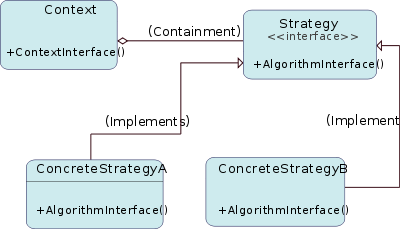
# Strategy TasarımKalıbı

Strategy tasarım kalıbı (policy tasarım kalıbı olarak da bilinir), hangi algoritmanın kullanılacağına çalışma zamanında karar verilme durumunda kullanılır.

|  |
| --- |
|  |

**UML Gösterimi**

[](http://en.wikipedia.org/wiki/File:Strategy_Pattern_Diagram_ZP.svg)

[http://bits.wikimedia.org/static-1.21wmf12/skins/common/images/magnify-clip.png](http://en.wikipedia.org/wiki/File:Strategy_Pattern_Diagram_ZP.svg)

[](http://en.wikipedia.org/wiki/File:Strategy_Pattern_in_UML.png)

[http://bits.wikimedia.org/static-1.21wmf12/skins/common/images/magnify-clip.png](http://en.wikipedia.org/wiki/File:Strategy_pattern_in_LePUS3.gif)

**Örnek**

/\*\* The classes that implement a concrete strategy should implement this.

\* The Context class uses this to call the concrete strategy. \*/

interface Strategy {

int execute(int a, int b);

}

/\*\* Implements the algorithm using the strategy interface \*/

class Add implements Strategy {

publicint execute(int a, int b) {

System.out.println("Called Add's execute()");

return a + b; // Do an addition with a and b

}

}

class Subtract implements Strategy {

publicint execute(int a, int b) {

System.out.println("Called Subtract's execute()");

return a - b; // Do a subtraction with a and b

}

}

class Multiply implements Strategy {

publicint execute(int a, int b) {

System.out.println("Called Multiply's execute()");

return a \* b; // Do a multiplication with a and b

}

}

/\*\* Configured with a ConcreteStrategy object and maintains a reference to a Strategy object \*/

class Context {

private Strategy strategy;

public Context(Strategy strategy) {

this.strategy = strategy;

}

publicintexecuteStrategy(int a, int b) {

returnthis.strategy.execute(a, b);

}

}

/\*\* Tests the pattern \*/

classStrategyExample {

public static void main(String[] args) {

Context context;

// Three contexts following different strategies

context = new Context(new Add());

intresultA = context.executeStrategy(3,4);

context = new Context(new Subtract());

intresultB = context.executeStrategy(3,4);

context = new Context(new Multiply());

intresultC = context.executeStrategy(3,4);

System.out.println("Result A : " + resultA );

System.out.println("Result B : " + resultB );

System.out.println("Result C : " + resultC );

}

}