method in interface
2. Try with resources enhancements
3. Safe varargs annotations
4. <> operator enhancements
5. Factory method to create unmodifiable collections
6. Stream api enhancements
7. JPMS(Java platform module system)
8. JShell
9. JLink(Java Linker)
10. HTTP/2 client
11. Process API Updates
12. G1GC

**JShell**

* It’s a REPL(Read-Evaluate-Print-Loop) tool to learn java concept in fun way or we can test the code snippet in easy way. No need to write main method with class.
* It’s not maint for main coding .it just for code testing.
* To execute one method for getting the data from DB. Store that code into db.jsh file. To load that file use command /open db.jsh and run using method name like test();
* Load jar file using command >jshell –v –class-path c:\xyz\mysql.jar;

**JPMS**

* In jar file based application we are using jar file. So at the time of execution if jvm is not able to find the class then it will not execute the programme.

NoClassDefFoundError

Version conflict

Monolithic structure

Security issue(all packages will expose)

Heavy weight(rt.jar file have to maintain)

* Module is also a group of packages similar to jar file but every module contains one configurable file that is module-info.java.in that every thing we have to mention inside this configuration file.
* > javac --module-source-path src -d out -m moduleA

> java --module-path out -m moduleA/pack1.Test

> java –module-path out -- show-module-resolution –m moduleA/pack1.Test

* Optional dependency mandatory at compile time but optional at run time.
* Cyclic dependency not allowed.
* Module graph is represent the dependency between module.
* Aggregator module provides the multiple common modules. here we have to use transitive module.eg:travelAgent module
* two jar files can contain the same package name but different module can’t contain package with the same name. So no chance for version conflicts.

**JLink**

* To run simple program we required rt.jar file(4500 classess) 400+mb JRE . But we can create our own customized jre using JLINK for light weight application.
* Copy required module from jdk to out folder and run the command.

>jlink -- module-path out – add-modules demoModule,java.base – output durgajre

**HTTP/2 Client**

* Used to send http request from java program(not from the browser) to webserver . Before that we are using HttpUrlConnection that so many problems.

**Process Update API**

* Used to find current running JVM processId.

**Private Method in Interface**

* For code reusability private method in interface came. For that suppose we have multiple default methods in interface and written common code then common code write into private method and use that private method wherever we required common code.

**Try with Resources Enhancement**

* Until 1.8 we can’t declare already created resources in try block, we have to create and declare inside try block or take new local variable using created resources. So that no need to use finally block ,automatically resources will be close but it’s allowed in 1.9.

**Factory Method to create Unmodifiable Collection**

**->Collections.unmodifiableList(l)//1.8**

**List.of(“A”,”B”,”C”);//1.9**

* **It will throw unsupportedOperationException**
* **Null not allowed**

**Stream API Enhancements**

* Two Default methods in Stream Interface :

takeWhile() : If condition is failed then it terminates the execution.

dropWhile() : Exactly opposite of takeWhile() method

* Two Static methods in Stream Interface to iterate two arguments or three arguments or check nullpointerexception explicitly: Stream.iterator() and Stream.ofNullable().

**<> Operator**

* We can use <> operator for anonymous inner classes.

**SafeVarargs annotation**

* If we are using varargs with generics then heap pollution(one type of variable is going to point another type of object at run time) problem can come. So if we are confident than heap pollution problem will never come then we can suppress the warning using SafeVarargs annotation.
* Until 1.8 we can use for static, final methods and constructor. but from 1.9 we can use for private method also.

**G1GC**

* Serial GC ,Parellel GC and CMS GC works on generation based. But Heap is divided into multiple regions also so G1GC will work on regions where maxm no of object are eligible for GC.
* Until 1.8 Parellel GC was the default garbage collector but 1.9 onwards G1GC is the default GC and it will delete more no object in less time.

