

## What's new in Java 8

<b>Course Name</b>	What's new in Java 8
<b>Course Description</b>	Java 8, released on March 2014 introduces a number of new features to the Java programming language. Lambda expressions, a functor (function pointer) like feature was well expected and is finally here to make code more convenient and concise. Streams (not IO streams) are wrappers around existing data structures that provide a convenient API to iterate, filter, map/reduce and query a data structure and have nice properties that collections lack. JDK 8 also introduces a new enhanced well-defined and convenient API for dealing with date and time. This training program introduces these new features to Java programmers who are not yet familiar with the new JDK.
<b>Target Population</b>	Experienced Java Programmers that are familiar with at least JDK 5 or a later version
<b>Pre-requisites</b>	<ul style="list-style-type: none"> <li>• Java programming language skills with JDK 5 and later</li> <li>• Familiarity with Generics - Programmers must be able to define generic instance and static methods.</li> <li>• Familiarity with the Java Collections Framework. Programmers must know how to work with lists, maps and sets</li> </ul>
<b>Course Objectives</b>	<p>Upon completion of the course, participants will be able to:</p> <ul style="list-style-type: none"> <li>• Utilize Lambda expressions both in new and re-factored legacy code</li> <li>• Utilize streams to perform operations on underlying data structures and to simplify code that is used to filter , map, reduce , aggregate and query collections...</li> <li>• Utilize and feel more comfortable using the new data and time features of the Java programming language JDK 8</li> </ul>
<b>Course Topics</b>	<b>Module 1 – Overview: Pre-Lambda Java</b> <ul style="list-style-type: none"> <li>o Installing Java 8</li> </ul>



	<ul style="list-style-type: none"> <li>o Installing an Eclipse IDE that supports Java 8</li> <li>o Review: Classic Pre-Lambda Java handlers and anonymous classes <ul style="list-style-type: none"> <li>o Separate class</li> <li>o Main class that implements the interfaces</li> <li>o Named Inner classes</li> <li>o Anonymous inner classes</li> </ul> </li> <li>o Review of making Generic classes and methods (since JDK 5)</li> <li>o <b>Exercise:</b> <ul style="list-style-type: none"> <li>o Program a simple Swing application using each of the traditional ways</li> <li>o</li> </ul> </li> </ul>
	<p><b>Module 2 - Lambda Expressions 1: Basics</b></p> <ul style="list-style-type: none"> <li>o Motivation and the big idea</li> <li>o The new lambda options <ul style="list-style-type: none"> <li>o Interpretation</li> <li>o The basic form</li> <li>o Type inferencing</li> <li>o Expression as lambda body</li> <li>o Omitting the parenthesis</li> <li>o Comparing lambda approaches to the alternatives</li> <li>o Using effectively final variables</li> <li>o The <code>@FunctionalInterface</code> Annotation</li> <li>o Method reference</li> <li>o The <code>java.util.function</code> package</li> </ul> </li> <li>o <b>Exercise</b> <ul style="list-style-type: none"> <li>o Sorting an array of Strings - by length, reverse length, first character, String with 'e'</li> <li>o Improve the sorting with lambdas</li> <li>o Using Predicates to sort the strings</li> <li>o Using Generics to sort an array of any element type</li> </ul> </li> </ul>
	<p><b>Module 3 - Lambda Expressions 2: The building blocks</b></p> <ul style="list-style-type: none"> <li>o Lambda building blocks in <code>java.util.function</code> <ul style="list-style-type: none"> <li>o XunaryOperation</li> <li>o XbinaryOperator</li> <li>o Xpredicate</li> </ul> </li> </ul>



	<ul style="list-style-type: none"> <li>o XConsumer</li> <li>o Generic Versions <ul style="list-style-type: none"> <li>o Predicate</li> <li>o Function</li> <li>o BinaryOperator</li> <li>o Consumer</li> <li>o Supplier</li> </ul> </li> <li>o variable scoping rules for lambda</li> <li>o <b>Exercise</b> <ul style="list-style-type: none"> <li>o Create a static method that accepts a List of Strings and a Predicate and return a new List containing values that pass a lambda expression test</li> <li>o Generify the previous method so it takes a list of any element type</li> <li>o The same as with the Predicate, only this time the method is called transformedList and it takes a list of strings and a Function</li> <li>o Generify the transformedList method</li> </ul> </li> </ul>
	<p><b>Module 4 - Lambda Expressions 3: Wrap up</b></p> <ul style="list-style-type: none"> <li>o Variable Scoping <ul style="list-style-type: none"> <li>o local variables</li> <li>o instance variables</li> <li>o "this"</li> </ul> </li> <li>o Method References – Details <ul style="list-style-type: none"> <li>o Class::staticMethod</li> <li>o variable::instanceMethod</li> <li>o Class::instanceMethod</li> <li>o Class::new</li> </ul> </li> <li>o New features in Java 8 Interfaces <ul style="list-style-type: none"> <li>o Default methods</li> <li>o Static methods</li> </ul> </li> <li>o Lambda returning methods <ul style="list-style-type: none"> <li>o From Predicate: and, or, negate, isEqual</li> <li>o From Function: andThen, compose, identify</li> <li>o From Consumer: chain</li> <li>o custom methods</li> </ul> </li> </ul>
	<p><b>Module 5 - Streams in Java 8: Part 1</b></p> <ul style="list-style-type: none"> <li>o Overview of Streams</li> <li>o Building Streams</li> <li>o Outputting streams into arrays or Lists</li> <li>o Core Stream Methods <ul style="list-style-type: none"> <li>o Overview</li> <li>o forEach</li> </ul> </li> </ul>



	<ul style="list-style-type: none"> <li>o map</li> <li>o filter</li> <li>o findList</li> <li>o Lazy evaluation and short-circuit operations</li> <li>o <b>Exercise</b> <ul style="list-style-type: none"> <li>o Looping a List of String using <code>forEach</code></li> <li>o Looping with <code>forEach</code> and <code>System.out::println</code> method references</li> <li>o Looping with <code>map</code> and <code>collect</code></li> <li>o Looping with <code>filter</code> and <code>collect</code></li> <li>o Looping with chained filters</li> <li>o Looping with lazy evaluation</li> <li>o Converting a list to an array using the <code>String[]::new</code> method reference</li> </ul> </li> </ul>
	<p><b>Module 6 - Streams in Java 8: Part 2</b></p> <ul style="list-style-type: none"> <li>o Additional Stream methods <ul style="list-style-type: none"> <li>o <code>reduce</code>, <code>sum</code></li> <li>o <code>limit</code>, <code>skip</code></li> <li>o <code>sorted</code>, <code>min</code>, <code>max</code>, <code>distinct</code></li> <li>o <code>noneMatch</code>, <code>allMatch</code>, <code>anyMatch</code>, <code>count</code></li> </ul> </li> </ul>



	<ul style="list-style-type: none"> <li>o Parallel Streams</li> <li>o Infinite Streams <ul style="list-style-type: none"> <li>o bounded streams with on-the-fly calculated values</li> </ul> </li> <li>o Grouping Stream Elements <ul style="list-style-type: none"> <li>o fancy uses of collect</li> </ul> </li> <li>o <b>Exercise</b> <ul style="list-style-type: none"> <li>o Using a single <code>stream.reduce</code> (no map) to uppercase and concatenate a list of strings</li> <li>o Using map and <code>reduce</code> to do the same</li> <li>o Produce a CSV string using <code>reduce</code> and <code>orElse</code></li> <li>o Creating a static method for producing a list of a given size of random numbers using <code>Stream.generate(Supplier&lt;?&gt;)</code> and <code>collect</code></li> <li>o Creating a static method that produces a list of numbers that go in order by a step size using <code>Stream.iterate(Integer, UnaryOperator)</code> and <code>collect</code></li> <li>o Compute the sums of some ints - in serial using <code>stream().reduce</code> Vs. in parallel</li> <li>o using <code>stream().parallel().reduce</code> (all-ways the same result)</li> <li>o Compute the product of some doubles in serial using <code>stream().reduce(Double, BinaryOperator&lt;Double&gt;)</code> and in parallel using <code>stream().parallel().reduce(Double, BinaryOperator&lt;Double&gt;)</code> (not always the same result)</li> </ul> </li> </ul>
	<p><b>Module 7 - The new Java 8 Date &amp; Time Approach</b></p> <ul style="list-style-type: none"> <li>o Date-Time Overview</li> <li>o Why a new API?</li> <li>o Advantages of the new API</li> <li>o Design Principles</li> <li>o Concepts</li> <li>o New packages and commonly used classes</li> </ul>



	<ul style="list-style-type: none"> <li>o Method Naming Conventions</li> <li>o Formatting and Parsing</li> <li>o Temporal Adjusters and Temporal Queries</li> <li>o Legacy Date-Time Code Interoperability</li> <li>o Summary</li> </ul>
<b>Course Duration</b>	3 Days (24 hours)

