adapter-Any real example of Adapter Pattern

Ask Question

I want to demonstrate use of Adapter Pattern to my team. I've read many books and articles online. Everyone is citing an example which are useful to understand the concept (Shape, Memory Card, Electronic Adapter etc.), but there is no real case study.

Can you please share any case study of Adapter Pattern?

p.s. I tried searching existing questions on stackoverflow, but did not find the answer so posting it as a new question. If you know there's already an answer for this, then please redirect.

oop design-patterns adapter software-design

edited Feb 10 '16 at 13:29

Dave Schweisguth

22.6k 7 59 88

asked Jun 18 '12 at 8:55

AksharRoop

1,355 1 12 28

4 Well if you want to demo it. You should have have a ready made example of it in your environment, in fact several. Otherwise why would you want to demo it? – Tony Hopkinson Jun 18 '12 at 8:59

1 Several examples here. stackoverflow.com/questions/1673841/… – r4. Jun 18 '12 at 8:59

1 @TonyHopkinson Aim is to make people aware of this design pattern with real example. – AksharRoop Jun 18 '12 at 10:14

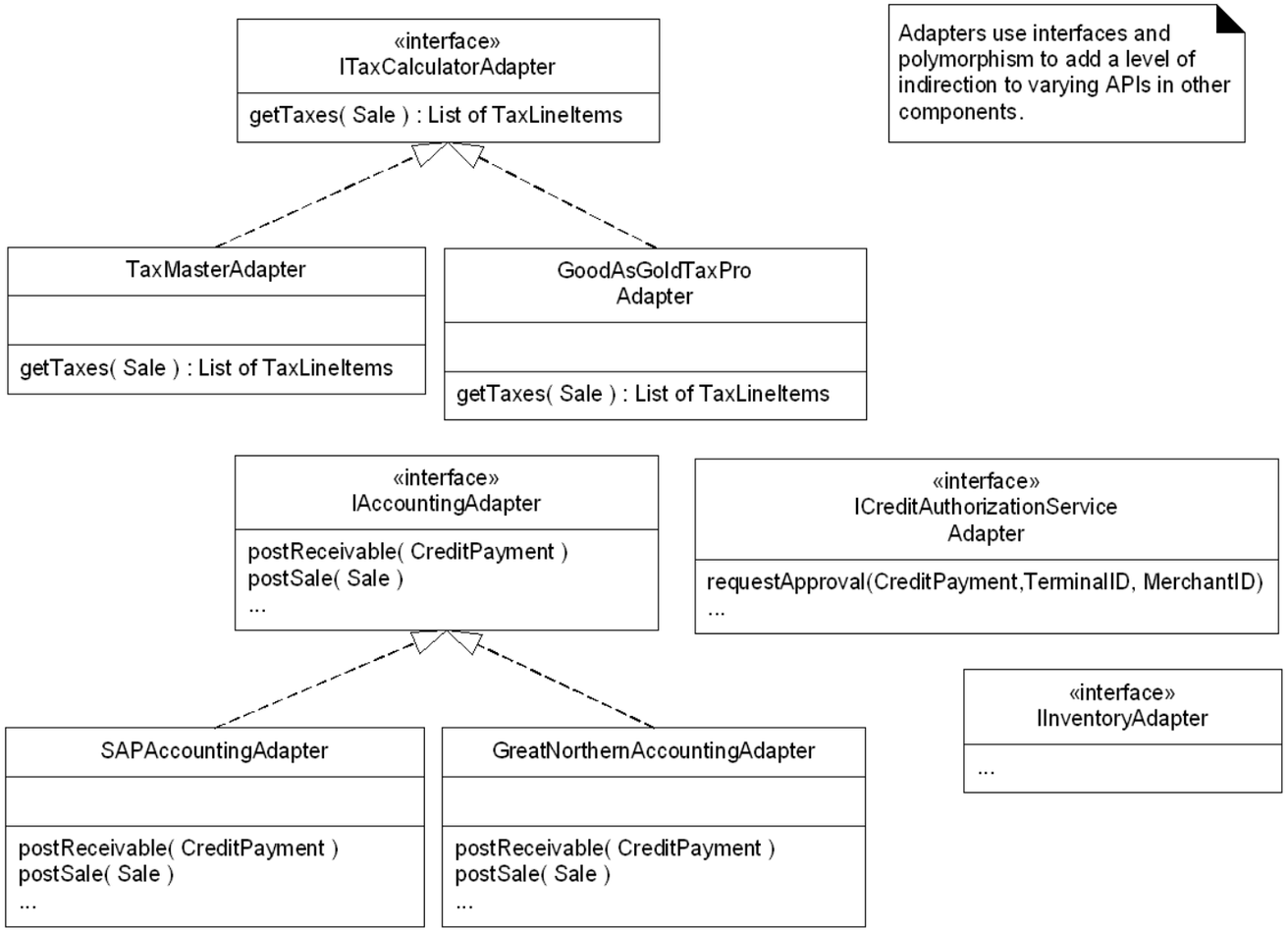
10 @AksharRoop Design Pattern is meant to be a solution to a problem, not a solution looking for a problem. Post

12 Answers

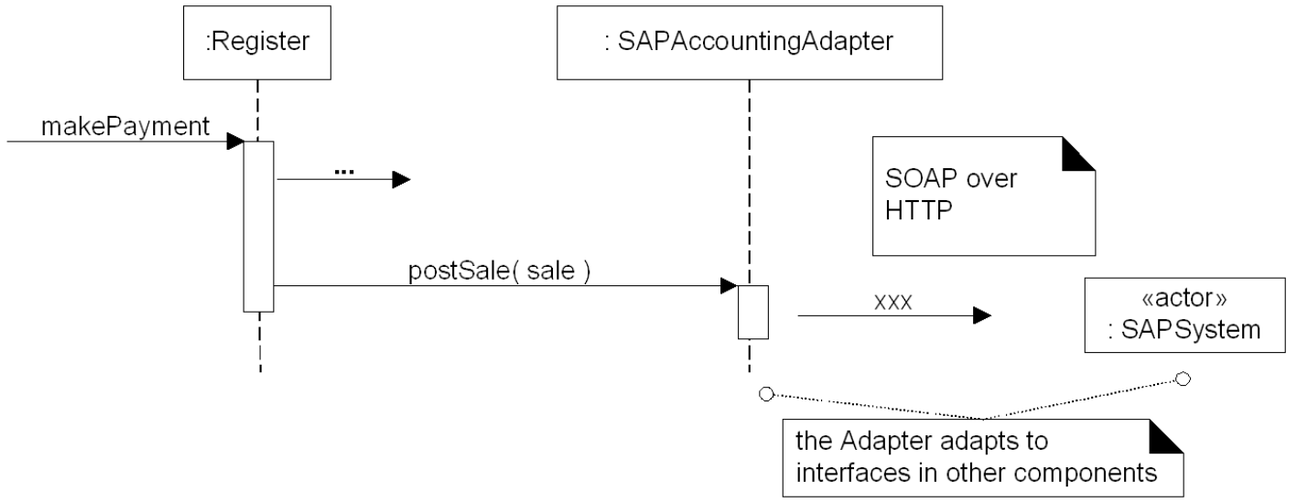
Many examples of Adapter are trivial or unrealistic (Rectangle vs. LegacyRectangle, Ratchet vs. Socket, SquarePeg vs RoundPeg, Duck vs. Turkey). Worse, many don't show **multiple Adapters for different Adaptees** (someone cited Java's Arrays.asList as an example of the adapter pattern). Adapting an interface of *only one class* to work with another seems a weak example of the GoF Adapter pattern. This pattern uses inheritance and polymorphism, so one would expect a good example to show *multiple implementations of adapters for different adaptees*.

The *best example* I found is in Chapter 26 of Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition). The following images are from the instructor material provided on an FTP site for the book.

The first one shows how an application can use multiple implementations (adaptees) that are functionally similar (e.g., tax calculators, accounting modules, credit authorization services, etc.) but have different APIs. We want to avoid hard-coding our domain-layer code to handle the different possible ways to calculate tax, post sales, authorize credit card requests, etc. Those are all external modules that might vary, and for which we can't modify the code. The adapter allows us to do the hard-coding in the adapter, whereas our domain-layer code always uses the same interface (the IWhateverAdapter interface).



We don't see in the above figure the actual adaptees. However, the following figure shows how a polymorphic call to `postSale(...)` in the IAccountingAdapter interface is made, which results in a posting of the sale via SOAP to an SAP system.



edited May 23 '17 at 12:18
Community ♦
1 1

answered Nov 10 '12 at 16:19
Fuhrmanator
4,456 3 29 63

this example using sessions is quite good too (although the implementation is not completely right, I think, using statics): community.sitepoint.com/t/phpunit-testing-cookies-and-sessions/… – Alejandro Moreno Sep 11 '14 at 14:31

and of course, the implementation in PHP: github.com/alex-moreno/DesignPatternsPHP/tree/master/Adapter – Alejandro Moreno Sep 11 '14 at 14:34

Convert an Interface into another Interface.

In order to connect power, we have different interfaces all over the world. Using Adapter we can connect easily like wise.

edited Jan 27 at 12:47

answered Jul 16 '15 at 23:10

Premraj

22.3k 9 121 100

How to turn a french person into a normal person...

```
public interface IFrenchPerson
{
    string Nom { get; set; }
}

public class Person : IPerson
{
    public string Name { get; set; }
}

public class FrenchPerson : IFrenchPerson
{
    public string Nom { get; set; }
}

public class PersonService
{
    public void PrintName(IPerson person)
    {
        Debug.Write(person.Name);
    }
}

public class FrenchPersonAdapter : IPerson
{
    private readonly IFrenchPerson frenchPerson;

    public FrenchPersonAdapter(IFrenchPerson frenchPerson)
    {
        this.frenchPerson = frenchPerson;
    }

    public string Name
    {
        get { return frenchPerson.Nom; }
        set { frenchPerson.Nom = value; }
    }
}
```

Example

```
var service = new PersonService();
var person = new Person();
```

edited Nov 21 '16 at 18:08

EmptyData

1,097 13 29

answered Aug 27 '15 at 14:15

CountZero

2,862 2 33 44

5 I am french and I feel insulté that you do not consider me a real person. (JK) – ZeroUltimax Aug 29 '17 at 12:35

4 @ZeroUltimax I'm pretty sure this code won't compile in Quebec. – Fuhrmanator Oct 20 '17 at 14:34

Any coder without knowledge of Adapters, would have solved the problem easily. How does knowledge of adapter theory help to save time, or make the solution better? Is the ultimate point to use a special class, instead of using just a method? – Rowan Gontier Jan 10 at 23:19

What if you don't control the interface and need to adapt one of your classes to a 3rd party library? Lot's of other good reasons which are outside the scope of this answer. – CountZero Jan 18 at 15:26

Here is an example that simulate converting `analog data` to `digit data`.

It provide an adapter that convert float digit data to binary data, it's probably not useful in real world, it just help to explain the concept of adapter pattern.

Code

AnalogSignal.java

```
package eric.designpattern.adapter;

public interface AnalogSignal {
    float[] getAnalog();

    void setAnalog(float[] analogData);

    void printAnalog();
}
```

DigitSignal.java

```
    void setDigit(byte[] digitData);

    void printDigit();
}
```

FloatAnalogSignal.java

```
package eric.designpattern.adapter;

import java.util.Arrays;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class FloatAnalogSignal implements AnalogSignal {
    private Logger logger = LoggerFactory.getLogger(this.getClass());
    private float[] data;

    public FloatAnalogSignal(float[] data) {
        this.data = data;
    }

    @Override
    public float[] getAnalog() {
        return data;
    }

    @Override
    public void setAnalog(float[] analogData) {
        this.data = analogData;
    }

    @Override
    public void printAnalog() {
        logger.info("{}", Arrays.toString(getAnalog()));
    }
}
```

BinDigitSignal.java

```
package eric.designpattern.adapter;
```

```java
public class BinDigitSignal implements DigitSignal {
    private Logger logger = LoggerFactory.getLogger(this.getClass());
    private byte[] data;

    public BinDigitSignal(byte[] data) {
        this.data = data;
    }

    @Override
    public byte[] getDigit() {
        return data;
    }

    @Override
    public void setDigit(byte[] digitData) {
        this.data = digitData;
    }

    @Override
    public void printDigit() {
        logger.info("{}", Arrays.toString(getDigit()));
    }
}
```

AnalogToDigitAdapter.java

```java
package eric.designpattern.adapter;

import java.util.Arrays;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * <p>
 * Adapter - convert analog data to digit data.
 * </p>
 *
 * @author eric
 * @date Mar 8, 2016 1:07:00 PM
 */
public class AnalogToDigitAdapter implements DigitSignal {
    public static final float DEFAULT_THRESHOLD_FLOAT_TO_BIN = 1.0f; // default
```

```
    private float threshold;
    private boolean cached;

    public AnalogToDigitAdapter(AnalogSignal analogSignal) {
        this(analogSignal, DEFAULT_THRESHOLD_FLOAT_TO_BIN);
    }

    public AnalogToDigitAdapter(AnalogSignal analogSignal, float threshold) {
        this.analogSignal = analogSignal;
        this.threshold = threshold;
        this.cached = false;
    }

    @Override
    public synchronized byte[] getDigit() {
        if (!cached) {
            float[] analogData = analogSignal.getAnalog();
            int len = analogData.length;
            digitData = new byte[len];

            for (int i = 0; i < len; i++) {
                digitData[i] = floatToByte(analogData[i]);
            }
        }

        return digitData;
    }

    // not supported, should set the inner analog data instead,
    @Override
    public void setDigit(byte[] digitData) {
        throw new UnsupportedOperationException();
    }

    public synchronized void setAnalogData(float[] analogData) {
        invalidCache();
        this.analogSignal.setAnalog(analogData);
    }

    public synchronized void invalidCache() {
        cached = false;
        digitData = null;
    }
```

```
    // float -> byte convert,
    private byte floatToByte(float f) {
        return (byte) (f >= threshold ? 1 : 0);
    }
}
```

Code - Test case

AdapterTest.java

```
package eric.designpattern.adapter.test;

import java.util.Arrays;

import junit.framework.TestCase;

import org.junit.Test;

import eric.designpattern.adapter.AnalogSignal;
import eric.designpattern.adapter.AnalogToDigitAdapter;
import eric.designpattern.adapter.BinDigitSignal;
import eric.designpattern.adapter.DigitSignal;
import eric.designpattern.adapter.FloatAnalogSignal;

public class AdapterTest extends TestCase {
    private float[] analogData = { 0.2f, 1.4f, 3.12f, 0.9f };
    private byte[] binData = { 0, 1, 1, 0 };
    private float[] analogData2 = { 1.2f, 1.4f, 0.12f, 0.9f };

    @Test
    public void testAdapter() {
        AnalogSignal analogSignal = new FloatAnalogSignal(analogData);
        analogSignal.printAnalog();

        DigitSignal digitSignal = new BinDigitSignal(binData);
        digitSignal.printDigit();

        // adapter
        AnalogToDigitAdapter adAdapter = new AnalogToDigitAdapter(analogSignal);
        adAdapter.printDigit();
```

```
    }
}
```

Dependence - via maven

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.8.2</version>
</dependency>

<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.13</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.13</version>
</dependency>
<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.16</version>
</dependency>
```

How to test

Just run the unit test.

edited Nov 14 '16 at 3:12

mkobit

17k 6 70 86

answered Mar 8 '16 at 5:05

Eric Wang

6,711 3 56 83

Real-world examples might be a language translator or a mobile charger. More here in this youtube video:

Youtube - Adapter Design pattern: Introduction

edited Aug 13 '17 at 14:21

Ebrahim Poursadeqi
1,362 2 9 22

answered May 31 '14 at 3:26

babu
79 1 1

You can find a PHP implementation of the Adapter pattern used as a defense against injection attacks here:

http://www.php5dp.com/category/design-patterns/adapter-composition/

One of the interesting aspects of the Adapter pattern is that it comes in two flavors: A class adapter relying on multiple inheritance and an object adapter relying on composition. The above example relies on composition.

answered Jul 30 '14 at 9:41

Bill
67 3

One Real example is Qt-Dbus.

The qt-dbus has a utility to generate the adaptor and interface code from the xml file provided. Here are the steps to do so.

```
 1. Create the xml file - this xml file should have the interfaces
that can be viewed by the qdbus-view in the system either on
the system or session bus.

    2.With the utility - qdbusxml2cpp , you generate the interface adaptor code.
This interface adaptor does the demarshalling of the data that is
received from the client. After demarshalling, it invokes the
```

```
in the point number 2, the adaptor interface does the demarshalling
and calls the adaptee - user defined methods.
```

You can see the complete example of Qt-Dbus over here -

http://www.tune2wizard.com/linux-qt-signals-and-slots-qt-d-bus/

answered Dec 9 '14 at 9:38

dexterous_stranger
2,726 3 28 63

A real example can be reporting documents in an application. Simple code as here.

Adapters i think are very useful for programming structure.

```
class WordAdaptee implements IReport{
    public void report(String s) {
        System.out.println(s +" Word");
    }
}

class ExcellAdaptee implements IReport{
    public void report(String s) {
        System.out.println(s +" Excel");
    }
}


class ReportAdapter implements IReport{
    WordAdaptee wordAdaptee=new WordAdaptee();
    @Override
    public void report(String s) {
        wordAdaptee.report(s);
    }
}

interface IReport {
    public void report(String s);
```

```
        IReport iReport=new ReportAdapter();

        //we want to write a report both from excel and world
        iReport.report("Trial report1 with one adaptee");  //we can directly
write the report if one adaptee is avaliable

        //assume there are N adaptees so it is like in our example
        IReport[] iReport2={new ExcellAdaptee(),new WordAdaptee()};

        //here we can use Polymorphism here
        for (int i = 0; i < iReport2.length; i++) {
            iReport2[i].report("Trial report 2");
        }
    }
}
```

Results will be:

```
Trial report1 with one adaptee Word
Trial report 2 Excel
Trial report 2 Word
```

edited Jun 3 '15 at 22:19

lilott8

731 9 27

answered Nov 17 '14 at 12:33

huseyin

734 9 12

This is actually a proxy. An adapter and adaptee have different interfaces. They don't implement the same interface. That's what a proxy does. – dvallejo Jul 7 '15 at 20:23

You can use the Adapter design pattern when you have to deal with different interfaces with similar behavior (which usually means classes with similar behavior but with different methods). An example of it would be a class to connect to a Samsung TV and another one to connect to a Sony TV. They will share common behavior like open menu, start playback, connect to a network and etc but each library will have a different implementation of it (with different method names and signatures). These different vendor specific implementations are called **Adaptee** in the UML diagrams.

The **Adapters** will then implement the **Target** interface delegating its method calls to the **Adaptees** that are passed to the **Adapters** via constructor.

For you to realize this in Java code, I wrote a very simple project using exactly the same example mentioned above using adapters to deal with multiple smart TV interfaces. The code is small, well documented and self explanatory so dig on it to see how a real world implementation would look like.

Just download the code and import it to Eclipse (or your favorite IDE) as a Maven project. You can execute the code by running **org.example.Main.java**. Remember that the important thing here is to understand how classes and interfaces are assembled together to design the pattern. I also created some fake **Adaptees** in the package **com.thirdparty.libs**. Hope it helps!

https://github.com/Dannemann/java-design-patterns

edited Feb 21 at 19:58

answered Feb 21 at 19:25

Jaabax

43 7

Use Adapter when you have an interface you cannot change, but which you need to use. See it as you're the new guy in an office and you can't make the gray-hairs follow your rules - you must adapt to theirs. Here is a real example from a real project I worked on sometime where the user interface is a given.

You have an application that read all the lines in a file into a List data structure and displayed them in a grid (let's call the underlying data store interface IDataStore). The user can navigate through these data by clicking the buttons "First page", "Previous page", "Next page", "Last Page". Everything works fine.

Now the application needs to be used with production logs which are too big to read into memory but the user still needs to navigate through it! One solution would be to implement a Cache that stores the first page, next, previous and last pages. What we want is when the user clicks "Next page", we return the page from the cache and update the cache; when they click last page, we return last page from cache. In the background we have a filestream doing all the magic. By so doing we only have four pages in memory as opposed to the entire file.

You can use an adapter to add this new cache feature to your application without the user noticing it. We

And who knows, tomorrow the boss wants to start reading the files from a database table. All you do is still extend IDataStore to SQLDataStore as you did for Cache, setup the connection in the background. When they click Next page, you generate the necessary sql query to fetch the next couple hundred rows from the database.

Essentially, the original interface of the application did not change. We simply adapted modern and cool features to work it while preserving the legacy interface.

answered Dec 14 '15 at 1:46

Justice O.

745 7 18

I dont understand? It sounds like you just used an existing interface and implemented the methods? Where is the different interface you need to adapt to and the adapter class? – berimbolo Feb 8 '17 at 13:36

@berimbolo Your confusion is valid as above example does not talk about adapter pattern clearly. – rahil008 Sep 8 '17 at 12:57

@Justice o's example does not talk about adapter pattern clearly. Extending his answer - We have existing interface IDataStore that our consumer code uses and we cannot change it. Now we are asked to use a cool new class from XYZ library that does what we want to implement, but but but, we cannot change that class to extend our IDataStore, seen the problem already ? Creating a new class - ADAPTER, that implements interface our consumer code expects, i.e. IDataStore and by using class from the library whose features we need to have - ADAPTEE, as a member in our ADAPTER, we can achieve what we wanted to.

answered Sep 8 '17 at 13:09

rahil008

86 5

This is an example of adapter implementation:

```
//nokia chargement adapted to samsung
    chargementNokia(x:boolean){
        const old= new SamsungCharger();
        let y:number = x ? 20 : 1;
        old.charge(y);
      }
}


class SamsungCharger {
      charge(x:number){
            console.log("chrgement x ==>", x);
      }
}


function main() {
      //charge samsung with nokia charger
      const adapter = new SamsungAdapter();
      adapter.chargementNokia(true);
}
```

answered Oct 20 '17 at 13:06

Omar Marzougui

9 1