Day 6 Lambda, Streams & Method Reference

Week1 Revisit

Lang Fundamentals,

OOPs Concepts

Coding Examples for Collections, Exception Handling, JDBC

Class, Objects.

In Java, everything is Object. (java.lang – default – Object)

Package, Class, Interface, Method (functions written inside a class)

Primitive data types (variables )

Derived Data types

Access & Non-Access Modifers.

Exception & Error

Types of Exception (Checked (compile-time) & UnChecked (runtime) Exception) – Throwable

Ways of handling exception (using throws keyword, using try/catch/finally block)

Collection API (java.util package) – Collection [I], List, Set, Map – Group of objects.

Utility Class – Collections (sort, filter)

JDBC – 5 Steps in interacting with any RDBMS.

Lambda – Is anonymous (nameless) function.

//Normal Method (Method signature + Method body)

Public void display() // method signature or method definition line

{ //Method body starts here

System.out.println(“I’m display method”);

} //method body ends here.

()=>{}

()=>System.out.println(“Im lambda”);

Java 8 Features (Lambda, Streams, Functional Interface, Method References)

1. Functional Interface (An interface with only one abstract method) (java.util.function – This package contains Functional Interfaces only) [predicate, supplier, function, bifunction, consumer]
2. Using Lambda ()->{} [Thin Arrow Syntax], Using Method Reference. (::)
3. Types of method reference [Static method reference, instance method reference, constructor method reference]

These concepts enables Functional Programming.

Multiple operations in a single line.

Public int add (int a, int b) {

Return (a+b);

}

(a,b)-> (a+b);

n->n\*n;

<https://www.geeksforgeeks.org/java/lambda-expressions-java-8/>

<https://www.w3schools.com/java/java_lambda.asp>

List numbers = new ArrayList() ;

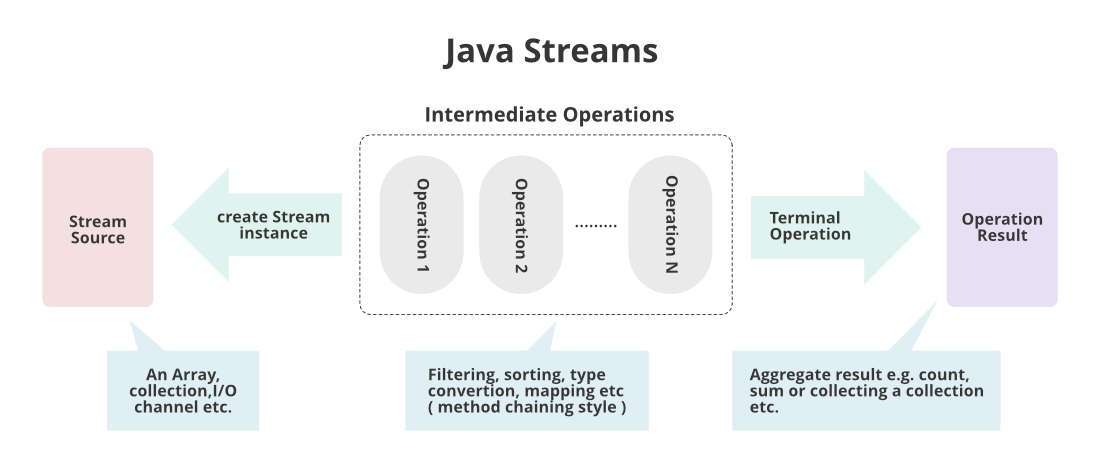
Functional Interface can have n number of static & default methods. (Comparator, Comparable, Runnable, Cloneable )

Streams = Flow of Data. They won’t change the source or original.

<https://www.geeksforgeeks.org/java/stream-in-java/>

<https://www.geeksforgeeks.org/java/java-8-stream-tutorial/>

1. Intermediate operation [sort, filter, search]
2. Terminal Operation [collect, display, count, min, max]



Create a Project for Trainers. (firstName, lastName, gender, email, mobile, dateOfBirth), Create a table in data base and Do the CRUD operation using JDBC. Use collections and streams where ever possible.

1. Create a java project in eclipse.
2. Add packages like model (Trainer) , dao(repository) [Data Access Object – findAll(), findById(),findByEmail(),findByFirstName(),findByMobile()],
3. Create a Starter class to perform CRUD operations.

CREATE TABLE `trainer` (

`id` int NOT NULL AUTO\_INCREMENT,

`first\_name` varchar(45) NOT NULL,

`last\_name` varchar(45) DEFAULT NULL,

`gender` varchar(1) DEFAULT NULL,

`email` varchar(250) DEFAULT NULL,

`mobile` bigint DEFAULT NULL,

`date\_of\_birth` date DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `email\_UNIQUE` (`email`),

UNIQUE KEY `mobile\_UNIQUE` (`mobile`)

)

insert into trainer (id, first\_name, last\_name, gender, email, mobile, date\_of\_birth) values (1, 'Fernando', 'Yurinov', 'M', 'fyurinov0@chicagotribune.com', '8731493924', '1996-05-23');

insert into trainer (id, first\_name, last\_name, gender, email, mobile, date\_of\_birth) values (2, 'Cyndia', 'Daysh', 'F', 'cdaysh1@smh.com.au', '7291674119', '1971-12-10');

insert into trainer (id, first\_name, last\_name, gender, email, mobile, date\_of\_birth) values (3, 'Rachel', 'Christopher', 'F', 'rchristopher2@yolasite.com', '6802146006', '1981-02-10');

insert into trainer (id, first\_name, last\_name, gender, email, mobile, date\_of\_birth) values (4, 'Ardisj', 'Janny', 'F', 'ajanny3@unc.edu', '6847919248', '1990-08-05');

insert into trainer (id, first\_name, last\_name, gender, email, mobile, date\_of\_birth) values (5, 'Ertha', 'Laflin', 'F', 'elaflin4@stanford.edu', '8796055928', '2010-02-14');

insert into trainer (id, first\_name, last\_name, gender, email, mobile, date\_of\_birth) values (6, 'Ki', 'Ragate', 'F', 'kragate5@ucla.edu', '8276535536', '2003-12-02');

insert into trainer (id, first\_name, last\_name, gender, email, mobile, date\_of\_birth) values (7, 'Cort', 'McGillicuddy', 'M', 'cmcgillicuddy6@cloudflare.com', '9389218497', '1976-10-21');

insert into trainer (id, first\_name, last\_name, gender, email, mobile, date\_of\_birth) values (8, 'Somerset', 'Ferran', 'M', 'sferran7@slideshare.net', '8497810785', '1973-05-09');

insert into trainer (id, first\_name, last\_name, gender, email, mobile, date\_of\_birth) values (9, 'Charlie', 'Sodeau', 'M', 'csodeau8@cbsnews.com', '9694024004', '1977-06-11');

insert into trainer (id, first\_name, last\_name, gender, email, mobile, date\_of\_birth) values (10, 'Taddeusz', 'June', 'M', 'tjune9@xing.com', '8291642124', '1976-11-03');