Objective

Optional Lecture:

Lambda

- Functional Interface

Lamda

Functional interface:

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

Example: Runnable Interface only contains run() method.

Refer to Example0.java

Lambda Expression:

Used to create anonymous classes with one method.

This feature added from Java 8

Functional Interface

```
interface NumericTest {
  boolean computeTest(int n);
}
```

Is a functional interface since it has one and only one abstract method.

Another example:

```
interface ProcessString{
  String process(String str);
}
```

Lambda

Lambda Arrow Operator

```
(argument) -> body
```

Lambda Arrow Operator

```
(n) -> n+2
(str)-> System.out.println(str);
()-> System.out.println(“Hello”);
```

Left side: The parameters required by the expression. It can also be empty.

Right side: Body of the lambda

Lambda_example1.java

```
public class Example1{
  public static void main(String args[]) {
    NumericTest isNegative = (n) -> (n < 0);
    // Output: true
    System.out.println(isNegative.computeTest(-5));

    NumericTest isEven = (n) -> (n % 2) == 0;
    // Output: false
    System.out.println(isEven.computeTest(5));
  }
}
//----
interface NumericTest {
  // Takes an integer and returns a boolean
  boolean computeTest(int n);
}
```

Refer to Example1.java

Lambda_example2.java

```
public class Example2{
  public static void main(String args[]) {
    ProcessString x =(str)->"Good Morning "+str+ "!";
    ProcessString y =(str)-> "Len: " + str.length();

    System.out.println(x.process("Gary"));
    System.out.println(y.process("Gary"));
  }
}
//----
interface ProcessString{
    String process(String str);
}
```

Refer to Example2.java

Block Lambda Expression

Lambda expression can consist of multiple line of codes, then we should use {}; to enclose the expression. Note to the semi-colon after the closing curly brace.

```
(str) -> {
  String msg = "Good Morning ";
  msg += str;
  msg += str.length();
  return msg;
};
```

Lambda_example3.java

```
public class Example3{
  public static void main (String args[]) {
    ProcessString reverse = (str) -> {
      String res = "";
      for(int i = str.length()-1; i >= 0; i--)
        res += str.charAt(i);
      return res;
    };
    String res= reverse.process("CPSC1181_002"));
    System.out.println(res);
  }
}
//----
interface ProcessString{
      String process(String str);
}
```

Refer to Example3.java

Lambda_example3.java

```
public class Example3{
  public static void main (String args[]) {
    ProcessString reverse = (str) -> {
      String res = "";
      for(int i = str.length()-1; i >= 0; i--)
        res += str.charAt(i);
      return res;
    };
    String res=  reverse.process("CPSC1181_002"));
    System.out.println(res);
  }
}
//----
interface ProcessString{
     String process(String str);
}
```

Refer to Example3.java

Generic Functional Interfaces

A lambda expression cannot be generic.

But the functional interface associated with a lambda expression can.

Refer to this example after we covered Generic Classes.

Refer to Example4.java

Lambda Expressions as arguments

Pass Lambda expression them as arguments.

To pass lambda expressions as parameters, just make sure the functional interface type is compatible with the required parameter.

Refer to Example5.java

Lambda Shortcut for Comparator

```
Comparator<Member> sCom = (m1,m2) ->
           m1.getName().compareTo(m2.getName());
Collections.sort(list, sCom);
```

Instead of

```
class mySort implements Comparator<Member>{
  public int compare(Member m1, Member m2){
    return m1.getName().compareTo(m2.getName());
  }
}
Collections.sort(list, new mySort());
```

Refer to TheComparatorTest.java

Lambda Shortcut for EventHandler

Lambda:

```
decButton.setOnAction(e -> {
   System.out.println("handled by Anonymous class");
});
```

Anonymous class:

```
decButton.setOnAction(new EventHandler<ActionEvent>() {
   @Override
   public void handle(ActionEvent e) {
      System.out.println("handled by Anonymous class");
   }
});
```

Refer to Exmaple6

More Tutorials:

Refer to the following link for more example of event handling using inner class, anonymous inner class, and Lambda expression.

https://taylorial.com/cs1021/EventHandlingFX.htm