**Immutable**

1. **Does Declaring an object "final" makes it immutable?**

Only declaring primitive types as final makes them immutable. Making objects final means that the object handler cannot be used to target some other object but the object is still mutable.

1. **Why Char array is preferred over String for storing password?**

String is immutable in java and stored in String pool. Once it’s created it stays in the pool until unless garbage collected, so even though we are done with password it’s available in memory for longer duration and there is no way to avoid it. It’s a security risk because anyone having access to memory dump can find the password as clear text.

1. **Why String is popular HashMap key in Java?**

Since String is immutable, its hashcode is cached at the time of creation and it doesn’t need to be calculated again. This makes it a great candidate for key in a Map and it’s processing is fast than other HashMap key objects. This is why String is mostly used Object as HashMap keys.

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

When a string is created and if the string already exists in the pool, the reference of the existing string will be returned, instead of creating a new object. If string is not immutable, changing the string with one reference will lead to the wrong value for the other references.

1. **To Cache its Hashcode**

If string is not immutable, one can change its hashcode and hence not fit to be cached.

1. **Security**

String is widely used as parameter for many java classes, e.g. network connection, opening files, etc. Making it mutable might possess threats due to interception by the other code segment.

<https://www.programcreek.com/2013/04/why-string-is-immutable-in-java/>

1. **Immutable classes and how to write own immutable class**

<https://www.journaldev.com/129/how-to-create-immutable-class-in-java>

Immutable class are the class, whose object cannot be modified once created

**To create an immutable class in java, you have to do following steps.**

1. Declare the class as final so it can’t be extended.

(if you don’t declare class as final you need to make sure each and every method of class is declared as final ) class as final strong immutability , method as final weak immutability

1. Make all fields private so that direct access is not allowed.
2. Don’t provide setter methods for variables
3. Make all mutable fields final so that its value can be assigned only once.
4. Initialize all the fields via a constructor performing deep copy. (so that even if the actual assign changes clone variable don’t change) ex hashmap as member variable
5. Perform cloning of objects in the getter methods to return a copy rather than returning the actual object reference. (so that the variable cannot be changed) ex hashmap as member variable

**Benefits**

1. Immutable objects are by default thread safe, can be shared without synchronization in concurrent environment.
2. Immutable object simplifies development, because its easier to share between multiple threads without external synchronization.
3. Immutable object boost performance of Java application by reducing synchronization in code.
4. Another important benefit of Immutable objects is reusability, you can cache Immutable object and reuse them, much like String literals and Integers. You can use static factory methods to provide methods like valueOf(), which can return an existing Immutable object from cache, instead of creating a new one.

Apart from above advantages, immutable object has disadvantage of creating garbage as well. Since immutable object cannot be reused and they are just a use and throw. String being a prime example, which can create lot of garbage and can potentially slow down application due to heavy garbage collection, but again that's extreme case and if used properly Immutable object adds lot of value.