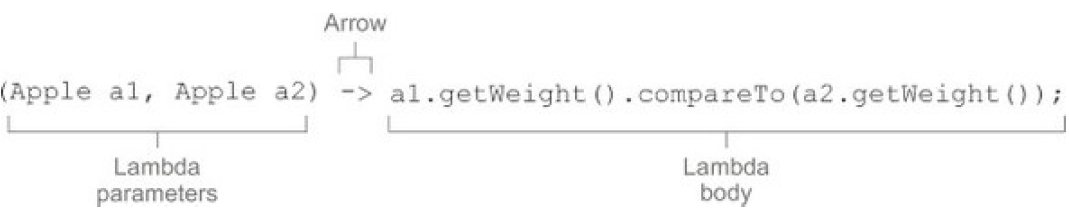
Java 8

**Lambda**

1. The basic syntax of a lambda are:

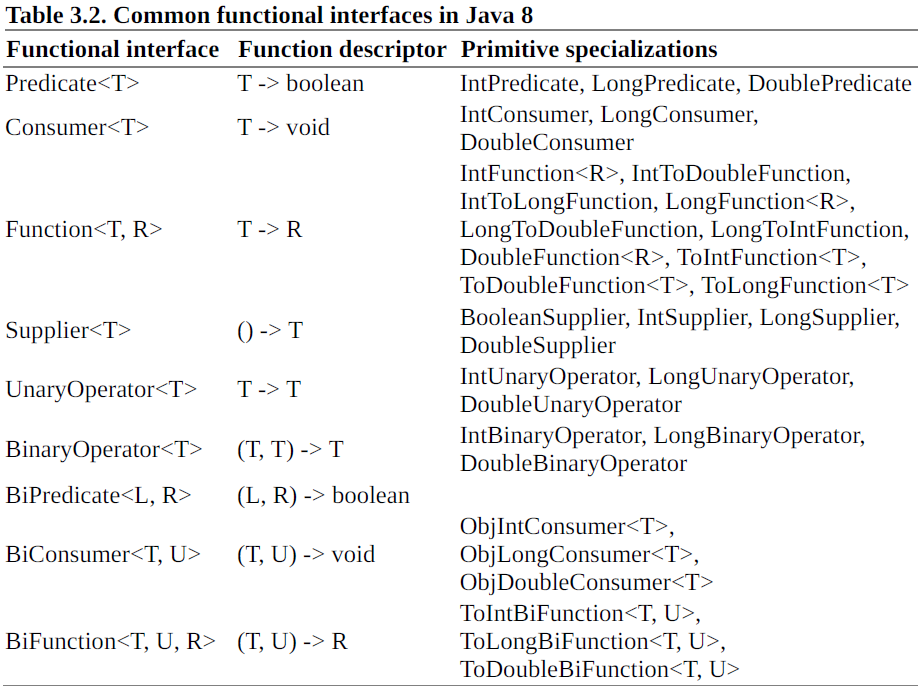
* (parameters) -> expression
* (parameters) -> {statements;} //(note the curly braces for statements)

1. Example:



* A list of parameters.
* An arrow.
* The body of the lambda.

1. Common functional interfaces



Note: none of the functional interfaces allow for a checked exception to be thrown. You have two

options if you need a lambda expression to throw an exception:

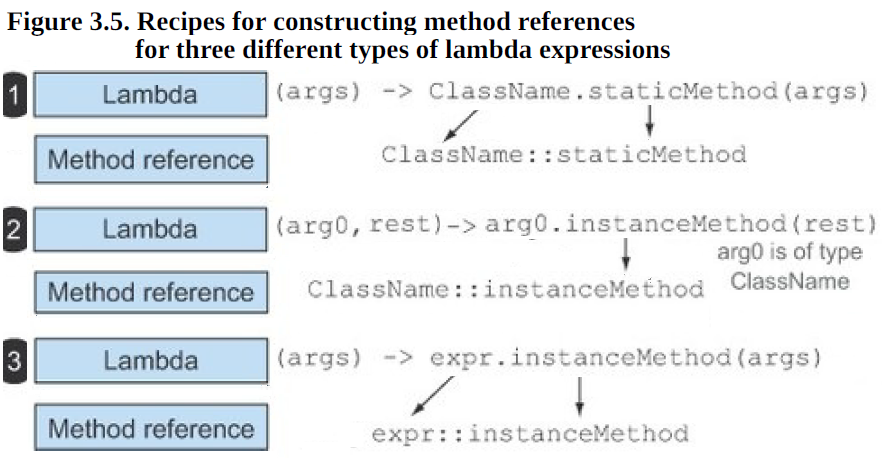
* define your own functional interface that declares the checked exception, or
* wrap the lambda with a try/catch block.

**Method references**

Method references let you reuse existing method definitions and pass them just like lambdas.

There are three main kinds of method references:

1. **static method** (for example, the method parseInt of Integer, written **Integer::parseInt**)
2. **instance method of an arbitrary type** (for example, the method length of a String, written **String::length**)
3. **instance method of an existing object** (for example, suppose you have a local variable expensiveTransaction that holds an object of type Transaction, which supports an instance method getValue; you can write **expensiveTransaction::getValue**)



Constructor references

**Stream**

1. Stream vs collection

* Collections are about data:

collections are data structures, they’re mostly about storing and accessing elements with specific time/space complexities (for example, an ArrayList vs. a LinkedList).

* Streams are about computations:

streams are about expressing computations such as filter, sorted, and map.

1. Streams consume from a data-providing source such as collections, arrays, or I/O resources.
2. Streams support database-like operations and common operations from functional programming languages to manipulate data, such as filter, map, reduce, find, match, sort, and so on
3. Pipelining— Many stream operations return a stream themselves, allowing operations to be chained and form a larger pipeline.
4. Internal iteration— In contrast to collections, which are iterated explicitly using an iterator, stream operations do the iteration behind the scenes for you.
5. 34343

5.4. Reducing

