Overview of Basic Java 8
CompletableFuture Features (Part 1)

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

• Understand the basic completable futures features

```
Class CompletableFuture<T>

java.lang.Object
    java.util.concurrent.CompletableFuture<T>

All Implemented Interfaces:
CompletionStage<T>, Future<T>

public class CompletableFuture<T>
    extends Object
    implements Future<T>, CompletionStage<T>

A Future that may be explicitly completed (setting its value and status), and may be used as a CompletionStage, supporting dependent functions and actions that trigger upon its completion.

When two or more threads attempt to complete, completeExceptionally, or cancel a CompletableFuture, only one of them succeeds.

In addition to these and related methods for directly manipulating status and results, CompletableFuture implements interface CompletionStage with the following policies:
```
Basic CompletableFuture
Future Features
Basic Completable Future Features

- Basic completable future features

See [github.com/douglasraigschmidt/LiveLessons/tree/master/Java8/ex8](https://github.com/douglasraigschmidt/LiveLessons/tree/master/Java8/ex8)
Basic Completable Future Features

- Basic completable future features
- Support the Future API

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/Future.html](docs.oracle.com/javase/8/docs/api/java/util/concurrent/Future.html)
Basic Completetable Future Features

- Basic completable future features
- Support the Future API
- Can (time-) block & poll

```java
String f1 = "62675744/15668936";
String f2 = "609136/913704";

ForkJoinTask<BigFraction> f = commonPool().submit(() -> {
    BigFraction bf1 =
        new BigFraction(f1);
    BigFraction bf2 =
        new BigFraction(f2);
    return bf1.multiply(bf2);
});
...

BigFraction result = f.get();
// f.get(10, MILLISECONDS);
// f.get(0, 0);
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html](http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html)
Basic Completable Future Features

- Basic completable future features
- Support the Future API
  - Can (time-) block & poll
  - Can be cancelled & tested if canceled/done

```java
String f1 = "62675744/15668936";
String f2 = "609136/913704";

ForkJoinTask<BigFraction> f = commonPool().submit(() -> {
    BigFraction bf1 =
        new BigFraction(f1);
    BigFraction bf2 =
        new BigFraction(f2);
    return bf1.multiply(bf2);
});

... 
if (!f.isDone() || !f.isCancelled())
    f.cancel();
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html](docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html)
Basic Completable Future Features

- Basic completable future features
- Support the Future API
  - Can (time-) block & poll
  - Can be cancelled & tested if canceled/done
  - cancel() doesn’t interrupt the computation by default..

```java
String f1 = "62675744/15668936";
String f2 = "609136/913704";

ForkJoinTask<BigFraction> f =
  commonPool().submit(() -> {
    BigFraction bf1 =
      new BigFraction(f1);
    BigFraction bf2 =
      new BigFraction(f2);
    return bf1.multiply(bf2);
  });

... if (!f.isDone() || !f.isCancelled())
  f.cancel();
```

Basic Completable Future Features

- Basic completable future features
- Support the Future API
- Define a `join()` method

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#join](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#join)
Basic Completetable Future Features

- Basic completetable future features
- Support the Future API
- Define a join() method
  - Behaves like get() without using checked exceptions

```java
futures.stream().map(Future::join).collect(toList())
```

Future::join can be used as a method reference in a Java 8 stream.
Basic Completable Future Features

- Basic completable future features
- Support the Future API
- Define a join() method
  - Behaves like get() without using checked exceptions

Mixing checked exceptions & Java 8 streams is ugly..
Basic Completetable Future Features

- Basic completetable future features
  - Support the Future API
  - Define a join() method
  - Can be completed explicitly

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#complete
Basic Completetable Future Features

- Basic completetable future features
- Support the Future API
- Define a join() method
- Can be completed explicitly
  - i.e., sets result returned by get()/join() to a given value

```java
CompletableFuture<...> future =
    new CompletableFuture<>();

new Thread (()
    -> {
        ...
        future.complete(...);
    }).start();

... System.out.println(future.join());
```

After complete() is done calls to join() will unblock
Basic Completable Future Features

- Basic completable future features
- Support the Future API
- Define a `join()` method
- Can be completed explicitly
  - i.e., sets result returned by `get()`/`join()` to a given value

```java
CompletableFuture<...> future = new CompletableFuture<>();

final CompletableFuture<Long> zero = CompletableFuture.completedFuture(0L);

new Thread (() -> {
...
    future.complete(zero.join());
}).start();

... System.out.println(future.join());
```

A completable future can be initialized to a value/constant

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#completedFuture](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#completedFuture)
End of Overview of Basic Java 8 Completable Future Features (Part 1)