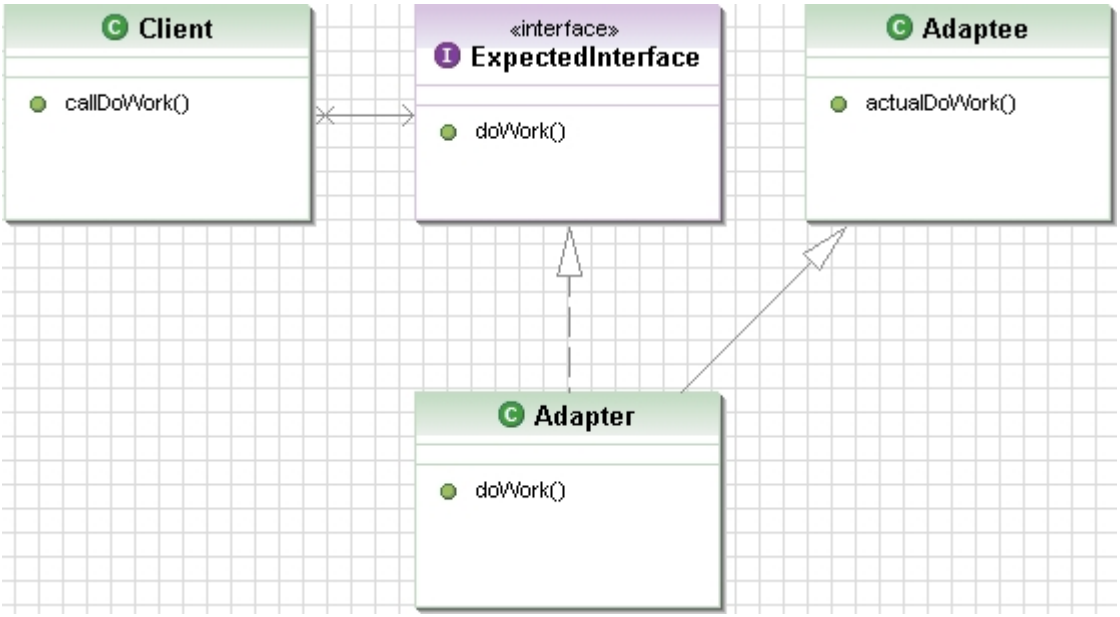
Adapter pattern

Adapter pattern เป็นดีไซน์แพตเทิร์น ที่ใช้เชื่อมการทำงานระหว่างคลาสที่เรียก และคลาสที่ถูกเรียก ซึ่ง Adapter pattern นั้นเป็นตัวช่วย ให้สองอินเตอร์เฟสที่เข้ากันไม่ได้ทำงานร่วมกันได้ ในการใช้รูปแบบการออกแบบอะแดปเตอร์

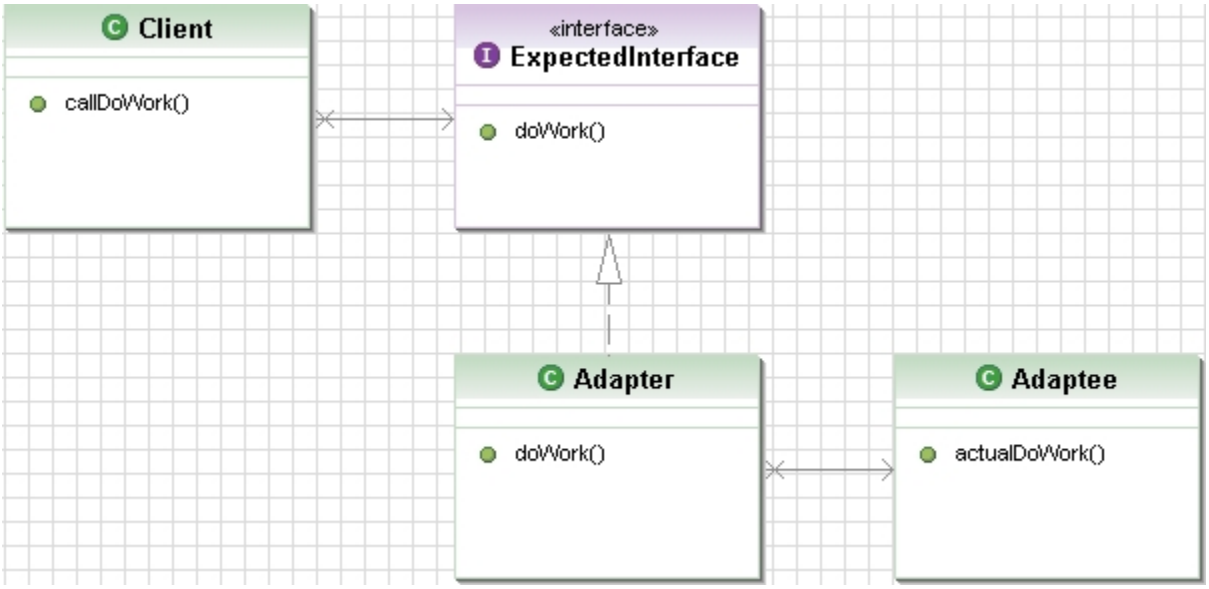
การนำไปใช้ มี 2 แบบ

การสร้างคลาสสำหรับทำหน้าที่เป็นอะแด็ปเตอร์สามารถทำได้สองวิธีดังนี้

1.วิธีการแบบ inheritance ให้คลาสอะแด็ปเตอร์ซับคลาสจากคลาสที่จะถูกใช้งานจริงและให้อิมพลิเมนท์อินเตอร์เฟสตามที่คลาสผู้เรียกคาดหวัง ความสัมพันธ์ระหว่างผู้แปลงและผู้ถูกแปลงจะเป็นในแบบ is-a ดังภาพประกอบ

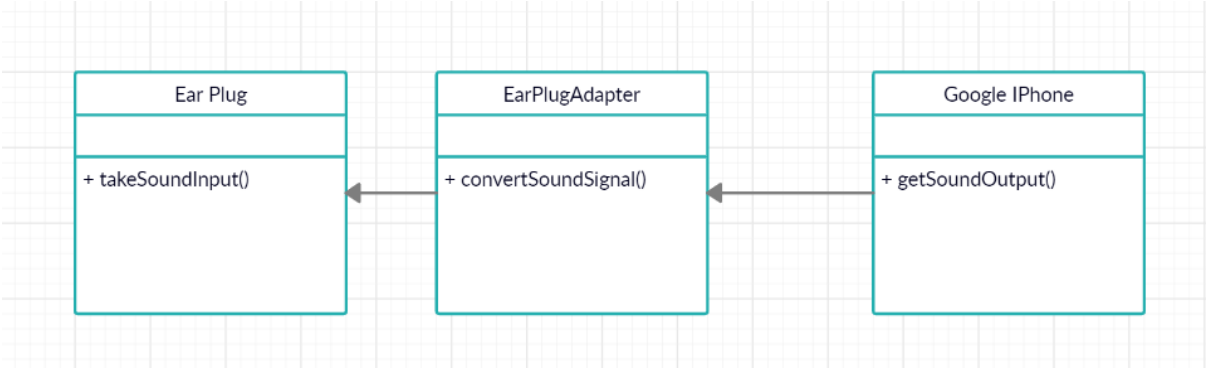


2.วิธีการแบบ delegation ให้คลาสอะแด็ปเตอร์อิมพลิเมนท์อินเตอร์เฟสตามที่คลาสผู้เรียกคาดหวังแต่ไม่ต้องซับคลาส คลาสอะแด็ปเตอร์เรียกเมธอดของคลาสที่จะถูกใช้งานผ่านทางอ็อบเจกต์ของคลาสที่ถูกใช้งานนั้น ความสัมพันธ์ระหว่างผู้แปลงและผู้ถูกแปลงจะเป็นในแบบ has-a ดังภาพ



1.1 EarPlugAdapter เวอร์ชั่นที่ 1

กรณีตัวอย่าง เราจะสร้างหูฟัง(ที่อุดหู Ear plug) ที่รับสัญญาณเสียงจาก Iphone ได้ แต่ Googlephone ไม่สามารถที่เข้าไปสัญญาณรับเสียงของ Ear plug นั้นได้ ไคลแอนซ์ (Client)จึงสั่งให้สร้างตัวแปลง Adapter เพื่อให้ googlephone สามารถใช้งาน Earplug ร่วมกับ iphone ได้



```java
package v1;

public class Client {
    Run | Debug
    public static void main (String[] args) {
        System.out.println("Adapter Design Pattern Example V1 ");

        // Created instances of devices
        Iphone iPhone = new Iphone();
        Googlephone googlePhone = new Googlephone();
        EarPlug earPlug = new EarPlug();

        // Ear plug is able to take iPhone sound signal
        String soundSignal = iPhone.getSoundOutput();
        earPlug.takeSoundInput(soundSignal);

        // Ear plug is not able to take google phone sound signal
        Integer soundSignals = googlePhone.getSoundOutput();

        // Created adapter to convert signals expected by client
        EarPlugAdapter earPlugAdapter = new EarPlugAdapter();
        earPlug.takeSoundInput(earPlugAdapter.convertSoundSignal(soundSignals));

    }
}
```

เราจะสร้าง Iphone ที่ส่งสัญญาณออกมาเป็น 1001010101 เป็นตัวแปรสตริง ส่งเข้าไปยัง Earplug จากนั้น Earplug จะได้รับสัญญาณของ Iphone เพื่อเล่นเสียง จากนั้นเราสร้าง Googlephone ที่สามารถส่งสัญญาณออกมาเป็น 1001010101 เช่นกันแต่เป็นตัวแปรชนิด Integer ซึ่งทำให้ Earplug ไม่สามารถรับสัญญาณนั้นได้ ดังนั้นจึงสร้าง EarplugAdapter เป็นตัวแปลงสัญญาณ เพื่อให้ Earplug นั้นเข้ามารองรับ ซึ่งจะรับค่า Integer ออกมาแปลงเป็น String กลับไป

```java
package v1;


public class Iphone {

    public String getSoundOutput(){

        return "1001010101";
    }
}
```

```java

package v1;
public class Googlephone {

    public Integer getSoundOutput(){

        return 1001010101;
    }
}
```

```java
package v1;

public class EarPlug {

    public void takeSoundInput(String inputSignal){
        System.out.println("playing input signal");
        System.out.println(inputSignal);
    }

}
```

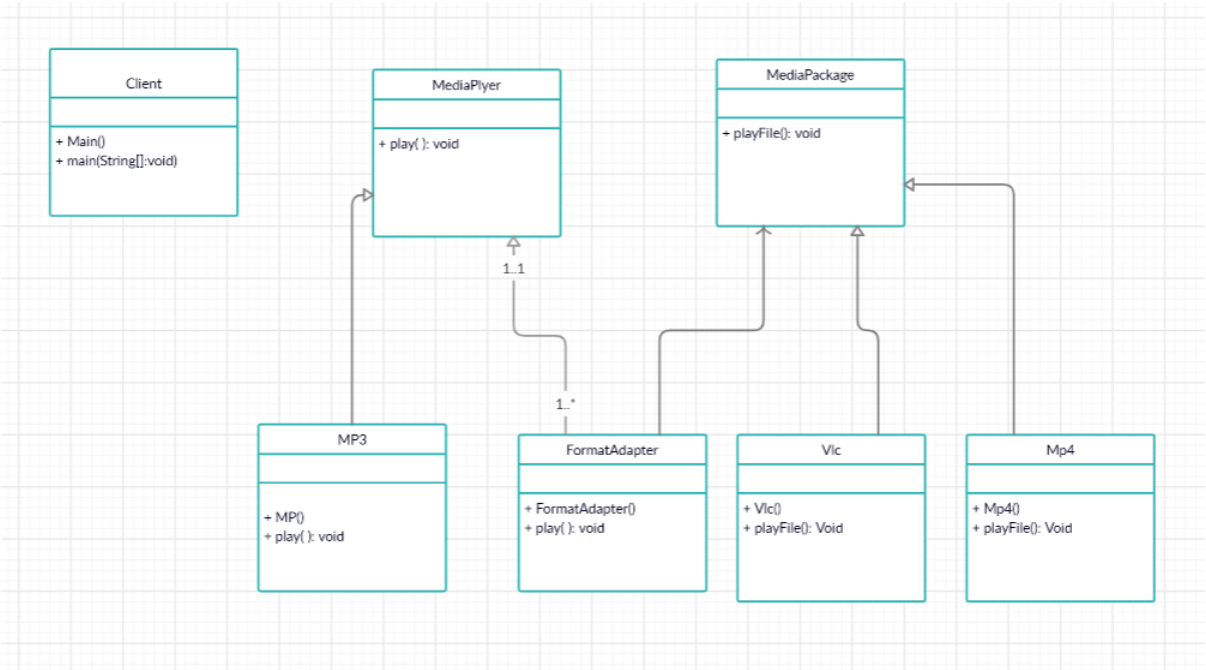
```java
package v1;


public class EarPlugAdapter {

    public String convertSoundSignal(Integer soundSignal){
        return soundSignal.toString();
    }
}
```

ตัวอย่างผลลัพธ์

```
Adapter Design Pattern Example V1
playing input signal
1001010101
playing input signal
1001010101
```

จะเห็นได้ว่าข้างต้น ก็ยังไม่ได้ใช้คุณสมบัติของ Adapter Design เราจะมาลองอีกตัวอย่างที่จะนำการ Interface เข้ามาทำให้เห็นภาพมากขึ้น

1.2 Media V2



MediaPlayer และ MediaPackage คลาส MP3 เป็นการใช้งานอินเทอร์เฟซ MediaPlayer และเรามี VLC และ MP4 เป็นการใช้งานอินเทอร์เฟซ MediaPackage เราต้องการใช้การใช้งาน MediaPackage เป็นอินสแตนซ์ MediaPlayer ดังนั้นเราต้องสร้างอะแดปเตอร์เพื่อช่วยในการทำงานกับสองคลาสที่เข้ากันไม่ได้ อะแดปเตอร์จะมีชื่อว่า FormatAdapter และต้องใช้งานอินเทอร์เฟซ MediaPlayer นอกจากนี้คลาส FormatAdapter ต้องมีการอ้างอิงไปยัง MediaPackage ซึ่งเป็นอินเตอร์เฟสที่เข้ากันไม่ได้

เราจะสร้าง 2 อินเตอร์เฟส

MediaPackage.java ×

v2 > MediaPackage.java > MediaPackage

```
1  package v2;
2
3  public interface MediaPackage {
4      void playFile(String filename);
5  }
```

MediaPlayer.java ×

v2 > MediaPlayer.java > MediaPlayer

```
1  package v2;
2
3  public interface MediaPlayer {
4      void play(String filename);
5  }
```

เราจะสร้างคลาสเพื่อทำการ implement

MP3.java

```java
package v2;

public class MP3 implements MediaPlayer{
    @Override
    public void play(String filename) {
        System.out.println("Playing MP3 File " + filename);
    }
}
```

MP4.java

```java
package v2;

public class MP4 implements MediaPackage {
    @Override
    public void playFile(String filename) {
        System.out.println("Playing MP4 File " + filename);
    }
}
```

VLC.java

```java
package v2;

public class VLC implements MediaPackage {
    @Override
    public void playFile(String filename) {
        System.out.println("Playing VLC File " + filename);
    }
}
```

ที่นี่เราต้องการใช้ instances VLC และ MP4 เป็น instances MediaPlayer ดังนั้นเราต้องการ Adapter

v2 > FormatAdapter.java > ...

```java
package v2;

public class FormatAdapter implements MediaPlayer {
    private MediaPackage media;
    public FormatAdapter(MediaPackage m) {
        media = m;
    }
    @Override
    public void play(String filename) {
      System.out.print("Using Adapter --> ");
      media.playFile(filename);
    }
  }
```

Client.java

v2 > Client.java > Client

```java
package v2;

public class Client {
    Run | Debug
    public static void main(String[] args) {
        MediaPlayer player = new MP3();
        player.play("file.mp3");
        player = new FormatAdapter(new MP4());
        player.play("file.mp4");
        player = new FormatAdapter(new VLC());
        player.play("file.avi");
    }
}
```

หลังจากการทำงานของโปรแกรมเราสามารถเห็นผลลัพธ์ MP4 และ VLC สามารถใช้เป็นอินสแตนซ์ MediaPlayer ผ่านอะแดปเตอร์

```
Playing MP3 File file.mp3
Using Adapter --> Playing MP4 File file.mp4
Using Adapter --> Playing VLC File file.avi
PS D:\ปี4เทอม1\SE\v1>
```

แหล่งอ้างอิง

https://th.wikipedia.org/wiki/อะแด็ปเตอร์แพตเทิร์น#ตัวอย่างโปรแกรม

https://en.wikipedia.org/wiki/Adapter_pattern#UML_class_diagram
https://medium.com/@phayao/design-pattern-101-adapter-pattern-62ac0da9ab4f
https://www.youtube.com/watch?time_continue=7&v=9XHU5pRDUDk&feature=emb_logo