Day 5 Revisit

- Created Bitbucket Repo for Classroom trainings, Conceptual Problem statement, assessment
- Classroom Training – Mono Repo concept (Only one branch with one folder for each participants – No branching strategy.
- Conceptual Problem statement – Used branching strategy (Main, dev branch)
- Learned Team code management.
- Creating Pull request, Adding review comments, accepting or rejecting PR.
- Merging to main/master branch.
- JDBC concepts
- Important classes and interfaces of java.sql package
- Important interfaces are Driver, Connection, Statement/PreparedStatement/CallableStatement/ResultSet
- Important classes DriverManager, Date
- When to use statement and/or preparedstatement
- Types of insert technique in DB table (Partial Insert, Full insert)
- AI – auto Increment in MySQL (It will create a sequence using that it will auto generate the ID column)
- Soft Delete (using a boolean column to update the status of the record/row – isActive) & Hard Delete (Deleting particular row permanently)
- createdAt, lastUpdated (Audit Columns) – Timestamps
- Read Operation (Read All[always returns many records], ReadById [always return single record], Read with Limit – This is suitable and helps to improve performance)
- Modify Operation [Insert, Update, Delete]
- Simple CRUD operation using MySQL DB table.
- Completed Assessment1 and closed all sub task of sprint1.

Agenda

Java 8 Features

- Lambda
- Streams
- Functional Interface
- Optional class

Types of Comments in Java (Single line, Multi line, Documentation comment)

```
//  -- Single Line comment

/*  Multi line

 *comment

*/


/** Document
```

*Comment

*/

Annotations –

What – Annotations are meta data.

Metadata – Data about the data.

In Java, Annotation always start with @ symbol.

Types of Annotations –

1) Built-in Annotations (Pre-defined Annotations) Annotations given by java lang creators
2) User-defined/ Custom Annotations

Based on Usage

1) Class Level Annotation -- @Entity, @Bean
2) Method Level Annotation -- @Override, @Deprecated
3) Property/Field Level Annotation -- @Id, @NotNull
4) Parameter level Annotation -- @RequestBody, @PathVariable

Creating custom annotation in Java -- https://www.geeksforgeeks.org/how-to-create-your-own-annotations-in-java/

Using JavaDoc tool to generate API documentation

https://www.geeksforgeeks.org/what-is-javadoc-tool-and-how-to-use-it/

POJO – Plain Old Java Objects (A class which is not extending any class nor implementing any interfaces)

Types of Classes

1) Abstract Class (Incomplete/Non-concrete class) – Its incomplete class
2) Inner Static Class
3) Normal Glass/Simple Class/ Concrete class/ POJO (A class with properties and methods)
4) Built-in/Pre-defined/Creator Developed Class
5) Custom/User-defined Classes
6) Wrapper Classes (Byte, Integer, Float, Character, Double, Boolean, Long, Short)
7) Bean/Entity Class (A class with properties, constructor an getter, setter methods)
8) Servlet Class – It runs on the server not locally
9) Starter Class – A Class with a main method

Types of Interfaces

1) Collection Interface – Used to deal with group of objects
2) Marker Interface (Empty Interface) – Serializable
3) Functional Interface (Added in Java 8)

Serialization – Process of storing the state of an object in to a flat file system.

Serialization Example in Java

https://www.geeksforgeeks.org/serialization-in-java/

Functional Interface – An interface with only one abstract method

All the interfaces created inside the java.util.function package is all functional interface

Consumer, Predicate, Supplier, Function

Abstract method/ Incomplete/ Non-Concrete Method

Void add(); // method signature line or Method declaration line.

Java 8 Features

- Functional Interface -- An Interface with only one abstract method (All interfaces inside java.util.function)
- Functional Interface can have more than one default and static methods in it.
- Lambda – Inline functions (Functions written in a single line) [Simplified /easy way of writing code]
- Method Reference
- Streams [Efficient way of handling group of data]
- @FunctionalInterface -- annotation is marked on the functional interfaces

Another important use of annotation –

Instead of providing xml based configuration, we can use annotation for that.

Lambda – It's java implementation of arrow functions.

//simple function

// access modifier return_type function_name()

{ //function body starts here

} //function body ends here

Public void add () {

System.out.println("Add function");

}

Lambda – is also called as anonymous (nameless) function

Public void () {

}

Lambda syntax

()->{}

(a,b)->{a+b}; -- lambda with two parameters

(a)->{System.out.println("a:"+a)}; - Lambda with only one parameter

a->System.out.println("a:"+a);

https://www.javatpoint.com/java-lambda-expressions

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

Method Reference -- It's a java8 feature used to simplify the lambda.

Streams - In Java, flow of data.

Out – represent output stream

In – input stream

Err – error stream

Sort, Filter, Search

Functional Programming – Its an approach, doing many task in a single line.

- POP (Procedure Oriented Programming)
- OOP (Object Oriented Programming)
- FP (Functional Programming)

Sync & Async Programming

Synchronous – Step by step.

1) Printing some text in console (5 ms)
2) Opening a txt file and reading content of it (200 ms)
3) Inserting 3 record in to a database table (400 ms)
4) Getting an name input from user and displaying it in the standard console (200 ms)

Async (Parallel Programming)

Functional Programming

Boxing – Is a process of converting primitive to its corresponding Object representation.

AutoBoxing & Auto Unboxing

Optional Class –

https://www.javatpoint.com/java-8-optional

https://www.javaguides.net/2018/07/java-8-optional-class.html

Week 1 Conceptual Problem statement Expectations

Day 1: Complete all the UML diagrams for the TMS (Training Management System)

Day 2: Create Team wise Repository in Bitbucket, Writing the Starter Class of the TMS (CLI version)

Day 3: Create Collections for various Entities used in TMS (Trainers, TOC, Company, SME)

Day 4: Design the Collection which we are going to use for different entity in TMS.

Day 5: Create MySQL database Table for all Entities of TMS, insert sample data. Also write CRUD operation on each entity.

Week 2 Conceptual Problem statement Expectations

Day 6: Update the CRUD operation code to use Streams, Lambda, Functional Interface and Method Reference.