Interview Questions

JAVA-111: Session 8

Answering interview questions is crucial in your journey of applied learning. Review them to ensure your understanding of important topics covered in the previous session and to prepare yourself for upcoming challenges. Remember that it's important to answer these questions on your own before viewing the solution. The solutions are hyperlinked to community posts on our platform.

**Note: The questions below have been sourced from previous interviews**

Questions

1. [Why is String immutable in Java?](https://crio.do/learn/crio-community/topic/why-is-string-immutable-in-java/165982)
2. [Discuss the key differences between StringBuilder and String classes. When would you use one over the other?](https://crio.do/learn/crio-community/topic/discuss-the-key-differences-between-stringbuilder-and-string-classes-when-would-you-use-one-over-the-other/163953)
3. [What is the difference between compareTo() and equals()?](https://crio.do/learn/crio-community/topic/what-is-the-difference-between-compareto-and-equals/188087)
4. [What is Java String Pool?](https://crio.do/learn/crio-community/topic/what-is-java-string-pool/252722)
5. [Explain the substring() method in Java.](https://crio.do/learn/crio-community/topic/explain-the-substring-method-in-java/252775)
6. [How do you convert a String to an integer in Java?](https://crio.do/learn/crio-community/topic/how-do-you-convert-a-string-to-an-integer-in-java/252774)

**1. Why is String immutable in Java?**

**Answer:**  
Strings in Java are **immutable** because once a String object is created, its value **cannot be changed**. This immutability offers several benefits:

* **Security**: Used in network connections, file paths, etc., immutable strings prevent tampering.
* **Thread Safety**: No need to synchronize String operations since they can't be modified.
* **String Pooling**: Immutability allows the JVM to **internally reuse** common strings from the **String Pool**, improving performance and memory usage.
* **Caching of hashcode**: Hashcode is cached for immutable strings which makes them fast in collections like HashMap.

**2. Discuss the key differences between StringBuilder and String classes. When would you use one over the other?**

| **Feature** | **String** | **StringBuilder** |
| --- | --- | --- |
| Mutability | Immutable | Mutable |
| Thread-Safe | Yes (due to immutability) | No (use StringBuffer if needed) |
| Performance | Slower for concatenation (creates new objects) | Faster (modifies in-place) |
| Use Case | Fixed text, secure/constant data | Dynamic string operations in loops or intensive updates |

**Use String** when:

* Working with constant or secure text.
* You don’t need to modify the string.

**Use StringBuilder** when:

* You need to **build/modify** strings frequently (e.g., loops, concatenation).

**3. What is the difference between compareTo() and equals()?**

| **Method** | **Purpose** | **Return Type** | **Usage** |
| --- | --- | --- | --- |
| equals() | Checks **value equality** | boolean | str1.equals(str2) |
| compareTo() | Compares **lexicographically** | int | str1.compareTo(str2) |

* equals() returns true if two strings have the **same content**.
* compareTo() returns:
  + 0 if strings are equal,
  + a **positive** number if str1 > str2,
  + a **negative** number if str1 < str2.

**4. What is Java String Pool?**

**Answer:**  
The **Java String Pool** is a special memory region in the heap where **interned strings** are stored.

* When you create a string using a **string literal**, Java checks the pool:
  + If the string exists, the **reference is reused**.
  + If not, it is added to the pool.

Example:

java

CopyEdit

String s1 = "Java";

String s2 = "Java";

System.out.println(s1 == s2); // true (same reference)

This reduces memory usage and improves performance.

**5. Explain the substring() method in Java.**

**Answer:**  
The substring() method returns a new string that is a **part of the original string**.

Syntax:

java

CopyEdit

String sub = original.substring(startIndex, endIndex);

* startIndex is **inclusive**, endIndex is **exclusive**.
* If you omit endIndex, it returns from startIndex to the **end of the string**.

Example:

java

CopyEdit

String s = "HelloWorld";

System.out.println(s.substring(0, 5)); // Hello

System.out.println(s.substring(5)); // World

**6. How do you convert a String to an integer in Java?**

**Answer:**  
Use the Integer.parseInt() or Integer.valueOf() methods.

java

CopyEdit

String number = "123";

int num1 = Integer.parseInt(number); // returns primitive int

Integer num2 = Integer.valueOf(number); // returns Integer object

* Make sure the string represents a valid integer, or it throws NumberFormatException.