CSE 1325 - Object-Oriented Programming

Lambda Expressions

Alex Dillhoff

University of Texas at Arlington

Lambda Expressions

A **lambda expression**, or **closure**, is an anonymous method.

That is, a function without a name.

Lambda Expressions

Consider the method defined below:

```
boolean thisIsTrue() {
    return true;
}
```

Lambda Expressions

An equivalent lambda expression would be written as

```
() -> true
```

Lambda Expressions

Lambda expressions can take on parameters as well. The following expression determines if the input is odd.

```
(num) -> num % 2 == 1
```

Lambda Expressions

Consider the sorting examples using interfaces.

Instead of implementing the `Comparable` interface or creating an object directly, we can pass a lambda expression instead.

Lambda Expressions

Example: `LambdaSortExample.java`

Functional Interfaces

Lambda expressions implement methods defined by a **functional interface**.

A functional interface is one that contains only a single `abstract` method.

An example of this is the `Runnable` interface.

Functional Interfaces

Consider the following interface.

```
interface OddInterface {
    boolean isOdd(int num);
}
```

Functional Interfaces

A reference to this interface can be created and assigned using a lambda expression.

```
OddInterface oddInterface = (num) -> num % 2 == 1;
```

Functional Interfaces

The method assigned by the lambda expression can be evaluated as follows:

```
oddInterface.isOdd(13);
```

Block Lambdas

Lambdas can also contain multiple lines of code. In this case they are called **block lambdas**.

We will use block lambdas in some of the GUI examples.

Block Lambdas

Example: Starting the controller on the event thread.

```
SwingUtilities.invokeLater(() -> {
    new MyGUIController();
});
```

Lambdas and Exceptions

Lambda expressions can throw exceptions.

If they are *checked* exceptions, the exception type must be specified in the function interface that the lambda is implementing.

Scope of Lambdas

Lambda expressions can read variables outside of their scope, but they cannot modify them.

The next slide shows an example of both a valid and invalid access to a variable.

Scope of Lambdas

```
int numPlayers = 2;

PlayerInterface p = (num) -> {
    // This is valid
    num = numPlayers;

    // This is not
    numPlayers++;
};
```

Method References

It is convenient to pass a reference to a method (akin to a function pointer in C).

This can also be seen in examples using calls that take functional interfaces.

Method References

Taking the previous example...

```
SwingUtilities.invokeLater(() -> {
    new MyGUIController();
});
```

Method References

We can simplify this with a method reference to the constructor...

```
SwingUtilities.invokeLater(MyGUIController::new);
```