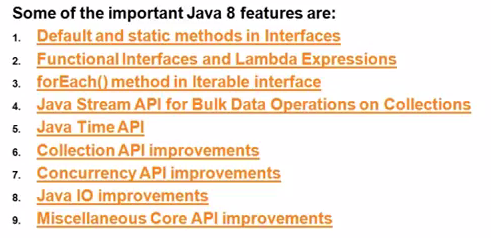
**Java 1.8 Features:**

1. Lambda Expression
2. Functional Interface:
   1. Predicate
   2. Functional
   3. Consumer
3. Default Methods in Interface
4. Static Methods in Interface
5. Method Reference & Constructor Reference by double colon (: :) operator
6. Stream API
7. Date & Time API (Joda API) joda.org



**Objective of Java 8:**

1. To Simplify Programming
2. To utilize the benefits of Functional programming in Java

Java is Object Oriented Program but 1.8 enables us to get the benefit of functional programming.

1. To enable parallel programming/processing.

**Lambda Expression:**

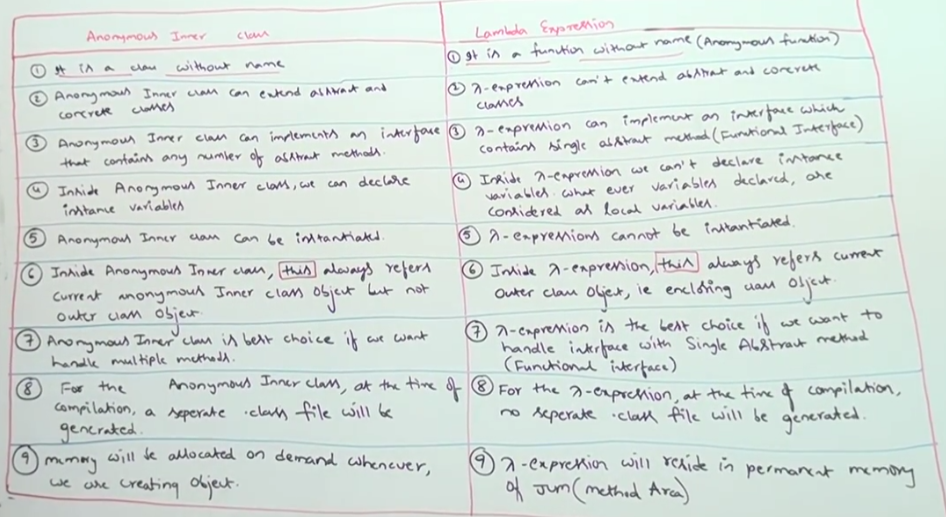
1. Enables functional programming in java
2. Write more readable, maintainable and concise code.
3. To use APIs very easily & effectively.
4. Enable Parallel Processing.
5. We can handle procedures/functions just like values or arguments.
6. Easier to use updated APIs and Libraries.
7. We can resolve complexity of anonymous inner class until some extent.

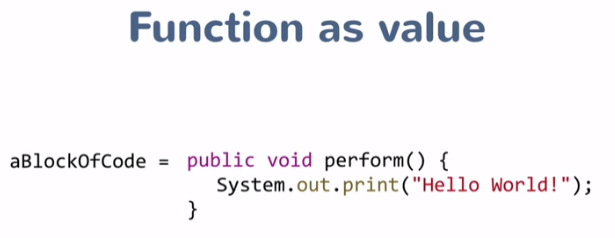
Functional Interface: Interface having single abstract method.

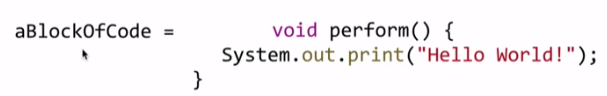
Local variable referenced from lambda expression must be final or effectively final.

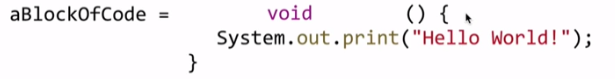
Steps to write the Lambda Expression

1. Remove the Modifiers
2. Remove the Method name
3. Remove the return type
4. If the body is one line. Remove the curly braces

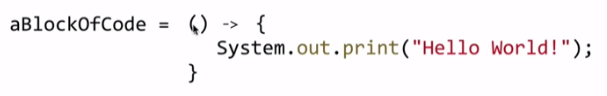






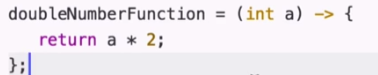
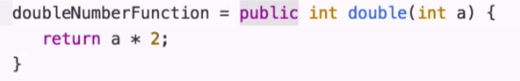








**Example2:**



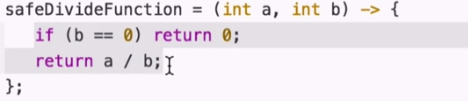
**For one line code, we can skip the return itself.**



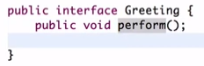


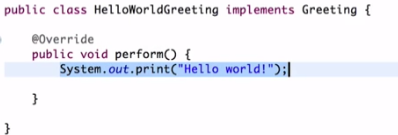






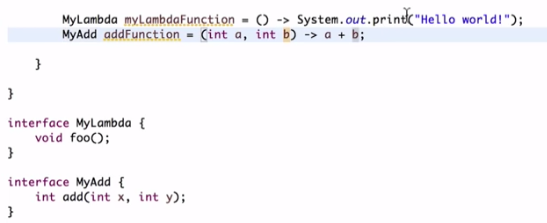








Create an interface with exactly same signature as the void method so that the lambda expression can be of that interface type.

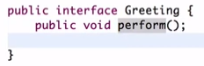


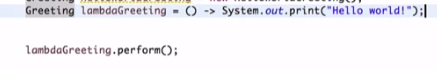
As we already have Greeting interface with single method with no argument and no return type which is matching the lambda expression we can use it without creating a new one.



When we are using an interface to declare Lambda expression then that interface should be a functional interface.

The functional interface should have only one method which matches the lambda expression.

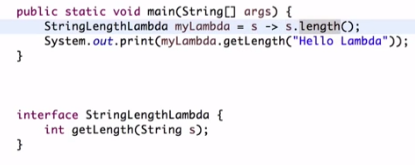




In above code lambdaGreeting is the lambda expression of type greeting interface.

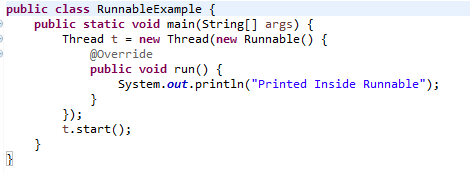
For one argument, we don’t even need the parenthesis.

Ex:



**Thread Creation using Lambda Expression:**

**Without Lambda Expression:**

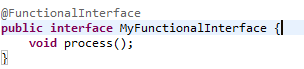


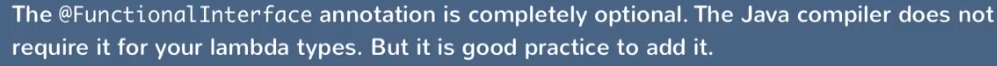
**With Lambda Expression:**



Java 8 gives a clue about functional interface if we declare an interface with @FuntionalInterface.

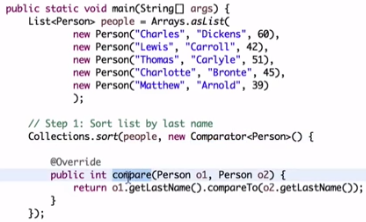
Ex:





Collection Sorting with Comparator and lambda expression:

**Earlier Code**



**New Code:**

Collections.sort(list, (p1, p2)->p1.getFname().compareTo(p2.getFname()));

Any Interface with single abstract method can be called as a functional interface.

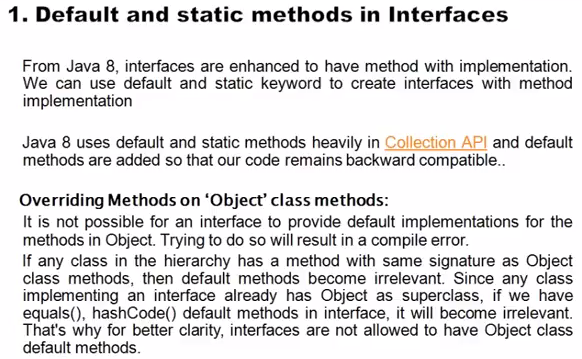
Java 8 Provides some interfaces which can be used in the Lambda Interface.

1. Predicate
2. Consumer
3. BiFunction
4. BiConsumer

**We can add Default and Static methods in the interface from Java 8 onwards**.

We can override the default methods but cannot override the static methods.

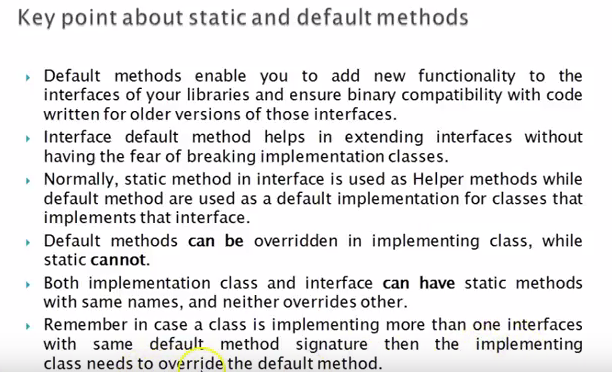
Collections.min(list)🡪It will return the minimum element from the list.



In Java 8 all the interface with single abstract methods are made functional interface.

Ex: Comparator Interface have some additional methods in Java 8. However, it was not there in earlier version of Java. So, to implement the new collection API features of Java 8 we need to implement the methods defined in Comparator which we were not doing earlier.

In Java 8 default methods are there in interface so that without implementing the interface we can directly use those methods. In this way, it won’t affect already implemented code (using comparator).

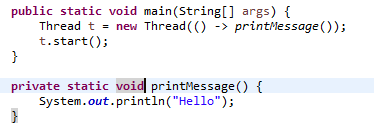


Java 8 introduces default so that list and collection interfaces can have default implementation of for each method. Class implementing these interfaces need not implement the same.

Default method allows us to add new functionality to interface without breaking the classes that implement that interface.

**Method Reference**

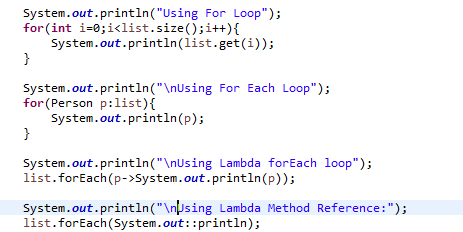
**Earlier:**





**Lambda expression with no input argument and calling a method with no argument is replaced with Method Reference.**

**Iterator:**



**Stream:**

