# Java SE 9

## API Updates

LUXOFT TRAINING

LXFT LISTED NYSE

# 1. Private methods for interfaces



From Java 8 we can add **static** and **default** methods to the interface.

Now you can also add **private** methods.

# 2. try with resources



You can declare AutoCloseable resource **outside** the try block and it will be closed automatically.

Just nice syntax update…

# 3. Immutable collections



Added set of constructors for List, Set, and Map.

*Providing APIs for creating small, unmodifiable collections satisfies a large set of use cases, and it helps keep the specification and implementations simple. Unmodifiable collections avoid the need to make defensive copies, and they are more suitable for parallel processing.*

*JEP-269*

# 4. Diamond operator



Now you can use the Diamond operator ( **< >** ) with anonymous inner classes.

Just nice syntax update…

# 5. Process API



The **java.lang.ProcessHandle** class returns information about each process as provided by the operating system including process id, arguments, command, start time, etc.

A ProcessHandle can return the process' parent, and the direct children, and to all descendants via a stream of ProcessHandles.

# 6. Stream API update



2 new methods added to the **java.util.stream.Stream** interface:

```
default Stream<T> dropWhile(Predicate<? super T> predicate)

default Stream<T> takeWhile(Predicate<? super T> predicate)
```

# 6. Stream API update



```
default Stream<T> takeWhile(Predicate<? super T> predicate)
```

Takes first n elements from the stream while predicate is **true**.

# 6. Stream API update



```
default Stream<T> dropWhile(Predicate<? super T> predicate)
```

Drops first n elements of the stream while predicate is **true**.

# 7. Optional update



3 new methods to **java.util.Optional** class

```
public Optional<T> or(Supplier<? extends
      Optional<? extends T>> supplier)

public void ifPresentOrElse(Consumer<? super T> action,
      Runnable emptyAction)

public Stream<T> stream()
```

# 8. Stack-Walking API



Efficient standard API for stack walking that allows easy filtering of, and lazy access to, the information in stack traces.

```
public <T> T walk(Function<? super Stream<StackFrame>, ? extends T> function)

public Class<?> getCallerClass()
```

# Task

Check all the examples by yourself and write down 5 - 10 use cases where you will use the new API (40 min).