Mobile Application Development (Android)
Java 8 Lambda Expressions

Waterford Institute of Technology

October 2, 2016

John Fitzgerald

Java 7 Interface

Definition

One or more abstract method declarations

- All methods are public by default.
- Unnecessary to add **abstract** or **public** modifiers.

```
public interface TextWatcher {
  // abstract method declaration
  void afterTextChanged();
}
```

Java 7 Interface

Use

```
public class ResidenceActivity implements TextWatcher {
  // Implement abstract method declared in interface
  @Override
  void afterTextChanged() {
    System.out.println("We're watching you");
  }
}
```

Java 8 Interface

Additional features

Constants

```
public interface TextWatcher {
   // const declaration
   int id = 1;

   // abstract method declaration
   void afterTextChanged();
}
```

Java 8 Interface

Additional features

Static methods

```
public interface TextWatcher {
  // const declaration
  int id = 1;

  // static method
  static long beforeTextChanged() {
    return new Date().getTime();
  }

  // abstract method declaration
  void afterTextChanged();
}
```

Java 8 Interface

Additional features

Default methods

```
public interface TextWatcher {
  // const declaration
  int id = 1;

  // static method
  static long beforeTextChanged() {
    return new Date().getTime();
  }

   // default method
  default void onTextChanged() {
    System.out.println("I hate change");
  }

  // abstract method declaration
  void afterTextChanged();
}
```

Java 8 Interface

Option to override default method

```
public class ResidenceActivity implements TextWatcher {
  // Optional: Implement default method
  @Override
  public void onTextChanged() {
    System.out.println("Adapt or die");
  }

  // Implement abstract method
  @Override
  void afterTextChanged() {
    System.out.println("We're watching you");
  }
}
```

Java 8 Interface

Overriding and hiding methods

Consider these 2 interfaces:

```
interface Top {
    default String name() { return "unnamed"; }
}
interface Left extends Top {
    default String name() { return getClass().getName(); }
}
```

Java 8 Interface

Overriding and hiding methods

Class may inherit both interfaces.

- Duplicate method names triggers error.
- Resolved by overriding Left.name() or Top.name().

```
public class MultipleInheritance implements Left, Top {

}
```

Java 8 Interface

Overriding and hiding methods

Class inherits both.

- Duplicate method names triggers error.
- Resolved by overriding Left.name() or Top.name().

```
public class MultipleInheritance implements Left, Top {

  @Override
  public String name() {
    // TODO Auto-generated method stub
    return Left.super.name();
  }
}
```

Java 8 Interface

Anonymous inner class

- A synthetic class created.
- Implements TextWatcher interface.
- Implements interface methods.
- Instantiates object.
- Passes object reference as parameter.
- This satisfies parameter type specfication.

```
public ResidenceActivity() {
  textView.addTextChangedListener(new TextWatcher() {
    @Override
    public void afterTextChanged() {
      System.out.println("Text changed");
    }
  });
}
```

Java 8 Interface

Functional interface

- An interface containing a single abstract method.
- May contain static or default methods.
- May contain constants.
- May contain overridden Object methods.

```
interface AFunctionalInterface {
  int compare(String o1, String o2);
}

interface AnotherFunctionalInterface{
  boolean equals(Object obj);
  int compare(String o1, String o2);
}
```

Java 8 Interface

Functional interface

Is TextWatcher a functional interface?

```
public interface TextWatcher {

  // const declaration
  int id = 1;

  // static method
  static long beforeTextChanged() {
    return new Date().getTime();
  }

  // abstract method
  void afterTextChanged();

  // default method
  default void onTextChanged() {
    System.out.println("I hate change");
  }
}
```

Java 8 Interface

Functional interface

Is Android's TextWatcher a functional interface?

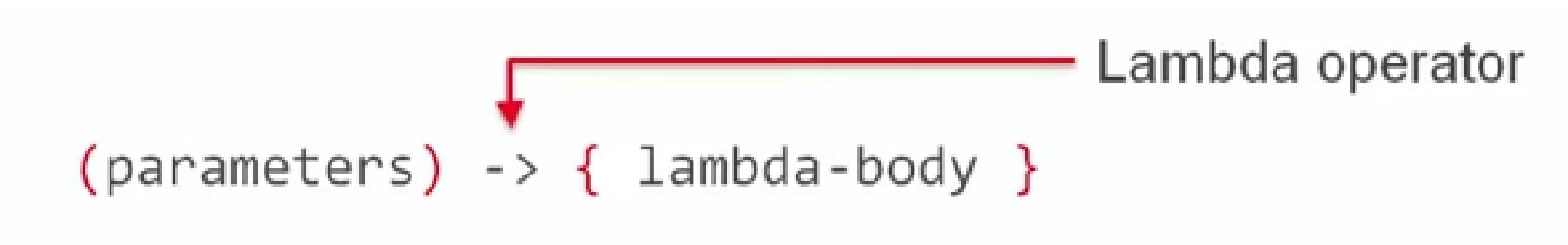
```
public interface TextWatcher {

  void afterTextChanged(Editable s);

  void beforeTextChanged(CharSequence s, int start, int count, int after);

  void onTextChanged(CharSequence s, int start, int before, int count);

}
```

Java 8 Lambdas

Anonymous functions

Lambda:

- Block of code capable of capable of expressing behaviour.
- Not associated with any class.
- Does not have a name.
- May be thought of as anonymous function.
- An alternative to an anonymous inner class.



Java 8 Lambdas

Anonymous functions

Java method

```
int add(int x, int y) {
  return x + y;
}
```

Behaviourally similar lambda (but not directly substitutable).

```
(int x, int y) -> {return x + y;}
```

Java 8 Lambdas

Anonymous functions

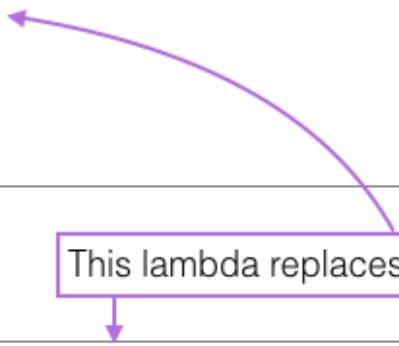
Anonymous inner class

```
    textView.addTextChangedListener(new TextWatcher() {

      @Override
      public void afterTextChanged() {
        System.out.println("Text changed");
      }
    });
  }
}
```

Replaced with lambda expression

```
textView.addTextChangedListener(() -> System.out.println("Text changed"));
```

```
textView.addTextChangedListener(new TextWatcher() {
    @Override
    public void afterTextChanged() {
      System.out.println("Text changed");
    }
  });
```



This lambda replaces this anonymous class

```
textView.addTextChangedListener(() -> System.out.println("Text changed"));
```

addTextChangedListener expects a **TextWatcher** type that implements a single abstract method that takes no parameters and does not return a value. The applied lambda satisfies these requirements.

Referenced Material

1. Oracle Chapter 9 Interfaces

`http://docs.oracle.com/javase/specs/jls/se8/html/jls-9.html#jls-9.3`

[Accessed 2016-09-28]

2. Oracle Overriding and Hiding Methods

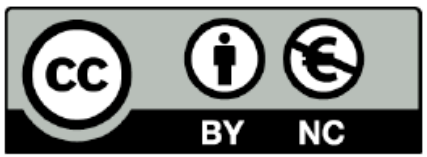
`https://docs.oracle.com/javase/tutorial/java/IandI/override.html`

[Accessed 2016-09-28]

3. Oracle Multiple Inheritance of State, Implementation and Type

`https://docs.oracle.com/javase/tutorial/java/IandI/multipleinheritance.html`

[Accessed 2016-09-28]



Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit