Dasari Vamsi

# FUNCTIONAL INTERFACES IN JAVA — EXPLAINED SIMPLY

The foundation of Lambdas & Streams

dasarivamsi.netlify.app

Dasari Vamsi

# What is a Functional Interface?

➡️ An interface with exactly ONE abstract method.

➡️ Used to represent a single functionality.

➡️ Can have default & static methods (but still only ONE abstract method).

Dasari Vamsi

# Why are they important?

- ✅ Enable Lambda Expressions
- ✅ Power Java Streams API
- ✅ Reduce boilerplate code
- ✅ Promote clean & readable functional style

dasarivamsi.netlify.app

Dasari Vamsi

# Examples (Built-in)

Java provides many:

- Runnable → run()
- Callable → call()
- Comparator → compare()
- Consumer → accept()
- Function → apply()
- Supplier → get()
- Predicate → test()

Dasari Vamsi

# Example Code

```
@FunctionalInterface
interface Greeting {
    void sayHello(String name);
}

public class Main {
    public static void main(String[] args) {
        Greeting g = (name) ->
System.out.println("Hello " + name);
        g.sayHello("Java");
    }
}
```

Output 👉 Hello Java

Dasari Vamsi

# Key Notes

⚡ Marked with @FunctionalInterface (optional, but good practice).

⚡ If more than 1 abstract method → compile-time error.

⚡ Widely used in Streams, Lambdas, Event Handling.

Dasari Vamsi

# Wrap-Up

🚀 Functional Interfaces are the backbone of Java's functional programming.

Next time you use a lambda or a stream operation, remember:

👉 A functional interface is working behind the scenes!

8/8

Dasari Vamsi

# HELPED YOU?

👉 Drop a comment or DM me!

dasarivamsi.netlify.app