Functional Programming

Functions and Lambda Expressions

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Table of Contents

Have a Question?

Software University

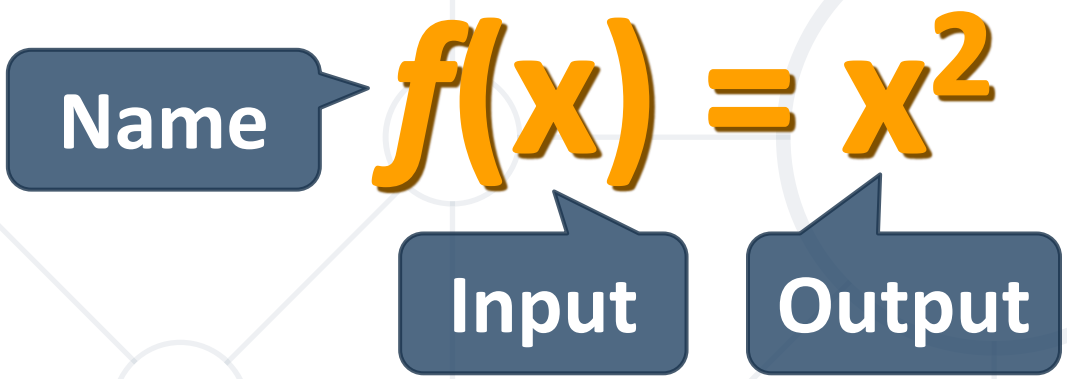
sli.do

#java-advanced

Lambda Expressions

What is a Function?

- Mathematic function



A **function** is a special relationship where **each** input has a **single** output

x	f(x)
3	9
1	1
0	0
4	16
-4	16

Lambda Expressions (1)

- Lambda expression - **unnamed function**
  - Has parameters and a body

Lambda Syntax

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

- Use the lambda operator **->** Read as "**goes to**"

Lambda Expressions (2)

- **Implicit** lambda expression

```
(msg) -> { System.out.println(msg); }
```

Parameters can be enclosed in **parentheses ()**

The body can be enclosed in **braces {}**

- **Explicit** lambda expression

```
String msg -> System.out.println(msg);
```

Declares parameters' type

Lambda Expressions (3)

- Can have a different number of parameters:
  - **Zero** parameters

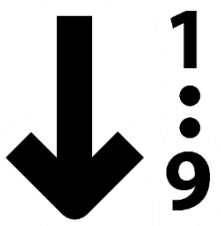
```
() -> { System.out.println("Hello!"); }
() -> { System.out.println("How are you?"); }
```

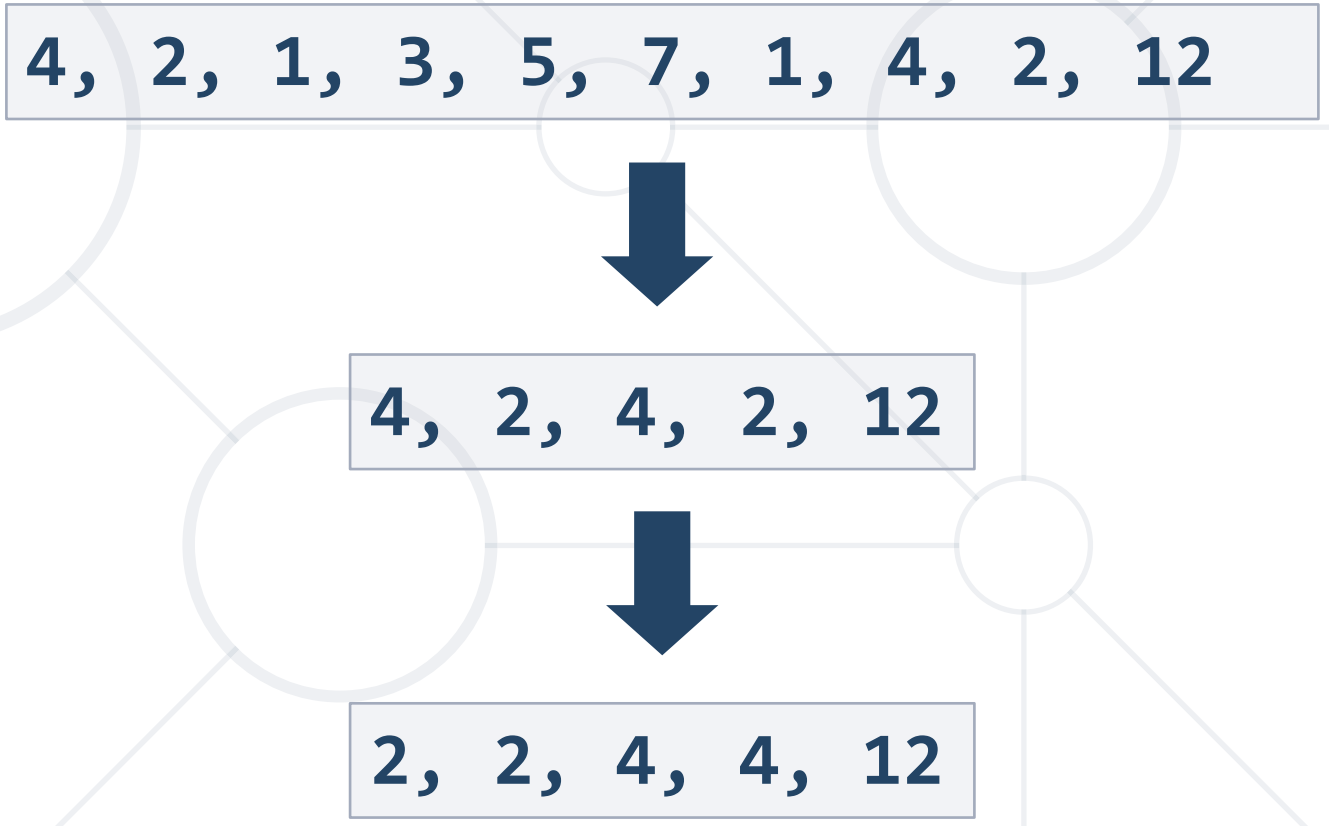
  - **More** parameters

```
(int x, int y) -> { return x + y; }
(int x, int y, int z) -> { return (y - x) * z; }
```

Problem: Sort Even Numbers

- **Read** Integers from the console
- **Print** the even numbers
- **Sort** the even numbers
- **Print** the sorted numbers





Check your solution here: https://judge.softuni.bg/Contests/1513/Functional-Programming-Lab

Solution: Sort Even Numbers

```
//TODO: Read numbers and parse them to List Of Integers

numbers.removeIf(n -> n % 2 != 0);

// TODO: Print the even numbers

numbers.sort((a, b) -> a.compareTo(b));

//TODO: Print the sorted numbers
```

Check your solution here: https://judge.softuni.bg/Contests/1513/Functional-Programming-Lab

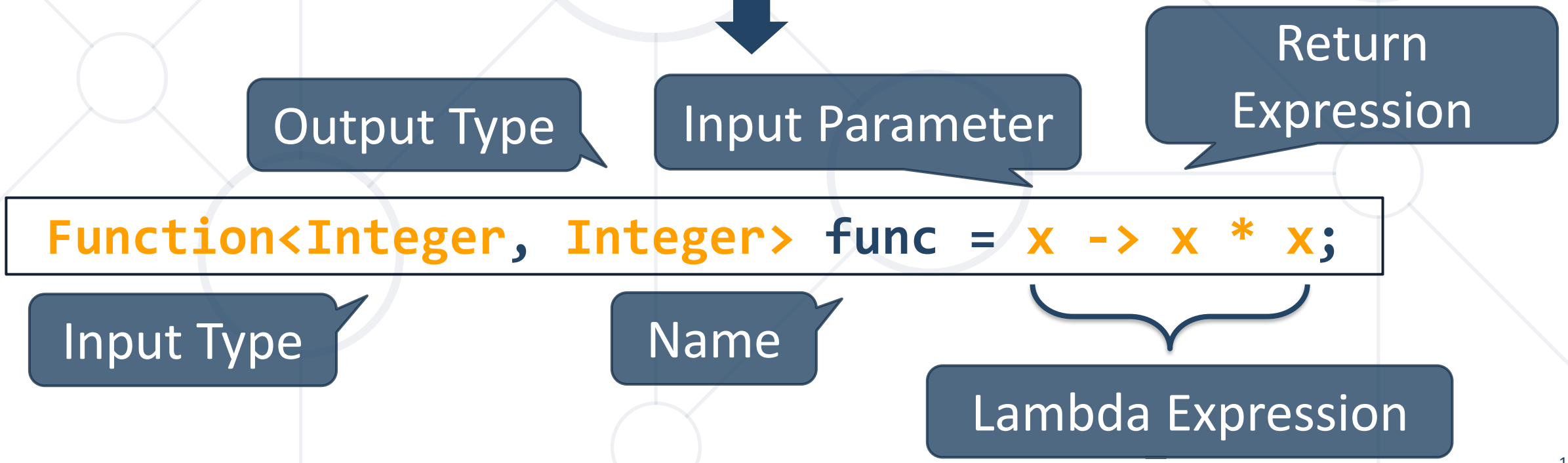
Functions

Mathematical and Java Functions

Java Functions

- In Java, we can create functions analogical to mathematical functions

$$f(x) = x^2$$



Function<T, R>

- In Java **Function<T,R>** is an interface that accepts a parameter of type **T** and returns a variable of type **R**

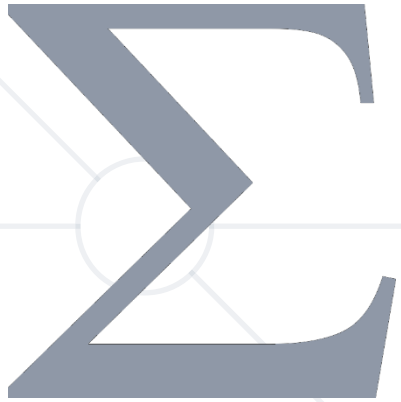
```
int increment(int number) {
    return number + 1;
}
```

- We use function with **.apply()**

```
Function<Integer, Integer> increment =
                            number -> number + 1;
int a = increment.apply(5);
int b = increment.apply(a);
```

Problem: Sum Numbers

- **Read** numbers from the console
- **Print** their count
- **Print** their sum
- Use a **Function**



```
4, 2, 1, 3, 5, 7, 1, 4, 2, 12
```

```
Count = 10
Sum = 41
```

Check your solution here: https://judge.softuni.bg/Contests/1513/Functional-Programming-Lab

Solution: Sum Numbers

```
// TODO: Read input
if (input.length < 2) {
    System.out.println("Count = " + input.length);
    System.out.println("Sum = " + input[0]);
} else {
    Function<String, Integer> parser = x -> Integer.parseInt(x);
    int sum = 0;
    for (String s : input) sum += parser.apply(s);
    // TODO: Print output
}
```

Check your solution here: https://judge.softuni.bg/Contests/1513/Functional-Programming-Lab

Other Function Types

Special Functions

Consumer<T>

- In Java **Consumer<T>** is a void interface:

```
void print(String message) {
    System.out.println(message);
}
```

- We use a Consumer with **.accept()**:

```
Consumer<String> print =
        message -> System.out.print(message);
print.accept("Peter");
```

Supplier<T>

- In Java `Supplier<T>` takes no parameters:

```
int genRandomInt() {
  Random rnd = new Random();
  return rnd.nextInt(51);
}
```

- We use a Supplier with `.get()`:

```
Supplier<Integer> genRandomInt =
                    () -> new Random().nextInt(51);
int rnd = genRandomInt.get();
```

Predicate<T>

- In Java `Predicate<T>` evaluates a condition:

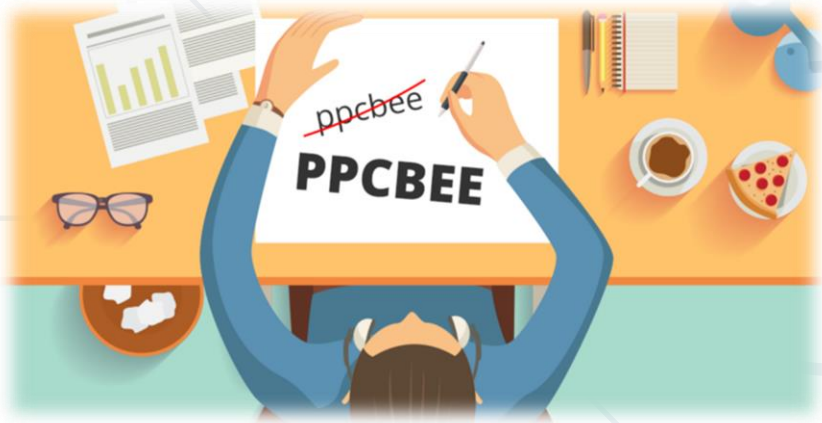
```
boolean isEven(int number) {
    return number % 2 == 0;
}
```

- We use the Predicate with `.test()`:

```
Predicate<Integer> isEven =
                    number -> number % 2 == 0;
System.out.println(isEven.test(6)); // true
```

Problem: Count Uppercase Words

- Read text from the console
- Find the words starting with an Uppercase letter
- Print the count and the words
- Use a **Predicate**




```
The following example shows how to use Predicate
```

```
2
The
Predicate
```

Check your solution here: https://judge.softuni.bg/Contests/1513/Functional-Programming-Lab

Solution: Count Uppercase Words

```
// TODO: Read text
Predicate<String> checkerUpperCase =
                word -> Character.isUpperCase(word.charAt(0));
ArrayList<String> result = new ArrayList<>();
for (int i = 0; i < textAsList.length; i++) {
  if (checkerUpperCase.test(textAsList[i]))
    result.add(textAsList[i]);
}
// TODO: Print results
```

Check your solution here: https://judge.softuni.bg/Contests/1513/Functional-Programming-Lab

Problem: Add VAT

- Read some items' prices from the console
- Add **VAT** of **20%** to all of them


```
1.38, 2.56, 4.4
```

```
Prices with VAT:
1,66
3,07
5,28
```

Check your solution here: https://judge.softuni.bg/Contests/1513/Functional-Programming-Lab

Solution: Add VAT

```
// TODO: Read input
List<Double> numbers = new ArrayList<>();
for (String s : input)
  numbers.add(Double.parseDouble(s));
UnaryOperator<Double> addVat = x -> x * 1.2;
System.out.println("Prices with VAT:");
for (Double str : numbers)
  System.out.println(String.format("%1$.2f",
                                   addVat.apply(str)));
```

Check your solution here: https://judge.softuni.bg/Contests/1513/Functional-Programming-Lab

Bi Functions

Using Functions With More Parameters

BiFunctions

- `BiFunction <T, U, R>`

```
BiFunction <Integer, Integer, String> sum = (x, y) -> "Sum is" + (x + y);
```

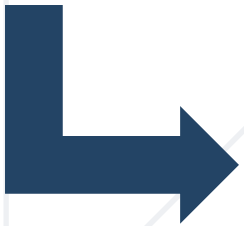
Two input parameters

- Analogically you can use:
  - `BiConsumer <T, U>`
  - `BiPredicate <T, U>`

Problem: Sum Numbers

- Read numbers from the console
- Print their count
- Print their sum
- Use **BiFunctions**

```
4, 2, 1, 3, 5, 7, 1, 4, 2, 12
```



```
Count = 10
Sum = 41
```

Check your solution here: https://judge.softuni.bg/Contests/1513/Functional-Programming-Lab

Solution: Sum Numbers

```
// TODO: Read input
int length = input.length;
int sum = Integer.parseInt(input[0]);
if (input.length >= 2) {
  BiFunction<Integer, String, Integer> parser =
                                  (x, y) -> x + Integer.parseInt(y);
  for (int i = 1; i < input.length; i++)
    sum = parser.apply(sum, input[i]);
}
// TODO: Print output
```

Check your solution here: https://judge.softuni.bg/Contests/1513/Functional-Programming-Lab

Passing Functions to Method

- We can pass `Function<T,R>` to methods:

```
static int operation(int number, Function<Integer, Integer> function) {
    return function.apply(number);
}
```

- We can use the method like that:

```
int a = 5;
int b = operation(a, number -> number * 5);  // b = 25
int c = operation(a, number -> number - 3);  // c = 2
int d = operation(b, number -> number % 2);  // d = 1
```

Problem: Filter by Age

- Read from console **n** people with their age
- Read a condition and an age so to **filter** them
- Read **format type** for the output
- Print all people that fulfill the condition

Peter	20
George	18
Raddy	29
Maria	32
Ivan	16

- Condition - "older"
- Age - 20
- Format - "name age"

Peter	20
Raddy	29
Maria	32

Solution: Filter by Age (1)

```
//TODO: Read info from the console
Predicate<Integer> tester = createTester(condition, age);
Consumer<Map.Entry<String, Integer>> printer =
                                    createPrinter(format);
printFilteredStudent(people, tester, printer);
```

Check your solution here: https://judge.softuni.bg/Contests/1513/Functional-Programming-Lab

Solution: Filter by Age (2)

```
static Consumer<Map.Entry<String, Integer>> createPrinter(String format) {
  Consumer<Map.Entry<String, Integer>> printer = null;
  switch (format) {
    case "name age":
      printer = person -> System.out.printf("%s - %d%n",
                                person.getKey(), person.getValue());
      break;  //TODO: Add more cases
  }
  return printer;
}
```

Check your solution here: https://judge.softuni.bg/Contests/1513/Functional-Programming-Lab

Solution: Filter by Age (3)

```
static Predicate<Integer> createTester(String condition, Integer age) {
  Predicate<Integer> tester = null;
  switch (condition) {
    case "younger":
      tester = x -> x <= age;
      break;  //TODO: Add more cases
  }
  return tester;
}
```

Check your solution here: https://judge.softuni.bg/Contests/1513/Functional-Programming-Lab

Solution: Filter by Age (4)

```
static void printFilteredStudent(
                LinkedHashMap<String, Integer> people,
                Predicate<Integer> tester,
                Consumer<Map.Entry<String, Integer>> printer) {
  for (Map.Entry<String, Integer> person : people.entrySet()) {
    if (tester.test(people.get(person.getKey())))
      printer.accept(person);
  }
}
```

Check your solution here: https://judge.softuni.bg/Contests/1513/Functional-Programming-Lab

Summary

- Lambda expressions are anonymous methods
- **Function<T,R>** is a function that returns R type
- **Consumer<T>** is a void function
- **Supplier<T>** gets no parameters
- **Predicate<T>** evaluates a condition
- **BiFunction<T, U, R>** accepts two parameters
- Functions can be passed like variables to methods

Questions?

SoftUni

Software University | SoftUni Svetlina | SoftUni Creative | SoftUni Digital | SoftUni Foundation | SoftUni Kids

SoftUni Diamond Partners

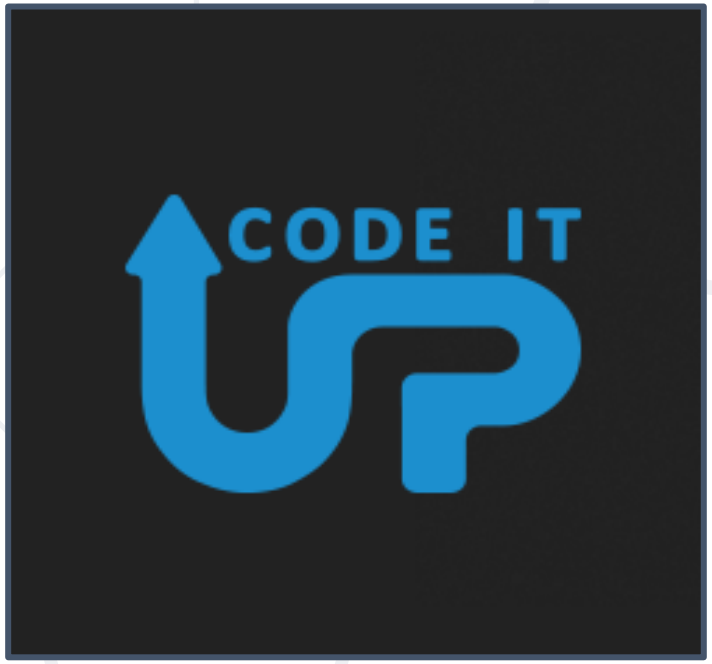
Software University

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

Educational Partners

Software University



VIRTUAL RACING SCHOOL

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

Software University

FORUM

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg
- © Software University – https://softuni.bg