Streams (Java 8) & Lambdas (Java 8)

Lambda Expression / Lambda functions

↳ concise way to represent one method interface using an expression.

↳ Anonymous function

↳ code → more compact & readable.

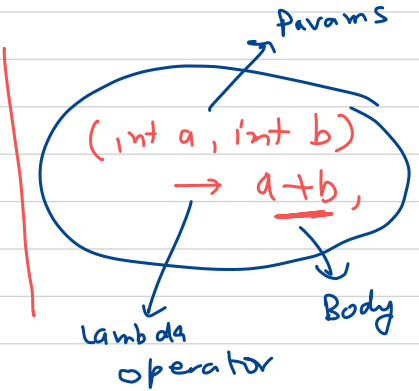
```
int Sum(int a, int b){
    ret a+b; }
```

Syntax

(parameters) → expression

or

(parameter) → { statements; }



Often used with Functional Interface

↓

an interface that contains a single abstract method.

Functional Interface in Java.

- Comparator
  - ↳ compare()
- Runnable
  - ↳ run()
- Callable
  - ↳ call()

More –

java.util.function

- → Function
- → Predicate
- → Consumer
- → Supplier

Java 8 Streams

↳ Stream in Java is a wrapper around a Data Source.

↳ Chain of operations on Collections | Arrays in functional & declarative way

→ Procedural Prog ✓
→ OOPS ✓
→ Functional | Declarative → Declare what to do, not how to do.
→ Reactive

Intermediate ops

↳ filter (no → no % 2 == 0)

↳ map (x → 2x)

filtering (x → x % 10 == 0)

↓

Map (x → 2x)

5, 10, 20, 22, 25, 80

↳ [20, 40, 160]

multiple of 10 and double them.

Common Functional Interface

① Predicate<T>

boolean test(T t)

- used for filtering / matching
- input is one obj, output is boolean.

- filter (s → s.length() > 5),
- filter (n → n % 7 == 0),

② Consumer

void accept(T t)

Example:

s → System.out.println(s);

③ Supplier

T get()

④ Function<T, R>

input type

result type

R apply(T t)

String

R → Integer

Function<String, Integer> f = s → s.length()

f.apply(=);