Overview of Java Futures

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

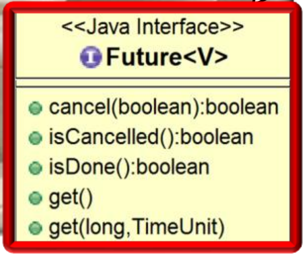
Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

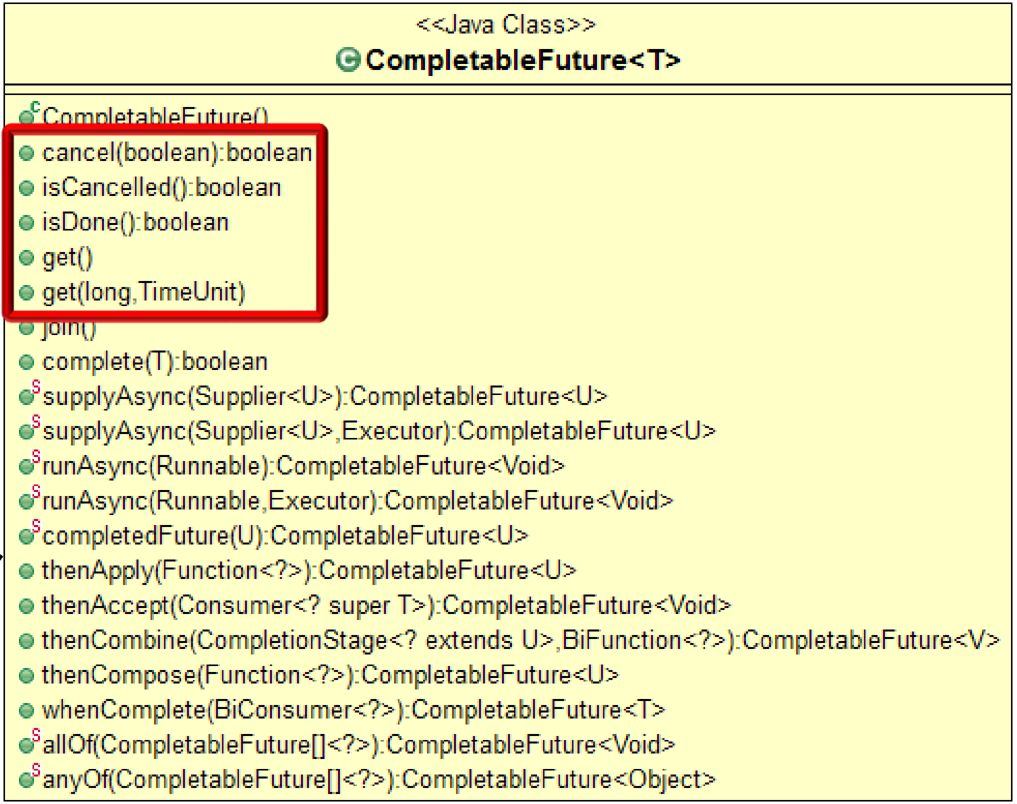
Learning Objectives in this Part of the Lesson

- 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



<<Java Interface>>
Future<V>

- cancel(boolean):boolean
- isCancelled():boolean
- isDone():boolean
- get()
- get(long,TimeUnit)



<<Java Class>>
CompletableFuture<T>

- CompletableFuture()
- cancel(boolean):boolean
- isCancelled():boolean
- isDone():boolean
- get()
- get(long,TimeUnit)
- join()
- complete(T):boolean
- supplyAsync(Supplier<U>):CompletableFuture<U>
- supplyAsync(Supplier<U>,Executor):CompletableFuture<U>
- runAsync(Runnable):CompletableFuture<Void>
- runAsync(Runnable,Executor):CompletableFuture<Void>
- completedFuture(U):CompletableFuture<U>
- thenApply(Function<?>):CompletableFuture<U>
- thenAccept(Consumer<? super T>):CompletableFuture<Void>
- thenCombine(CompletionStage<? extends U>,BiFunction<?>):CompletableFuture<V>
- thenCompose(Function<?>):CompletableFuture<U>
- whenComplete(BiConsumer<?>):CompletableFuture<T>
- allOf(CompletableFuture[]<?>):CompletableFuture<Void>
- anyOf(CompletableFuture[]<?>):CompletableFuture<Object>

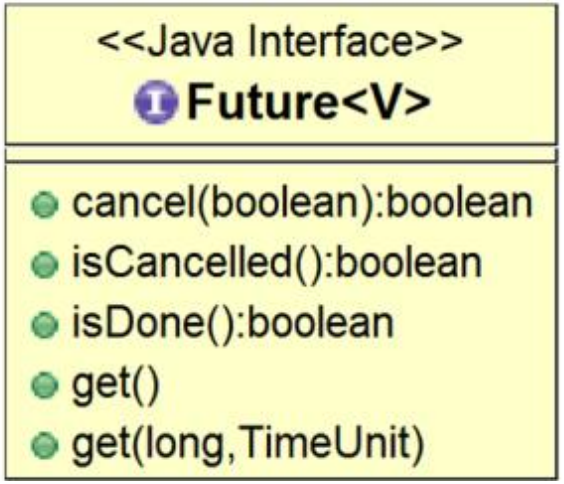
See en.wikipedia.org/wiki/Java_version_history

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

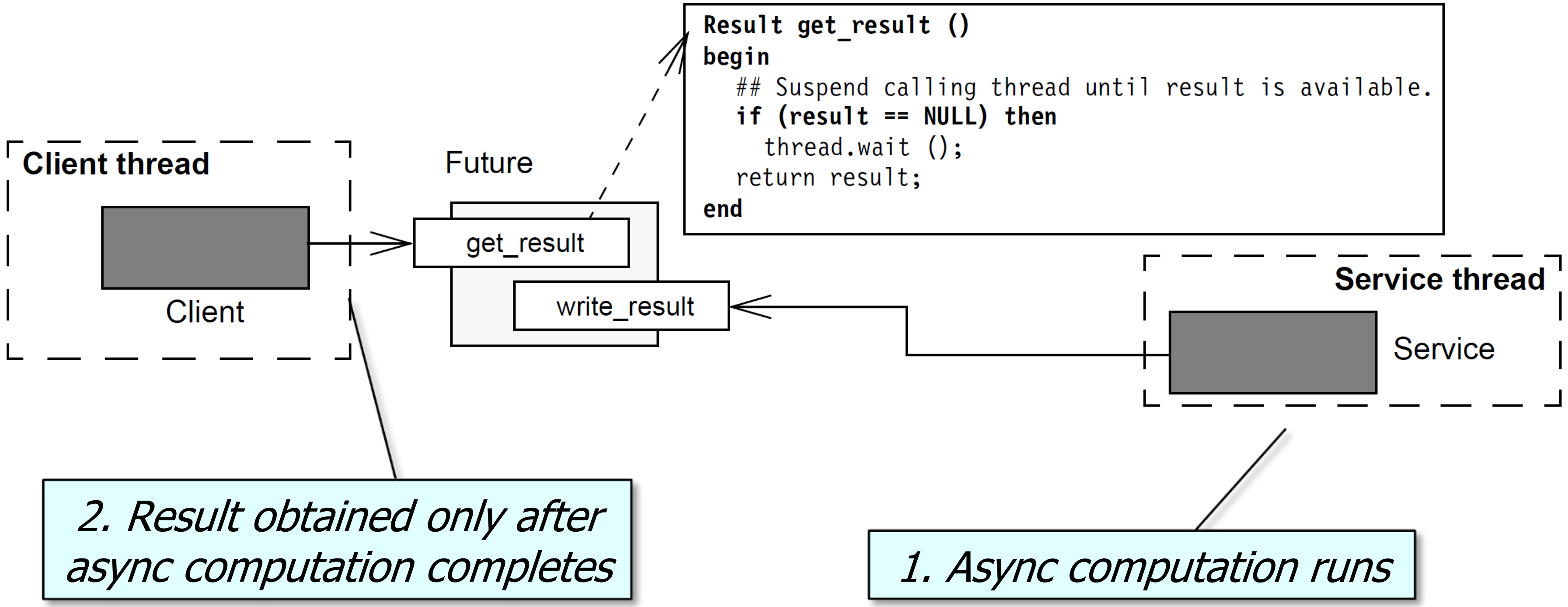


See docs.oracle.com/javase/8/docs/api/java/util/concurrent/Future.html

A Human Known Use of Java Futures

A Human Known Use of Java Futures

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

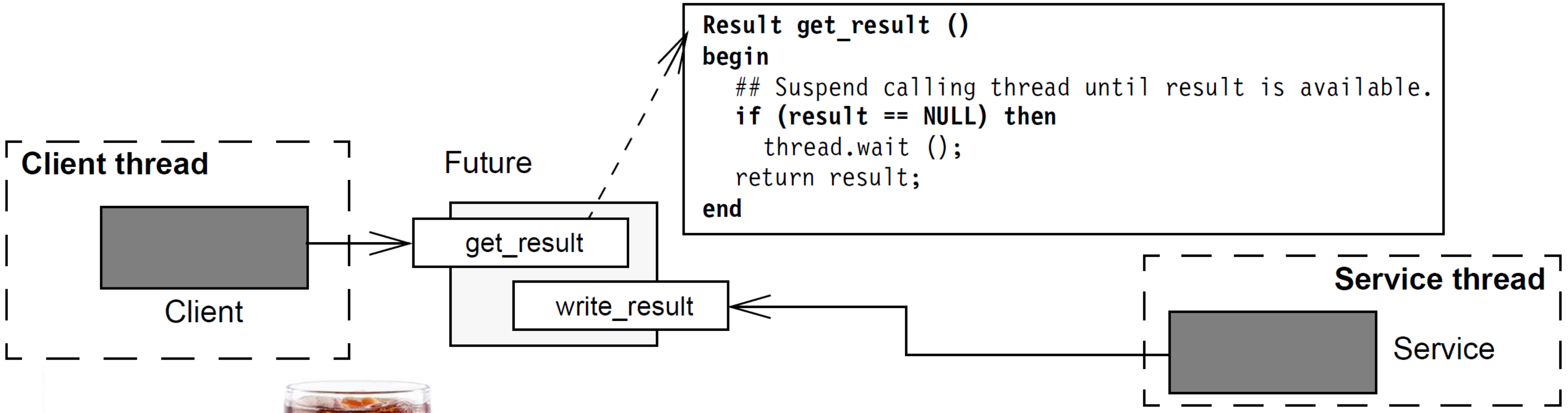

Client thread
Client
Future
get_result
write_result
Service thread
Service
2. Result obtained only after async computation completes
1. Async computation runs

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

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



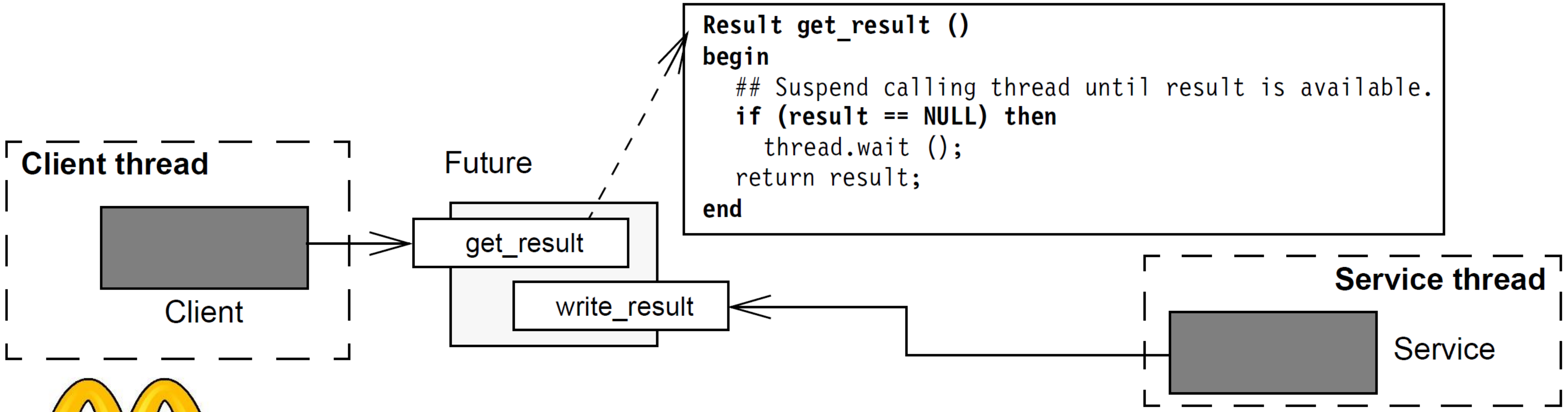
```
Result get_result ()
begin
  ## Suspend calling thread until result is available.
  if (result == NULL) then
    thread.wait ();
  return result;
end
```

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

See www.citygrafx.com/table-numbers-table-markers

A Human Known Use of Java Futures

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



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

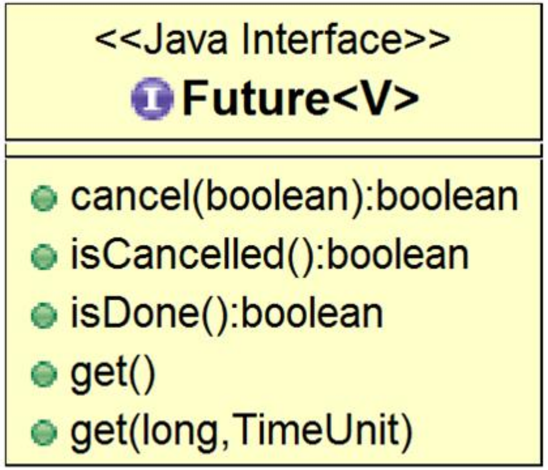
Table tent #'s are a human-known-use of futures!

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

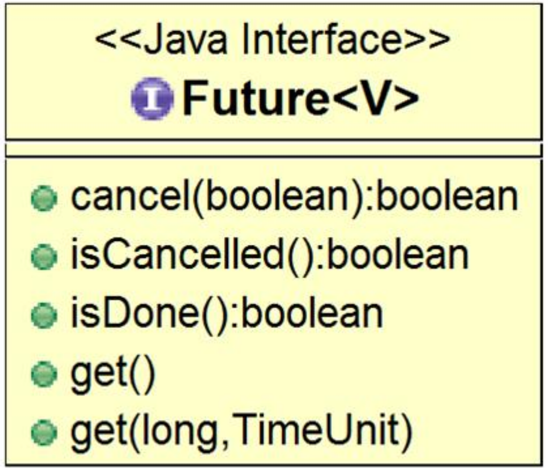


cancel(boolean):boolean
isCancelled():boolean
isDone():boolean
get()
get(long,TimeUnit)

See en.wikipedia.org/wiki/Java_version_history

Overview of the Java Future API

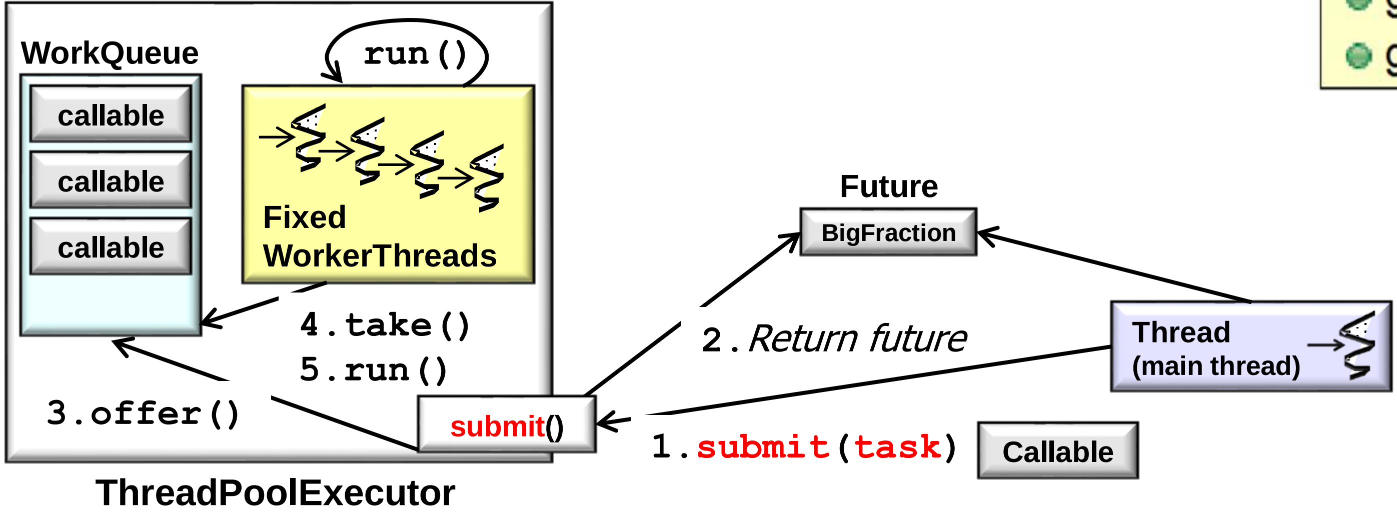
- Java Future methods can manage a task's lifecycle after it's submitted to run asynchronously



<<Java Interface>>

Future<V>

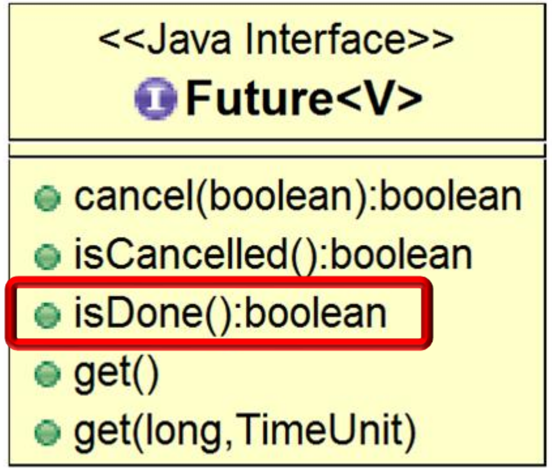
- cancel(boolean):boolean
- isCancelled():boolean
- isDone():boolean
- get()
- get(long,TimeUnit)

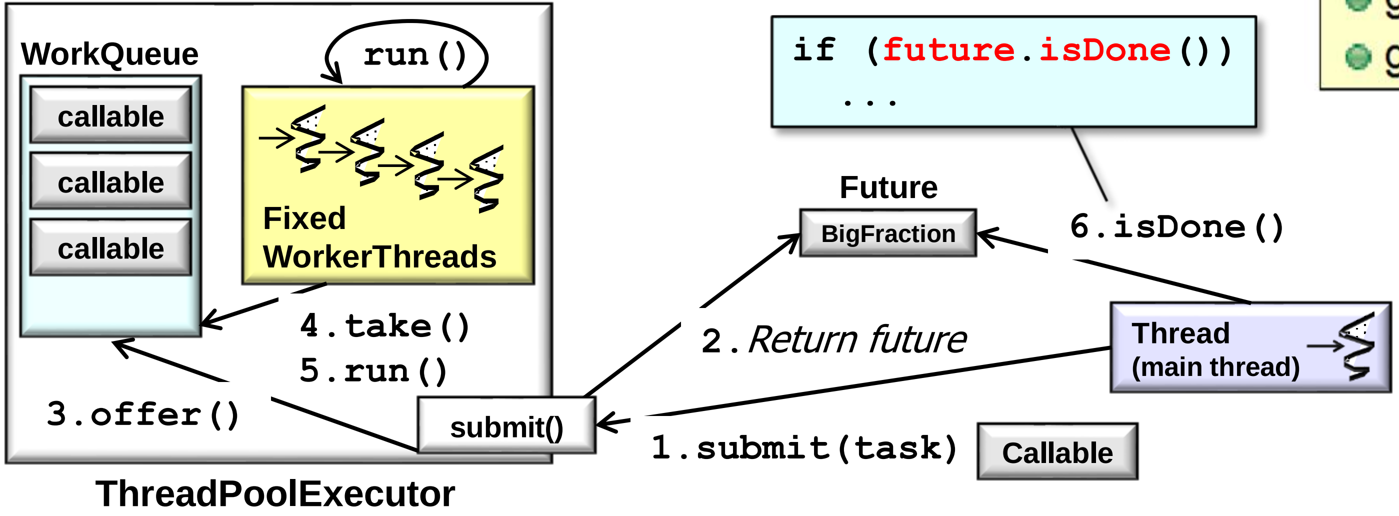


See docs.oracle.com/javase/8/docs/api/java/util/concurrent/Future.html

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

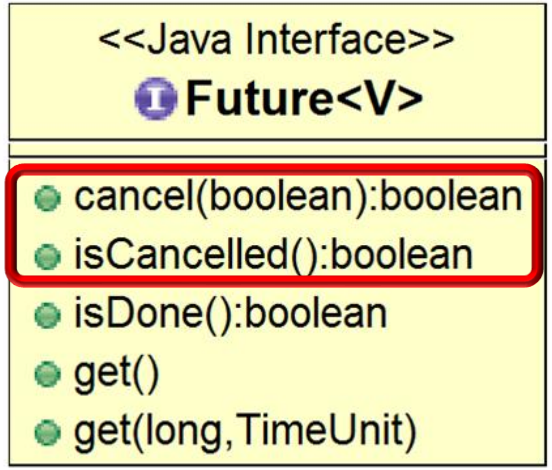




```
if (future.isDone())
  ...
```

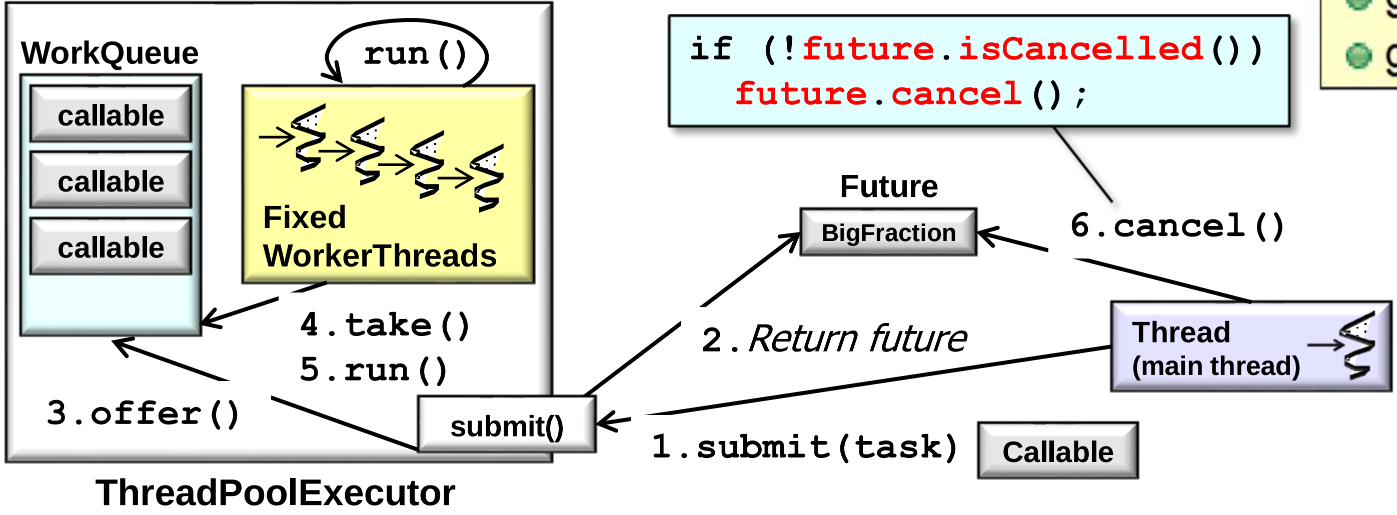
Overview of the Java Future API

- 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



<<Java Interface>>
Future<V>

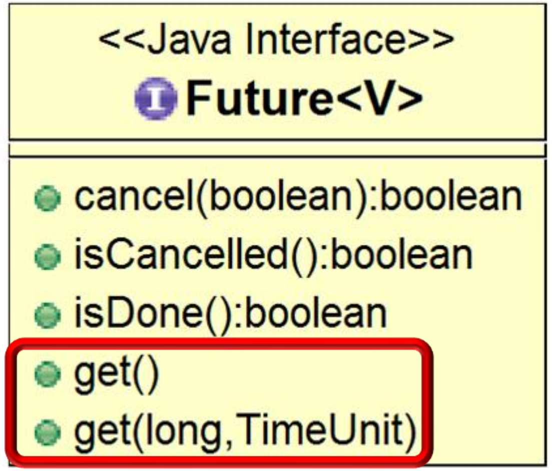
- cancel(boolean):boolean
- isCancelled():boolean
- isDone():boolean
- get()
- get(long,TimeUnit)



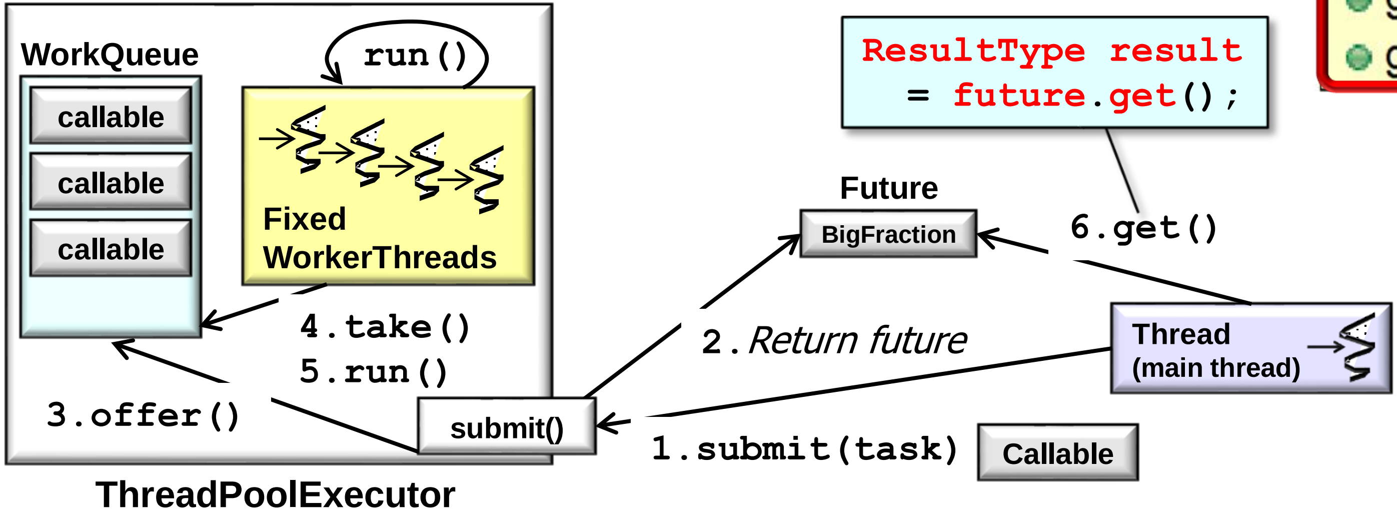
```
if (!future.isCancelled())
  future.cancel();
```

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

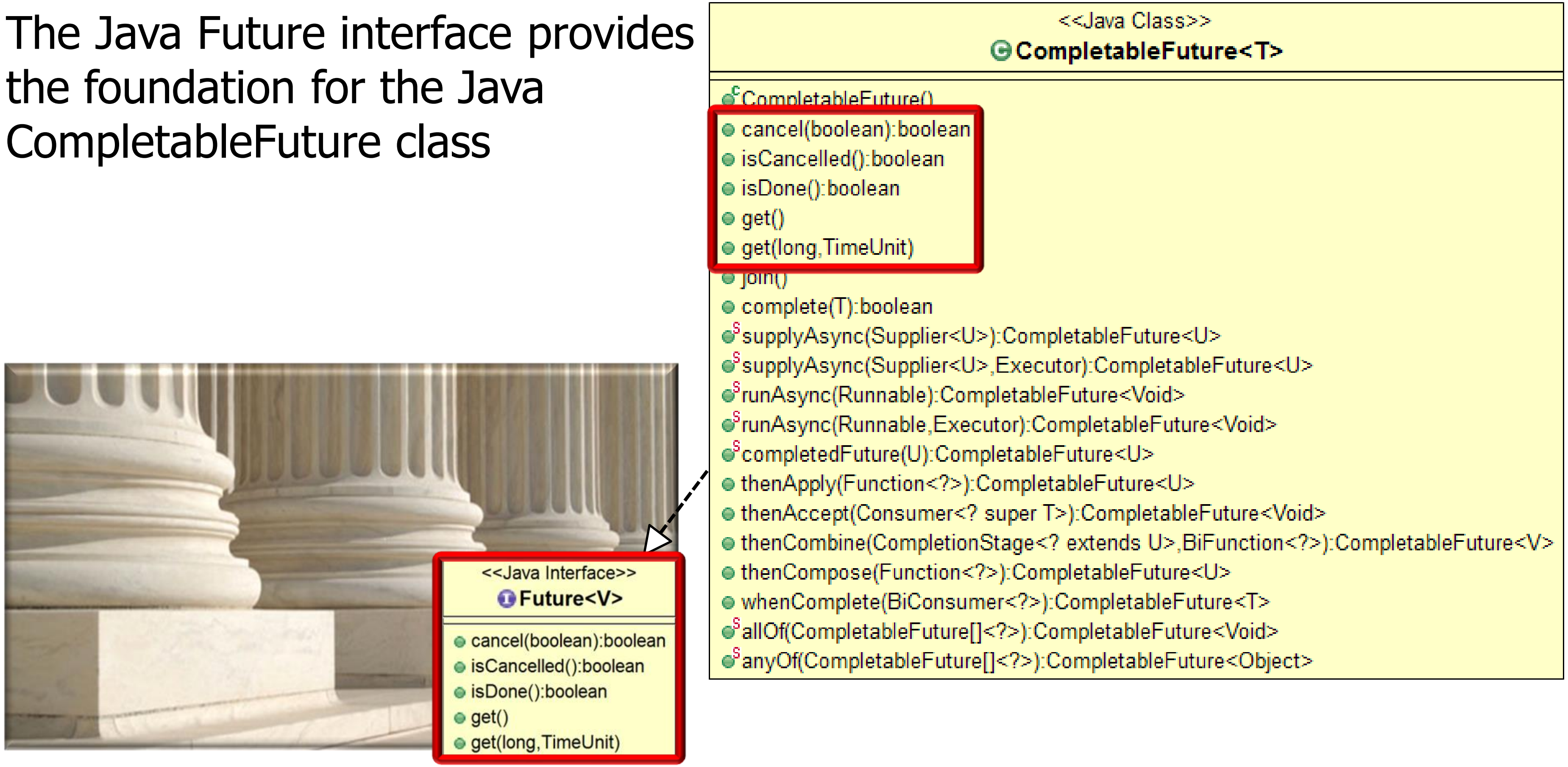


```
ResultType result
  = future.get();
```



Overview of the Java Future API

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



<<Java Class>>
CompletableFuture<T>

- CompletableFuture()
- cancel(boolean):boolean
- isCancelled():boolean
- isDone():boolean
- get()
- get(long,TimeUnit)
- join()
- complete(T):boolean
- supplyAsync(Supplier<U>):CompletableFuture<U>
- supplyAsync(Supplier<U>,Executor):CompletableFuture<U>
- runAsync(Runnable):CompletableFuture<Void>
- runAsync(Runnable,Executor):CompletableFuture<Void>
- completedFuture(U):CompletableFuture<U>
- thenApply(Function<?>):CompletableFuture<U>
- thenAccept(Consumer<? super T>):CompletableFuture<Void>
- thenCombine(CompletionStage<? extends U>,BiFunction<?>):CompletableFuture<V>
- thenCompose(Function<?>):CompletableFuture<U>
- whenComplete(BiConsumer<?>):CompletableFuture<T>
- allOf(CompletableFuture[]<?>):CompletableFuture<Void>
- anyOf(CompletableFuture[]<?>):CompletableFuture<Object>

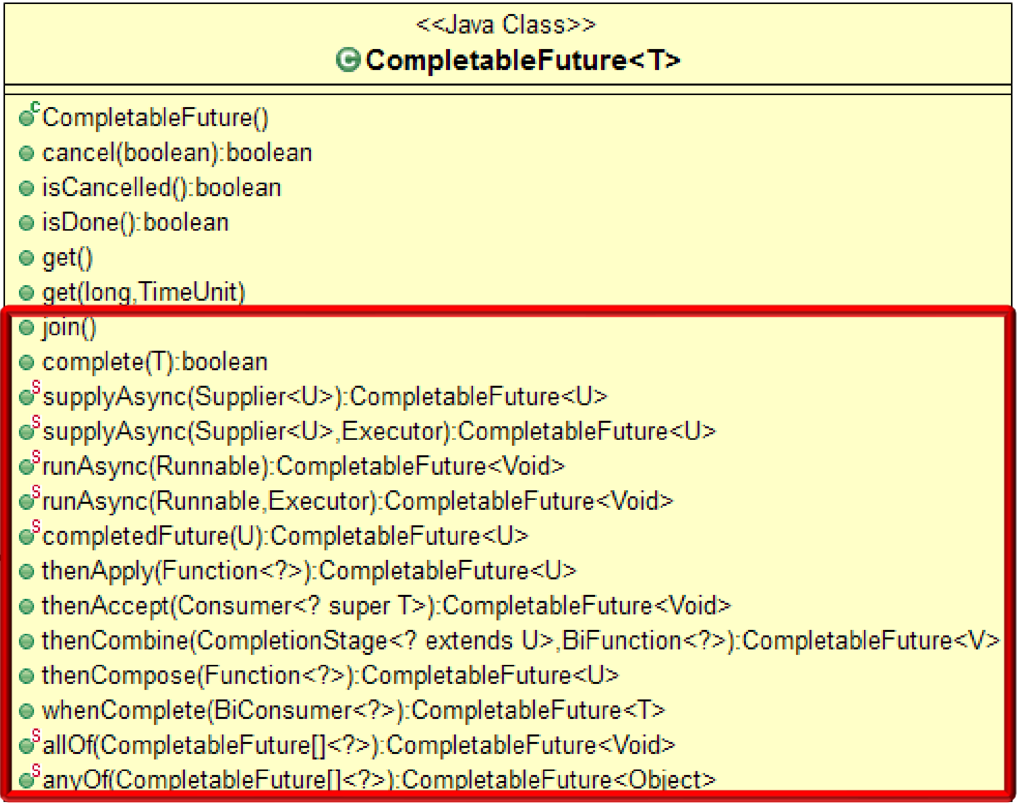
<<Java Interface>>
Future<V>

- cancel(boolean):boolean
- isCancelled():boolean
- isDone():boolean
- get()
- get(long,TimeUnit)

See en.wikipedia.org/wiki/Java_version_history

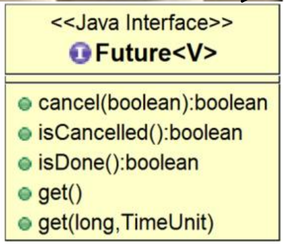
Overview of the Java Future API

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



```
<<Java Class>>
CompletableFuture<T>

CompletableFuture()
cancel(boolean):boolean
isCancelled():boolean
isDone():boolean
get()
get(long,TimeUnit)
join()
complete(T):boolean
supplyAsync(Supplier<U>):CompletableFuture<U>
supplyAsync(Supplier<U>,Executor):CompletableFuture<U>
runAsync(Runnable):CompletableFuture<Void>
runAsync(Runnable,Executor):CompletableFuture<Void>
completedFuture(U):CompletableFuture<U>
thenApply(Function<?>):CompletableFuture<U>
thenAccept(Consumer<? super T>):CompletableFuture<Void>
thenCombine(CompletionStage<? extends U>,BiFunction<?>):CompletableFuture<V>
thenCompose(Function<?>):CompletableFuture<U>
whenComplete(BiConsumer<?>):CompletableFuture<T>
allOf(CompletableFuture[]<?>):CompletableFuture<Void>
anyOf(CompletableFuture[]<?>):CompletableFuture<Object>
```



```
<<Java Interface>>
Future<V>

cancel(boolean):boolean
isCancelled():boolean
isDone():boolean
get()
get(long,TimeUnit)
```

See upcoming lessons on the completable futures framework

End of Overview of Java Futures