Adapter Design Pattern

- Imagine you have to use a method which has been defined in the interface.And it has not been implemented.
- But the same process is in another class and you don't have the source code of that class.
- In this type of situation you can use this design pattern.Create another class(adapter class) and implement the method in the interface using that class which you don’t have the source code.
- Example code in below

```
package com.adapter;

public interface Cable
{
   void charge(String chargingNote);
}
```

Interface

```
package com.adapter;

public class Iphone
{
   private Cable cable;

   public Cable getCable()
   {
       return cable;
   }

   public void setCable( Cable cable )
   {
       this.cable = cable;
   }
   public void chargePhone(String chargingNote){
       cable.charge( chargingNote );
   }
}
```

You need to charge this Iphone

```
package com.adapter;

public class TypeCcable
{
   public void typeCChargingNote(String chargingNote){
       System.out.println(chargingNote);
   }
}
```

But you don’t have a lightning cable support adapter,but you have a Type C cable and an adapter.

```
package com.adapter;

public class ChargingAdapter implements Cable
{
   TypeCcable typeCcable = new TypeCcable();
   @Override
   public void charge(String chargingNote){
       typeCcable.typeCChargingNote( chargingNote );
   }
}
```

Using the adapter to connect Type C cable with the Iphone

```
//Adapter design pattern
Iphone iphone = new Iphone();
Cable cable = new ChargingAdapter();
iphone.setCable( cable );
iphone.chargePhone( "Phone is charging" );
```

Create a cable connecting Type C cable and type C to lightning adapter.Now you can charge a Iphone using default cable