

**Why Is String Immutable in Java?**

The key benefits of keeping this class as immutable are caching, security, synchronization, and performance.

**Security :** The *String* is widely used in Java applications to store sensitive pieces of information like usernames, passwords, connection URLs, network connections, etc. It’s also used extensively by JVM class loaders while loading classes.Hence securing *String* class is crucial regarding the security of the whole application in general.

**Synchronization/ Thread Safe :** Being immutable automatically makes the *String* thread safe since they won’t be changed when accessed from multiple threads. Hence **immutable objects, in general, can be shared across multiple threads running simultaneously. They’re also thread-safe** because if a thread changes the value, then instead of modifying the same, a new *String* would be created in the *String* pool. Hence, *Strings* are safe for multi-threading.

**Hashcode Caching :** Since *String* objects are abundantly used as a data structure, they are also widely used in hash implementations like *HashMap*, *HashTable*, *HashSet*, etc. When operating upon these hash implementations, *hashCode()* method is called quite frequently for bucketing.

The immutability guarantees *Strings* that their value won’t change. So **the***hashCode()***method is overridden in***String***class to facilitate caching, such that the hash is calculated and cached during the first***hashCode()***call and the same value is returned ever since.**

**This, in turn, improves the performance of collections that uses hash implementations when operated with***String***objects.**

On the other hand, mutable *Strings* would produce two different hashcodes at the time of insertion and retrieval if contents of *String* was modified after the operation, potentially losing the value object in the *Map*.

**Performance/Saves heap memory size** : *String* pool exists because *Strings* are immutable. In turn, it enhances the performance by saving heap memory and faster access of hash implementations when operated with *Strings.*

Since *String* is the most widely used data structure, improving the performance of *String* have a considerable effect on improving the performance of the whole application in general.