1. Threading
2. JRE JDK JVM
3. JDBC
4. Collections
5. OOPS
6. Singleton
7. Java 8 features
8. Comparator vs comparable
9. Files
10. Synchronization vs Asynchronization
11. StringBuilder vs StringBuffer
12. Equals and == (Heap memory, String pool)
13. Date Time
14. Java Regular Expression
15. Exception Handling
16. Enum
17. Inner class
18. Typecasting
19. Access modifiers
20. Java IO
21. Primitive vs Non primitive
22. Wrapper class Auto Boxing vs Un Boxing
23. Design Pattern
24. Serialization
25. Why Strings are immutable

Strings are immutable in many programming languages, including Java, Python, and C#. There are several reasons why strings are designed to be immutable:

1. \*Thread Safety\*: Immutability ensures that strings cannot be modified after creation. This property makes strings inherently thread-safe because multiple threads can safely read the same string without worrying about concurrent modifications.

2. \*Caching\*: Since strings are immutable, they can be cached and reused. This optimization can improve performance and reduce memory usage, especially in scenarios where the same string is used multiple times.

3. \*Security\*: Immutable strings prevent unintended modifications, which can be crucial in security-sensitive applications. For example, if strings representing sensitive data (such as passwords or cryptographic keys) were mutable, they could be inadvertently modified, compromising security.

4. \*Hashing\*: Immutable strings allow for efficient hashing and comparison operations. Once a string is hashed, its hash value remains constant, simplifying hash table implementations and string comparison algorithms.

5. \*Predictability\*: Immutability leads to predictable behavior. Since strings cannot change, developers can rely on their values remaining constant throughout the program's execution, which reduces complexity and potential bugs.

6. \*Optimizations\*: Immutable strings enable compiler and runtime optimizations. For example, string interning techniques can be used to optimize memory usage by reusing identical string literals.

Overall, immutability provides several benefits in terms of performance, thread safety, security, and predictability, making it a common design choice for strings in many programming languages.