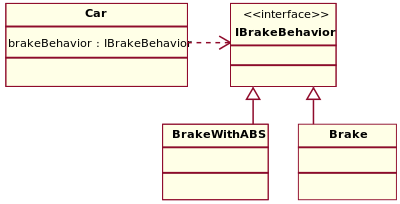
**strategy pattern** (also known as the **policy pattern**) is a [software design pattern](https://en.wikipedia.org/wiki/Design_pattern_(computer_science)) that enables an [algorithm](https://en.wikipedia.org/wiki/Algorithm)'s behavior to be selected at runtime. The strategy pattern

Structure

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

[](https://en.wikipedia.org/wiki/File:StrategyPattern_IBrakeBehavior.svg)

Accelerate and [brake](https://en.wikipedia.org/wiki/Brake) behaviors must be declared in each new [car model](https://en.wikipedia.org/wiki/Car_model).

According to the strategy pattern, the behaviors of a class should not be inherited. Instead they should be encapsulated using interfaces. As an example, consider a car class. Two possible functionalities for car are *brake* and *accelerate*.

1) **Family of Algorithms**- The definition says that the pattern defines the family of algorithms- it means we have functionality (in these algorithms) which will do the same common thing for our object, but in different ways.

2) **Encapsulate each one of them**- The pattern would force you to place your algorithms in different classes (encapsulate them). Doing so would help us in selecting the appropriate algorithm for our object.

3) **Make them interchangeable-**The beauty with strategy pattern is we can select at run time which algorithm we should apply to our object and can replace them with one another

ISortingStrategy object would decide which algorithm to call. The great thing is suppose we realize that one of our algorithm is flawed we simply need to change the sorting algorithm reference in GetSortingOption method and in doing so we need not to change anything in the client code ( i.e the Program class). We can even decide the algorithm at runtime. Suppose during the peak hours when the number of railway passengers increase we can have another customized algorithm (say HugeDataSorting) in place. Based on the number of passengers the ISortingStrategy object would keep changing the reference to HeapSort or HugeDataSorting.