Interface

Ett interface är en koppling mellan dig och tekniken. Du måste veta hur interfacet fungerar men inte hur det som ligger bakom fungerar.

Använder du en app för att leta spara kontakter måste du veta hur man lägger in personer, tar bort personer o.s.v.

Du har ett interface mellan dig och tekniken. Om den som gjort programmet använder databas, filer, XML eller annat behöver du inte bry dig om.

Samma sak med klasser i Java eller t.ex. C#.

Exempel

Du gör ett interface för att hantera kontakter.

```
public interface MyContacts {
   void addContact();
}
```

Ett företag gör en klass som använder interfacet

```
public class MyClass implements MyContacts
{
   public void addContact()
   {
         enLista.add(enPerson)
   }
}
```

Ett annat företag gör en annan klass som använder interfacet

```
public class AnotherClass implements MyContacts
{
   public void addContact()
   {
         String sql = "INSERT INTO Customers (CustomerName, Country)
          VALUES ('Kalle','Norway')";
   }
}
```

Du vill nu använda någon av de här klasserna.

Vilken du än använder så vet du att du kan använda metoden addContact() för att lägga till kontakt.

Så du kan skapa använda klassen MyClass och skriva:

MyClass minKlass = new MyClass();

Och sedan:

minKlass.addContact();

Du vet att den metoden finns och att den lägger till en ny kontakt.

Du kan också använda den andra klassen.

AnotherClass minKlass = new AntotherClass();

Och sedan:

minKlass.addContact();

Du vet att addContact lägger till kontakter. Du vet inte hur men du vet att metoden finns och hur den fungerar.

De som gjort klasser enligt interfacet kan ändra hur de fungerar internt bara de fortsätter fungera som du tror.