



Java Certified #12

A question lead guide to prepare Java certification



Handling Date, Time, Text, Numeric and Boolean Values

What do the following print?

```
public class DividedDuration {  
    public static void main( String[] args ) {  
        var day = Duration.ofDays(2);  
        System.out.print(day.dividedBy( 8 ));  
    }  
}
```

- PToD
- PToH
- PT6H
- An exception is thrown.
- Compilation fails.

PT6H

The correct answer is indeed **PT6H**.

Explanation:

1. **Duration.ofDays(2):**

- The `Duration.ofDays(2)` method creates a `Duration` object representing **2 days**.
- Internally, this is stored as **48 hours** (2 days × 24 hours/day).

2. **The `dividedBy(int divisor)` Method:**

- This method divides the duration by the provided divisor.
- In this case: 48 hours ÷ 8 = 6 hours

3. **Result:**

- The result is a new `Duration` object representing **6 hours**.
- This is printed in the ISO-8601 duration format: `PT6H`, where:
 - `P` indicates a period.
 - `T` separates the time component.
 - `6H` represents 6 hours.

4. **Code Output:**

- The program successfully prints:
PT6H



Working with Streams and Lambda expressions

What do the following print?

```
public class StreamReduce {  
    public static void main( String[] args ) {  
        Stream<String> stream= Stream.of("J","a","v","a");  
        System.out.print(stream.reduce( String::concat ));  
    }  
}
```

- Java
- null
- Optional[Java]
- Compilation fails

Optional[Java]

Explanation:

1. **Stream.reduce() Without an Identity:**

- The `reduce` method is called without an identity value. When this happens, the result is wrapped in an `Optional`. This is done to handle the case where the stream is empty, in which case `Optional.empty()` is returned.
- The method signature being used here is:
`Optional<T> reduce(BinaryOperator<T> accumulator);`

2. **String::concat:**

- The `String::concat` method is passed as the accumulator. It concatenates the elements of the stream in the order they are encountered.
- The stream contains the elements "J", "a", "v", and "a", so they are concatenated to form "Java".

3. **Output:**

- Since the result of `reduce` is wrapped in an `Optional`, the output is:
`Optional[Java]`

4. **Key Observations:**

- If the stream were empty, the result would be `Optional.empty()`.
- If an identity value (e.g., "") had been provided, the output would be a plain string (Java).

Code Behavior Recap:

The program processes the stream and concatenates its elements, printing the result wrapped in an `Optional`.



Handling Date, Time, Text, Numeric and Boolean Values

Which StringBuilder variable fails to compile?

```
public class StringBuilderInstantiations {  
    public static void main( String[] args ) {  
        var stringBuilder1= new StringBuilder();  
        var stringBuilder2 = new StringBuilder(10);  
        var stringBuilder3 = new StringBuilder("Java");  
        var stringBuilder4 = new StringBuilder(new char[]{'J','a','v','a'});  
    }  
}
```

- **stringBuilder1**
- **stringBuilder2**
- **stringBuilder3**
- **stringBuilder4**
- **None of them**

stringBuilder4

Explanation:

1. `stringBuilder1 = new StringBuilder();`
 - This is valid.
 - Creates a `StringBuilder` with an initial capacity of 16 (default capacity).
2. `stringBuilder2 = new StringBuilder(10);`
 - This is valid.
 - Creates a `StringBuilder` with an initial capacity of 10.
3. `stringBuilder3 = new StringBuilder("Java");`
 - This is valid.
 - Creates a `StringBuilder` initialized with the content "Java" and a capacity of $4 + 16 = 20$ (content length + default capacity).
4. `stringBuilder4 = new StringBuilder(new char[]{'J', 'a', 'v', 'a'});`
 - **This is invalid and fails to compile.**
 - The `StringBuilder` constructor does not accept a `char[]` as an argument. It only accepts a `String` argument or an integer for capacity.

Compilation Error for `stringBuilder4`:

The compiler will report:

Error: no suitable constructor found for `StringBuilder(char[])`

—



<https://bit.ly/javaOCP>