Java

Lambda Expressions and Functional Programming

Dalton State College

T. Gonzalez

Lambda Expressions

A lambda expression is an anonymous method, that is, a method without a name.

Lambda Expression Syntax

A lambda expression consists of:

1. A comma-separated list of formal parameters enclosed in parentheses.
   - The types of the parameters can be omitted.
   - The parentheses can be omitted if there is only one parameter.

2. The arrrow token. `->`

3. A body, which consists of a single expression or a statement block.
   - If the body consists a single expression, the expression is evaluated and the result is returned.
   - If the body uses statements, then the statements must be enclosed in curly braces.
   - Method calls do not have to be enclosed in braces.

Lambda Expression Examples

```
n -> 3 * n + 2

(x, y) -> x > y

z -> System.out.println(z)

x -> {if(1 < x && x < 5)
          return true;
      else if(x == 2)
          return false;
      else
          return true;}
```

Using Lambda Expressions to Process Lists and Maps

Lists such as an `ArrayLists` can be used processed using lambda expressions.

Lambda expressions can be used in conjunction with the following methods:

- `replaceAll()`
- `removeIf()`
- `forEach()`

The `replaceAll()` Method

Given a `List` named `list` and a lambda expression `lambda` the statement `list.replaceAll(lambda);` calls the lambda expression once for every item in the list and then stores the result back in the list.

The value returned by the lambda expression must be the same type as the type of the list.

See ReplaceAllDemo.java.

The `removeIf()` Method

Given a `List` named `list` and a lambda expression `lambda` the statement `list.removeIf(lambda);` calls the lambda expression once for every item in the list and then removes the item when the result of the lambda expression is **`true`**.

See RemoveIfDemo.java.

Method References

Method references are compact, easy-to-read lambda expressions for methods that already have a name.

Lambda expression: `x -> System.out.println(x)`

Method reference: `System.out::println`

There are four types of method references:

Reference to a static method	ContainingClass::staticMethodName
Reference to an instance method of a particular object	containingObject::instanceMethodName
Reference to an instance method of an arbitrary object of a particular type	ContainingType::methodName
Reference to a constructor	ClassName::new

The `forEach()` Method

Given a `List` named `list` and a lambda expression `lambda` the statement `list.forEach(lambda);` calls the lambda expression once for every item in the list.

See ForEachDemo.java.

In-Class Problem

Write a program that declares and initializes an `List` of `Integers` with at least five elements. Use a lambda expression to replace all of the elements in the list with the length of the list.

In-Class Problem

Write a program that declares and initializes a `List` of `Integers`. Use a lambda expression to remove all odd multiples of three from the list.

In-Class Problem

Write a program that declares and initializes a `List` of `OrderedPair` objects. Use a lambda expression to call the `updateY()` method in each of the objects in the list.

In-Class Problem

Write a program that reads the data from the file name_company_data.csv into a `HashMap`. Use lambda expressions to do the following in order:

- Remove all information of people who have a lowercase a or e in their name.
- Capitalize the following letters in company names: i, o, u, and y.
- Print the resulting map.