Douglas C. Schmidt

<u>d.schmidt@vanderbilt.edu</u>

www.dre.vanderbilt.edu/~schmidt



Professor of Computer Science

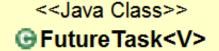
Institute for Software Integrated Systems

Vanderbilt University Nashville, Tennessee, USA



Learning Objectives in this Part of the Lesson

- Understand how Java FutureTask conveys a result from a computation running in a thread to thread(s) retrieving the result
- Recognize key methods in Java FutureTask



- FutureTask(Callable<V>)
- FutureTask(Runnable,V)
- isCancelled():boolean
- isDone():boolean
- cancel(boolean):boolean
- get()
- get(long,TimeUnit)
- run():void

 Key methods in FutureTask class FutureTask<V> implements RunnableFuture<V> { public FutureTask

```
public boolean cancel (boolean
                                     mayInterruptIfRunning) {...}
                                public V get() {...}
                                protected void done() {...}
See docs.oracle.com/javase/8/docs/api/java/util/concurrent/FutureTask.html
```

public void run() {...}

(Callable < V > callable) { . . . }

public boolean isCancelled() {...}

public boolean isDone() {...}

- Key methods in FutureTask, e.g. Constructs a future task with
 - a callable to run later
 - The run() method invokes call() on the callable

```
RunnableFuture<V> {
public FutureTask
```

class FutureTask<V> implements

(Callable < V > callable) { . . . }

public void run() {...}

public boolean isCancelled() {...}

public boolean isDone() {...} public boolean cancel (boolean

mayInterruptIfRunning) {...}

public V get() {...} protected void done() {...}

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

- Key methods in FutureTask, e.g.
 - Constructs a future task with a callable to run later
 - Runs the task
 - i.e., forward to the call() method on the callable

```
class FutureTask<V> implements
          RunnableFuture<V> {
```

public void run() {...}

public boolean isCancelled() {...}

public boolean isDone() {...}
public boolean cancel(boolean

mayInterruptIfRunning) {...}
public V get() {...}

protected void done() {...}
...

Sets the future task to the result of its computation unless it's been cancelled

- Key methods in FutureTask, e.g.
 - Constructs a future task with a callable to run later
 - Runs the task
 - Checks whether the task was cancelled or finished normally

```
class FutureTask<V> implements
    RunnableFuture<V> {
   public FutureTask
        (Callable<V> callable) {...}
```

public boolean isCancelled() {...}

public boolean isDone() {...}
public boolean cancel(boolean

public void run() {...}

public V get() {...}
protected void done() {...}

mayInterruptIfRunning) {...}

•

- Key methods in FutureTask, e.g.
 - Constructs a future task with a callable to run later
 - Runs the task
 - Checks whether the task was cancelled or finished normally
 - Cancels a future task

```
class FutureTask<V> implements
      RunnableFuture<V> {
  public FutureTask
        (Callable < V > callable) { . . . }
  public void run() {...}
  public boolean isCancelled() {...}
  public boolean isDone() {...}
  public boolean cancel (boolean
      mayInterruptIfRunning) {...}
  public V get() {...}
  protected void done() {...}
```

- Key methods in FutureTask, e.g.
 - Constructs a future task with a callable to run later
 - Runs the task
 - Checks whether the task was cancelled or finished normally
 - Cancels a future task
 - true if thread executing this task should be interrupted
 - false if in-progress tasks are allowed to complete

```
class FutureTask<V> implements
      RunnableFuture<V> {
 public FutureTask
        (Callable < V > callable) { . . . }
 public void run() {...}
 public boolean isCancelled() {...}
 public boolean isDone() {...}
 public boolean cancel (boolean
      mayInterruptIfRunning) {...}
 public V get() {...}
 protected void done() {...}
```

- Key methods in FutureTask, e.g.
 - Constructs a future task with a callable to run later
 - Runs the task
 - Checks whether the task was cancelled or finished normally
 - Cancels a future task
 - Gets the result
 - Blocks until task finishes

```
class FutureTask<V> implements
          RunnableFuture<V> {
   public FutureTask
          (Callable<V> callable) {...}
   public void run() {...}
   public boolean isCancelled() {...}
   public boolean isDone() {...}
```

public boolean cancel (boolean

protected void done() {...}

public V get() {...}

mayInterruptIfRunning) {...}

10

- Key methods in FutureTask, e.g.
 - Constructs a future task with a callable to run later
 - Runs the task
 - Checks whether the task was cancelled or finished normally
 - Cancels a future task
 - Gets the result
 - Blocks until task finishes
 - There's also a timed version of get(), just like Future

```
class FutureTask<V> implements
    RunnableFuture<V> {
```

public void run() {...}

public boolean isCancelled() {...}

public boolean isDone() {...}
public boolean cancel(boolean

mayInterruptIfRunning) {...}
public V get(long, TimeUnit) {...}

protected void done() {...}
...

- Key methods in FutureTask, e.g.
 - Constructs a future task with a callable to run later
 - Runs the task
 - Checks whether the task was cancelled or finished normally
 - Cancels a future task
 - Gets the result
 - Hook method called back in the run() hook method if the call() hook method is successful

class FutureTask<V> implements
 RunnableFuture<V> {
 public FutureTask

(Callable<V> callable) {...}
public void run() {...}

public boolean isCancelled() {...}

public boolean isDone() {...}
public boolean cancel(boolean

mayInterruptIfRunning) {...}

public V get() {...}
protected void done() {...}

•••

Subclasses may override this method, e.g., to invoke completion callbacks

End of Key Methods in Java FutureTask