Java: Lambda Cheat Sheet | Programming.Guide

7–8 minutes

Lambdas

```
() -> "Hello"
() -> System.out.println("Hello")
(String str) -> str.length()
(str) -> str.length()
str -> str.length()
(int i, int j) -> i + j
(i, j) -> i + j

() -> {
    System.out.println("Hello");
    System.out.println("World");
}


(int i) -> {
    System.out.println("Hello");
    return i;
}
```

Method References

```
// Static methods
Supplier<Thread> runtimeSup =
Thread::currentThread;

// Bound instance methods
Supplier<String> helloSup = "hello"::toUpperCase;
Consumer<String> printer = System.out::println;

// Unbound instance methods
Function<String, String> lower =
String::toLowerCase;

// Constructors
Supplier<String> stringSup = String::new;
Function<Integer, int[]> arrSup = int[]::new;
```

Standard Functional Interfaces

Interface	Type	
Runnable		→
BiConsumer<T, U>	T, U	→
BiFunction<T, U, R>	T, U	→ R
BinaryOperator<T>	T, T	→ T
BiPredicate<T, U>	T, U	→ boolean
BooleanSupplier		→ boolean
Callable<V>		→ V
Consumer<T>	T	→
DoubleBinaryOperator	double, double	→ double
DoubleConsumer	double	→
DoubleFunction<R>	double	→ R
DoublePredicate	double	→ boolean
DoubleSupplier		→ double
DoubleToIntFunction	double	→ int
DoubleToLongFunction	double	→ long
DoubleUnaryOperator	double	→ double
Function<T, R>	T	→ R
IntBinaryOperator	int, int	→ int
IntConsumer	int	→
IntFunction<R>	int	→ R
IntPredicate	int	→ boolean
IntSupplier		→ int
IntToDoubleFunction	int	→ double
IntToLongFunction	int	→ long
IntUnaryOperator	int	→ int
LongBinaryOperator	long, long	→ long
LongConsumer	long	→
LongFunction<R>	long	→ R

Interface	Type	
LongPredicate	long	→ boolean
LongSupplier		→ long
LongToDoubleFunction	long	→ double
LongToIntFunction	long	→ int
LongUnaryOperator	long	→ long
ObjDoubleConsumer<T>	T, double	→
ObjIntConsumer<T>	T, int	→
ObjLongConsumer<T>	T, long	→
Predicate<T>	T	→ boolean
Supplier<T>		→ T
ToDoubleBiFunction<T, U>	T, U	→ double
ToDoubleFunction<T>	T	→ double
ToIntBiFunction<T, U>	T, U	→ int
ToIntFunction<T>	T	→ int
ToLongBiFunction<T, U>	T, U	→ long
ToLongFunction<T>	T	→ long
UnaryOperator<T>	T	→ T

Custom Functional Interfaces

Declared like:

```
@FunctionalInterface
interface MyInterface {
    String method(String str);
}
```

Used like:

```
MyInterface doubler = str -> str + str;
String abab = doubler.method("ab");
```

- Functional interface: Any interface with a single abstract method
- Can have additional `default` methods
- The `@FunctionalInterface` annotation (which is optional) causes the compiler to complain if the interface is not a functional interface