Java 10

Link :- http://openjdk.java.net/projects/jdk/
Release Notes:- https://www.oracle.com/java/technologies/javase/10-relnote-issues.html#NewFeature

Features Lists :-

1. var keyword
2. Collectors API enhancement
3. Docker Awarnes
4. orElseThrow() in Optional
5. copyOf(Collection c) in List, Set, Map
6. Java based JIT compiler used to convert byte code to native code.

1. var :-
   - Java 10 has came with a very cool feature namely 'var'.
   - It removed the headache of the programmer to take care about the type of a variable.
   - Java identifies the type of variable via its value.
   - We can assign any kind of value in it like int,float, List, Set, Custom Object etc.
   - For ex :-
     - var i = 10;
     - var f = 10.10f
     - var l = new ArrayList<>()
     - var s = new HashSet<>()
     - var custom = new CustomObject()
   - var is not a keyword. We can use it as name of the any variable. For ex :-
     - var var = 10;
   - This is because it is possible that in previous versions of java any variable may be declared with var name. If this become keyword then compile time error will be occurred.
   - So to support legacy code, var is not created as keyword.

   - We can only use var at the time of variable declaration. This is allowed only at that place. Otherwise it is an compilation error.For ex :-
      - var i = 10;   i = (var) 10.10f;
      - List<var> l = new ArrayList<>();
      - var l = new ArrayList<var>();
   - All such kind of syntaxes are not supported.
   - var type variable can not be assigned with null. Because with null value java is not able to identify the type of the variable. For ex :-
      - var s = null;
   - We can not use var with assignment of lambda expression. For ex
      - Consumer<Integer> i = (i) -> {System.out.print(i)};   --->> Here we can not replace it as follows :-
      - var i = (i) -> {System.out.print(i)}; // Compilation Error
   - var type variables are not supported as class level variables whether it is initialized or not.
2. Collectors API :-
   - Now we can get unmodifiable collections also from collectors.
   - JDK-10 introduced 4 new methods in collectors API.
      - toUnmodifiableList()
      - toUnmodifiableMap()
      - toUnmodifiableSet()
3. Docker Awareness :-
   - Now JDK-10 is aware about the concept of containers.
   - Before JDK-10, JVM pulls the container specific configuration from OS.
   - But now we can define it using JAVA OPTIONS or JAVA-OPTS.
   - By default this feature is enabled. We can disable it by using following command line option :-
      - -XX : -UseContainerSupport
   -
4. orElseThrow() :-
   - orElseThrow() is introduced in java.util.Optional class.
   - It is totally similar to get() method of this class.
   - But it is more preferred.
   - Because it also have a overloaded version of it.
   - orElseThrow(Suplier<? extends X> exceptionSupplier) throws X

   - Now we have more control over exception that we want to through in case of absence of value.

5. copyOf() :-
   - copyOf() is introduced in List, Map & Set.
   - It returns unmodifiable collections as a response.
     - List<String> l = new ArrayList<>();   List<String> fix = List.copyOf(l);
6. Java based JIT compiler :-
   - Until JDK-8, JIT compiler was in c++.
   - But in JDK-9 it was introduced as Graal and started working on it.
   - Now it is available with stability in java-10 and can be enabled via following JVM options :-
     - -XX:+UnlockExperimentalVMOptions -XX:+UseJVMCICompiler
   - This is more efficient then JIT in c++.
   - It is more faster then JIT.
7. Root Certificates :-
   - Good news is that Oracle open-sourced the root certificate.
   - So build creation and security is more reliable.
   - Initially root cacerts in java was empty.
   - But now it contains set of root certificate that is used to establish the trust in certificate chains.
   - So as a result of it, critical security components such as TLS do not work under JDK-10.