Overview of Java Futures

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 Part of the Lesson

- Motivate the need for Java futures by understanding the pros & cons of synchrony & asynchrony
- Understand that Java futures provide the foundation for completable futures in Java

See en.wikipedia.org/wiki/Java_version_history
Learning Objectives in this Part of the Lesson

• Motivate the need for Java futures by understanding the pros & cons of synchrony & asynchrony

• Understand that Java futures provide the foundation for completable futures in Java

• Recognize a human known use of Java futures
Learning Objectives in this Part of the Lesson

• Motivate the need for Java futures by understanding the pros & cons of synchrony & asynchrony

• Understand that Java futures provide the foundation for completable futures in Java
  • Recognize a human known use of Java futures

• Know all the methods in the Future interface

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/Future.html
A Human Known Use of Java Futures
A Human Known Use of Java Futures

1. Async computation runs
2. Result obtained only after async computation completes

See en.wikipedia.org/wiki/Futures_and_promises
A Human Known Use of Java Futures

- A future is essentially a proxy that represents the result(s) of an async call.

```java
Result get_result ()
begin
    ## Suspend calling thread until result is available.
    if (result == NULL) then
        thread.wait ();
    return result;
end
```

**Table tent #’s are a human-known-use of futures!**

A Human Known Use of Java Futures

- A future is essentially a proxy that represents the result(s) of an async call

```java
Result get_result ()
begin
  ## Suspend calling thread until result is available.
  if (result == NULL) then
    thread.wait ();
  return result;
end
```

---

Table tent #'s are a human-known-use of futures!

---

e.g., McDonald’s vs Wendy’s model of preparing fast food
Overview of the Java Future API
Overview of the Java Future API

- Java 5 added async call support via the Java Future interface

See en.wikipedia.org/wiki/Java_version_history
Overview of the Java Future API

- Java Future methods can manage a task’s lifecycle after it’s submitted to run asynchronously.

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/Future.html
Overview of the Java Future API

- Java Future methods can manage a task’s lifecycle after it’s submitted to run asynchronously, e.g.
- A future can be tested for completion

```
1. submit(task)
2. Return future
3. offer()
4. take()
5. run()
6. isDone()
```

```
future.isDone()
...}
```

```
Future<V>
- isDone(): boolean
- get(): object
- get(long, TimeUnit): object
```

```
ThreadPoolExecutor
<table>
<thead>
<tr>
<th>WorkQueue</th>
</tr>
</thead>
<tbody>
<tr>
<td>callable</td>
</tr>
<tr>
<td>callable</td>
</tr>
<tr>
<td>callable</td>
</tr>
</tbody>
</table>

Fixed WorkerThreads

Thread (main thread)

Callable

BigFraction
Overview of the Java Future API

- Java Future methods can manage a task’s lifecycle after it’s submitted to run asynchronously, e.g.
  - A future can be tested for completion
  - A future be tested for cancellation & cancelled

```
1. submit(task)
2. Return future
3. offer()
4. take()
5. run()
6. cancel()

Runnable
Callable
Future<V>
Future.cancel(): boolean
Future.isCancelled(): boolean
Future.isDone(): boolean
Future.get():
Future.get(long, TimeUnit)
```

```
Future.run()
```

```
future.cancel();
```

```
if (!future.isCancelled())
future.cancel();
```
Overview of the Java Future API

- Java Future methods can manage a task’s lifecycle after it’s submitted to run asynchronously, e.g.
  - A future can be tested for completion
  - A future be tested for cancellation & cancelled
  - A future can retrieve a two-way task’s result

```java
Future<BigFraction> future = new ThreadPoolExecutor().submit((Callable) new Callable() {
    public BigFraction call() {
        return new BigFraction();
    }
}).get();
```

- WorkQueue
  - Submit task
  - Offer task
  - Take task
  - Run task

- Future
  - Set result
  - Get result

- Callable

- Thread (main thread)
  - Run

- ResultType
  - Get
  - Get with timeout

- Fixed WorkerThreads
  - Run

- `cancel(boolean)`, `isCancelled()`, `isDone()`
Overview of the Java Future API

- The Java Future interface provides the foundation for the Java CompletableFuture class.

See en.wikipedia.org/wiki/Java_version_history
Overview of the Java Future API

- The Java Future interface provides the foundation for the Java CompletableFuture class.
- However, the CompletableFuture class defines dozens of methods & more powerful capabilities.
End of Overview of Java Futures