Understand Other Properties of Java Functional Interfaces

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

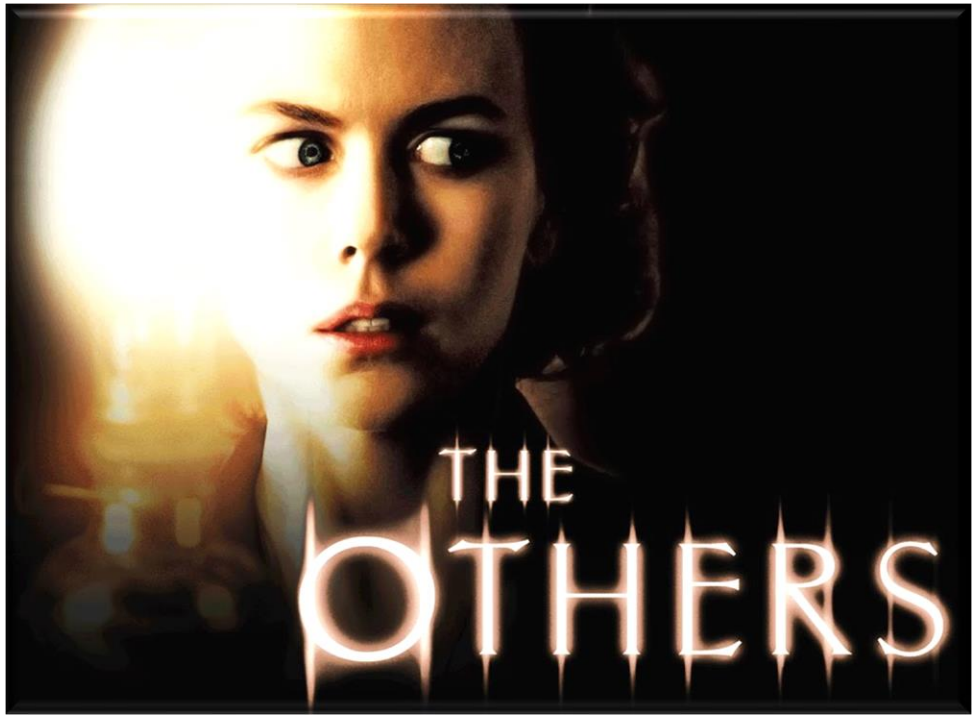
Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

Learning Objectives in this Lesson

- Understand foundational functional programming features in Java, e.g.,
  - Lambda expressions
  - Method & constructor references
  - Key functional interfaces
  - Other properties of functional interfaces



Other Properties of Functional Interfaces

Other Properties of Functional Interfaces

- Functional interfaces may also have default methods and/or static methods

```
#FunctionalInterface
interface Comparator<T> {
  int compare(T o1, T o2);

  boolean equals(Object obj);

  default Comparator<T> reversed()
  { return Collections.reverseOrder(this); }

  static <T extends Comparable<? super T>>
  Comparator<T> reverseOrder()
  { return Collections.reverseOrder(); }
  ...
```

See docs.oracle.com/javase/tutorial/java/IandI/defaultmethods.html

Other Properties of Functional Interfaces

- Functional interfaces may also have default methods and/or static methods, e.g.

```
#FunctionalInterface
interface Comparator<T> {
  int compare(T o1, T o2);

  boolean equals(Object obj);

  default Comparator<T> reversed()
  { return Collections.reverseOrder(this); }

  static <T extends Comparable<? super T>>
  Comparator<T> reverseOrder()
  { return Collections.reverseOrder(); }
  ...
```

A comparison function that imposes a total ordering on some collection of objects

See docs.oracle.com/javase/8/docs/api/java/util/Comparator.html

Other Properties of Functional Interfaces

- Functional interfaces may also have default methods and/or static methods

```
#FunctionalInterface
interface Comparator<T> {
  int compare(T o1, T o2);

  boolean equals(Object obj);

  default Comparator<T> reversed()
  { return Collections.reverseOrder(this); }

  static <T extends Comparable<? super T>>
  Comparator<T> reverseOrder()
  { return Collections.reverseOrder(); }
  ...
```

This annotation type indicates that this interface type declaration is intended as a functional interface.

See docs.oracle.com/javase/8/docs/api/java/lang/FunctionalInterface.html

Other Properties of Functional Interfaces

- Functional interfaces may also have default methods and/or static methods, e.g.

```
#FunctionalInterface
interface Comparator<T> {
  int compare(T o1, T o2);

  boolean equals(Object obj);

  default Comparator<T> reversed()
  { return Collections.reverseOrder(this); }

  static <T extends Comparable<? super T>>
  Comparator<T> reverseOrder()
  { return Collections.reverseOrder(); }
  ...
```

An abstract method in this functional interface

Other Properties of Functional Interfaces

- Functional interfaces may also have default methods and/or static methods, e.g.

```
#FunctionalInterface
interface Comparator<T> {
  int compare(T o1, T o2);

  boolean equals(Object obj);

  default Comparator<T> reversed()
  { return Collections.reverseOrder(this); }

  static <T extends Comparable<? super T>>
  Comparator<T> reverseOrder()
  { return Collections.reverseOrder(); }
  ...
```

A default method provides the default implementation, which can be overridden

See earlier lesson on "*Overview of Functional Interfaces: Function*"

Other Properties of Functional Interfaces

- Functional interfaces may also have default methods and/or static methods, e.g.

```
#FunctionalInterface
interface Comparator<T> {
  int compare(T o1, T o2);

  boolean equals(Object obj);

  default Comparator<T> reversed()
  { return Collections.reverseOrder(this); }

  static <T extends Comparable<? super T>>
  Comparator<T> reverseOrder()
  { return Collections.reverseOrder(); }
  ...
```

A static method provides the one-&-only implementation

Other Properties of Functional Interfaces

- Functional interfaces may also have default methods and/or static methods, e.g.

```
#FunctionalInterface
interface Comparator<T> {
  int compare(T o1, T o2);

  boolean equals(Object obj);

  default Comparator<T> reversed()
  { return Collections.reverseOrder(this); }

  static <T extends Comparable<? super T>>
  Comparator<T> reverseOrder()
  { return Collections.reverseOrder(); }
  ...
```

An abstract method that overrides a public java.lang.Object method does not count as part of the interface's abstract method count

See docs.oracle.com/javase/8/docs/api/java/lang/FunctionalInterface.html

End of Understand Other Properties of Java Functional Interfaces