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

1. Declare the class as final so it can’t be extended.
2. Make all fields private so that direct access is not allowed.
3. Don’t provide setter methods for variables
4. Make all **mutable fields final** so that its value can be assigned only once.
5. Initialize all the fields via a constructor performing deep copy.
6. Perform cloning of objects in the getter methods to return a copy rather than returning the actual object reference.

S.O.L.I.D stands for:

When expanded the acronyms might seem complicated, but they are pretty simple to grasp.

* **S** - Single-responsiblity principle

*A class should have one and only one reason to change, meaning that a class should only have one job.*

* **O** - Open-closed principle

*Objects or entities should be open for extension, but closed for modification.*

* **L** - Liskov substitution principle

The Liskov Substitution Principle (LSP) is the one here that is most unique to object-oriented programming.

*Let****q(x)****be a property provable about objects of****x****of type****T.****Then****q(y)****should be provable for objects****y****of type****S****where****S****is a subtype of****T.***

All this is stating is that every subclass/derived class should be substitutable for their base/parent class.

* **I** - Interface segregation principle

*A client should never be forced to implement an interface that it doesn’t use or clients shouldn’t be forced to depend on methods they do not use.*

* **D** - Dependency Inversion Principle

*Entities must depend on abstractions not on concretions. It states that the high level module must not depend on the low level module, but they should depend on abstractions.*