Strategy Design Pattern in Java – Example Tutorial

APRIL 2, 2018 BY PANKAJ — 27 COMMENTS

Strategy design pattern is one of the **behavioral design pattern**. Strategy pattern is used when we have multiple algorithm for a specific task and client decides the actual implementation to be used at runtime.

Strategy Pattern

Strategy pattern is also known as **Policy Pattern**. We define multiple algorithms and let client application pass the algorithm to be used as a parameter.

One of the best example of strategy pattern is `Collections.sort()` method that takes Comparator parameter. Based on the different implementations of Comparator interfaces, the Objects are getting sorted in different ways.

For our example, we will try to implement a simple Shopping Cart where we have two payment strategies – using Credit Card or using PayPal.

First of all we will create the interface for our strategy pattern example, in our case to pay the amount passed as argument.

`PaymentStrategy.java`

```
package com.journaldev.design.strategy;

public interface PaymentStrategy {

        public void pay(int amount);
}
```

Now we will have to create concrete implementation of algorithms for payment using credit/debit card or through paypal.

`CreditCardStrategy.java`

```
package com.journaldev.design.strategy;

public class CreditCardStrategy implements PaymentStrategy {

        private String name;
        private String cardNumber;
        private String cvv;
        private String dateOfExpiry;

        public CreditCardStrategy(String nm, String ccNum, String cvv, String
expiryDate){
                this.name=nm;
                this.cardNumber=ccNum;
                this.cvv=cvv;
                this.dateOfExpiry=expiryDate;
```

```
	}
	@Override
	public void pay(int amount) {
		System.out.println(amount +" paid with credit/debit card");
	}

}
```

PaypalStrategy.java

```
package com.journaldev.design.strategy;

public class PaypalStrategy implements PaymentStrategy {

	private String emailId;
	private String password;

	public PaypalStrategy(String email, String pwd){
		this.emailId=email;
		this.password=pwd;
	}

	@Override
	public void pay(int amount) {
		System.out.println(amount + " paid using Paypal.");
	}

}
```

Now our strategy pattern example algorithms are ready. We can implement Shopping Cart and payment method will require input as Payment strategy.

Item.java

```
package com.journaldev.design.strategy;

public class Item {

	private String upcCode;
	private int price;

	public Item(String upc, int cost){
		this.upcCode=upc;
```

```
                this.price=cost;
        }

        public String getUpcCode() {
                return upcCode;
        }

        public int getPrice() {
                return price;
        }

}
```

ShoppingCart.java

```
package com.journaldev.design.strategy;

import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.List;

public class ShoppingCart {

        //List of items
        List<Item> items;

        public ShoppingCart(){
                this.items=new ArrayList<Item>();
        }

        public void addItem(Item item){
                this.items.add(item);
        }

        public void removeItem(Item item){
                this.items.remove(item);
        }
```

Notice that payment method of shopping cart requires payment algorithm as argument and doesn't store it anywhere as instance variable.

Let's test our strategy pattern example setup with a simple program.

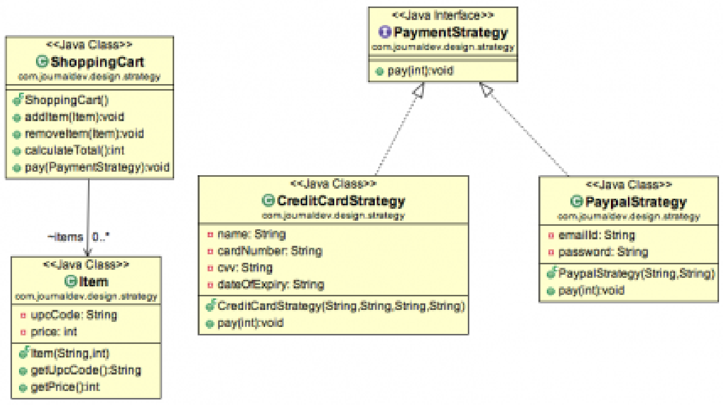
ShoppingCartTest.java

```
package com.journaldev.design.strategy;

public class ShoppingCartTest {

        public static void main(String[] args) {
                ShoppingCart cart = new ShoppingCart();

                Item item1 = new Item("1234",10);
                Item item2 = new Item("5678",40);

                cart.addItem(item1);
                cart.addItem(item2);

                //pay by paypal
                cart.pay(new PaypalStrategy("myemail@example.com", "mypwd"));

                //pay by credit card
                cart.pay(new CreditCardStrategy("Pankaj Kumar", "1234567890123456",
"786", "12/15"));
        }

}
```

Output of above program is:

```
50 paid using Paypal.
50 paid with credit/debit card
```

Strategy Design Pattern Class Diagram



Strategy Design Pattern Important Points

- We could have used composition to create instance variable for strategies but we should avoid that as we want the specific strategy to be applied for a particular task. Same is followed in Collections.sort() and Arrays.sort() method that take comparator as argument.
- Strategy Pattern is very similar to **State Pattern**. One of the difference is that Context contains state as instance variable and there can be multiple tasks whose implementation can be dependent on the state whereas in strategy pattern strategy is passed as argument to the method and context object doesn't have any variable to store it.
- Strategy pattern is useful when we have multiple algorithms for specific task and we want our application to be flexible to chose any of the algorithm at runtime for specific task.

That's all for Strategy Pattern in java, I hope you liked it.

« **PREVIOUS**
State Design Pattern in Java

NEXT »
Template Method Design Pattern in Java

FILED UNDER: DESIGN PATTERNS

Comments

Prabakar says

JUNE 9, 2018 AT 1:29 PM

using factory pattern also we can decide instance of PaymentStrategy .

Reply

Pitambar says

JUNE 5, 2018 AT 9:12 PM

Nice post!!!

Thank you very much for the

Reply

Ed Chen says

MAY 24, 2018 AT 9:13 PM

Can I change the PaymentStrategy interface to be an abstract class or an normal class?

Does it violate the spirit of the strategy pattern if I change the interface to class ?

I am kinda confusing by using an interface or using an class..

Reply

Murshid Alam says

MAY 9, 2018 AT 1:33 AM

Hi

public void pay(PaymentStrategy paymentMethod){

int amount = calculateTotal();

paymentMethod.pay(amount);

}

Here PaymentStrategy is an interface which is holding the reference of implemented class, is it not Run time Polymorphism .?

Strategy Design pattern and Runt time Polymorphism both are same..?

Reply

Alan Smith says

FEBRUARY 23, 2018 AT 2:28 PM

This example was not correct

Reply

sam says

MARCH 23, 2018 AT 6:13 AM

Why ?

Reply

Murat KG says

JANUARY 19, 2018 AT 8:43 PM

Thank you! Clear example

Reply

Shiva Shankar says

JANUARY 19, 2018 AT 8:54 AM

simply superb explanation. Thanks □

Reply

Arun SIngh says

OCTOBER 4, 2017 AT 7:32 AM

thats easy, thanks Pankaj

Reply

Manas says

SEPTEMBER 8, 2017 AT 9:15 AM

The design pattern is well exaplained really. Thanks a lot.

Reply

Boopathi says
OCTOBER 18, 2016 AT 11:33 PM
It would be more clear if we pass the interface reference.
public class ShoppingCartTest {
public static void main(String[] args) {
ShoppingCart cart = new ShoppingCart();
Item item1 = new Item("1234",10);
Item item2 = new Item("5678",40);
cart.addItem(item1);
cart.addItem(item2);
//pay by paypal
PaymentStrategy paymentStrategy;
paymentStrategy= new PaypalStrategy("myemail@example.com", "mypwd");
//pay by credit card
cart.pay(paymentStrategy);
paymentStrategy=new CreditCardStrategy("Pankaj Kumar", "1234567890123456", "786", "12/15");
cart.pay(paymentStrategy);
}
}
Reply

Rafal says
JUNE 21, 2016 AT 5:19 AM
Great !!!
Reply

Ashish Patel says
JANUARY 8, 2016 AT 7:20 PM
Thank you very much for such a very simple and clear definition&example of same.
Reply

Sachin Kakkar says
NOVEMBER 9, 2015 AT 12:38 AM
Can we say method overriding and overloading also implementation of Strategy design pattern as in overriding based on the object of class method gets invoked and this is decided at the run time. and in

the case of overloading method invocation based on parameters passed. Does both are also implementation of Strategy design pattern.
Reply

Rahul says
OCTOBER 25, 2015 AT 9:04 AM
Keep up the good work
Reply

Fuhrmanator says
AUGUST 5, 2015 AT 1:38 PM
I used your example in an answer on Stack Overflow: http://stackoverflow.com/a/30424503/1168342
Reply

Sushil Jain says
JUNE 23, 2015 AT 2:08 AM
Thank you very much for such a very simple and clear definition&example of same.
Reply

aziz says
MARCH 27, 2015 AT 4:57 AM
thanks
Reply

maheraj says
MARCH 18, 2015 AT 11:02 AM
nice explanation
Reply

Bhavani says
FEBRUARY 9, 2015 AT 12:35 AM

Very well explained.
Thanks
Reply

Ravi says
AUGUST 29, 2014 AT 10:32 AM
Thanks for your effort. Simple and clear.
Reply

raed says
AUGUST 18, 2014 AT 1:50 AM
well done , but as i know with Strategy Design Pattern the Object of the class will change his Type
i mean change the behaviour of the Object at the runtime ??
Reply

BG says
JULY 21, 2014 AT 4:28 PM
really nice explanation thanks for the doc
Reply

Gabriel says
JUNE 21, 2014 AT 6:21 AM
Thanks for explain the pattern in a simple and nice way.
Reply

Aashu says
JUNE 15, 2014 AT 11:02 PM
Can we say strategy pattern is replacement for Switch Case / If Else ?
Reply

parag says

MAY 21, 2014 AT 12:55 AM

Hi Pankaj,

Just want to say thanks for describing the design pattern so nicely.

Reply

Raja says

APRIL 25, 2014 AT 1:15 AM

Hello sir

thank you very much for this information.

Really you are the rock of Java…………..

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

Search for tutorials...

DOWNLOAD ANDROID APP

GET IT ON Google Play

DESIGN PATTERNS TUTORIAL

RECOMMENDED TUTORIALS

Java Tutorials

- Java IO
- Java Regular Expressions
- Multithreading in Java
- Java Logging
- Java Annotations
- Java XML
- Collections in Java
- Java Generics
- Exception Handling in Java
- Java Reflection
- Java Design Patterns
- JDBC Tutorial

Java EE Tutorials

- Servlet JSP Tutorial
- Struts2 Tutorial
- Spring Tutorial
- Hibernate Tutorial
- Primefaces Tutorial
- Apache Axis 2
- JAX-RS
- Memcached Tutorial

MANUFACTURING
KEEP IMPROVING
Strive for excellence, keep an open mind, to manufacture customer satisfied products.
Quality Wins Customers
Trust Creates Brand