


```

class A {
    private B b;

    A() {
        this.b = new D();
    }
}

```

interface

class A itself is responsible for creating object of B

No DI

```

class A {
    private B b;

    A(B _b) {
        this.b = _b;
    }
}

```

Now class A, is not responsible for creating an object of B, someone will separately create it and provide it to A.

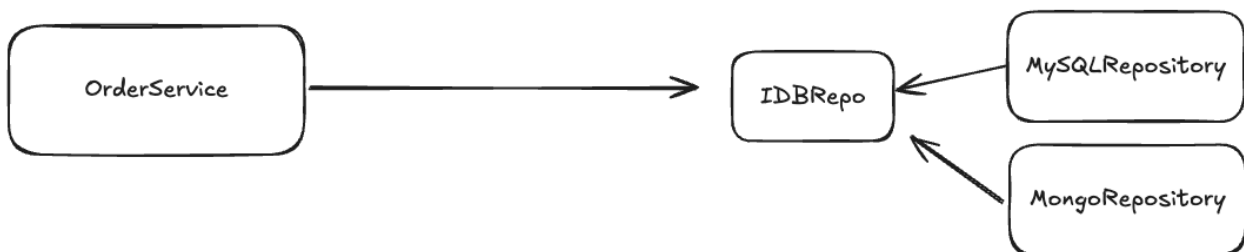
DI

class C implements B

class D implements B

A newObj = new A(new C());

A newObj1 = new A(new D());



class OrderService {

.....

```
private final IDBRepo repo;

public void doSomething() {
    ....
    repo.getAllOrders();
}

}
```

```
class A {
    public B b;

    constructor() {
        this.b = new C();
    }

    public void doSomething() {
        this.b = new D();
    }
}
```

Controller ----> Service ---> Gateway



```
11 @Bean
12 public Retrofit retrofit() {
13     return new Retrofit.Builder()
14         .baseUrl("https://fakestoreapi.in/api")
15         .addConverterFactory(GsonConverterFactory.create())
16         .build();
17 }
18
19 @Bean
20 public FakeStoreCategoryApi fakeStoreCategoryApi(Retrofit retrofit) {
21     return retrofit.create(FakeStoreCategoryApi.class);
22 }
```



This method is now responsible for providing an object of `FakestoreCategoryApi`