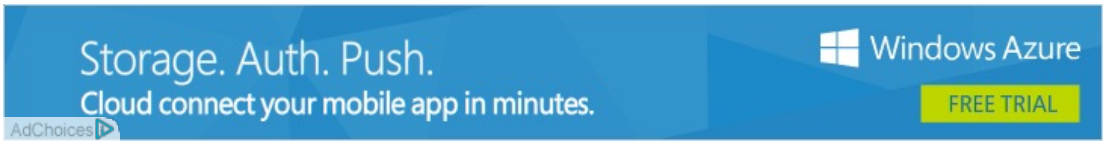
2

implementing class adapter pattern in java



while reading up on Adapter pattern in Head first design patterns ,I came across this sentence

'class adapter...because you need multiple inheritance to implement it,which is not possible in java'

Just to experiment ,I tried

```
interface MyNeededInterface{
    public void operationOne(MyNeededInterface other);
    public MyNeededInterface operationTwo();
}

public class ThirdPartyLibraryClass{
    public void thirdPartyOp();
}
```

Suppose I create

```
class ThirdPartyWrapper extends ThirdPartyLibraryClass implements MyNeededInterfac

    @Override
    public void operationOne(ThirdPartyWrapper other){
        this.thirdPartyOp();
        dosomeExtra();
    }
    @Override
    public ThirdPartyWrapper operationTwo(){
        int somevalue = doSomeThingElse();
        return new ThirdPartyWrapper(somevalue);
    }
}
```

In my code,I can use

```
MyNeededInterface myclass = createThirdPartyWrapper();
myclass.operationOne(someobj);
...
```

Is this not class adapter? I am a beginner when it comes to design patterns. So please excuse my ignorance/mistakes

thanks

mark

java | class | design-patterns | adapter

edited Jun 10 '11 at 11:54
Don Roby
20.9k 4 26 50

asked Jun 10 '11 at 11:35
markjason72
348 1 9 24

add comment (requires 50 reputation)

3 Answers

The class adapter pattern is not possible in Java because you can't extend multiple classes. So you'll

The class adapter pattern is not possible in Java because you can't extend multiple classes. So you'll have to go with the adapter pattern which uses composition rather than inheritance.

An example of the adapter pattern through composition can be found below:

```
interface Duck
{
    public void quack();
}

class BlackDuck implements Duck
{
   public void quack() { }
}

class Turkey
{
    public void gobble() { }
}

class TurkeyAdapter implements Duck
{
    private Turkey t;

    public TurkeyAdapter(Turkey t)
    {
        this.t = t;
    }

    public void quack()
    {
        // A turkey is not a duck but, act like one
        t.gobble();
    }
}
```

Now you can pass a `Turkey` to a method which is expecting a `Duck` through the `TurkeyAdapter`.

```
class DuckCatcher
{
    public void catch(Duck duck) { }
}
```

By using the adapter pattern the `DuckCatcher` is now also able to catch `Turkey(Adapter)`s and `Duck`s.

edited Jun 10 '11 at 12:05

answered Jun 10 '11 at 11:40


Kevin
3,293 9 25

1 This is the object adapter pattern. The OP wants to attempt implementing the class adapter pattern. – Vineet Reynolds Jun 10 '11 at 11:42

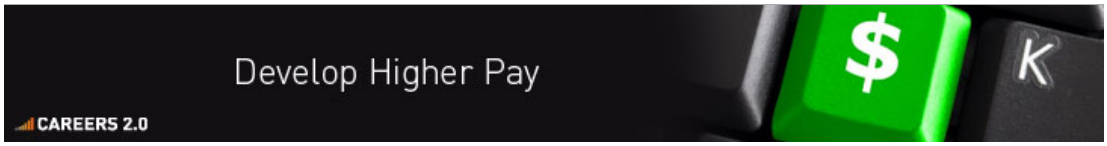
1 @Vineet: Could you expand? I'm not sure what the difference is between both? I thought there was just one general 'adapter pattern'. – Kevin Jun 10 '11 at 11:44

2 the Head First Patterns book, discusses the variant where inheritance (and not composition) is used to implement the class adapter. This is possible in C++ - see this example for instance. – Vineet Reynolds Jun 10 '11 at 11:49

@Vineet: Ah I see, thanks for the link. The adapter pattern through inheritance would be impossible in Java though. – Kevin Jun 10 '11 at 12:00

yes. You might want to edit your answer. – Vineet Reynolds Jun 10 '11 at 12:03

show **3** more comments



The full story in heads up is: **class** adapter pattern is *impossible* in Java just because Java does not provide *multiple inheritance*.

In their diagram, they show that the `Adapter` class *subclasses* both `Target` and `Adaptee` . Your example is (close to) the Object adapter pattern. The difference is that you implement the *Target* in your adapter class, rather then just subclassing the target (`MyNeededInterface` in your example)

edited Jun 10 '11 at 11:52

answered Jun 10 '11 at 11:45

Andreas_D
57.3k 2 49 101

1 Yet it achieves the same goal (adapting the adaptee) and clearly isn't the object adapter pattern. If you can't call it the class adapter pattern then what do you call it? – Pace Jun 10 '11 at 13:47

add comment (requires 50 reputation)

Yes, you can create a class adapter with an interface as long as you're only wrapping a single adaptee. With multiple inheritance you could take two or more adaptees and wrap them into a single interface.

answered Jun 10 '11 at 11:45

Pace
8,662 13 38

ok..that is a valid point...Thanks everyone for the responses.. – markjason72 Jun 11 '11 at 11:54

add comment (requires 50 reputation)

Not the answer you're looking for? Browse other questions tagged java class design-patterns adapter **or ask your own question.**