ORACLE

Lesson 1-4: Functional Interfaces in the java.util.function Package

java.util.function Package

- Well defined set of general purpose functional interfaces
  - All have only one abstract method
  - Lambda expressions can be used wherever these types are referenced
  - Used extensively in the Java class libraries
  - Especially with the Stream API
  - This section covers the generic interfaces
    - There are numerous type specific versions
    - `Double`, `Int`, `Long` and `Obj` forms
    - Review the package for more detail

Consumer<T>

Operation That Takes a Single Value And Returns No Result

- Also, `BiConsumer<T,U>` that takes two values and returns no result
- Default method available for composing functions
  - `andThen(Consumer after)`

```
String s -> System.out.println(s)

(k, v) -> System.out.println("key:" + k + ", value:" + v)
```

Supplier

A Supplier Of Results

- The opposite of a `Consumer`

```
() -> createLogMessage()
```

Function<T,R>

A Function That Accepts One Argument And Returns A Result

- Type of argument and result may be different
- Also, `BiFunction<T,U,R>` that accepts two arguments and returns a result
- Useful default methods for composing
  - `compose`, `andThen`

```
Student s -> s.getName()

(String name, Student s) -> new Teacher(name, s)
```

UnaryOperator<T>

- Specialised form of `Function`
- Single argument and result of the same type
  - `T apply(T a)`

```
String s -> s.toLowerCase()
```

BinaryOperator<T>

- Specialised form of `BiFunction`
- Two arguments and a result, all of the same type
  - `T apply(T a, T b)`

```
(String x, String y) -> {
  if (x.length() > y.length())
    return x;
  return y;
}
```

Predicate

A Boolean Valued Function Of One Argument

- Also, `BiPredicate`, that takes two arguments
- Has useful default and static methods for combination
  - `and()`, `or()`, `negate()`, `isEqual()`

```
Student s -> s.graduationYear() == 2011
```

```
Files.find(start, maxDepth,
  (path, attr) -> String.valueOf(path).endsWith(".js") &&
    attr.size() > 1024,
  FileVisitOption.FOLLOW_LINKS);
```

Section 4

Summary

- The function package provides a range of functional interfaces
- Used extensively in Streams
  - But also used in many other places
- Unlikely you will need to define your own extensions to the function package


ORACLE®