Lambda Expressions

CPSC 1181 – O.O.

Jeremy Hilliker

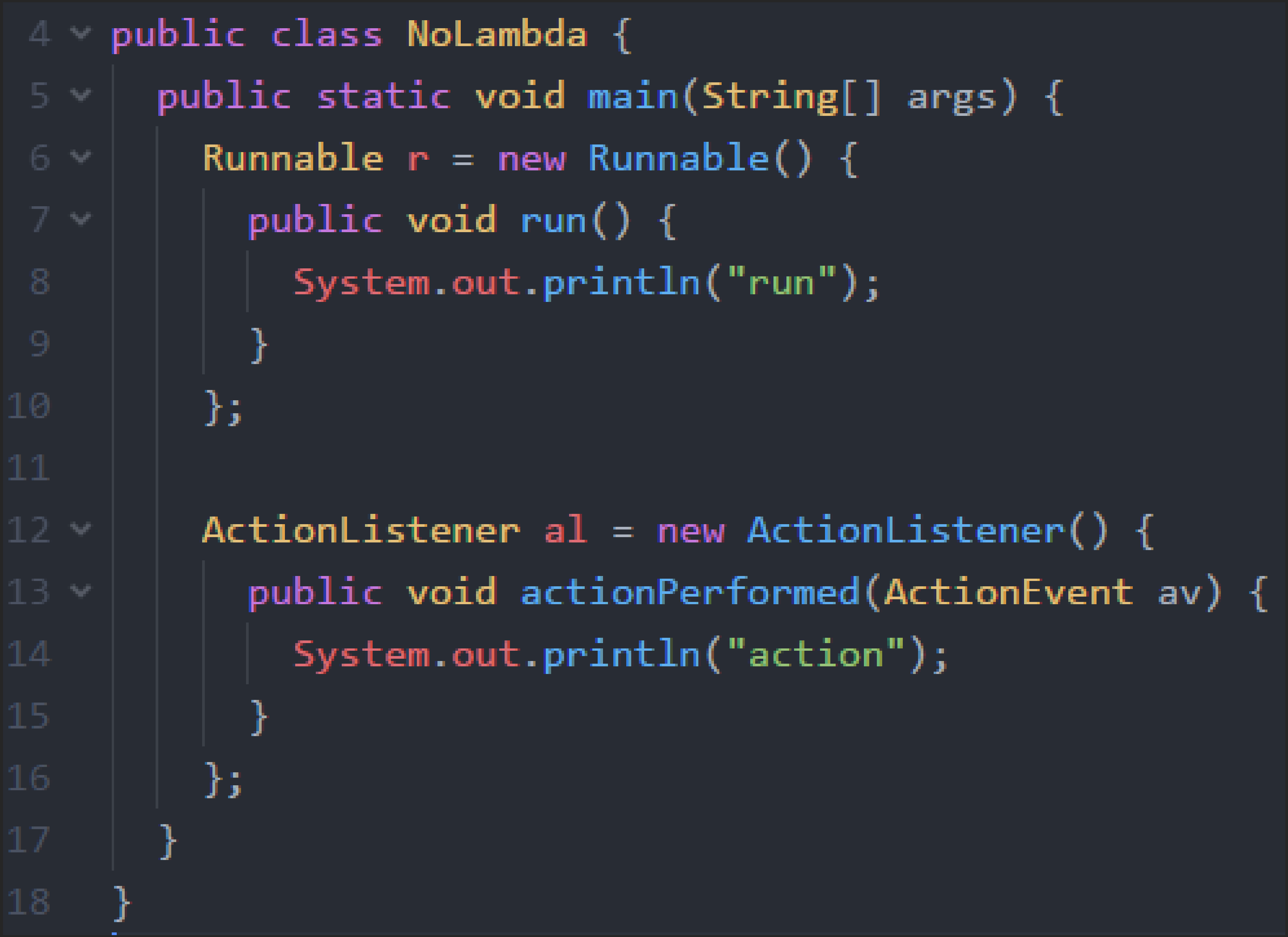
Summer 2017

Langara.

THE COLLEGE OF HIGHER LEARNING.

Problem

- Implementing an interface for a single method is clunky
- Lots of syntax, for something that is supposed to be easy & fast

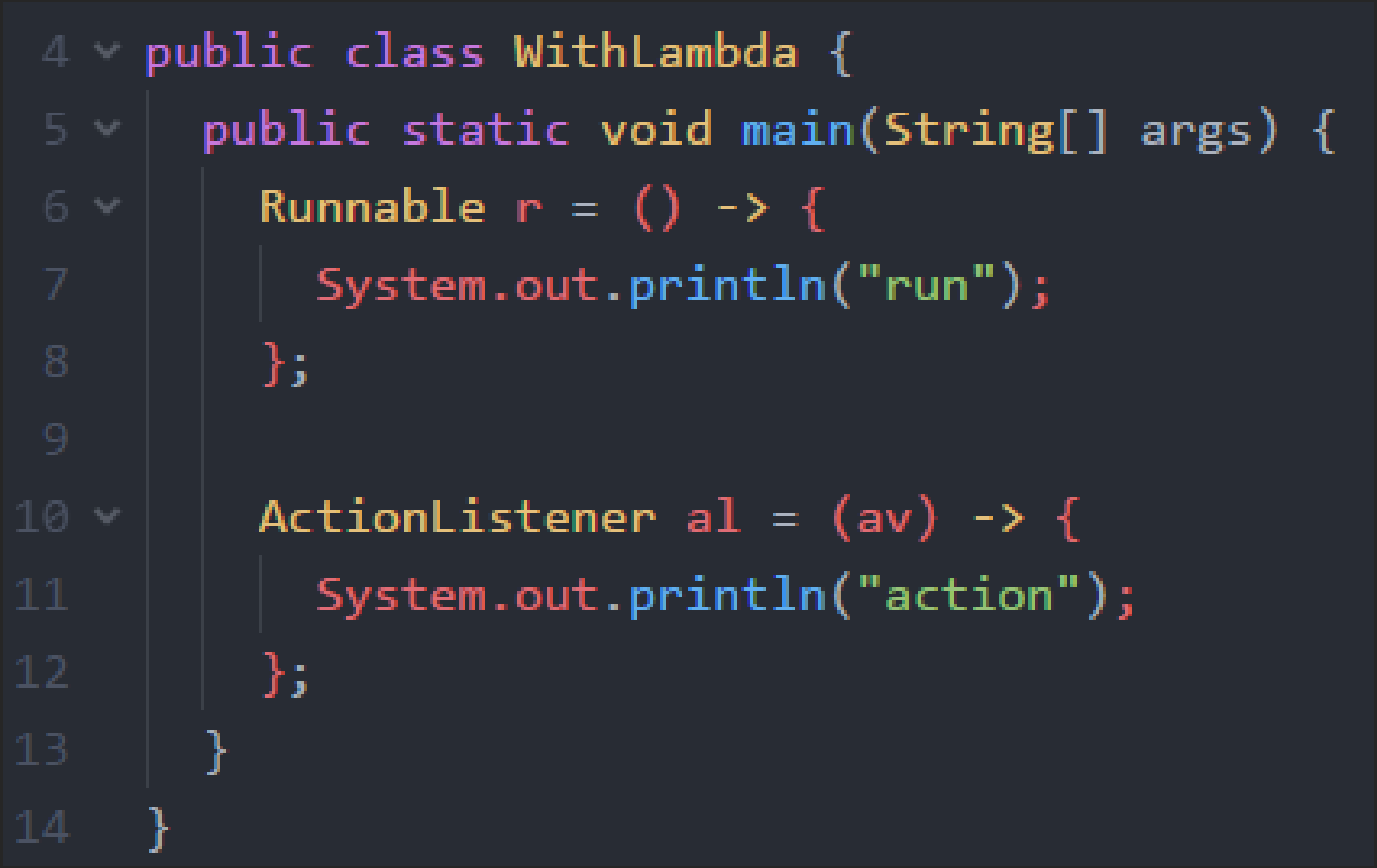


```java
public class NoLambda {
  public static void main(String[] args) {
    Runnable r = new Runnable() {
      public void run() {
        System.out.println("run");
      }
    };

    ActionListener al = new ActionListener() {
      public void actionPerformed(ActionEvent av) {
        System.out.println("action");
      }
    };
  }
}
```

Idea

- Introduce a language construct that allows you to implement a single method with little syntax
- In other languages: a ***lambda expression***

- ***Def'n: lambda expression***
  - An *expression* which evaluates to a *method* with *no identifier*



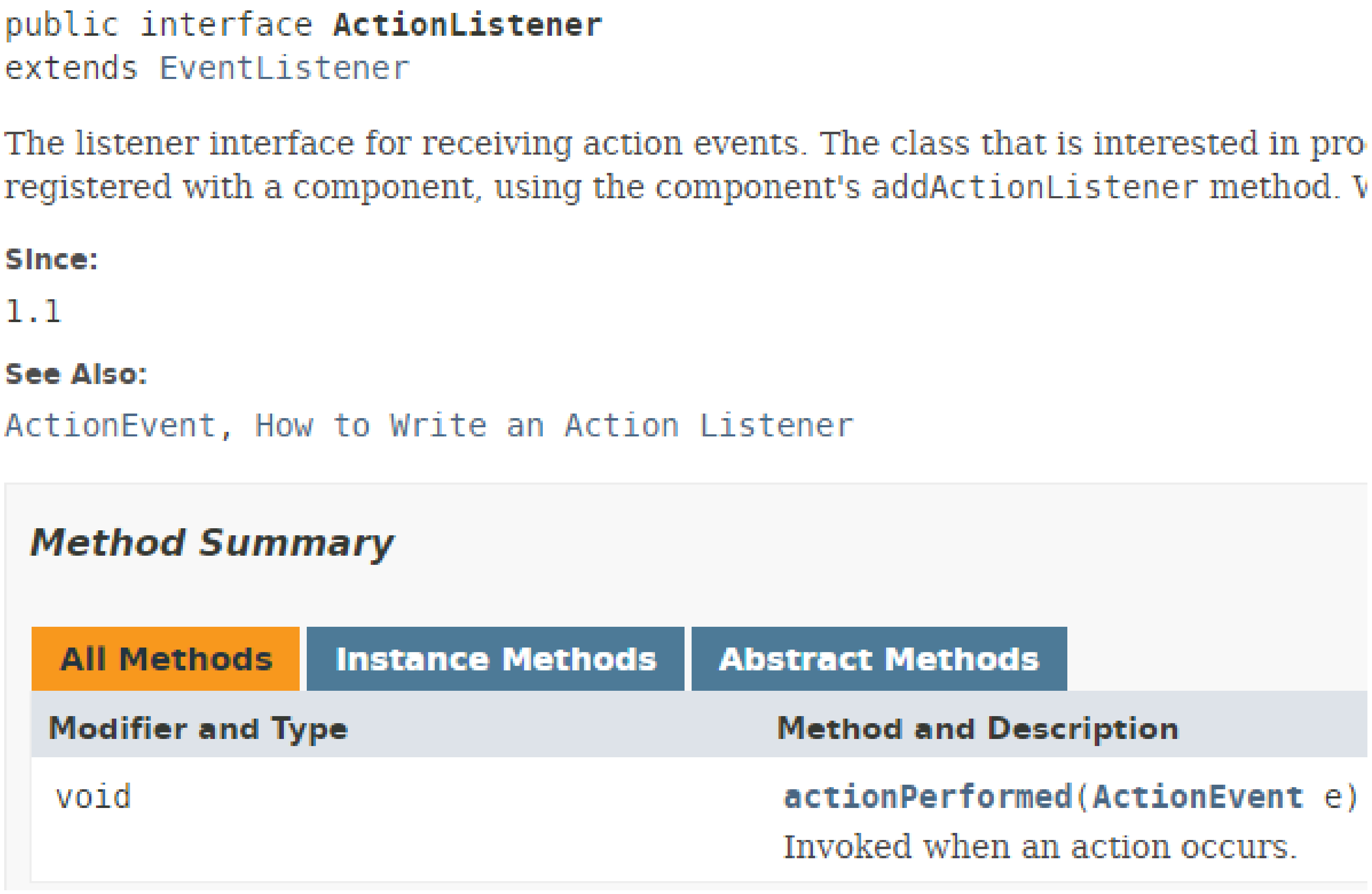
```java
public class WithLambda {
    public static void main(String[] args) {
        Runnable r = () -> {
            System.out.println("run");
        };

        ActionListener al = (av) -> {
            System.out.println("action");
        };
    }
}
```

How

- Instead of
  - declaring a class (inner or anonymous)
  - Implementing the method (with full signature)
- We just directly implement the method
- All types get inferred
  - Reasonably safe since there is only one method
  - Could be a problem is the thing we pass it to is overloaded

Where

- Can do this anywhere that gets assigned a "*functional interface*" (via substitution principle)
- ***Def'n: functional interface***
  - An interface with only one method
    - Runnable
    - Callable
    - ActionListener
    - All kinds of others
- Allows us to build very generic methods that are easy to use
  - they accept functional interfaces as arguments



```
public interface ActionListener
extends EventListener
```

The listener interface for receiving action events. The class that is interested in pro
registered with a component, using the component's `addActionListener` method. V

Since:

1.1

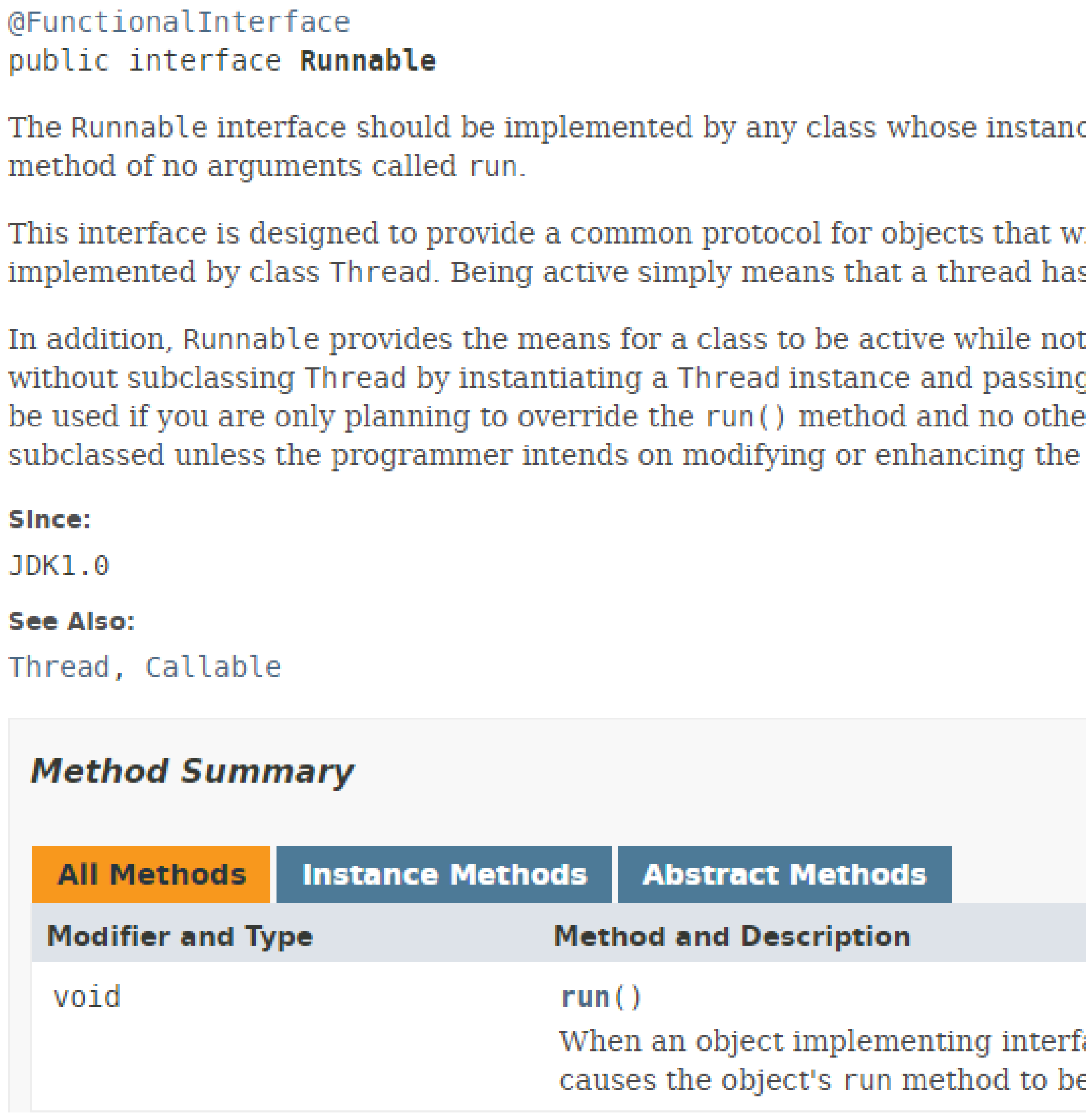
See Also:

ActionEvent, How to Write an Action Listener

Method Summary

All Methods	Instance Methods	Abstract Methods

Modifier and Type	Method and Description
`void`	**`actionPerformed(ActionEvent e)`** Invoked when an action occurs.



```
@FunctionalInterface
public interface Runnable
```

The `Runnable` interface should be implemented by any class whose instanc
method of no arguments called `run`.

This interface is designed to provide a common protocol for objects that w
implemented by class `Thread`. Being active simply means that a thread has

In addition, `Runnable` provides the means for a class to be active while not
without subclassing `Thread` by instantiating a `Thread` instance and passing
be used if you are only planning to override the `run()` method and no othe
subclassed unless the programmer intends on modifying or enhancing the

Since:

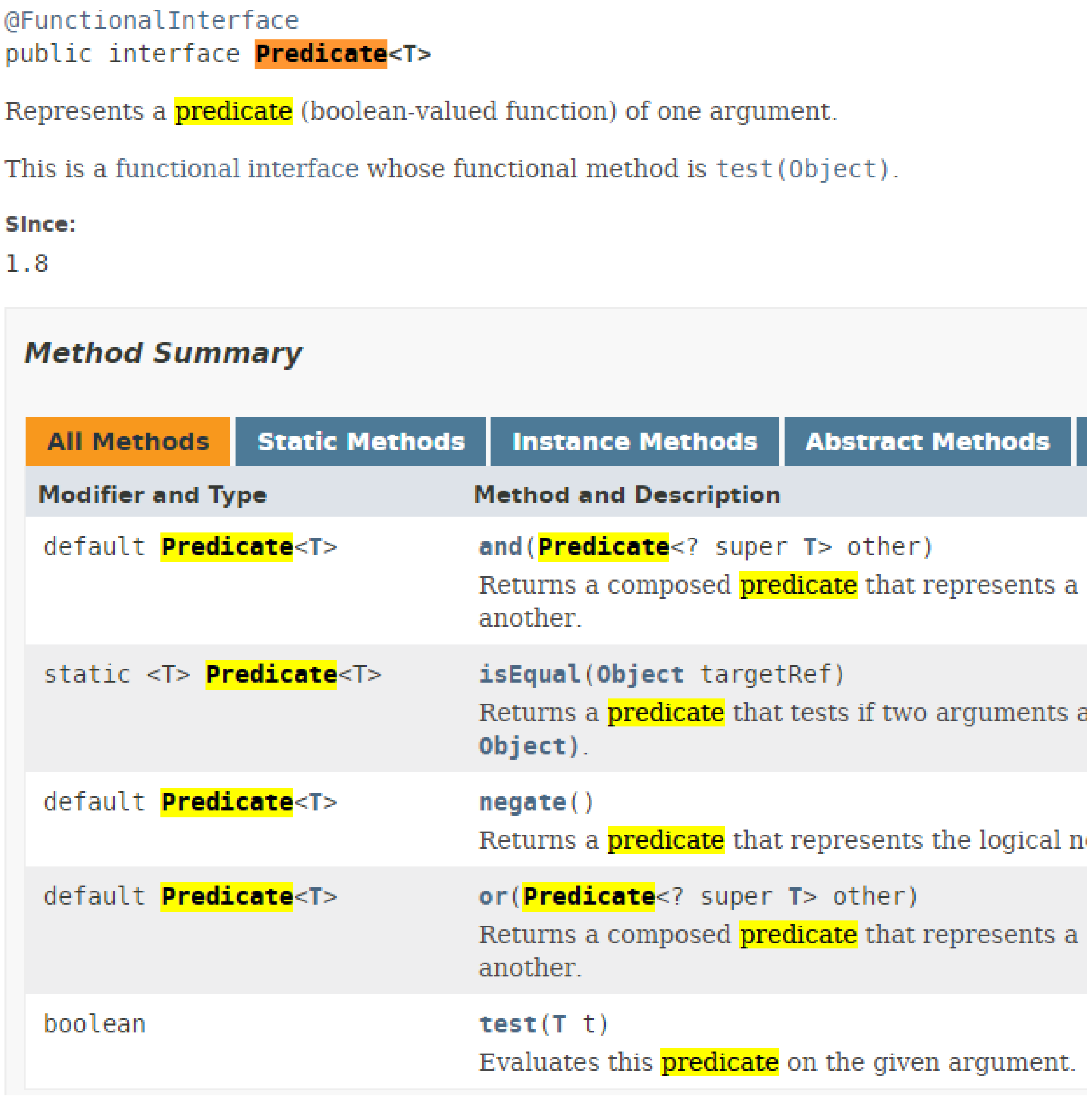
JDK1.0

See Also:

`Thread`, `Callable`

Method Summary

Modifier and Type	Method and Description
`void`	**`run()`** When an object implementing interf causes the object's `run` method to be



```
@FunctionalInterface
public interface Predicate<T>
```

Represents a predicate (boolean-valued function) of one argument.

This is a functional interface whose functional method is `test(Object)`.

Since:

1.8

Method Summary

All Methods | Static Methods | Instance Methods | Abstract Methods

Modifier and Type	Method and Description
`default Predicate<T>`	`and(Predicate<? super T> other)` Returns a composed predicate that represents a another.
`static <T> Predicate<T>`	`isEqual(Object targetRef)` Returns a predicate that tests if two arguments a `Object)`.
`default Predicate<T>`	`negate()` Returns a predicate that represents the logical n
`default Predicate<T>`	`or(Predicate<? super T> other)` Returns a composed predicate that represents a another.
`boolean`	`test(T t)` Evaluates this predicate on the given argument.

Syntax

- `Predicate.test(T t)`
  - `(t) -> {impl}`

- `Public void eg(Predicate p)`
  - `o.eg(new Predicate() {`
    `public void test(T t) { impl }});`
  - `o.eg((t) -> {impl});`