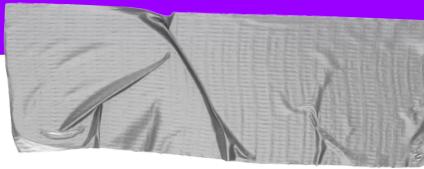




# Java Certified #5

A question lead guide to prepare Java certification



## Handling Date, Time, Text, Numeric and Boolean Values

Given:

```
LocalDate localDate = LocalDate.of( 2020, 8, 8 );  
Date date = java.sql.Date.valueOf( localDate );  
DateFormat formatter = new SimpleDateFormat(/* */ );  
System.out.println( formatter.format( date ) );
```

It's known that the given code prints out "August 08".

Which of the following should be inserted?

- MM d
- MM dd
- MMM dd
- MMMM dd
- None of the above

# MMMM dd

The number of symbol letters you specify also determines the format.

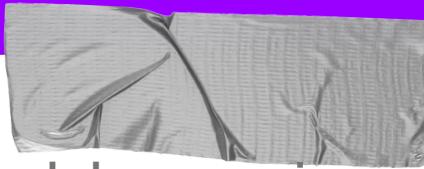
For example, if the `zz` pattern results in `PDT`, then the `zzzz` pattern generates `Pacific Daylight Time`.

The following table summarizes these rules:

Presentation	Number of Symbols	Result
Text	1 - 3	abbreviated form, if one exists
Text	$\geq 4$	full form
Number	minimum number of digits is required	shorter numbers are padded with zeros (for a year, if the count of 'y' is 2, then the year is truncated to 2 digits)
Text & Number	1 - 2	number form
Text & Number	3	text form

To print a month value in its full form, we need only four letters `M`. However, adding more letters doesn't hurt.

<https://docs.oracle.com/javase/tutorial/i18n/format/simpleDateFormat.html#datepattern>



## Working with Streams and Lambda expressions

Given:

```
StringBuilder result = Stream.of( "a", "b" )  
    .collect(  
        () -> new StringBuilder( "c" ),  
        StringBuilder::append, ( a, b ) -> b.append( a )  
    );
```

System.out.println( result );//What is the output of the given code fragment?

- abc
- bac
- cab
- cacb
- cbca

**cab**

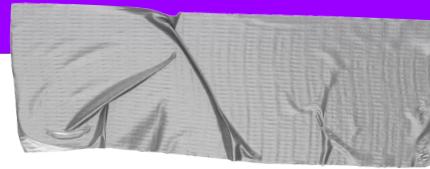
In the given code, the stream is sequential, meaning the combiner function doesn't have any effect.

The supplier function creates a `StringBuilder` instance with the initial value of `c` .

The accumulator function then appends two elements in the stream to this builder one by one,

As a result, the final value of the string builder is `cab` .

[https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/stream/Stream.html#collect\(java.util.functionSupplier,java.util.functionBiConsumer,java.util.functionBiConsumer\)](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/stream/Stream.html#collect(java.util.function.Supplier,java.util.function.BiConsumer,java.util.function.BiConsumer))



## Managing Concurrent Code Execution

Given:

```
ExecutorService service = Executors.newFixedThreadPool( 2 );
```

```
Runnable task = () -> System.out.println( "Task is complete" );
```

```
service.submit( task );
```

```
service.shutdown();
```

```
service.submit( task );
```

What happens when executing the given code fragment?

- It prints 'Task is complete once', then exits normally
- It prints 'Task is complete twice', then exits normally
- It exits normally without printing anything to the console
- It prints 'Task is complete' once and throws an exception
- It prints 'Task is complete' twice and throws an exception

# It prints Task is complete once and throws an exception

After an `ExecutorService` shuts down, it doesn't accept any more tasks.

Submitting tasks at this point will lead to a `RejectedExecutionException`.

Notice the first task is submitted before the `shutdown` method is called.

Therefore, the message `Task is complete` is always printed out once.

<https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/concurrent/ExecutorService.html>



<https://bit.ly/javaOCP>