WFS Notes

1. Java 7 and Java 8
2. Spring
3. Spring Microservices
4. Angular Framework

Java has done major changes in Java 5, Java 7 and Java 8

Java 5 features

1. for each loop
2. Enums
3. Generics
4. Var-Args
5. Static Imports
6. Annotations

Enums: Fixed set of constants, so that at compile time itself you can verify the valid values, ex:

Gender must be male or female

Account must be Savings, Current, Recurring, Fixed Deposit

Departments must be Accounts, Sales, Finance,..

Gender.java



Employee.java



EmployeeDemo.java



Varying Arguments

It can accept 0 or more arguments

int sum(int... x)



Output:



Java 7 Features

1. Diamond Operator
2. Multi-Catch block
3. Resource Management with try-with resource syntax
4. Strings in Switch case
5. Underscores in numbers

Diamond Operator <>:

Before Java 7:

List<Integer> list = new ArrayList<Integer>();

From Java 7:

List<Integer> list = new ArrayList<>();

Multi-Catch Statements:

In one catch you can have one or more exceptions to handle

catch(ArithmeticException | NumberFormatException e) { }

Resource Management with try-with resource statement

This features automatically closes the resources like files, buffers, databases etc.

It automatically flushes the streams while performing write operations

Syntax:

try (resource instances; resource instances) {

}

Example:

try (fw = new FileWriter(“abc.txt”)) { }

This makes the Java to automatically perform close after performing the operations inside the try



Above code automatically closes the streams

Strings in Switch & Underscores in Numbers



Java 8 Features:

1. New Date & Time API’s
2. Static & Default methods inside interface
3. Functional Interface
4. Lambda Expressions
5. Stream API’s

New Date & Time API’s

Java introduced 3 main classes to work on Date, Time and DateTime

1. LocalDate
2. LocalTime
3. LocalDateTime

All these 3 classes have some static methods to create date & time instances

now(): to create current date or time instance

of(): to create specific date or time instance

parse(): to take a string and convert to date & time instance

LocalDate Demo



Output:



LocalTime & LocalDateTime also works the same way



Output:

Time: 11:45:17.188

Date & Time: 2020-10-05T11:45:17.189

Format Date & Time: 05/10/2020 T 11:45:17

Changes in the interface

You can have methods with body in the interface

1. default methods: it will have some default implementations but not necessary to override
2. static methods: you can call directly from the interface name

Lambda Expression:

It is a simplified form of implementing anonymous class, it has to be applied only on the interfaces having single abstract method (Functional Interface)



Lambda expressions are better when the statement is single line expression, but it also allows you to have multiline statements

() -> 10 // a method that returns an int

() -> “Hello” // a method that returns a String

(x, y) -> x + y // a method that returns addition of x & y

() -> { System.out.println(“...”); return 10; }

() -> System.out.println() // a method returns void

Note: return statement is required when a method has a return type if you write more than one line in lambda expression

Functional Interface:

It allows you to pass code directly to a method, it is an interface with only one abstract methods.



Exercise:



1st requirement: get all the employees added into the collection

2nd requirement: get only the top 3 employees sorted based on id, salary, dob

Java Streams:

Streams are collection of data that you want to operate while processing the collection, Java Streams processes the collection of data in a declarative way like SQL.

SQL statements are declarative statements they are easier to write, to select items, to filter items, to sum items

select \* from employee;

select \* from employee order by name;

select \* from employee order by name desc;

You can simplify sorting, filtering, iterating in the streams