**Java 8**

* The main Objective of java8 is to reduce the number of lines of code by implementing functional programming.
* The main features include:

1. Lambda expressions
2. Functional Interface
3. Default methods and static methods
4. Predefined Functional interfaces like

Predicate

Function

Consumer

Producer etc

1. Double colon operator(::)

Method reference

Constructor reference

1. Streams
2. Date and Time API
3. Optional Class
4. Nashorn javascript

**Lambda Expressions:**

1. It is already available in LISP, Python, C, C++, Ruby, Scala. Finally Java also implemented this feature.
2. It is an anonymous function, without return type, without modifiers.
3. Public void m1()

{

System.out.println(“hello”);

}

1. As it doesn’t have name, modifier, return type remove them. But to convey that it is lambda expression, we need to use “->” symbol.

() -> System.out.println(“hello”);

1. Compiler gets the data type automatically sometimes. In that case we can remove the datatype.

(a,b) -> System.out.println(a+b);

1. If the function body contains only one line and it is return stmt, u can remove the keyword return. Below are the examples.

n->n\*n;

s->s.length();

**Functional Interfaces:**

* An interface contains only single abstract method is called functional interface.
* Collections.sort(lst, (emp1,emp2) -> (emp1.eno<emp2.eno)?-1:(emp1.eno>emp2.eno)?1:0);
* Collections.sort(lst, (e1,e2)->e1.ename.compareTo(e2.ename));

**Default Methods and Static Methods:**

* Without effecting implementation classes if we want to add new methods to the interface(extending interface functionality), we use default methods.
* Also default methods avoid the burden of implementing all unnecessary methods of interface in the implementation class. I will just implement whatever I need in my class.
* We can’t make Object class methods as default methods in our interfaces.
* Interface static methods can only be called using Interface name even if it is implementation class. By default these methods won’t be available in implementation classes.
* In interface you can define, public abstract or default or static methods only. All other methods are invalid.
* If same default method is available in two different interfaces and if we want to implement both of them in a single class, it is not possible. So we need to create one more interface which extending both of these and override this methods by calling A.super.m1() and then implement.
* From 1.8 onwards, default methods and public static methods are also allowed. Public static methods are useful for common utilities.

**Predefined Functional Interfaces:**

* These will be useful to use lambda expression in our daily programming. They include:

1. Predicate
2. Function
3. Consumer
4. Supplier

* Two argument Predefined Functional Interfaces:

1. BiPredicate
2. BiFunction
3. BiConsumer

* Primitive Functional Interfaces

1. IntPredicate
2. IntFunction
3. IntConsumer

**Predicate:**

* It is a functional interface contains only one public abstract boolean test(T t) abstract method.
* It is useful in lambda expressions to replace the conditional check code.

Ex:

Predicate p<Integer> p = i -> i>10;

sop(p.test(15)); -> true

**Function:**

* It is a functional interface contains only one public abstract <T2> apply(T1 t1, T2 t2) abstract method.
* It is useful in lambda expressions to chk and return some value.

Function<String, Integer> f = s1->s1.length();

Sop(f.apply(“string”));

**Consumer:**

* If we need to give input and perform some operations and don’t expect any output then we should go for Consumer.
* It contains void accept(T t) method.

Consumer<String> c = s1 -> sop(s1);

c.accept(“hello”);

**Supplier:**

* It won’t accept any argument but return some value.
* It contains R get() method.

Supplier<String> s = () -> {some random logic

return “output”

}

Sop(s.get());

**Streams:**

* To process the data present in the collections then Streams will be useful.
* filter method is used to filer some data. Eg., to get only even numbers from the list.
* map method is used to apply some function on every object. Eg., to convert all strings to upper class.

**Spring Boot**

* **how do I enable auto reload of spring boot changes?** -> By using spring boot developer tools. Add spring-boot-devtools dependency and restart ur app.
* **How do u configure two different envs?** -> using profiles.

Eg., application-prod.properties and application-dev.properies then use spring.profiles.active = prod in application.properties.

* Web.xml, dispatcher-servlet.xml, applicationContext.xml

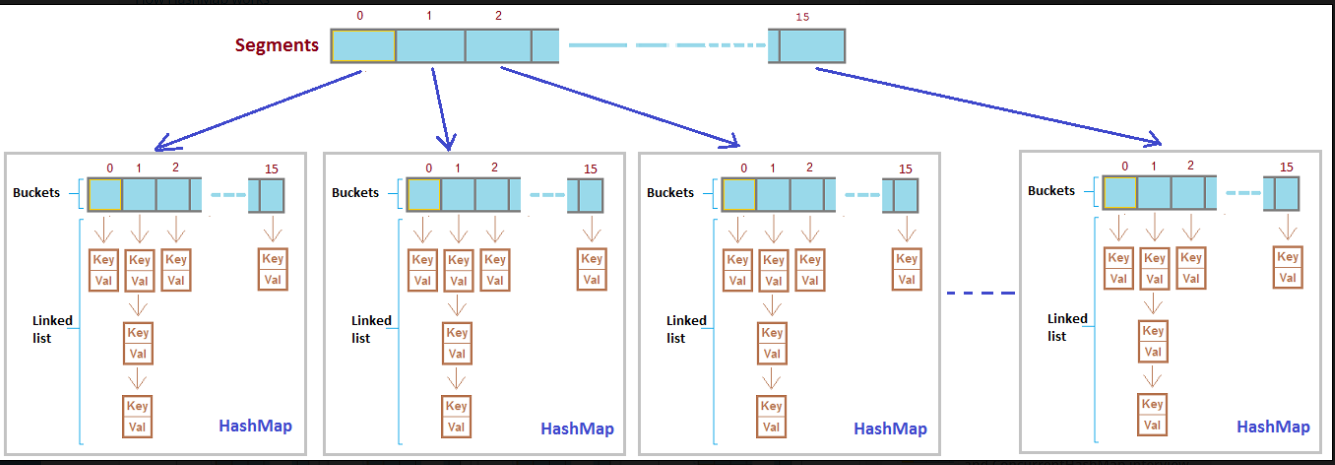
**Features:**

* **Auto configure**  - Detects and automatically configures the Spring applications based on the added dependencies
* **CLI** - Command Line Interface to start, test and stop Spring Boot applications from command prompt (Not discussed in this course)
* **Actuator** - Enables enterprise features and gives the insight of the application

**Int 1:**

* 1. **What is concurrent hashmap and when u use it?**
* **Ans:** Prerequisites:Need of ConcurrentMap  
  ConcurrentHashMap ConcurrentHashMap class is introduced in JDK 1.5, which implements ConcurrentMap as well as Serializable interface also. ConcureentHashMap is enhancement of HashMap as we know that while dealing with Threads in our application HashMap is not a good choice because performance wise HashMap is not upto the mark.
* Key points of ConcurrentHashMap:
* The underlined data structure for ConcurrentHashMap is Hashtable.
* ConcurrentHashMap class is thread-safe i.e. multiple thread can operate on a single object without any complications.
* At a time any number of threads are applicable for read operation without locking the ConcurrentHashMap object which is not there in HashMap.
* In ConcurrentHashMap, the Object is divided into number of segments according to the concurrency level.
* Default concurrency-level of ConcurrentHashMap is 16.
* In ConcurrentHashMap, at a time any number of threads can perform retrieval operation but for updation in object, thread must lock the particular segment in which thread want to operate.This type of locking mechanism is known as Segment locking or bucket locking.Hence at a time 16 updation operations can be performed by threads.
* null insertion is not possible in ConcurrentHashMap as key or value.
  1. **How concurrenthashmap contains segments?**

Ans: ConcurrentHashMap added one Array on top of it and each index of this additional array represents complete HashMap. Additional array is called Segment in ConcurrentHashMap.



* 1. **How concurrenthashmap decides in which segment data to be stored?**

**Ans:** Once we get the hashcode value with hashcode() method, this value will be sent to segments() method to identify the segment number.

* 1. **How hashset use hashmap where as hashset having only value but map having key, value pair? What is that value passed to map in case of hashset?**

**Ans:** A dummy Object class object(PRESENT) value to associate with an Object in the backing up for Hashmap value.

* 1. **What is the special advantage of in built FunctionalInterface Predicate instead of if clause?**
  2. **How haspmap identified a duplicate value?**
  3. **What is volatile?**
  4. **What is serialization and what is problem if we don’t give unique serialVersionUID?**
  5. **What are types of injections in spring? When you use what?**
  6. **byName, byType, constructor autowiring?**
  7. **How you inject a prototype bean into a singleton bean, what will be the behavior of such things?**
  8. **How you manage transaction management in spring?**
  9. **What is Optional class in java 8?**
  10. **Difference b/n abstract class and interface when comes to java8 as java8 interfaces can have default methods?**
  11. **What is the use of constructor in abstract class?**
  12. **What is intermediate and end operations in Streams? Can an end operation become an intermediary?**
  13. **Can we apply Streams directly on collections without converting them to Streams?**

**ANGULAR**

1. Class:
2. Constructor: only one can be created
3. Access modifier: public, private, protected
4. Public -> can be accessed outside the class and it is default, private -> can’t be used outside the class, protected -> can be used in inherited classes too.
5. Static variable is specific to the class but not to object.
6. Modularity: set of functionality
7. Import or export of a class or method for reusability
8. Type script course: <https://lex.infosysapps.com/viewer/lex_9436233116512678000/lex_auth_012712954616782848927>
9. Angular course: <https://lex.infosysapps.com/viewer/lex_20858515543254600000/lex_9393820929135647000>
10. https://lex.infosysapps.com/viewer/lex\_20858515543254600000/lex\_19957124143596520000
11. **npm install -g @angular/cli**
12. Angular was developed by google
13. Angular 1 to 7. From Angular 2, type script support introduced so we can design mobile apps.
14. Angular play ground

<https://lex.infosysapps.com/viewer/lex_auth_01257271100325068833>

<https://lex.infosysapps.com/viewer/lex_20858515543254600000/lex_30987236123954876000>

Features of Angular:

1. SPA
2. Modularity
3. Typescript support
4. Cross browser support
5. Web component support
6. Mobile app development
7. Easy to learn
8. Ide support
9. ECMA script 6
10. Performance

Ng new <MyApp> -> to create a new app.

Ng generate component <myComponent> -> to add a new component

ng serve - - open -> to start and open a project

**Template:**

1. Inline -> direct html code inside back ticks
2. External -> templateUrl -> include an external html file.

**Elements in template:**

1. Html, interpolation,{{}}, Template expressions – text inside interpolation -> {{expression}}, template statements (onclick events or etc.)
2. () -> one way databinding, html to TS

**Directives:**

1. Structural directives
2. Attribute directives

**ANGULAR**

1. Why type script instead of type script?

Ans: JS is interpreted not compiled, so we will get one by one and need fix one by one. Minimum obj orientation. JS is not data type specific, we will get errors at run time in case data type not matches.

TS is extension of ECMA script. Any valid JS is also a TS. Finally TS need to transpiled in JS as browser doesn’t understand it.

1. “let” keyword instead of “var” keyword.
2. <https://lex.infosysapps.com/viewer/lex_20858515543254600000>

**Creation of angular application:**

1. Install node js first
2. **npm install -g @angular/cli 🡪 install angular cli to provide basic structure of angular application**
3. ng new my-app 🡪 creates a new application
4. ng server - - open - - port 3000->
5. ng build
6. ng generate
7. Install Microsoft visual studio code from software center