Learning Objectives in this Part of the Lesson

- Understand foundational functional programming features in Java 8, e.g.,
  - Lambda expressions
  - Method & constructor references
  - Key functional interfaces
Learning Objectives in this Part of the Lesson

• Understand foundational functional programming features in Java 8, e.g.,
  • Lambda expressions
  • Method & constructor references
  • Key functional interfaces

These features form the basis for Java streams & concurrency/parallelism frameworks
Overview of Functional Interfaces
Overview of Functional Interfaces

- A functional interface is an interface that contains only one abstract method.

Overview of Functional Interfaces

• A functional interface is the type used for a parameter when a lambda expression or method reference is passed as an argument to a method

```
<T> void runTest(Function<T, T> fact, T n) {
    long startTime = System.nanoTime();
    T result = fact.apply(n);
    long stopTime = (System.nanoTime() - startTime) / 1_000_000;
    ...
}
runTest(ParallelStreamFactorial::factorial, n);
runTest(SequentialStreamFactorial::factorial, n);
...
Overview of Functional Interfaces

- A functional interface is the type used for a parameter when a lambda expression or method reference is passed as an argument to a method.

```java
<T> void runTest(Function<T, T> fact, T n) {
    long startTime = System.nanoTime();
    T result = fact.apply(n);
    long stopTime = (System.nanoTime() - startTime) / 1_000_000;
    ...}
```

`runTest(ParallelStreamFactorial::factorial, n);`
`runTest(SequentialStreamFactorial::factorial, n);`
...
Overview of Functional Interfaces

- A functional interface is the type used for a parameter when a lambda expression or method reference is passed as an argument to a method.

```java
<T> void runTest(Function<T, T> fact, T n) {
    long startTime = System.nanoTime();
    T result = fact.apply(n);
    long stopTime = (System.nanoTime() - startTime) / 1_000_000;
    ...
}
runTest(ParallelStreamFactorial::factorial, n);
runTest(SequentialStreamFactorial::factorial, n);
...
```

'Select' parameterizes the factorial implementation.

See docs.oracle.com/javase/8/docs/api/java/util/function/Function.html
Overview of Functional Interfaces

• A functional interface is the type used for a parameter when a lambda expression or method reference is passed as an argument to a method

```java
<T> void runTest(Function<T, T> fact, T n) {
    long startTime = System.nanoTime();
    T result = fact.apply(n);
    long stopTime = (System.nanoTime() - startTime) / 1_000_000;
    ...
}
runTest(ParallelStreamFactorial::factorial, n);
runTest(SequentialStreamFactorial::factorial, n);
...
```

Different factorial implementations can be passed as method reference params to the runTest() method

This is an example of behavior parameterization
Overview of Functional Interfaces

- A functional interface is the type used for a parameter when a lambda expression or method reference is passed as an argument to a method.

```java
<T> void runTest(Function<T, T> fact, T n) {
    long startTime = System.nanoTime();
    T result = fact.apply(n);
    long stopTime = (System.nanoTime() - startTime) / 1_000_000;
    ...
}
runTest(ParallelStreamFactorial::factorial, n);
```

```java
static BigInteger factorial(BigInteger n) {
    return LongStream.rangeClosed(1, n)
        .parallel()
        .mapToObj(BigInteger::valueOf)
        .reduce(BigInteger.ONE, BigInteger::multiply);
}
```
Summary of Common Functional Interfaces
Java defines many types of functional interfaces

<table>
<thead>
<tr>
<th>Interface Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
</tr>
<tr>
<td>BiConsumer&lt;T,U&gt;</td>
</tr>
<tr>
<td>BiFunction&lt;T,U,R&gt;</td>
</tr>
<tr>
<td>BinaryOperator&lt;T&gt;</td>
</tr>
<tr>
<td>BiPredicate&lt;T,U&gt;</td>
</tr>
<tr>
<td>BooleanSupplier</td>
</tr>
<tr>
<td>Consumer&lt;T&gt;</td>
</tr>
<tr>
<td>DoubleBinaryOperator</td>
</tr>
<tr>
<td>DoubleConsumer</td>
</tr>
<tr>
<td>DoubleFunction&lt;R&gt;</td>
</tr>
<tr>
<td>DoublePredicate</td>
</tr>
<tr>
<td>DoubleSupplier</td>
</tr>
<tr>
<td>DoubleToDoubleFunction</td>
</tr>
<tr>
<td>DoubleToLongFunction</td>
</tr>
<tr>
<td>DoubleUnaryOperator</td>
</tr>
<tr>
<td>Function&lt;T,R&gt;</td>
</tr>
</tbody>
</table>

See [docs.oracle.com/javase/8/docs/api/java/util/function/package-summary.html](docs.oracle.com/javase/8/docs/api/java/util/function/package-summary.html)
Java defines many types of functional interfaces

Some of these interfaces handle reference types

See www.oreilly.com/library/view/java-8-pocket/9781491901083/ch04.html
Java defines many types of functional interfaces

- Some of these interfaces handle reference types
- Other interfaces support primitive types

See [docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html](docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html)
Java defines many types of functional interfaces

- Some of these interfaces handle reference types
- Other interfaces support primitive types
- Avoids “auto-boxing” overhead

### Summary of Common Functional Interfaces

Java defines many types of functional interfaces

- Some of these interfaces handle reference types
- Other interfaces support primitive types.
- There’s an explosion of Java functional interfaces!
Java defines many types of functional interfaces

- Some of these interfaces handle reference types
- Other interfaces support primitive types.
- There’s an explosion of Java functional interfaces!
- However, learn these interfaces before trying to customize your own
Summary of Common Functional Interfaces

- Java defines many types of functional interfaces.
  - Some of these interfaces handle reference types.
  - Other interfaces support primitive types.
  - There’s an explosion of Java functional interfaces!

*We focus on the most common types of functional interfaces*
Summary of Common Functional Interfaces

- Java defines many types of functional interfaces.
  - Some of these interfaces handle reference types.
  - Other interfaces support primitive types.
  - There’s an explosion of Java functional interfaces!

All usages of functional interfaces in the upcoming examples are “stateless”!
End of Understand Java

Functional Interfaces:
Overview