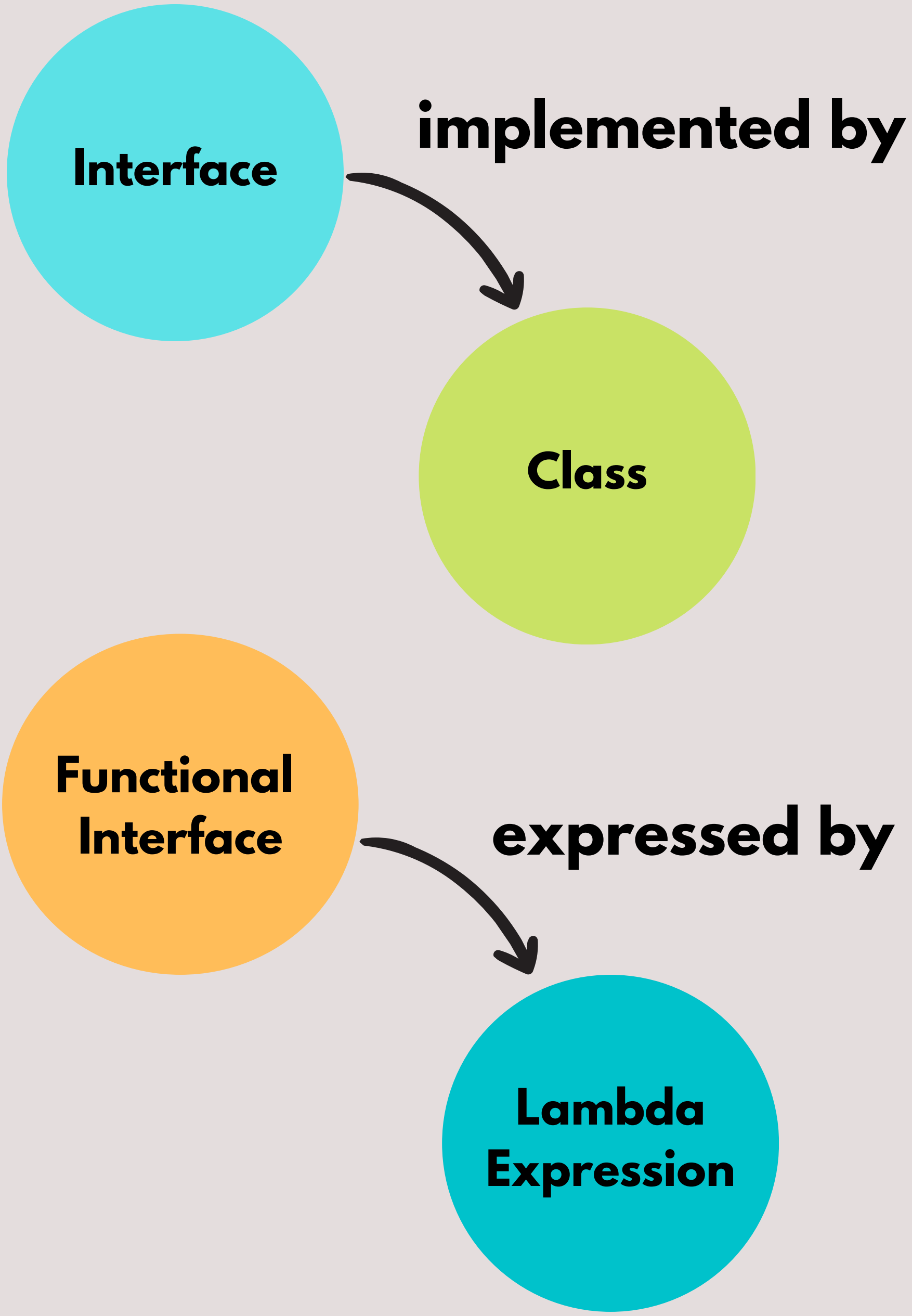
Functional Interface in Java 8



nivedhasankar

Next →

Difference between Interface and Functional Interface

Interface

1. An Interface implemented by class.
2. An Interface can have more than one abstract classes.
3. Implementation is done through typical object oriented programming.

Functional Interface

1. An Functional Interface expressed using Lambda Expression .
2. An Functional Interface can have only one abstract class.
3. Implementation is done through functional programming.

nivedhasankar

Next ⟶

Abstract Class

1. An abstract class is declared using the abstract keyword.
2. An abstract class can have abstract and non-abstract methods.
3. An abstract class can not be instantiated (we cannot create objects of abstract classes).

```
abstract class Grocery{
public abstract void Veggies();
//abstract method
public void Fruits{
System.out.print("apple");
//non- abstract method
}}
```

nivedhasankar

Next ⟶

Abstract Method

1. An abstract method is declared using the abstract keyword.

2. An abstract method can only be used in an abstract class, and it does not have a body.

```
abstract class Grocery{
public abstract void Veggies();
//abstract method
}
class  groceryShop extends Grocery{
public abstract void Veggies(){
System.out.print("Onion");
}}
class main(){
public static void main(String[] args){
groceryShop list = new groceryShop();
list.Veggies();
}}
```

nivedhasankar

Next ⟶

Functional Interface

1. **An interface that contains only one abstract method is called as functional interface.**

```
interface MyFunctionalInterface{
public void execute();
}
```

Whenever we define a Method in a interface by default that method is abstract. So, We don't mention abstract keyword in a method.

nivedhasankar

Next →

Functional Interface

2. **Functional Interface can have any number of default, static method but can contain only one abstract method.**

```
interface MyFunctionalInterface {
    public void execute();
        //abstract method
    public static  void  print1() {
      //static method
      System.out.print("static");
    }
     public default void print2() {
      //default method
        System.out.println("default");
    }
}
```

nivedhasankar

Next ⟶

Functional Interface

3. We can use Java 8 provides @FunctionalInterface annotation to mark an interface as a functional interface.

```
@FunctionalInterface
interface MyFunctionalInterface {
    public void execute();
        //abstract method
}
```

@FunctionalInterface annotation helps the compiler to validate whether the give interface is a valid functional interface are or not.

nivedhasankar

Next →

Lambda Expression

1. Provide the implementation of an interface that has a functional Interface.

2. It saves a lot of code.

3. We don't need to define the method again for providing the implementation.

```
(argument-list) -> {body}
```

Without Parameters:

```
() -> {body}
```

With Parameters:

```
(a1,a2) -> {body}
```

nivedhasankar

Next ⟶

Components of Lambda Expression

Java lambda expression consists of three components.

1) **Argument list:**
   It can be with arguments and without arguments.

2) **Arrow-token:**
   It is used to link arguments-list and body of expression.

3) **Body:**
   It contains expressions and statements for lambda expression.

nivedhasankar

Next →

How to Implement Functional Interface Using Lambda Expression

```
@FunctionalInterface
interface  A {
  void show();
}
 public class Main{
    public static void main(String[]
   args){
    A obj = () ->{
   System.out.println ("show");
  }
   obj.show();
   }
}
```

nivedhasankar

End