# Overview

https://java2blog.com/java-8-interview-questions/

## What are the important features of java 8?

* The main addition is functional programming constructs to object-oriented roots.
* Lambda Expression
* The stream API
* Functional Interface
* Method reference
* Optional
* Data API
* Nashorn, JavaScript Engine

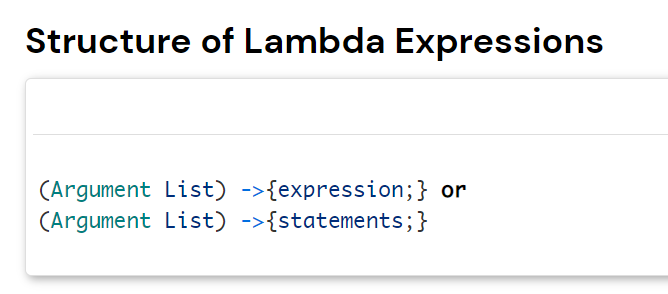
## What are main advantages of using Java 8?

* More compact code
* More readable and reusable code
* Parallel operations
* Java 8 can be used with distributed systems like Spark

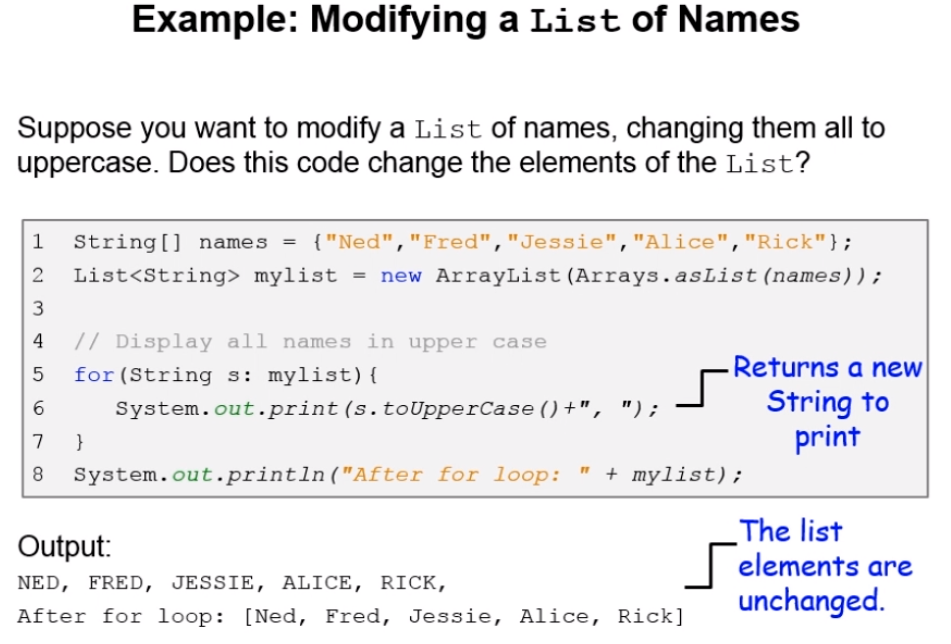
# Lambda Expressions

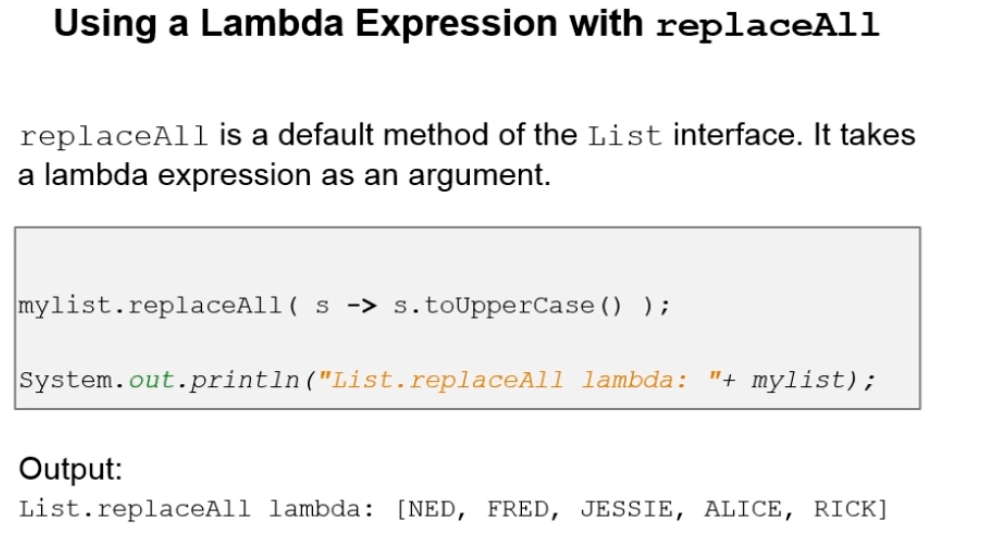
## What is lambda expression?

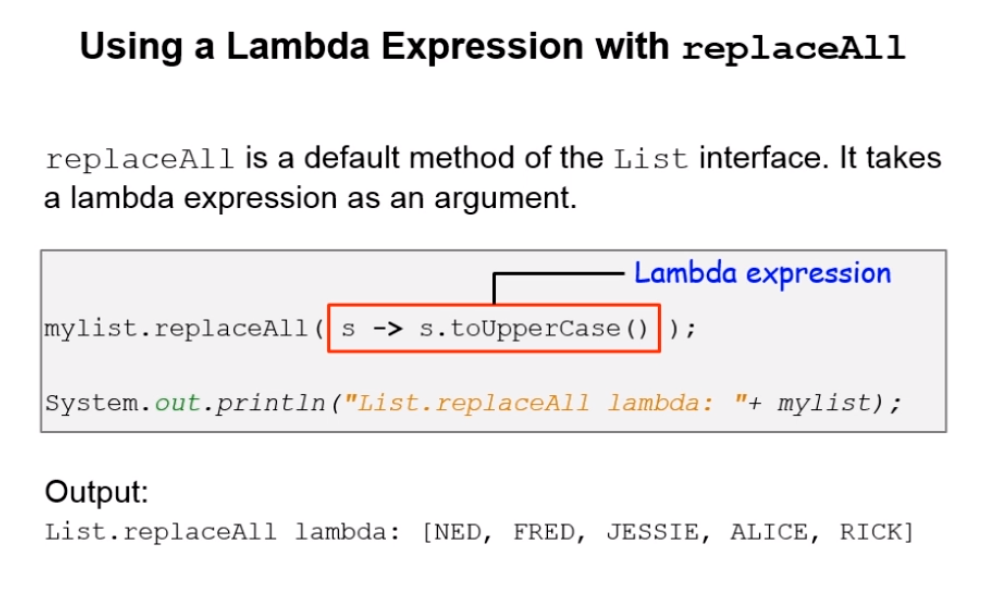
Block of code that we can pass around so that it can be executed once or multiple times.

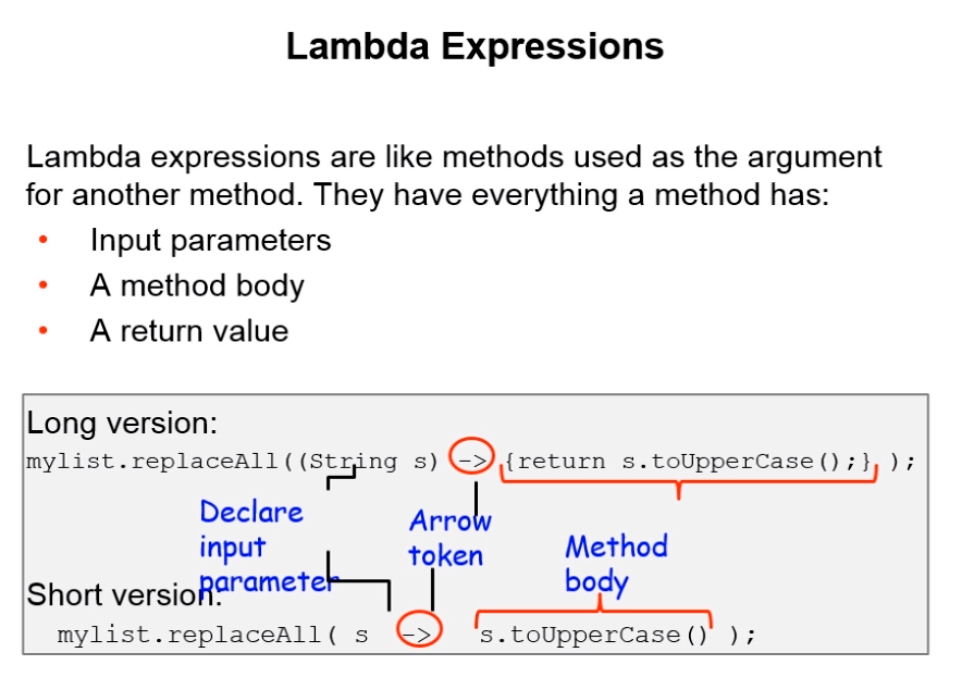


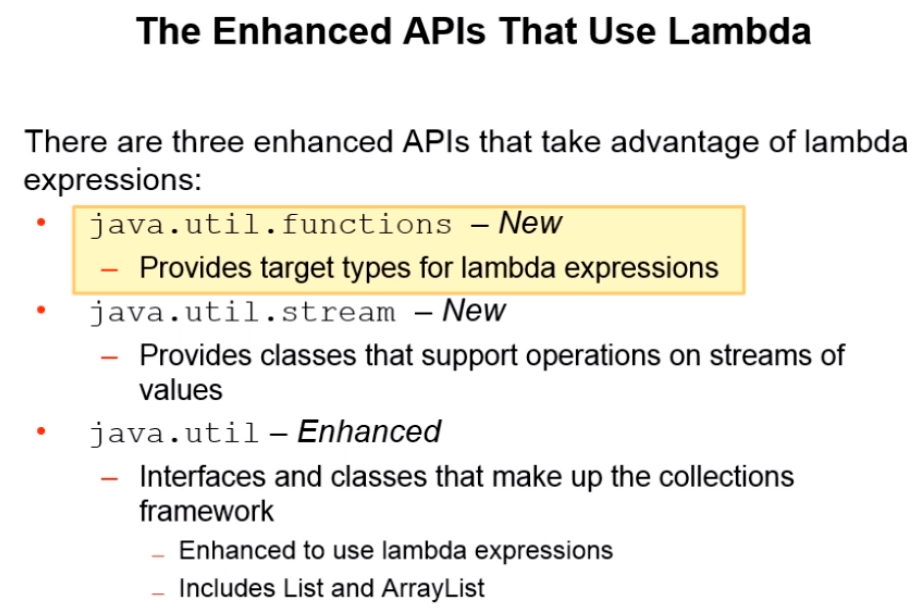
https://java2blog.com/lambda-expressions-in-java-8/

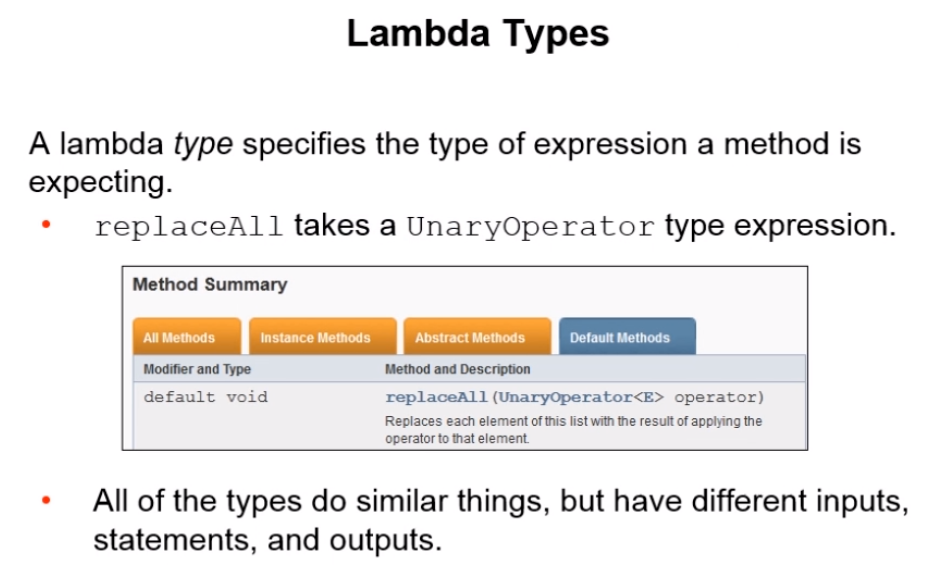


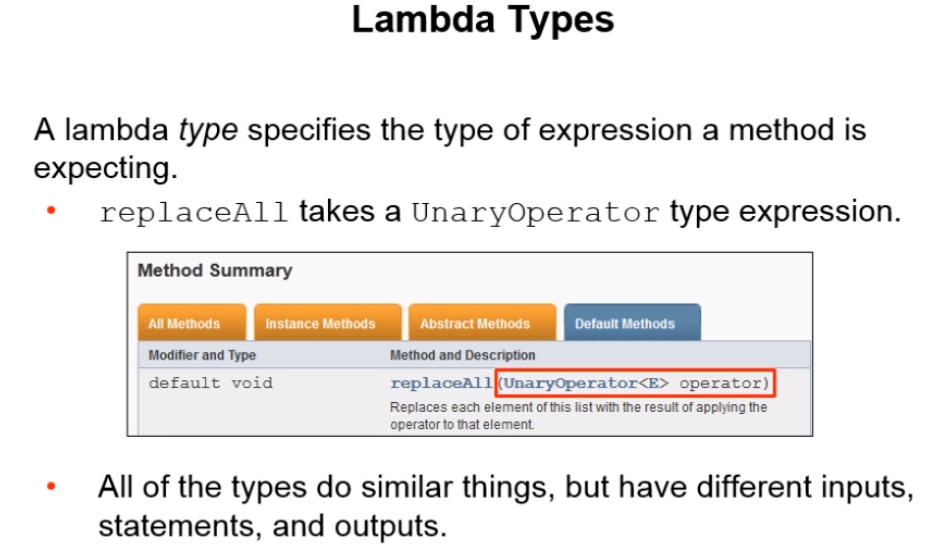


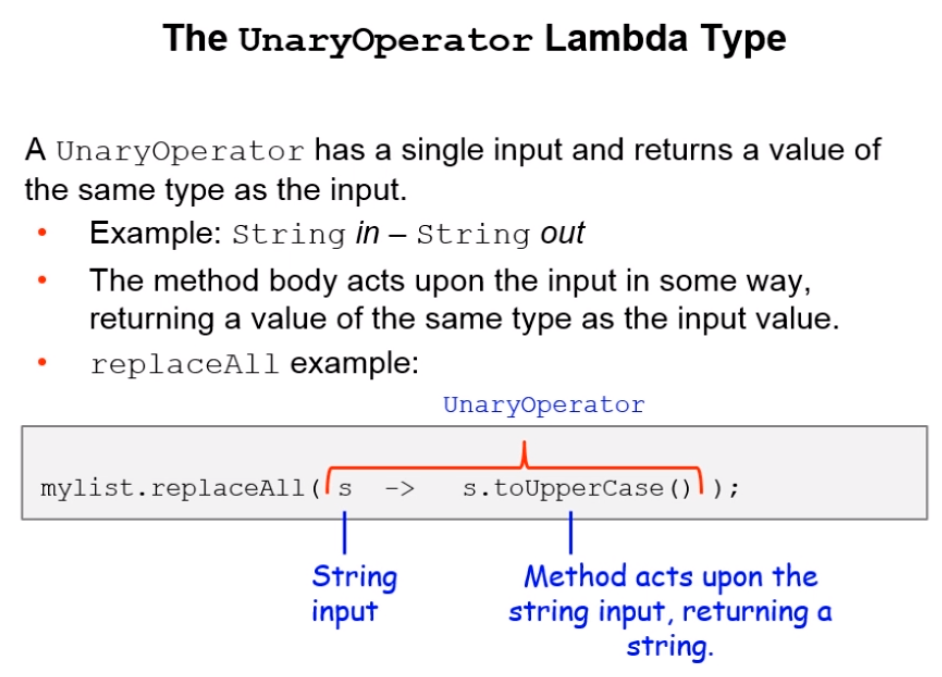


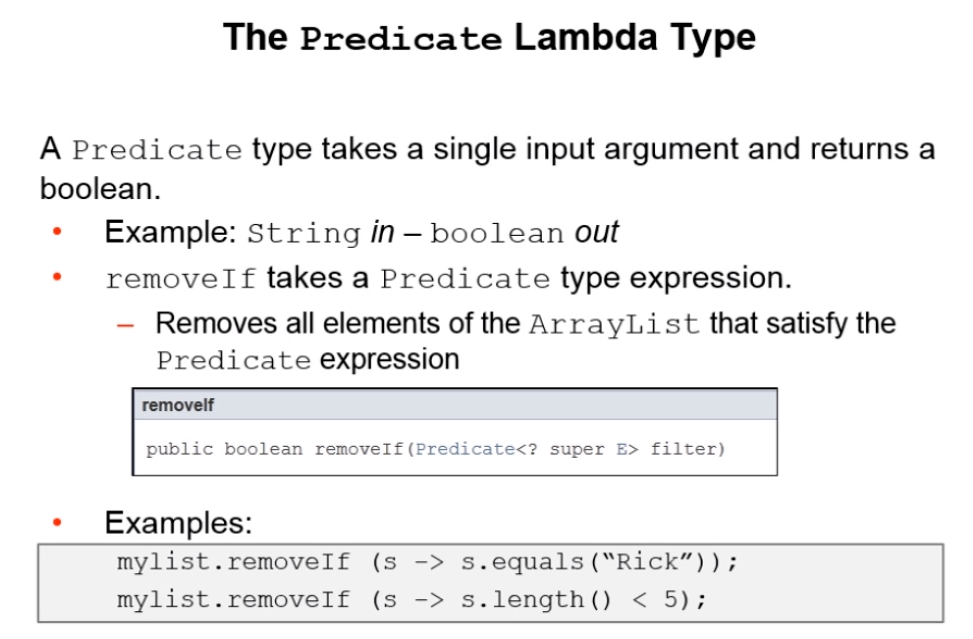




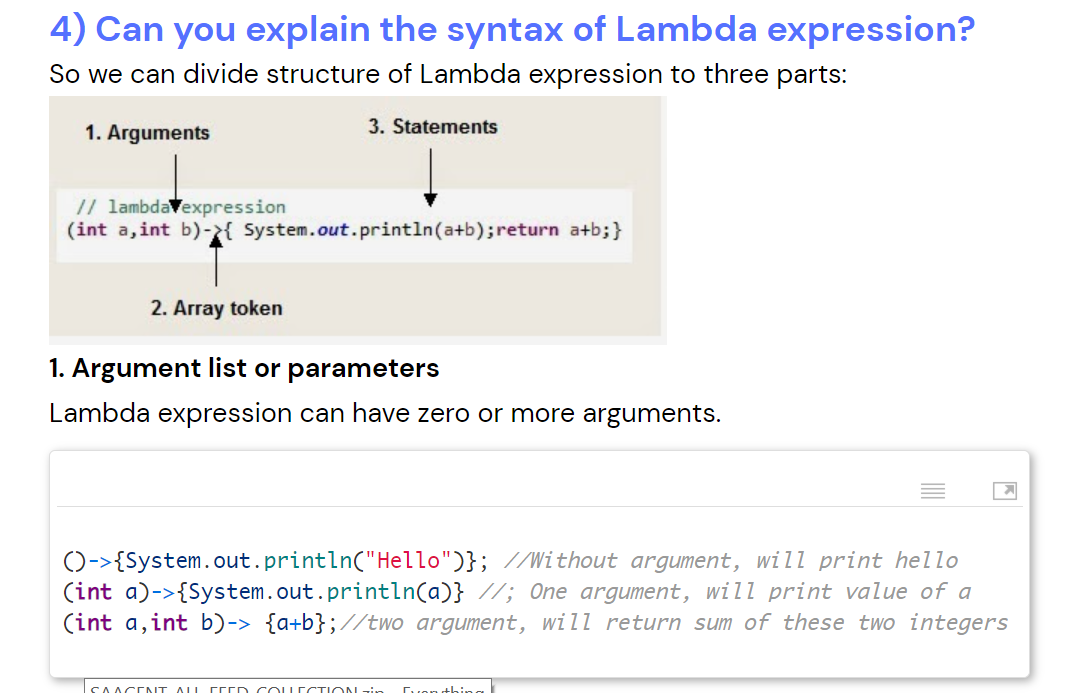


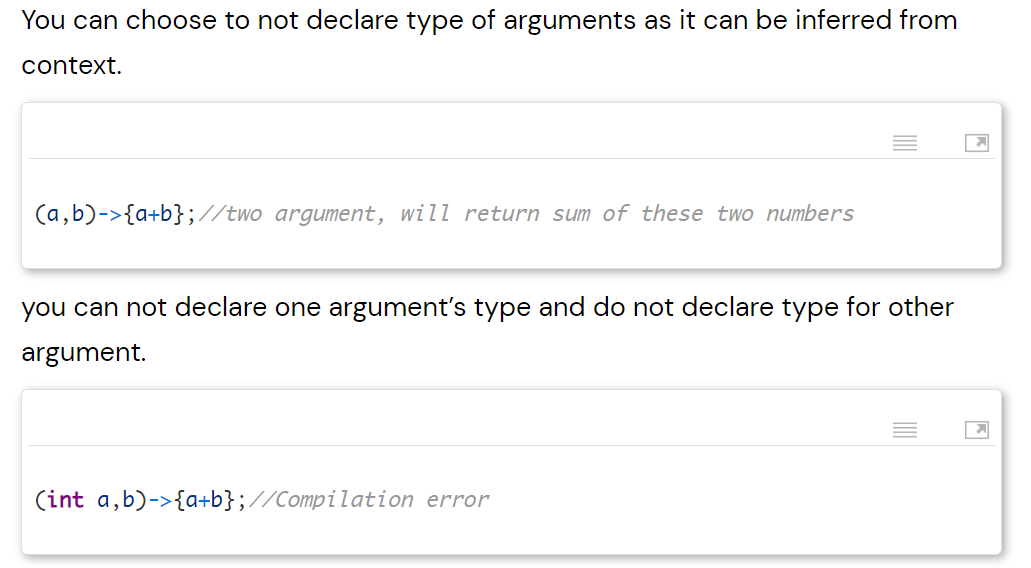


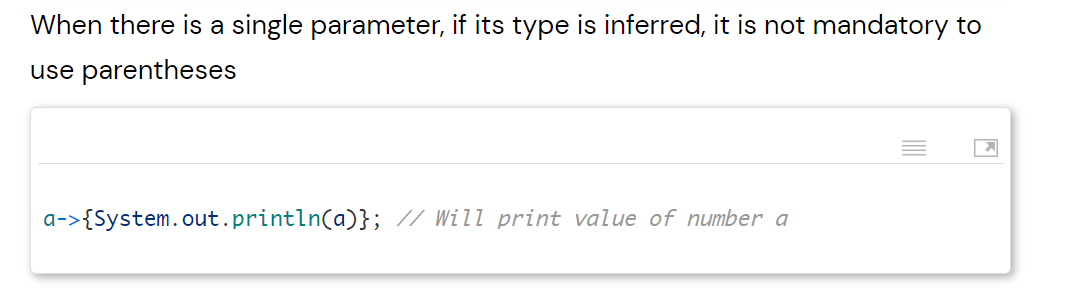


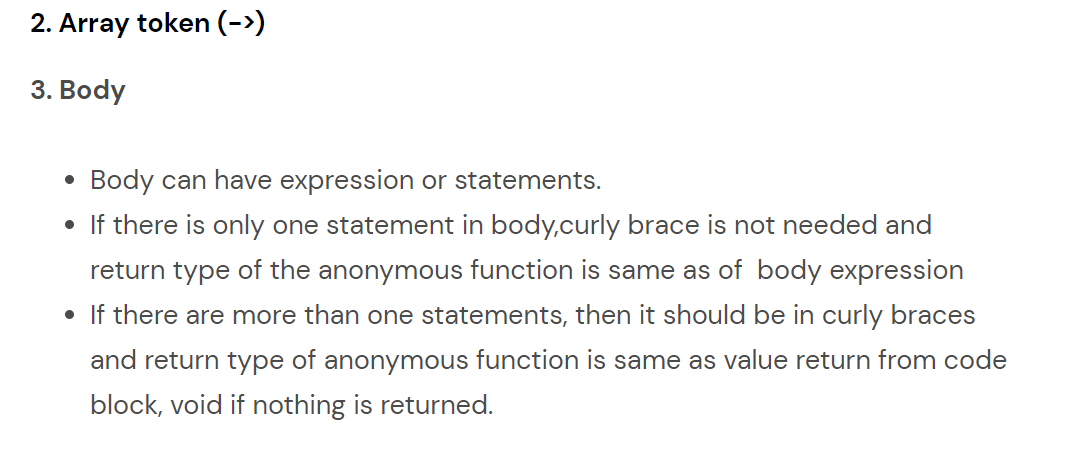


## Can you explain the syntax of Lambda expression?









## What are functional interfaces?

Functional interfaces are those interfaces which can have only one abstract method.It can have static method, default methods or can override Object’s class methods.

There are many functional interfaces already present in java such as Comparable, Runnable.

As we have only one method in Runnable, hence it is considered as functional interface.

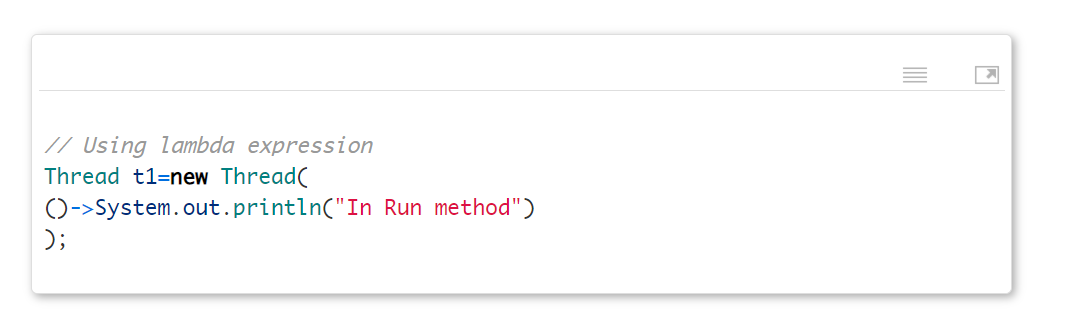
https://java2blog.com/java-8-functional-interface-example/

## How lambda expression and functional interfaces are related?

Lambda expressions can only be applied to abstract method of functional interface.

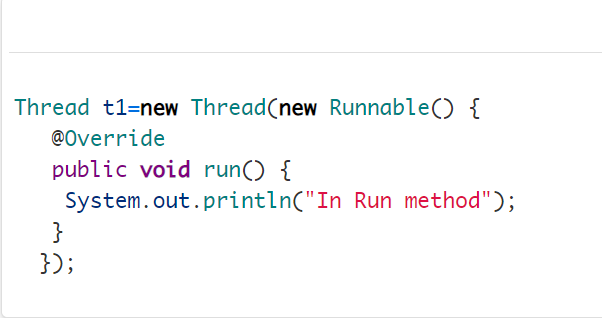
For example

Runnable has only one abstract method called run, so it can be used as below:



Here we are using Thread constructor which takes Runnable as parameter. As you can see we did not specify any function name here, as Runnable has only one abstract method, java will implicitly create anonymous Runnable and execute run method.

It will be as good as below code.

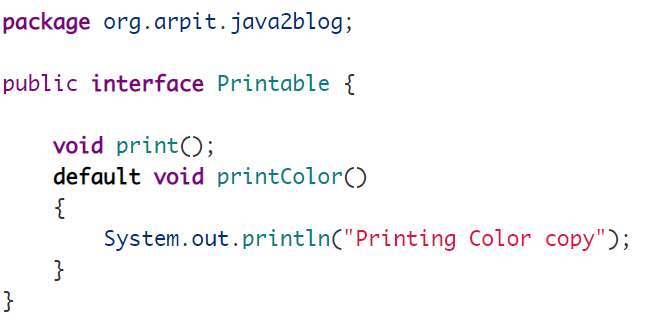


## Can you create your own functional interface?

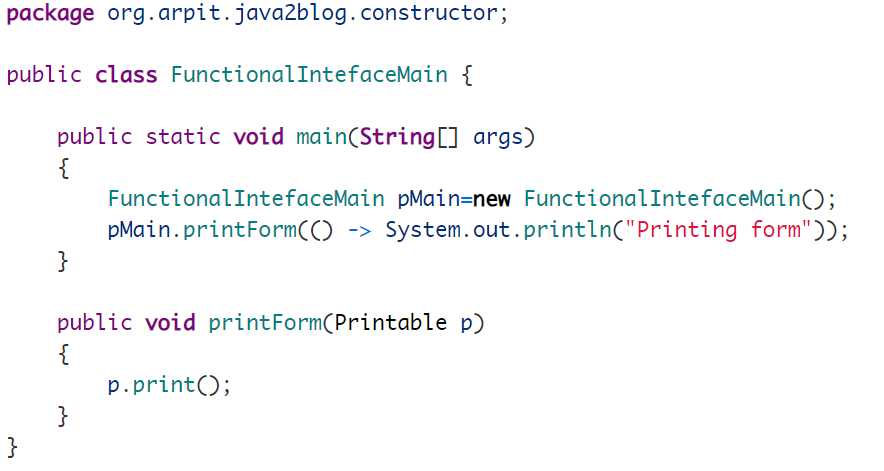
Yes, you can create your own functional interface. Java can implicitly identify functional interface but you can also annotate it with @FunctionalInterface.

Example:

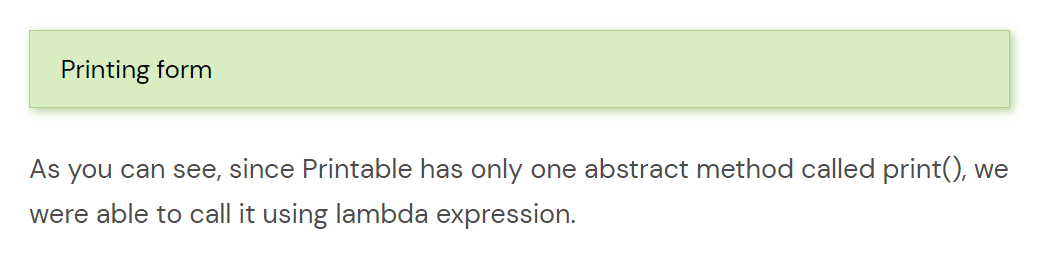
Create interface named "Printable" as below



Create main class named "FunctionalIntefaceMain"



When you run above program, you will get below output:



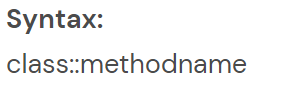
## What is method reference in java 8?

<https://java2blog.com/java-8-method-reference/>

Method reference is used refer method of functional interface.

It is nothing but compact way of lambda expression.

You can simply replace lambda expression with method reference.



## What is Optional? Why and how can you use it?

This class is basically introduced to avoid NullPointerException in java.

Optional class encapsulates optional value which is either present or not.

It is a wrapper around object and can be used to avoid NullPointerExceptions.

Let’s take a simple example



## What are defaults methods?

<https://www.java2blog.com/interface-default-methods-in-java-8/>

methods in interface which have body and prefixed by ***“default”*** keywords.

We have an interface and many classes implemented that interface.

Now if we add new methods to interface our java project will be full of compilation errors because we need to add these new methods to all classes. which are implementing that interface. So instead of adding normal method if we add default method, we will not get any compilation issue. So in this way we achieved backward compatibility.

Default methods introduced in java8 to maintain backward compatibility.

## Can we override default methods?

Yes.

## What about multiple inheritance in case of java?

Adding default implementation to the interface can give rise to ambiguity in multiple inheritance. As two interfaces can provide same default method and there can be ambiguity while calling. Java 8 will give you compile time exception when this kind of situation will arise.

## What is the difference between Predicate and Function?

Note: "predicate" == "filter criteria"

https://java2blog.com/java-8-predicate-examples/

|  |  |
| --- | --- |
| Predicate | Function |
| Functional interface | Functional interface |
| Predicate<T> is single argument function and either it returns true or false. This can be used as assignment target for lambda expression or method reference. | Function<T,R> is also single argument function but it returns an Object. Here T denotes type of input to the function and R denotes type of Result.  This can also be used as the assignment target for a lambda expression or method reference. |

1. **forEach() method in Iterable interface:**

**Iterable interface is added with new method forEach**

Doubts

In this case how will you delete an element from the collecti

<http://www.journaldev.com/2389/java-8-features-with-examples#iterable-forEach>

1. **default and static methods in Interfaces**

[**http://www.journaldev.com/2752/java-8-interface-changes-static-method-default-method**](http://www.journaldev.com/2752/java-8-interface-changes-static-method-default-method)

1. **Functional Interfaces @Functional interface annotation**
2. **Lambda Expressions**

We can visualize functional programming in the java object oriented world.

(argument) -> (body).

<http://www.journaldev.com/2763/java-8-functional-interfaces>

1. **Java Stream API for Bulk Data Operations on Collections.**

<http://www.journaldev.com/2774/java-8-stream>

1. **Java Time API**

[**http://www.journaldev.com/2800/java-8-date-localdate-localdatetime-instant**](http://www.journaldev.com/2800/java-8-date-localdate-localdatetime-instant)

1. **Collection API improvements**
2. **Concurrency API improvements**
3. **Java IO improvements**
4. **Miscellaneous Core API improvements**
5. **Nashorn, JavaScript Engine**

# Stream API (for collections)

## What is stream API in Java? IMP

Stream is abstraction to process collections of values and specifying what you want to do, leaving the scheduling of operations to

the implementation.

https://stackoverflow.com/questions/44180101/in-java-what-are-the-advantages-of-streams-over-loops

## What are the advantages of streams? IMP

* Streams are designed to work with large datasets, whereas arrays and collections etc which implements iterable are entirely in memory
* Streams are more declarative style or more expressive style. Using stream, we can declare our intent in code rather than describing how it is done

For example if you have employee details in an array and you want to filter employee details based on their age then we can write code like

*employee.filter( p -> p.age() < 19).collect(toList());*

It quite clearly says that you're filtering matching elements from a list,

*List<Employee> filtered = new ArrayList<> ();*

*for (Employee e: employee) {*

*if(e.age() < 19) {*

*filtered.add(p);*

*}*

*}*

*return filtered;*

"I'm doing a loop". The purpose of the loop is buried deeper in the logic

* Stream works will with functional programming
* Stream encourages loose coupling: stream-handling code doesn't need to know the source of the stream

## What are the disadvantages of streams?

Filters and maps cannot throw checked exceptions

**Performance**: A for loop through an array is extremely lightweight both in terms of heap and CPU usage.

**Readability**: as this is more of expressive way so people who are experience in functional programming can read it quite easily. If new team member arrives and if is more of procedural programming background, he will find it more difficult to understand. Because industry is full of procedural programmers.

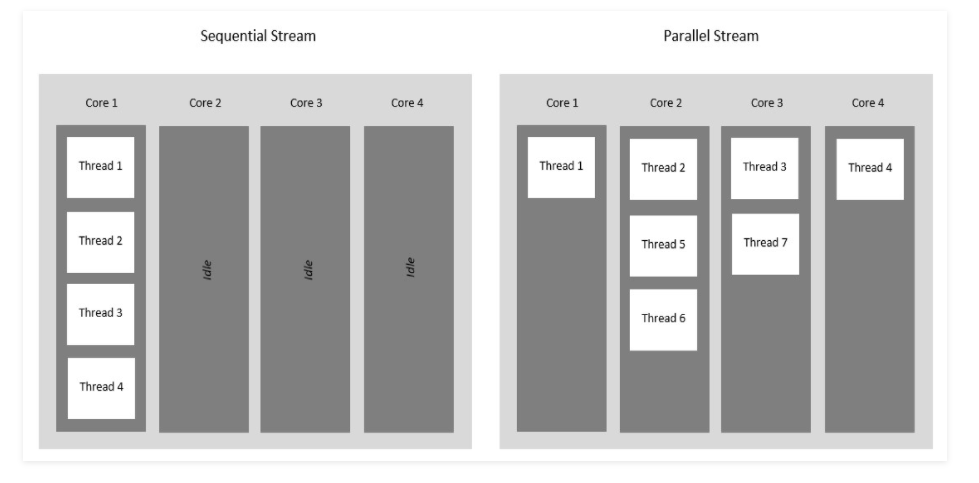
**Debugging**: debugging is difficult because more abstraction is involved.

## On which all things we can create stream?

collections, arrays, generators, or iterators

## What are parallel streams?

In case of stream all threads will run in a single core, whereas if we apply parallel stream on a collection the data will be divided, and threads will be executed parallelly in multiple cores.



## When we should use parallel streams?

When result is unaffected by order of execution and state of one element does not affect other, we should use parallel streams.

# Date and Time API

## What the issues with Old Date and time API?

**Thread Safety**: You might be already aware that java.util.Date is mutable and not thread safe. Even java.text.SimpleDateFormat is also not Thread-Safe. New Java 8 date and time APIs are thread safe.

**Performance**: Java 8 ‘s new APIs are better in performance than old Java APIs.

**More Readable**: Old APIs such Calendar and Date are poorly designed and hard to understand. Java 8 Date and Time APIs are easy to understand and comply with ISO standards.

## Can you provide some APIs of Java 8 Date and Time?

LocalDate, LocalTime, and LocalDateTime are the Core API classes for Java 8. As the name suggests, these classes are local to context of observer. It denotes current date and time in context of Observer.

## How will you get current date and time using Java 8 Date and Time API?

You can simply use now method of LocalDate to get today’s date.

LocalDate currentDate = LocalDate.now();

System.out.println(currentDate);

# Others

## Do we have PermGen in Java 8? Are you aware of MetaSpace?