The Java Consumer Functional Interface

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand the Consumer functional interface in Java & recognize how it can be used in conjunction with lambda expressions & method references

Interface Consumer<T>

Type Parameters:

`T - the type of the input to the operation`

All Known Subinterfaces:

`Stream.Builder<T>`

Functional Interface:

This is a functional interface and can therefore be used as the assignment target for a lambda expression or method reference.

See docs.oracle.com/javase/8/docs/api/java/util/function/Consumer.html

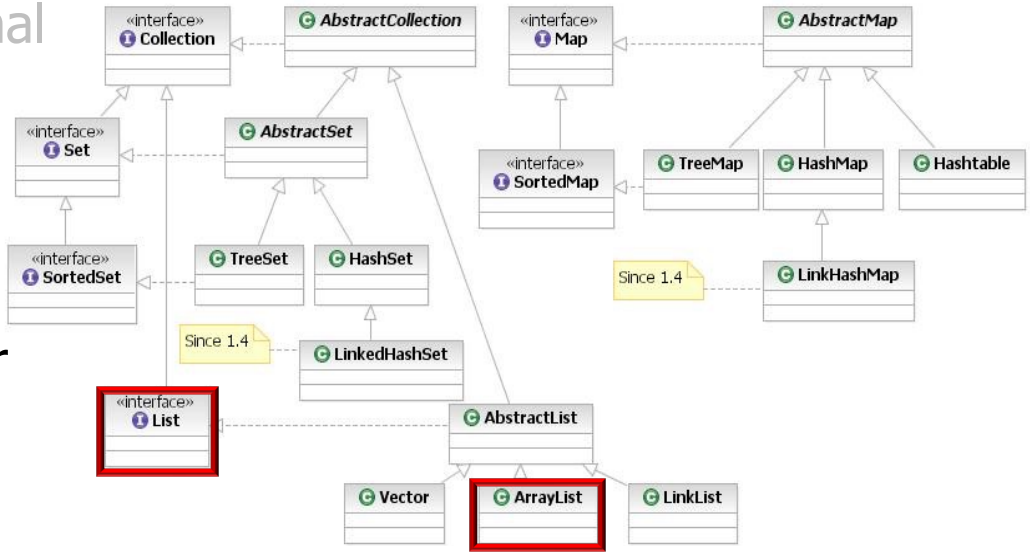
Learning Objectives in this Part of the Lesson

- Understand the Consumer functional interface in Java & recognize how it can be used in conjunction with lambda expressions & method references
- Know how to apply Java Consumer in a concise example

See github.com/douglascraigschmidt/ModernJava/tree/main/FP/ex14

Learning Objectives in this Part of the Lesson

- Understand the Consumer functional interface in Java & recognize how it can be used in conjunction with lambda expressions & method references
- Know how to apply Java Consumer in a concise example
  - This example shows the List interface & ArrayList class in the Java collection framework



See docs.oracle.com/javase/8/docs/technotes/guides/collections

Overview of the Consumer Functional Interface

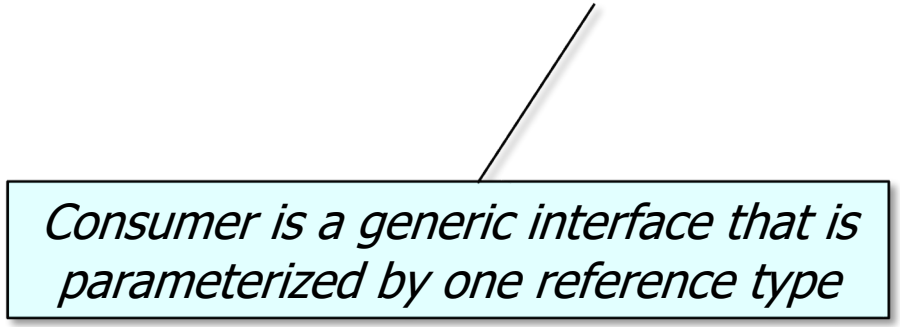
Overview of the Consumer Functional Interface

- A *Consumer* accepts a parameter & returns no results, e.g.,
  - `public interface Consumer<T> { void accept(T t); }`

See docs.oracle.com/javase/8/docs/api/java/util/function/Consumer.html

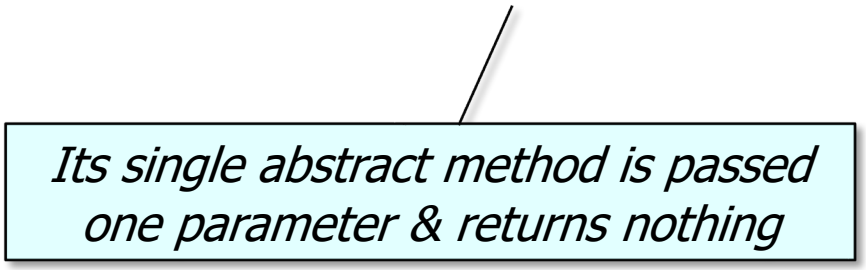
Overview of the Consumer Functional Interface

- A *Consumer* accepts a parameter & returns no results, e.g.,
  - `public interface Consumer<T> { void accept(T t); }`



Overview of the Consumer Functional Interface

- A *Consumer* accepts a parameter & returns no results, e.g.,
  - `public interface Consumer<T> { void accept(T t); }`



Applying the Consumer Functional Interface

Applying the Consumer Functional Interface

- This example shows how a modern Java Consumer interface can be used with the forEach() method to print out the values in a List

```
List<Thread> threads = Arrays.asList(new Thread("Larry"),
                                     new Thread("Curly"),
                                     new Thread("Moe"));
```

Create a list of threads with names of the three stooges

```
threads.forEach(System.out::println);
threads.sort(Comparator.comparing(Thread::getName));
threads.forEach(System.out::println);
```

See github.com/douglascraigschmidt/ModernJava/tree/main/FP/ex14

Applying the Consumer Functional Interface

- This example shows how a modern Java Consumer interface can be used with the forEach() method to print out the values in a List

```
List<Thread> threads = Arrays.asList(new Thread("Larry"),
                                     new Thread("Curly"),
                                     new Thread("Moe"));
```

Print out threads using forEach()

```
threads.forEach(System.out::println);
threads.sort(Comparator.comparing(Thread::getName));
threads.forEach(System.out::println);
```

See docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html#forEach

How Iterable Uses the Consumer Functional Interface

How Iterable Uses the Consumer Functional Interface

- Here's how the Iterable forEach() method uses the Consumer parameter passed to it

```
public interface Iterable<T> {
  ...
  default void forEach(Consumer<? super T> action) {
    for (T t : this) {
      action.accept(t);
    }
  }
```

See docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html#forEach

How Iterable Uses the Consumer Functional Interface

- Here's how the Iterable forEach() method uses the Consumer parameter passed to it

```
public interface Iterable<T> {
  ...
  default void forEach(Consumer<? super T> action) {
    for (T t : this) {
      action.accept(t);
    }
  }
```

System.out::println

The consumer parameter is bound to the System.out::println method reference

How Iterable Uses the Consumer Functional Interface

- Here's how the Iterable forEach() method uses the Consumer parameter passed to it

```
public interface Iterable<T> {
  ...
  default void forEach(Consumer<? super T> action) {
    for (T t : this) {
      action.accept(t);
    }
  }
```

```
System.out.println(t)
```

The accept() method is replaced by the call to System.out.println()

How Iterable Uses the Consumer Functional Interface

- Here's how the Iterable forEach() method uses the Consumer parameter passed to it

```
public interface Iterable<T> {
  ...
  default void forEach(Consumer<? super T> action) {
    for (T t : this) {
      action.accept(t);
    }
  }
```

This use of "this" triggers the creation of the iterator associated with the subclass!!

See docs.oracle.com/javase/8/docs/technotes/guides/language/foreach.html

End of the Consumer Java Functional Interface