ORACLE

# Lesson 1-7:
# Useful New Methods In JDK 8 That Can Use Lambdas

# Iterable Interface

- `Iterable.forEach(Consumer c)`

```
List<String> myList = ...
myList.forEach(s -> System.out.println(s));
```

```
myList.forEach(System.out::println);
```

# Collection Interface

- `Collection.removeIf(Predicate p)`

```
List<String> myList = ...
myList.removeIf(s -> s.length() == 0);
```

# List Interface

- List.replaceAll(UnaryOperator o)

```
List<String> myList = ...
myList.replaceAll(s -> s.toUpperCase());
```

```
myList.replaceAll(String::toUpperCase);
```

# List Interface

- `List.sort(Comparator c)`
- Replaces `Collections.sort(List l, Comparator c)`

```
List<String> myList = ...
myList.sort((x, y) -> x.length() – y.length());
```

# Logger Class

- This is a common problem
  - `logger.finest(createComplexMessage());`
- `createComplexMessage()` is always called, even when not required
  - Heisenberg's Uncertainty Principle in software
- New methods in `Logger` class
  - Takes a `Supplier` as an argument (which is a functional interface)
- Simple change to code has big impact on performance
  - `logger.finest(() -> createComplexMessage());`
- We now pass *how* to create the message, not the actual message



6

# Section 8

## Summary

- Use the new methods in JDK 8 to eliminate the frequent need for loops
- Remember that a Lambda provides behaviour, not a value
  - Very useful for conditional uses of data

# Lesson 1: Summary

# Summary

- Lambda expressions provide a simple way to pass behaviour as a parameter, or assign to a variable
- They can be used wherever a functional interface type is used
  - The Lambda provides the implementation of the single abstract method
- Method and constructor references can be used as shorthand
- Several useful new methods in JDK 8 that can use Lambdas


ORACLE®