Polymorphism vs Strategy pattern

Ask Question

Asked 4 years, 6 months ago Active 10 months ago Viewed 9k times

45

14

What is the difference between the `Strategy` pattern and `Polymorphism` in Java?

I'm confused that whatever is achieved via Strategy Pattern is basically possible by polymorphism. Correct me if I'm wrong in this regard.

Please, also provide me example to eradicate my confusion.

java design-patterns polymorphism strategy-pattern

share improve this question

edited Jul 28 '15 at 17:18

CKing

13.3k ● 4 ● 35 ● 71

asked Jul 24 '15 at 11:02

TryinHard

3,526 ● 2 ● 24 ● 48

6 It's apples and oranges since you can't have a Strategy Pattern *without* polymorphism as the interface must be implemented, and so your question is confusing to me. – Hovercraft Full Of Eels Jul 24 '15 at 11:05

2 I think he just wants an example in which the strategy pattern is more useful than polymorphism, this happens when you want to reuse different strategies for multiple object. Otherwise it really doesn't make sense and you should use polymorphism. – Joshua Byer Jul 24 '15 at 23:57

Related What pattern does Collections.sort with a Comparator use – CKing Feb 26 '17 at 6:00

add a comment

9 Answers

active oldest votes

43

+50

For me, the link from **CKing** post and the example in Wikipedia are clear enough, but I'll try to give you a new example. As they said, Strategy Pattern is mostly **a way to change the behaviour of an algorithm at runtime**. Of course you can achieve this in many different ways (such as holding a value and using switch-case, but it wouldn't be as nice as Strategy Pattern).

Let's say you're developing a turn-based strategy game with two kind of **Units**: **Infantry** and **Tank** (subclasses of Unit). Your terrain could be **Plains**, **Railroad** or **Forests**.

```
class Unit{
    MovementStrategy ms;
    final int baseMovement;
    int x,y;

    public Unit(int baseMovement){
        this.baseMovement = baseMovement;
    }

    abstract void fire();

    void moveForward(){
        x = x + ms.getHexagonsToMove(baseMovement);
    }

    void setMovementStrategy(MovementStrategy ms){
        this.ms = ms;
    }
}
```

Any Unit subclass must implement *fire()* method because **it's going to be completely different for them** (Tank shots heavy long-distance round and Infantry shot several short distance light bullets). In this example we use normal polymorphism/inheritance since *fire()* method will be really different for any unit, and **it won't change during the game**.

```
class Infantry extends Unit{
    public Infantry(){
        super(2);
    }
```

```
    void fire(){
        //whatever
    }
}

class Tank extends Unit{
    public Tank(){
        super(5);
    }

    void fire(){
        //whatever
    }
}
```

Units also are able to move, and have a field *baseMovement* that holds the number of hexagons it can walk. We're developing a strategy game, not a real world simulation, so we don't care how they move, we just want to add a value to their coordinates (in my example I only use X coordinate in order to get a simpler code). If all the terrain was the same, we wouldn't need any Strategy object... but **we need to change the behaviour of move() method at runtime!**

So, we implement a different **MovementStrategy** class for each of our kinds of Terrain, and we program our game to trigger a *setMovementStrategy()* to any unit that move on each hexagon. And we don't even need to write anything else in our Unit subclasses.

```
interface MovementStrategy{
    public int getHexagonsToMove(int base);
}

class PlainMovementStrategy implements MovementStrategy{
    public int getHexagonsToMove(int base){
        return base;
    }
}

class RailroadMovementStrategy implements MovementStrategy{
    public int getHexagonsToMove(int base){
        return base*3;
    }
```

```
}

class ForestMovementStrategy implements MovementStrategy{
    public int getHexagonsToMove(int base){
        return (int)(base/2);
    }
}
```

Now, when any **Unit** move inside a **Forest**, we call

```
unit.setMovementStrategy(new ForestMovementStrategy());
```

And as soon it goes to a **Plain**, we do:

```
unit.setMovementStrategy(new PlainMovementStrategy());
```

Now we're able to change how far away our units move depending on the Terrain, and we don't need to rewrite in any of the subclasses.

I hope this helps you a better understanding of the difference.

share improve this answer

edited May 23 '17 at 11:54
Community ♦
1 ● 1

answered Jul 27 '15 at 23:19

rolgalan
1,446 ● 13 ● 30

2 Just Wondering: is there a benefit to using an enum for this particular example over creating a new object every single time the location changes movement strategies? – D. Ben Knoble Jul 31 '15 at 19:23

1 Enum could work here, you can even add abstract methods to an enumeration. If you follow that route though, I suggest having the enum implement an interface. That way you can always swap out the enum implementation with another implementaiton without much refactoring of the calling class. – aglassman Jul 31 '15 at 20:08

add a comment

26

> I'm confused that whatever is achieved via Strategy Pattern is basically possible by polymorphism.

You can't drive a car without a steering wheel. That does not mean that a steering wheel is a car. Similarly, the Strategy pattern relies on polymorphism but that does not mean that they are the same thing.

The purpose of the Strategy pattern is to promote the use of composition (has-a) over inheritance (is-a). Instead of your class inheriting behavior from a super class, you define the behavior in a separate class and your class has-a reference to it.

As far as an example goes, take a look at this answer that does a good job.

share improve this answer

edited Apr 12 '17 at 7:31
Community ♦
1 ● 1

answered Jul 24 '15 at 11:19

CKing
13.3k ● 4 ● 35 ● 71

add a comment

10

> Polymorphism vs Strategy pattern with core java examples

- Basic difference : **Polymorphism** is programming language concept, and **Strategy pattern** is one of behavioral design pattern of GoF.
- **Polymorphism** is the provision of a single interface to entities of different types.
  Example: The steering wheel(i.e., the interface) is same no matter what type of actual steering mechanism is used. That is, the steering wheel works the same whether your car has manual steering, power steering, or rack-and-pinion steering. Therefore once you know how to operate the steering wheel, you can drive any type of car.
- In programming, **Polymorphism** implemented in two ways:
  - Early-Binding/Static/Compile-Time Polymorphism (ex: function overloading)
  - Late-Binding/Dynamic/Run-Time Polymorphism (ex: function overriding)

> Compile-time: the time period in which you, the developer, are compiling your code.
> Run-time: the time period which a user is running your piece of software.
> Source

- A **Strategy pattern** defines a set of algorithms that can be used interchangeably.
  - The Strategy pattern is a dynamic pattern (How do you want run a behavior in software?).
  - Example of core java: `java.util.Comparator#compare()`, executed by among others `Collections#sort()`.
  - **Modes of transportation** is analogous to **strategy design pattern**. We use car, bike, bus, local train and so on.. different strategies to go office day by day.

share improve this answer

edited Apr 2 '19 at 14:11

answered Jul 28 '15 at 12:55

Premraj

43.2k ● 21 ● 191 ● 137

2 java.util.Comparator#compare() is a great example!! Now I feel ashamed for not think of that first. I use it widely in my projects. – rolgalan Jul 28 '15 at 13:43

If java.util Comparator#compare is an example of the strategy pattern, then is the strategy pattern and polymorphism one and the same thing? If yes, then what is the difference between Polymorphism and Strategy pattern? – CKing Jun 22 '19 at 19:10

add a comment

7

If you are establishing an analogy where:

- in one case you have several overridable methods;
- in the other case you have a Strategy interface with several implementations,

then the difference is the degree of coupling, which is very strong in the first case, whereas in the second case any foreign code can participate in your class's logic by contributing its Strategy implementation.

share improve this answer

answered Jul 24 '15 at 11:17

Marko Topolnik

164k ● 23 ● 231 ● 353

add a comment

> **Q:** What is the difference between the Strategy pattern and Polymorphism in Java?

The questions is certainly confusing since initially there seems not to be any relation between these two ideas.

Polymorphism is a much broader concept in programming and yes the strategy pattern in Java uses a form of polymorphism catalogued as inclusion polymorphism to accomplish its intent, but this is by no means the only type of polymorphism that exists, neither it is the only way to implement the strategy pattern as I will demonstrate soon.

Also polymorphism is not something that only exists in Java or in object-oriented programming languages. Different forms of polymorphism exist in all the programming paradigms and not in all languages you are forced to use polymorphism to implement a strategy pattern (e.g. functional languages).

For further discussion on this topic please read this other answer where we have discussed whether polymorphism is possible without inheritance and I provide interesting references and examples to other types of polymorphism like parametric and ad-hoc polymorphism.

Ideally this will reveal to you that polymorphism is a bigger concept that goes beyond the boundaries of object-oriented programming and even beyond inheritance and subtyping.

> **Q:** I'm confused that whatever is achieved via Strategy Pattern is basically possible by polymorphism. Correct me if I'm wrong in this regard.

From my point of view the relations between these two concepts are: that the strategy pattern leverages the power of polymorphism available in languages like Java to implement its intent, and that polymorphism can be considered in itself a pattern.

For instance, consider this quote from the GoF book:

> If we assumed procedural languages, we might have included design patterns called 'inheritance', 'encapsulation' and 'polymorphism'.

It is just that we rarely think of polymorphism as a pattern, first because it implies many things and because it is implemented differently in different languages and also because it typically presents as some form of language feature.

In his book Elemental Design Patterns, Jason Mc C. Smith comments on the GoF quote above saying:

> Pattern are language-independent concepts; they take form and become concrete solutions when you implement them within a particular language with a given set of language features and constructs [...] This means that it is a bit strange to talk about "Java design pattern", "C++ design patterns", "Websphere design pattern" and so on, even though we all do it. It's a mildly lazy form of shorthand for what we really mean, or should mean: design patterns as implemented in Java, C++, WebSphere and so on, regardless of language or API.

So, as you can see, you are thinking in the Strategy pattern from the perspective of the Java implementation, but in other language paradigms such pattern may have been implemented in different ways, probably without using inheritance at all, for example, in pure functional programming languages this is most certainly implemented using high order functions and function composition.

As such, this would be an strategy pattern implementation without resorting to *inclusion polymorphism* at all. In a function composition strategy we may still be using other forms of polymorphism (e.g. parametric), but that is not a requirement for the strategy pattern

> Q: Please, also provide me example to eradicate my confusion.

As discussed above, in Java we are probably forced to use inclusion polymorphism to implement a Strategy pattern, but as explained above as well, patterns are not something that belong to a specific language, so if we think of the strategy pattern as a concept living outside any language boundaries then you will easily see other languages implement this in different ways.

In some hypothetical functional language I may have a function that reads some data from a file, maybe the file is encrypted and you need to provide a decryption strategy:

```
function readFile(path: String, decrypt: string -> string) {
    return decrypt(loadFromDisk(path));
}
```

And that `decrypt` argument is a function that serves the purpose of a strategy pattern, it encapsulates an interchangeable algorithm.

Now you can do

```
readFile("customers.txt", aes)
readFile("finance.txt", blowfish)
```

Where `aes` and `blowfish` are decryption function strategies.

There are dozens of languages that work like this, SML, Haskell, JavaScript, etc.

share improve this answer

edited Oct 16 '18 at 11:59

answered Jul 31 '15 at 15:20

Edwin Dalorzo

63k • 24 • 122 • 182

add a comment

3

First of all. Polymorphism can mean two different things. Most commonly polymorphism refers to a polymorphic type. However, you are asking for the pattern.

Polymorphic code can change itself each time it runs while the function of the code stays the same. An easy example is to do 1+3=4 instead of 5-1=4. Both achieves the same result using different code. This is useful for code that does not want to be recognized i.e computer viruses or cryptographic code.

Strategy pattern on the other hand uses a family of algorithms that can be interchanged. This might be used when translating text. First some code determines the language. if the language is swedish or spanish the text will be processed by, different functions of the same family, translateSwedish() or translateSpanish().

To round things up. Polymorphic code uses different code that achieves the same result. While Strategy uses different code to achieve better results.

share improve this answer

answered Jul 28 '15 at 9:20

Herman Wilén

165 • 10

add a comment

Consider this

we have animals and a strategy pattern object to describe how they move... for instance

fly/swim/walk

Given the large number of animals that use any of these methods (ie thousands of different animals fly), we need to use the same code for many different animals. That code should only exist in one place, so that it is easily changed and doesn't take up any unneeded space.

In this example, a straightforward polymorphism approach will result in massive code duplication. A more complex approach which places a intermediate class between animal and say robin fail to take in to consideration that how a animal moves is not really what defines it. Furthermore, it is possible that a animal has other strategy objects and they cannot all be made polymorphic through intermediate classes.

share improve this answer

answered Jul 24 '15 at 23:55

Joshua Byer

499 ● 4 ● 11

add a comment

2

One definition for Polymorphism is the provision of a single interface to entities of different types.

With that in mind say you have a “bird” interface and all of your bird classes must implement “laysEggs()” method, well no biggie that works. And as you keep coding your “bird paradise program” you now add the “fly()” and realize that overloading and overriding for penguin and kiwi is unnecessary as in real life they can't fly, yet you must still implement that method. This can become tedious and pointless as you are faced with Ostrich and others that can't fly. And even worst off when adding the method “swim()” because even fewer birds can swim. As you may already know, the strategy pattern solves this problem.

In lame-man’s terms you can think of Polymorphism as a conglomerate of practices, while the Strategy Pattern is the best practice for a specific case. Example: the Strategy Pattern is to be used when an algorithm's behavior needs to be selected at Runtime (via interchangeable algorithms). And although it is sort of true that whatever is achieved via Strategy Pattern is basically possible by polymorphism, without knowledge of the Strategy Pattern this would leave you with the “reinvent the wheel” problem to solve this specific problem. In conclusion, they are very different even if one is based on the other. I'll leave you to see “Ender Muab'Dib” code as it is well explained if you still want a code example from me just ask, cheers and hope I helped.

share improve this answer

answered Sep 17 '15 at 23:12

Omar Ayala

154 ● 3 ● 11

add a comment

2

Polymorphism is a principle and Strategy is a design pattern

From oracle documentation page

> The dictionary definition of polymorphism refers to a principle in biology in which an organism or species can have many different forms or stages. This principle can also be applied to object-oriented programming and languages like the Java language. Subclasses of a class can define their own unique behaviors and yet share some of the same functionality of the parent class.

Polymorphism can be achieved in compile time (method overloading) and run time (method overriding).

Strategy_pattern

1. Defines a family of algorithms,
2. Encapsulates each algorithm, and
3. Makes the algorithms interchangeable within that family.

Strategy can use run-time polymorphism principle to achieve the desired functionality.

Strategy pattern had one more component called Context in it's URL diagram. Refer to below SE posts:

Real World Example of the Strategy Pattern

Does this Java Strategy pattern have a redundant Context class?

Few more useful articles:

strategy by sourcemaking

share improve this answer

edited Sep 24 '17 at 18:43

answered Jun 7 '16 at 9:55

Ravindra babu
36.2k ● 6 ● 188 ● 162

In short, a strategy pattern can use polymorphism. It's not a question of one vs. the other. – Kokodoko Oct 10 '16 at 15:25

add a comment

Your Answer

Sign up or log in

Sign up using Google

Sign up using Facebook

Sign up using Email and Password

Post as a guest

Name

Email

Required, but never shown

Post Your Answer

By clicking "Post Your Answer", you agree to our terms of service, privacy policy and cookie policy

Not the answer you're looking for? Browse other questions tagged java design-patterns polymorphism strategy-pattern or ask your own question.

Blog

- Podcast: Make my Monolith a Micro
- When laziness is efficient: Make the most of your command line

Featured on Meta

- TLS 1.0 and TLS 1.1 removal for Stack Exchange services
- Did Stack Exchange cut the number of negative comments nearly in half between…

An account of my meeting with the Stack Overflow management team

86 people chatting

Android
49 secs ago - Graeme

Java
3 hours ago - KarelG

Linked

356 Runtime vs. Compile time

90 Real World Example of the Strategy Pattern

53 Is polymorphism possible without inheritance?

14 Does this Java Strategy pattern have a redundant Context class?

11 What's design pattern does Collections.sort use?

Related

1731 Difference between static class and singleton pattern?

157 What is the difference between the

template method and the strategy patterns?

566 Difference between DTO, VO, POJO, JavaBeans?

212 What is the difference between Strategy design pattern and State design pattern?

1720 What is a JavaBean exactly?

66 Difference between Strategy pattern and Command pattern

437 What are the differences between Abstract Factory and Factory design patterns?

51 Strategy Pattern V/S Decorator Pattern

Hot Network Questions

- Flexure of a Grid
- How to write the definition of each term in an equation with pointing arrows?
- Is this a triangle?
- How to remove spare tire / wheel
- Group Anagrams (C#)
- What does "D." stand for?
- Recovering from stumbling over words
- "Let's hear it for these ..." Why use "hear" rather than "listen", does that mean the audience do not need to pay much attention?
- Why isn't the center of the Earth cold?
- Why the network switch drop the frame acording to ethertype/length?

- Quoting GPL licensed text in presentation slides
- When was Alice Born?
- title page in book
- I place you back on the ground
- Corner modelling technique
- Can I ask a cop, "What happened?"
- Does Iran's sanction play a pivotal role in rejecting papers from a journal?
- What does the horizontal bar indicate in the DMM display?
- Busiest domestic-only airport?
- Why is the asteroid belt shaped like a triangle?
- Lonely Multiplication
- Double deck vs wide body airliner, why would anyone build a double deck one?
- Is it possible for master, tempdb, model and msdb to have a database_id other than 1,2,3,4 respectively?
- Buy house with down payment, no loan, sell instantly

Question feed