Solid Principles

```
nSingle Responsibility:
Account. Java: A class should have Single Responsibility
outerate Expans Paccount &
  int money;
  Public get Account Numberci;
  Public Set Account Numberc);
  public get Namec);
  boplic setwamecs;
  Public int deposite); 11 Account class
                           Should not take
                          responsibility of
ransaction. Java:
                             Transaction class
Public class Transaction ?
    public int withdrawers
```

Good Example:

```
Account · Java:
Public class Account &
    int money;
    Public get Account Numberco;
   Public Set Account Numberc);
   public get Namec);
    public set mameco;
    <del>Public int deposites;</del> Il Account class
 3
                             Should not take
                             responsibility of
Transaction. Java:
                               Transaction class
 Public class Transaction ?
      public int deposition,
     public int withdrawch,
  3
```

```
OPEN/CLOSE:
  (Open for Extension, Closed for modification)
Bad Example: Calculatorisara:
Public class coloulators
  Switch (operation) ?
    case "add":
        return num it numz;
    Case " sub":
         return num 1 - num 2:
If we want to add another feature
i.e., Multiplication. then we are modifying
```

the existing code

This is not olcay.

Good Example: Galculator. Java: Public class calculator ? public void perform cint num, cint num, Operation operation) Operation. Perform (num, num2); Operation. Java: Public interface Operation { public world perform (rht num, int num Add Operation. Java: public class Addaperation implements operation (@ override public Yoid perform (int a, inb b), return a +b: Sub operation. Java: public class subaperation implements operation (@override

public Yoid perform (int a, inb b);
return a-b; we can add trulliplication

```
LISKOV Substitution Principle:
       Both Parent & child class pointers should
 be interchangeable.
Bad Example:
Loan Payment · Java:
Public interface LoanPayment &
     public world do Paymenter;
     Public void Forceclose (oanc);
3
Homeloan Payment. Java:
 public class Homeloan Payment implements
                                   Loan Payment
   public world do Paymentes;
    public Void do Repaymentes?
   Public void Forcectose (oanc);
Credit Loan Payment. Java:
Public class credit-Loan Pogment Implements
                                  Loanpayment
   public world do Paymentes:
   Public Void do Repaymentes (throw Error;)
   Public void Forceclose Coancil
```

3

throw Errors

```
Loanclosure service. Java:
Public Class Loan Closure Service &
  LoanPayment homeloan = new
                               Home Loan ();
   hometoan. Forceclose Coan C) 11 no error;
  LoanPayment creditioan = new
                              creditLoanc?
   creditioan. force close Coan () 1
                          // Frror.
Violation of Liskov Substitution Principle
Good Example:
   Loan Payment · Java:
   Public interface LoanPayment &
```

Public (rold do Paymente);

Secure Loan Java:

Public Interface Secure Loan extends

Loan Payment of

Public Void force Close Loan();

}

```
Homeloan Payment. Java:
 public class Homeloan Payment implement
                                 Secretoan
    public world do Paymentes;
   Public void Forcectose (oanc);
Credit Loan Payment. Java:
                                   Implement
 Public class credit-Loan Pogment
                                   Loanpoyment
   Public void de Paymentes;
Loan closure service. Java
public class Loanclosure service 2
  public Secureloan secureloan?
  public Loanclosureservice C Secureloun
                                  secureloun)
     this. secure loan: secure loan;
  secure Loun. Force close (oanc);
```

Interface Segregation:

Donot bottlehecic the client with multiple implementation. Segregate the interpaces with corresponding functionally

Bad Example:

```
public interface DAOInterface i
public void Open connections in
public void create Records;
public void delete Records;
public void OpenFiles;
```

public class absimplemention implements

PHOINTERFACE (

```
public void Open connections in Public void create Records; public void delete Records; public void Open Files;

DB don't need Files!
```

```
public class file, Implemention implements
   DAO Interface (
     public void Open Filec);
     public void create Recordes;
      public vois delek Recordes;
      public Void Open connectionin
          File don't need DR
It's a bottlenecyl; Therefore, let
segregate interfaces.
 public interface operations &
     public void create Recorde? 1
     public void derete Recorderi
 Public interface DBConnection?
      public void open connection () 1
 public interface fice Operations
     Public boid openfile()1
```

```
public class DAO implement
               fle operation, operations
public class PBO implements
               DB connection, operations
Dependency Injection:
     one class should not have
 dependency internally
  Bad Example: Calculatorisara:
  Public class coloulators
     Switch (operation) {
       case "add":
          Add operation 9 = new Add operation ()
                 a. perform
       Case " sub" !
        suboperation as new suboperation()
             a. perform
      calculator depends on Addoperation
        9 suboperation class
```

```
Good Example:
Galculator. Java:
 public ciass calculators
    public void perform cint num; cintnumz,
                            Operation operation)
      Operation perform (num, num2),
 3
Operation. Java:
 Public interface Operation {
     public woid perform ( the num, int num, int num,
Add Operation. Java:
 public class Addaperation implements
                           operation (
    @ override
    public Yoid performitht a, inb b),
        return a +b:
 Sub operation. Java:
   public class subaperation implements
```

Public class subaperation implements

Operation (

public Yoid perform(int a, inb b);

return a b;

No dependency,