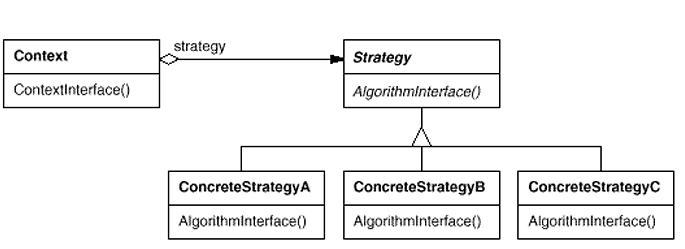
Strategy Pattern – Behavioral - 2024

GOF : Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.

# Structure Diagram



Common Examples of Strategy Patterns are

1. Compressing files with different compression strategies like zip, rar, lgma etc.
2. Encrypt String with different Cipher algorithm strategy
3. Sorting numbers with strategies like Quick Sort, Merge Sort etc.
4. In Spring JPA, fetching strategies like EAGER and LAZY.
5. Payment Strategy like Card payment, UPI Payment etc.
6. Text editor with multiple formatting strategies like Bold, Italics etc.
7. Interest calculation for different types of loan like Home Loan, Car Loan, Personal Loan
8. In case C3 application, it MUPs, CPD or DI(Dell Identity) strategy for authentication.

To some extent, strategy pattern looks similar to Factory pattern.

Difference between Strategy and Factory pattern

* A factory pattern is a creational pattern. A factory pattern is used to create objects of a specific type. The factory pattern allows you to encapsulate object creation.
* A strategy pattern is an operational pattern. A strategy pattern is used to perform an operation (or set of operations) in a particular manner. The strategy pattern allows you to polymorphically change behavior of a class.

The State Design Pattern is similar to the Strategy Design Pattern as both involve encapsulating different behavior into separate classes. However, the State pattern focuses on managing the internal state of an object and transitioning between states, while the Strategy pattern focuses on interchangeable algorithms.

Working examples given below.

public interface SortingStrategy {

void sort(int[] numbers);

}

public class BubbleSort implements SortingStrategy {

public void sort(int[] numbers) {

// Bubble sort implementation

}

}

public class QuickSort implements SortingStrategy {

public void sort(int[] numbers) {

// Quick sort implementation

}

}

public class SortingContext {

private SortingStrategy strategy;

public SortingContext(SortingStrategy strategy) {

this.strategy = strategy;

}

public void setStrategy(SortingStrategy strategy) {

this.strategy = strategy;

}

public void sortNumbers(int[] numbers) {

strategy.sort(numbers);

}

}

public class Client {

public static void main(String[] args) {

int[] numbers = {5, 1, 3, 2, 4};

SortingContext context = new SortingContext(new BubbleSort());

context.sortNumbers(numbers);

context.setStrategy(new QuickSort());

context.sortNumbers(numbers);

}

}

As of Java 8, we can use Lambda expressions to implement strategy pattern.

public interface SortingStrategy {

void sort(int[] numbers);

}

public class SortingContext {

private SortingStrategy strategy;

public SortingContext(SortingStrategy strategy) {

this.strategy = strategy;

}

public void sortNumbers(int[] numbers) {

strategy.sort(numbers);

}

}

You can have test client like the below.

public void bubbleSort(int[] numbers) {  
 System.*out*.println("Bubble sort sorting ...");  
}  
  
public void check() {  
 int[] numbers = {5, 1, 3, 2, 4};  
  
 SortingStrategy strategy = (nums) -> bubbleSort(nums);  
 SortingContext context = new SortingContext(strategy);  
 context.sortNumbers(numbers);  
}

Other types of Examples are given below.

CompressionStrategy.java  
//Strategy Interface  
public interface CompressionStrategy {  
 public void compressFiles(ArrayList<File> files);  
}

CompressionContext.java

public class CompressionContext {  
 private CompressionStrategy strategy;  
  
 public CompressionContext(CompressionStrategy strategy) {  
 this.strategy = strategy;  
 }  
  
 //use the strategy  
 public void createArchive(ArrayList<File> files) {  
 strategy.compressFiles(files);  
 }  
}

RarCompressionStrategy.java

public class RarCompressionStrategy implements CompressionStrategy {  
 public void compressFiles(ArrayList<File> files) {  
 System.out.println("Compressing files using RAR strategy ...");  
 }  
}

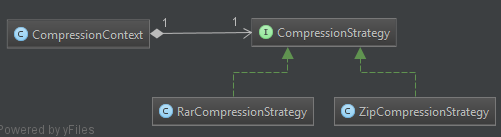
ZipCompressionStrategy.java

public class ZipCompressionStrategy implements CompressionStrategy {  
 public void compressFiles(ArrayList<File> files) {  
 System.out.println("Compressing files using ZIP strategy ...");  
 }  
}

Client.java

public class Client {  
 public static void main(String[] args) {  
 CompressionContext ctx = new CompressionContext(new ZipCompressionStrategy());  
  
 //get a list of files...  
 ArrayList<File> files = new ArrayList<File>();  
 File file1 = new File("a.doc");  
 File file2 = new File("b.doc");  
 files.add(file1);  
 files.add(file2);  
  
 ctx.createArchive(files);  
 }  
}

UML Diagram



Another Example on Text Editor Formatting

public interface TextFormattingStrategy {

String format(String text);

}

public class BoldFormatting implements TextFormattingStrategy {

public String format(String text) {

return "<b>" + text + "</b>";

}

}

public class ItalicFormatting implements TextFormattingStrategy {

public String format(String text) {

return "<i>" + text + "</i>";

}

}

public class UnderlineFormatting implements TextFormattingStrategy {

public String format(String text) {

return "<u>" + text + "</u>";

}

}

public class TextEditor {

private TextFormattingStrategy formattingStrategy;

public void setFormattingStrategy(TextFormattingStrategy formattingStrategy) {

this.formattingStrategy = formattingStrategy;

}

public String formatText(String text) {

return formattingStrategy.format(text);

}

}

public class Client {

public static void main(String[] args) {

TextEditor editor = new TextEditor();

editor.setFormattingStrategy(new BoldFormatting());

String formattedText = editor.formatText("Hello World");

System.out.println(formattedText);

editor.setFormattingStrategy(new ItalicFormatting());

formattedText = editor.formatText("Hello World");

System.out.println(formattedText);

editor.setFormattingStrategy(new UnderlineFormatting());

formattedText = editor.formatText("Hello World");

System.out.println(formattedText);

}

}