Interface Default and Static Methods

Static Method

DEFINITION/WHAT

Starting with Java 8, interfaces can have static and default methods that, despite being declared in an interface, have a defined behavior.

NEED/WHY

- Need 1: The static method is available only through and inside of an interface.
- Need 2 : It can't be overridden by an implementing class

IMPLEMENTATION/HOW IT INTERNALLY WORKS

To call it outside the interface the standard approach for static method call can be used

REAL TIME EXAMPLE

To add new functionality to the existing interface without forcing all implementing classes to create an implementation of the new methods.

Default Method

DEFINITION/WHAT

Default methods are declared using the new default keyword. These are accessible through the instance of the implementing class and can be overridden.

NEED/WHY

- Need 1: Accessible through the instance of the implementing class

IMPLEMENTATION/HOW IT INTERNALLY WORKS

Despite being declared in an interface, have a defined behavior.

REAL TIME EXAMPLE

Possible to add new functionality to the existing interface without forcing all implementing classes

forEach() method in Iterable interface

forEach() method

Definition/What

Java 8 has introduced forEach method in *java.lang.Iterable* interface so that while writing code we focus on business logic

Need/Why

- The static method is available only through and inside of an interface.
- Need 1: To avoid ConcurrentModificationException
- Need 2 : While writing code only focus on business logic

Implementation/How it internally works

To call it outside the interface the standard approach for static method call can be used

The forEach method takes java.util.function.Consumer object as an argument, so it helps in having our business logic at a separate location that we can reuse.

Real Time Example

To traverse through a Collection

Functional Interfaces

Functional Interface

Definition/What

An interface with exactly one abstract method becomes a Functional Interface

Need/Why

- The static method is available only through and inside of an interface.
- Need 1: The possibility to use lambda expressions to instantiate them

Implementation/How it internally works

Since functional interfaces have only one method, lambda expressions can easily provide the method implementation. We just need to provide method arguments and business logic

Real Time Example

java.lang.Runnable with a single abstract method run() is a great example of a functional interface.

Lambda Expressions

Lambda Expressions

DEFINITION/WHAT

Lambda expressions enable you to treat functionality as method argument, or code as data.

NEED/WHY

- Need-1: A function that can be created without belonging to any class.
- Need-2: lambda expression can be passed around as if it was an object and executed on demand.

IMPLEMENTATION/HOW IT INTERNALLY WORKS

Lambda expressions basically express instances of functional Interface.

Lambda expressions implement the only abstract function and therefore implement functional interfaces

REAL TIME EXAMPLE

A function that can be created without belonging to any class

I/O Stream

Definition/What

A stream is a flow of data from a source or to a sink. Types of streams are files, memory, and pipes between threads or processes.

Need/Why

- Need-1: To initiates the flow of data, also called an input stream
- Need-2: To Terminates the flow of data, also called an output stream

Implementation/How it internally works

Java technology supports two types of streams, character and byte. Input and output of character data is handled by readers and writers. Input and output of byte data is handled by input streams and output streams: Normally, the term stream refers to a byte stream. The terms reader and writer refer to character streams.

Real Time Example

Performs a copy file operation using a built-in buffer.

Performs a copy file operation using a manual buffer

Calendar

DEFINITION/WHAT

The **Calendar** class is an abstract class that provides methods for converting between a specific instant in time and a set of calender fields**.**

NEED/WHY

Need-1: Implementing a concrete calendar system outside the package.

Need-2: Manipulating the calendar fields, such as getting the date of the next week

IMPLEMENTATION/HOW IT INTERNALLY WORKS

Calendar object produces all the calendar field values needed to implement the date-time formatting for a particular language and calendar style.

REAL TIME EXAMPLE

An instant in time can be represented by a millisecond value that is an offset from the Epoch, January 1, 1970 00:00:00.000 GMT (Gregorian)

Locale

DEFINITION/WHAT

A Locale object represents a specific geographical, political, or cultural region. An operation that requires a Locale to perform its task is called locale-sensitive **and uses the Locale to tailor information for the user.**

NEED/WHY

Need-1: To represent a specific geographical, political, or cultural region.

IMPLEMENTATION/HOW IT INTERNALLY WORKS

The Locale class implements IETF BCP 47, with support for the LDML (UTS#35, "Unicode Locale Data Markup Language") BCP 47-compatible extensions for locale data exchange.

REAL TIME EXAMPLE

An operation that requires a **Locale** to perform its task is called locale-sensitive and uses the **Locale** to tailor information for the user