Advanced Java CompletableFuture Features: Applying Factory Methods

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

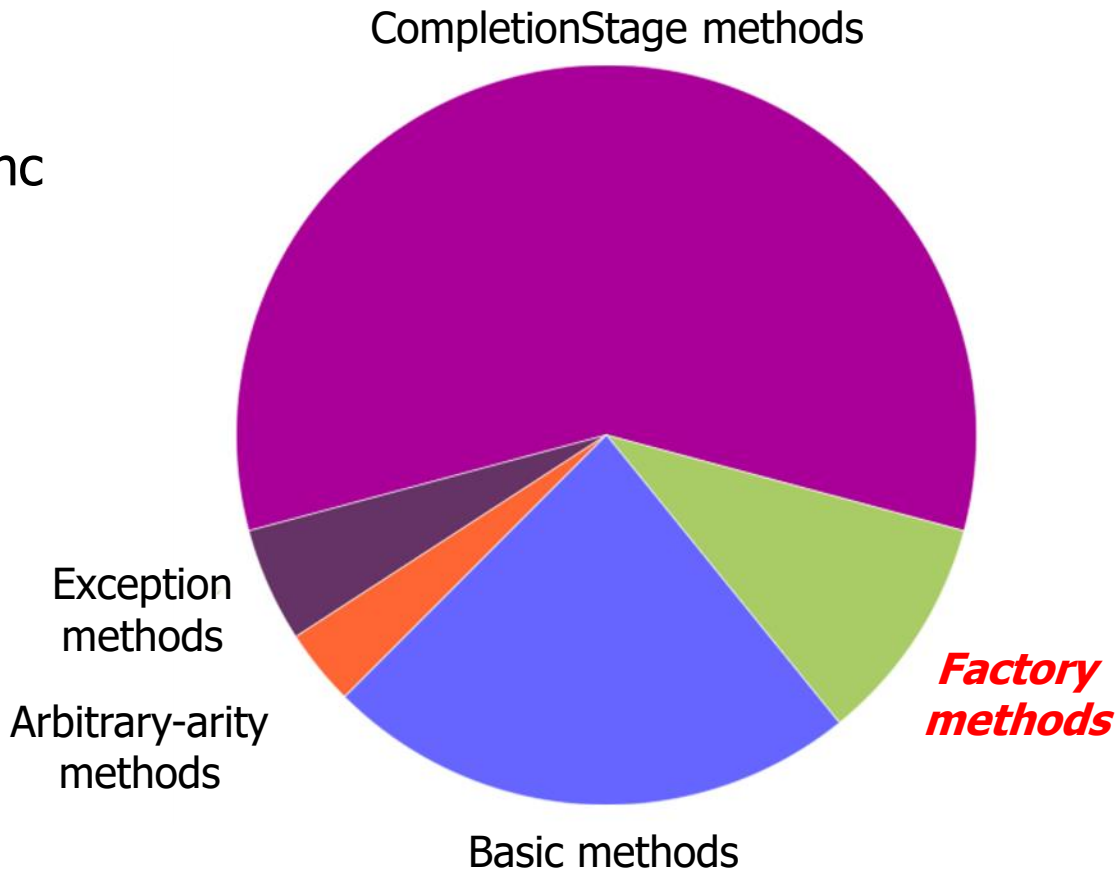
Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand advanced features of completable futures, e.g.
  - Factory methods initiate async computations
    - Applying factory methods



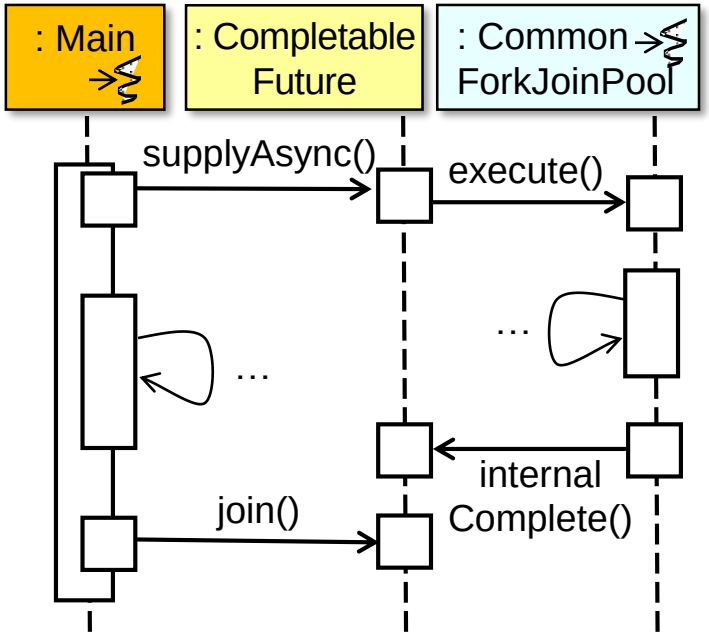
Applying Completable Future Factory Methods

Applying CompletableFuture Factory Methods

- Using supplyAsync() to multiply big fractions

```
String f1 = "62675744/15668936";
String f2 = "609136/913704";
CompletableFuture<BigFraction> future =
  CompletableFuture
    .supplyAsync(() -> {
      BigFraction bf1 =
        new BigFraction(f1);
      BigFraction bf2 =
        new BigFraction(f2);

      return bf1.multiply(bf2);
    });
...
System.out.println(future.join().toMixedString());
```



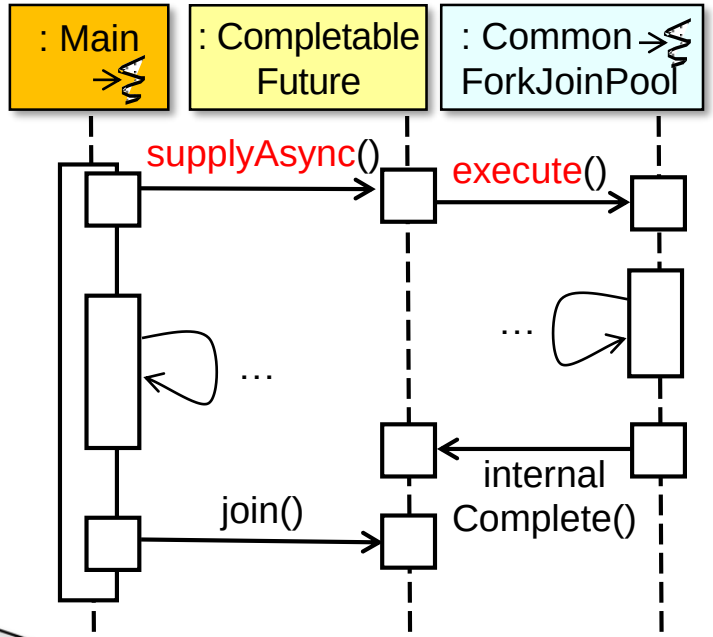
See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex8

Applying CompletableFuture Factory Methods

- Using supplyAsync() to multiply big fractions

```
String f1 = "62675744/15668936";
String f2 = "609136/913704";
CompletableFuture<BigFraction> future =
  CompletableFuture
    .supplyAsync(() -> {
      BigFraction bf1 =
        new BigFraction(f1);
      BigFraction bf2 =
        new BigFraction(f2);

      return bf1.multiply(bf2);
    });
...
System.out.println(future.join().toMixedString());
```



Arrange to execute the supplier lambda in common fork-join pool

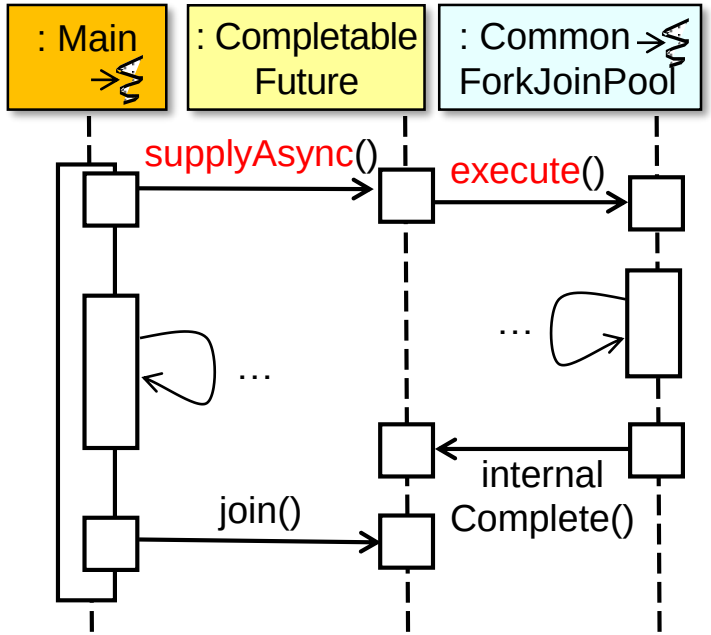
See docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#supplyAsync

Applying CompletableFuture Factory Methods

- Using supplyAsync() to multiply big fractions

```
String f1 = "62675744/15668936";
String f2 = "609136/913704";
CompletableFuture<BigFraction> future =
  CompletableFuture
    .supplyAsync(() -> {
      BigFraction bf1 =
        new BigFraction(f1);
      BigFraction bf2 =
        new BigFraction(f2);

      return bf1.multiply(bf2);
    });
...
System.out.println(future.join().toMixedString());
```



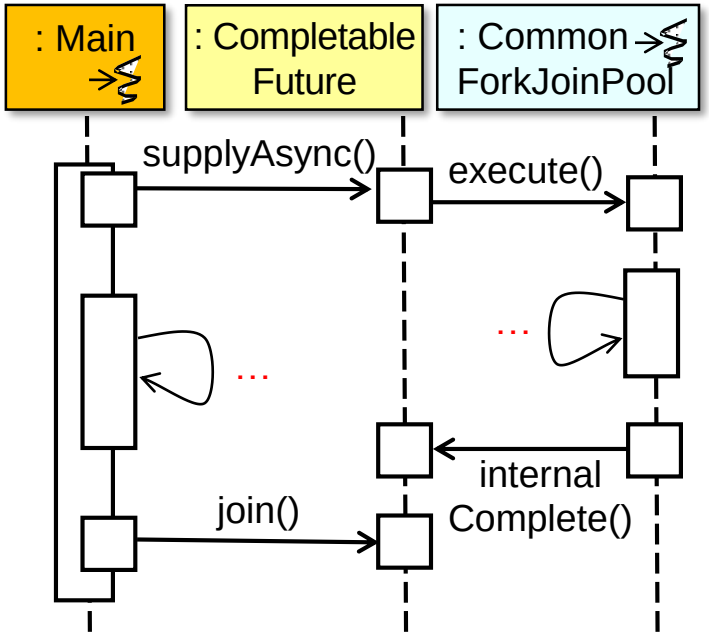
Define a supplier lambda that multiplies two BigFractions

Applying CompletableFuture Factory Methods

- Using supplyAsync() to multiply big fractions

```
String f1 = "62675744/15668936";
String f2 = "609136/913704";
CompletableFuture<BigFraction> future =
  CompletableFuture
    .supplyAsync(() -> {
      BigFraction bf1 =
        new BigFraction(f1);
      BigFraction bf2 =
        new BigFraction(f2);

      return bf1.multiply(bf2);
    });
...
System.out.println(future.join().toMixedString());
```



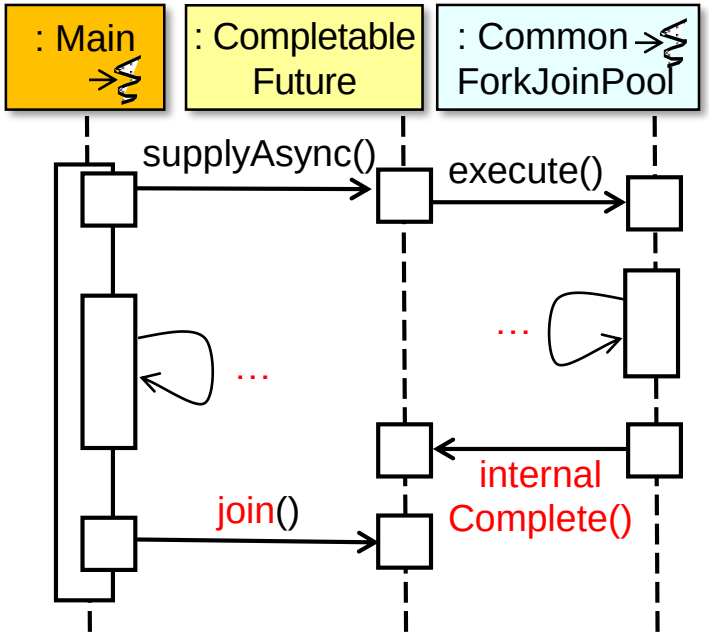
These computations run concurrently

Applying CompletableFuture Factory Methods

- Using supplyAsync() to multiply big fractions

```
String f1 = "62675744/15668936";
String f2 = "609136/913704";
CompletableFuture<BigFraction> future =
  CompletableFuture
    .supplyAsync(() -> {
      BigFraction bf1 =
        new BigFraction(f1);
      BigFraction bf2 =
        new BigFraction(f2);

      return bf1.multiply(bf2);
    });
...
System.out.println(future.join().toMixedString());
```



join() blocks until result is complete

Applying CompletableFuture Factory Methods

- Using supplyAsync() to multiply big fractions

```
String f1 = "62675744/15668936";
String f2 = "609136/913704";
CompletableFuture<BigFraction> future =
  CompletableFuture
    .supplyAsync(() -> {
       BigFraction bf1 =
         new BigFraction(f1);
       BigFraction bf2 =
         new BigFraction(f2);

       return bf1.multiply(bf2);
    });
...
System.out.println(future.join().toMixedString());
```

Calling CompletableFuture.supplyAsync() avoids the use of threads in this example!

Applying CompletableFuture Factory Methods

- Using supplyAsync() to multiply big fractions

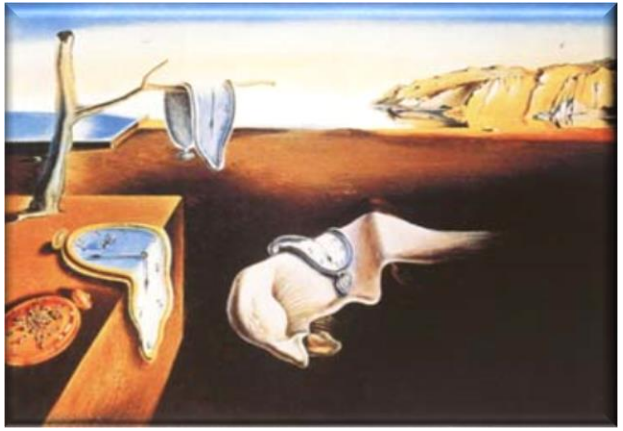
```
String f1 = "62675744/15668936";
String f2 = "609136/913704";
CompletableFuture<BigFraction> future =
  CompletableFuture
    .supplyAsync(() -> {
      BigFraction bf1 =
        new BigFraction(f1);
      BigFraction bf2 =
        new BigFraction(f2);

      return bf1.multiply(bf2);
    });
...
System.out.println(future.join().toMixedString());
```

There's no need to explicitly complete the future since supplyAsync() returns one

Applying CompletableFuture Factory Methods

- Using supplyAsync() to multiply big fractions



```
String f1 = "62675744/15668936";
String f2 = "609136/913704";
CompletableFuture<BigFraction> future =
  CompletableFuture
    .supplyAsync(() -> {
       BigFraction bf1 =
         new BigFraction(f1);
       BigFraction bf2 =
         new BigFraction(f2);

       return bf1.multiply(bf2);
    });
...
System.out.println(future.join().toMixedString());
```

However, we still must fix the problem with calling join() explicitly..

End of Advanced Java CompletableFuture Features: Applying Factory Methods