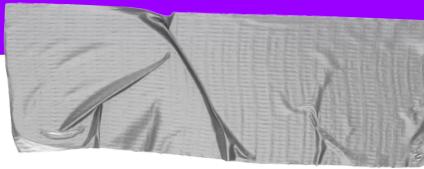




Java Certified #8

A question lead guide to prepare Java certification



Handling Date, Time, Text, Numeric and Boolean Values

Given:

```
String textBlock = """"
```

```
j \
```

```
a \t
```

```
v \s
```

```
a \
```

```
""";
```

```
System.out.println(textBlock.length());
```

What is the output?

- 10
- 11
- 12
- 14

12

The answer is 12.

Let's decompose this text block.

The first line is '`j \`': a letter '`j`', a space '', and an escaping end line '`\`' (so no break line).
So this first line has a length of `2`.

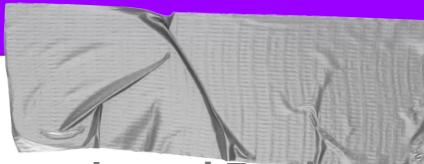
The second line is '`a \t`': a letter '`a`', a space '', a tab '`\t`', and a break line '`\n`' (implicit).
So this second line has a length of `4`.

The third line is '`v \s`': a letter '`v`', a space '', a space '`\s`', and a break line '`\n`' (implicit).

So this second line has a length of `4`.

The fourth line is '`a \`': a letter '`a`', a space '', and an escaping end line '`\`' (so no break line).
So this second line has a length of `2`.

Oracle Java text block documentation: <https://docs.oracle.com/en/java/javase/15/text-blocks/index.html>



Handling Date, Time, Text, Numeric and Boolean Values

Given:

```
Period p = Period.between( LocalDate.of( 2023, Month.MAY, 4 ), LocalDate.of( 2024, Month.MAY, 4 ) );
```

```
System.out.println( p );
```

```
Duration d=Duration.between(LocalDate.of( 2023, Month.MAY, 4 ), LocalDate.of( 2024, Month.MAY, 4 ) );
```

```
System.out.println( d );
```

What is the output?

- P1Y
- PT8784H
- PT8784H
P1Y
- UnsupportedTemporalTypeException
- P1Y
UnsupportedTemporalTypeException

P1Y

UnsupportedTemporalTypeException

The output is:

P1Y

```
java.time.temporal.UnsupportedTemporalTypeException: Unsupported unit: Seconds  
    at java.base/java.time.LocalDate.until(LocalDate.java:1636)  
    at java.base/java.time.Duration.between(Duration.java:489)
```

Period works well with LocalDate, while Duration no.

Because duration works with Time and according to JavaDoc:

"The specified temporal objects must support the SECONDS unit. For full accuracy, either the NANOS unit or the NANO_OF_SECOND field should be supported."

Duration javadoc:

<https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/time/Duration>

Period javadoc:

<https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/time/Period>



Packaging and Deploying Java Code

Consider the following methods to load an implementation of MyService using ServiceLoader. Which of the methods are correct? (Choose all that apply)

- `MyService service = ServiceLoader.load(MyService.class).findFirst().get();`
- `MyService service = ServiceLoader.load(MyService.class).iterator().next();`
- `MyService service = ServiceLoader.services(MyService.class).getFirstInstance();`
- `MyService service = ServiceLoader.getService(MyService.class);`

```
MyService service = ServiceLoader.load(MyService.class).findFirst().get();  
MyService service = ServiceLoader.load(MyService.class).iterator().next();
```

Option "ServiceLoader.load(MyService.class).findFirst().get();"

is correct because it uses `ServiceLoader.load()` to find the first available service provider for `MyService` and retrieves it if present.

Option "ServiceLoader.load(MyService.class).iterator().next();"

is correct as it uses `ServiceLoader.load()` to get an iterator over available service providers for `MyService` and retrieves the next element if available.

Option "ServiceLoader.services(MyService.class).getFirstInstance();"

is incorrect because there is no `services()` method in `ServiceLoader` class that provides a `getFirstInstance()` method.

Option "ServiceLoader.getService(MyService.class);"

is incorrect because `ServiceLoader` does not have a method named `getService()`.

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



<https://bit.ly/javaOCP>