JAVA 8

Why do we use
@FunctionalInterface?

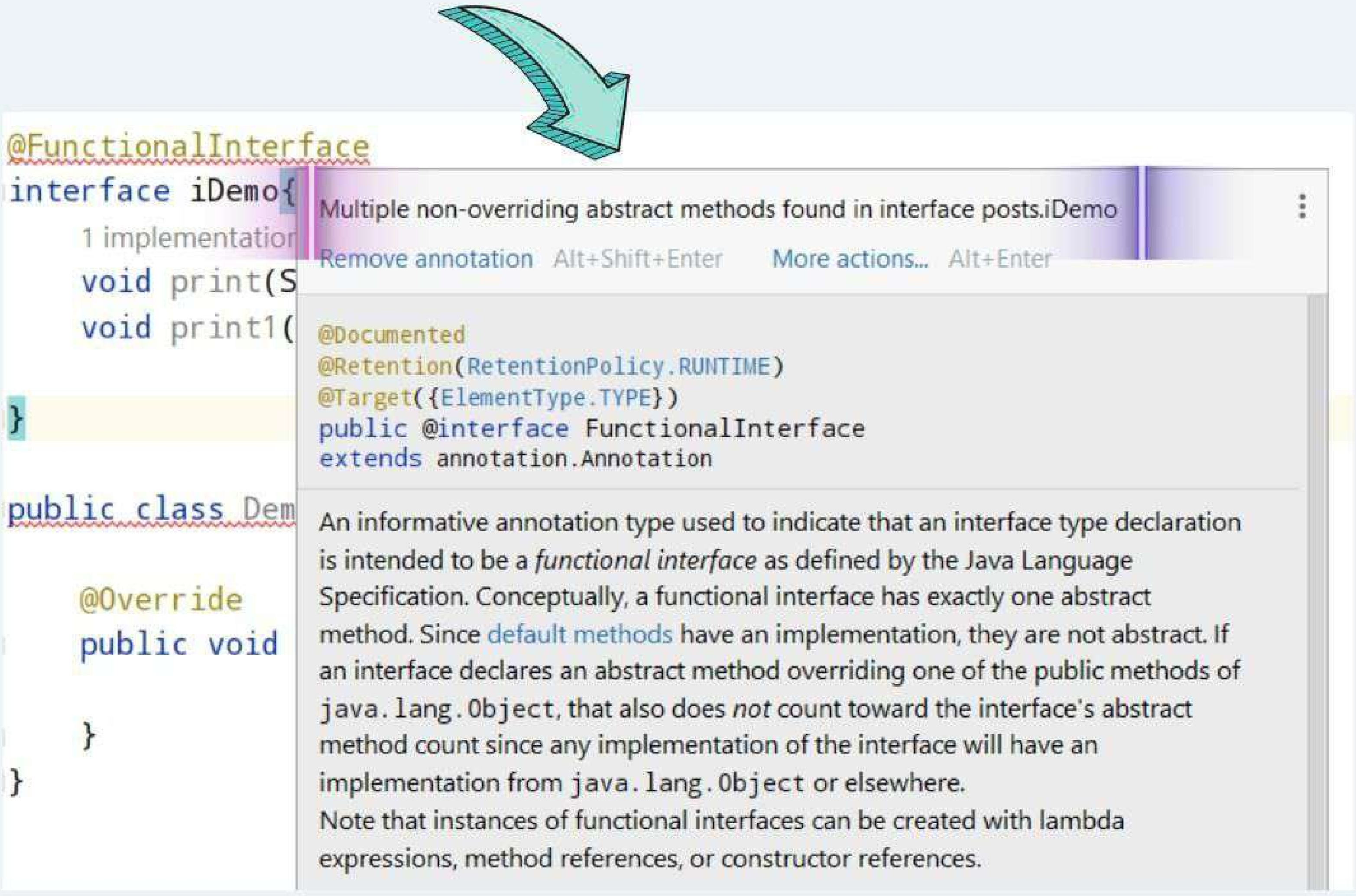
@techy.tacos

This annotation is used to indicate that the interface is intended to be a functional interface. The compiler will return a meaningful error if you define an interface using the @FunctionalInterface annotation and it isn't a functional interface. It is completely optional and there is no strict rule to annotate a functional interface with this annotation but consider it as a good practice.

Examples :

- If we are trying to declare more than 1 abstract method using @FunctionalInterface then straightforward we get the compile time error .

@techy.tacos



```
@FunctionalInterface
interface iDemo{
    1 implementation
    void print(S
    void print1(

}

public class Dem

    @Override
    public void

    }
}
```

Multiple non-overriding abstract methods found in interface posts.iDemo

```
@Documented
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.TYPE})
public @interface FunctionalInterface
extends annotation.Annotation
```

An informative annotation type used to indicate that an interface type declaration is intended to be a *functional interface* as defined by the Java Language Specification. Conceptually, a functional interface has exactly one abstract method. Since default methods have an implementation, they are not abstract. If an interface declares an abstract method overriding one of the public methods of `java.lang.Object`, that also does *not* count toward the interface's abstract method count since any implementation of the interface will have an implementation from `java.lang.Object` or elsewhere.
Note that instances of functional interfaces can be created with lambda expressions, method references, or constructor references.