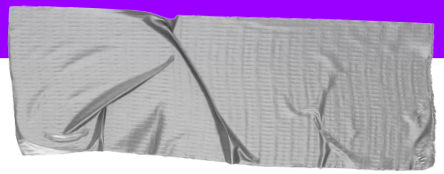
Java Certified #24

A question lead guide to prepare Java certification



Handling Exceptions

Given:

```
public class BoomBoom implements AutoCloseable {
   public static void main( String[] args ) {
      try ( BoomBoom boomBoom = new BoomBoom() ) {
         System.out.print( "bim " );
         throw new Exception();
      } catch ( Exception e ) { System.out.print( "boom " ); }
   }
   @Override
   public void close() throws Exception {
      System.out.print( "bam " );
      throw new RuntimeException();}}
```

What is printed?

- ➔ **bim boom bam**
- ➔ **bim bam boom**
- ➔ **bim boom**
- ➔ **bim bam followed by an exception**
- ➔ **Compilation fails.**

bim bam boom

1. Entering `try` block

- `"bim "` is printed.
- An `Exception` is thrown.

2. Before handling the exception, `close()` is called

- The `try-with-resources` mechanism **ensures that the `close()` method is called before the `catch` block executes.**
- Inside `close()`, `"bam "` is printed.
- The `close()` method **throws a `RuntimeException`**, but this does **not replace the original `Exception` from the `try` block.** Instead, it becomes a **suppressed exception**.

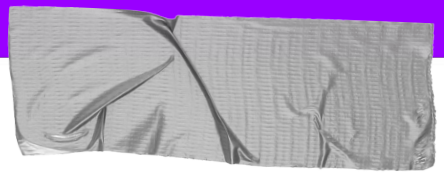
3. Handling the Exception in `catch`

- The `catch (Exception e)` block **only catches the original `Exception`** thrown inside `try`.
- `"boom "` is printed.

4. What happens to the `RuntimeException` from `close()`?

- **It is suppressed and does not propagate.**
- The JVM allows the `catch (Exception e)` block to complete execution normally without rethrowing suppressed exceptions.
- **No exception is printed or thrown at runtime.**

Final Printed Output: bim bam boom



Handling Exceptions

Given:

```
Integer frenchRevolution = 1789;
Object o1 = new String( "1789" );
Object o2 = frenchRevolution;
frenchRevolution = null;
Object o3 = o2.toString();
System.out.println( o1.equals( o3 ) );
```

What is printed?

- ➔ **true**
- ➔ **false**
- ➔ **2,Teddy Riner,Mouse,1,15.99**
  3,Sebastien Loeb,Monitor,1,199.99
- ➔ **A NullPointerException is thrown.**
- ➔ **A ClassCastException is thrown.**
- ➔ **Compilation fails.**

true

Let's analyze the code step by step to determine the output.

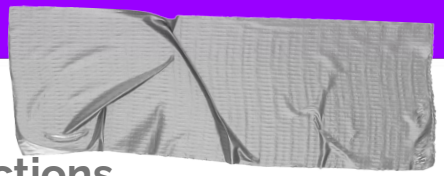
Code Breakdown:

```
Integer frenchRevolution = 1789;  // Autoboxing: Integer.valueOf(1789)

Object o1 = new String("1789");   // A String object with "1789"

Object o2 = frenchRevolution;     // o2 refers to the Integer object 1789

frenchRevolution = null;          // The original reference is set to null, but o2 still holds the Integer 1789

Object o3 = o2.toString();        // o2 is an Integer, so o3 = "1789" (String)

System.out.println(o1.equals(o3)); // Comparing o1 ("1789") with o3 ("1789")
```

Key Points:

1. `o1.equals(o3)` is equivalent to `"1789".equals("1789")`, since:
   - `o1` is a `String` with `"1789"`.
   - `o3` is also a `String` with `"1789"` (converted via `Integer.toString()`).
   - The `String.equals()` method compares values, and since both strings are `"1789"`, it returns **true**.

Final Output: `true`



Working with Arrays and Collections

Given:

```
var cabarets = new TreeMap<>();
cabarets.put( 1, "Moulin Rouge" );
cabarets.put( 2, "Crazy Horse" );
cabarets.put( 3, "Paradis Latin" );
cabarets.put( 4, "Le Lido" );
cabarets.put( 5, "Folies Bergère" );
System.out.println( cabarets.subMap( 2, true, 5, false ) );
```

What is printed?

- ➔ **{2=Crazy Horse, 3=Paradis Latin, 4=Le Lido, 5=Folies Bergère}**
- ➔ **var d[] = new int[4];**
- ➔ **{}**
- ➔ **An exception is thrown at runtime.**
- ➔ **Compilation fails.**

{2=Crazy Horse, 3=Paradis Latin, 4=Le Lido}

1. Understanding `TreeMap.subMap(fromKey, fromInclusive, toKey, toInclusive)`

- The `subMap(K fromKey, boolean fromInclusive, K toKey, boolean toInclusive)` method returns a **view** of a portion of the map.
- It includes entries **starting from `fromKey`** (if `fromInclusive` is `true`).
- It includes entries **up to `toKey`** (if `toInclusive` is `true`).

2. Extracting the Relevant Entries

The `TreeMap` is sorted in ascending order:

{1=Moulin Rouge, 2=Crazy Horse, 3=Paradis Latin, 4=Le Lido, 5=Folies Bergère}

Now, applying: cabarets.subMap(2, true, 5, false)

- **Start from key `2`**, **including** it (`true`).
- **Stop before key `5`**, **excluding** it (`false`).
- This includes the keys **2, 3, and 4**.

The resulting subMap: {2=Crazy Horse, 3=Paradis Latin, 4=Le Lido}

Final Answer:

`{2=Crazy Horse, 3=Paradis Latin, 4=Le Lido}`

udemy Pro

https://bit.ly/javaOCP

https://bit.ly/jfullstack27