Agenda:

1) Functional Interfaces
2) lambdas
3) Streams.

# FUNCTIONAL INTERFACE.

↳ Abstract Methods.

⇒ Method without body.

⇒ Single Abstract Method. (SAM)

```
interface A {
    void fun();
}
```

⇒ FI.

```
interface A {
    void fun1();
    void fun2();
}
```

✗

```
interface A {
    void fun1();
    default void fun2() {
        ...
    }
}
```

⇒ @FunctionalInterface ⇒ Not mandatory
↳ Good to Have.

⇒ Examples.

- Runnable
- Callable
- Comparable
- Comparator

...

⇒ Some important functional Interfaces.

1) Consumer
2) BiConsumer
3) Predicate
4) BiPredicate
5) Function
6) BiFunction
7) BinaryOperator

## Consumer & BiConsumer

Consumer<T> → Any data type

↓

void accept(T t);

BiConsumer<T, V>

⇓

void accept(T t, V v);

## Predicate & BiPredicate

Predicate<T>

↓

bool test(T t)

BiPredicate<T, V>

↓

bool test(T t, V v);

# Function & Bifunction

Function<T, V>

↓

V apply(T t);

Function<T, U, V>

↓

V apply(T t, U u);

BinaryOperator<T>

↓

T apply(T $t_1$, T $t_2$);

# STREAMS.

List<Integer> nums = [- - - - - -]

↓

Stream

⇒ A wrapper on data sources for doing operation on items present inside the data source.