**Day 10: 29-10-2025:**

**Java 8:**

**Method reference :** A method reference in Java 8 is a short notation of lambda expression that call as existing method directly rather linking explicitly.

Using this concept we can refer a method by its name instead of writing full lambda expression. With help of method reference we can make code core concise and readable.

In other word a method reference is a compact way of passing behaviour (method execution) as an argument to a method or functions.

Syntax to use method reference

ClassName::methodName; method must be static

ObjectName::methodName: method can be non static

List<String> ll = Arrays.asList(“A”,”B”,”C”);

To retrieve data from collection using lambda

ll.stream().forEach(v->System.out.println(v));

To retrieve data from collection using method reference

ll.stream().forEach(System.out::println);

Optional class : The Optional class in java 8 is a container object use to represent the present or absent a specific value in method or any class objects. It helps to avoid NullPointerException by providing few pre defined method to handle safe way null value.

Before Java 8 we were using Date class, Calendar class, SimpleDateFormat class etc.

There is two Date class is there

1 part of util package : to find system date and time

2. part of sql package : to store date information in database.

SimpleDateFormat class is part of text package.

All these date classes not a thread safe and they are mutable.

From Java 8 onward all date and time related classes kept in time package.

LocalDate

LocalTime

LocalDateTime

ZoneDateTime

Java 9 Features

1. From Java 9 onward interface can contains private methods.

**try with resource**

**Java 7 features the classes which internally implements Closeable interface automatically call close() method to close the resource. Now a day finally block not required.**

**Resource close in reverse order.**

try(){

}catch(Exception e) {

}

Java new features

Sealed classes

Sealed class and interface allow you to control which other classes or interfaces can extends or implements them.

Normally in java we can extends or implements any class or interfaces unless it is marked as final.

In java 15 onward sealed keyword introduce

If we use sealed keyword with class or interface it mark as class or interface are sealed.

We can use permits keyword. this keyword help use whichever class or interface can extends or implements sealed class/interface.

**final** : can’t inherits

**sealed** : can’t inherits

**sealed with permits** : can inherits with permitted class or interfaces.

**non-sealed**: by default any class or interface can inherits.

Records

Enhanced switch statemen using lambda style

Text block