SOLID Principle:

S(Single responsibility Principle - SRP) : class should be having one and only one responsibility

* Open a database connection
* Fetch data from database
* Write the data in an external file

The issue with this class is that it handles lot of operations. Suppose any of the following change happens in future.

* New database
* Adopt ORM to manage queries on database
* Change in the output structure

ex : db coonction data,save store data in one class , so break it down in three class

O: Open for extension and closed for modification :

Ex1 : new Service class creation in txn mobiquity

Ex 2: Payment type addition : so create interface Payment and implement this interface for adding new payment way ( vi Paytm , freecharge etc )

**L -(Liskov substitution principle -LSP)** :” Derived types must be completely substitutable for their base types”. Vehicle interface has method runWithFuel(); , Car , motorcycle, Etc . It will fail when Cycle will implement .

**I :( Interface Segregation Principle) :** “Clients should not be forced to implement unnecessary methods which they will not use”

Ex : Ipayment interface : saveCardData, saveOtherInfo , CardPayment : saveCardData , CashPayment : saveOtherInfo but no need to override saveCardData method implementation

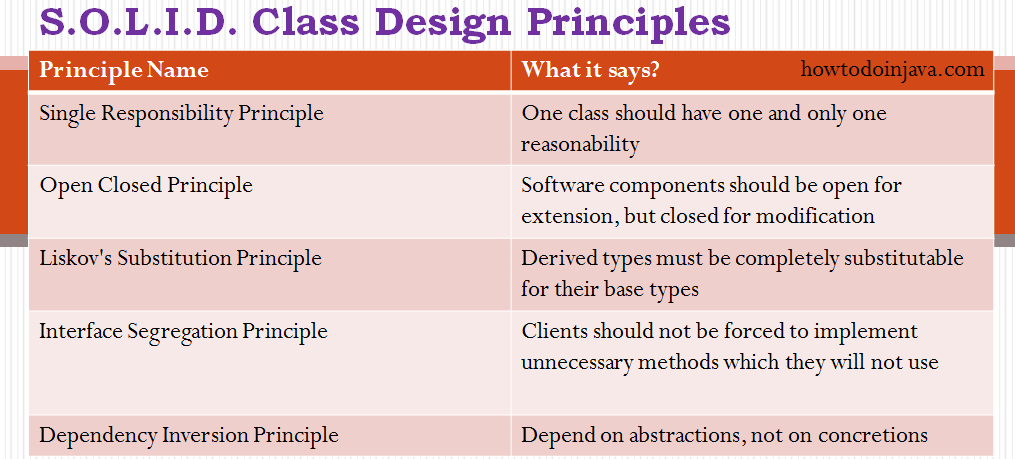
**D: (Dependency Inversion Principle) :** “Depend on abstractions, not on concretions”.

High-level modules should not depend on low-level modules. Both should depend on abstractions.

Abstractions should not depend on details. Details should depend on abstractions.

In the above code In spite of Injecting MySQLConnection class in PasswordReminder class but it depends on MySQLConnection.High-level module PasswordReminder should not depend on low-level module MySQLConnection.

<https://hackernoon.com/solid-principles-simple-and-easy-explanation-f57d86c47a7f>



Architecture : MVC, WEBAPI etc

For Good Application

1. Architecture

2. Design principle (SOLID, ops )

3. Design Pattern (Structural, creational, behavioral)

References :

<https://www.youtube.com/watch?v=HLFbeC78YlU&list=PL6n9fhu94yhXjG1w2blMXUzyDrZ_eyOme> ( for introduction)

<https://www.youtube.com/watch?v=hCsqBIyT1pI>

<https://howtodoinjava.com/best-practices/5-class-design-principles-solid-in-java/>

<https://springframework.guru/principles-of-object-oriented-design/liskov-substitution-principle/> LISKOV principle