Strategy Design Pattern

In Strategy pattern, a class behavior or its algorithm can be changed at run time. This type of design pattern comes under behavior pattern. Select different algorithms at runtime.

In Strategy pattern, we create objects which represent various strategies and a context object whose behavior varies as per its strategy object. The strategy object changes the executing algorithm of the context object.

**Implementation**

We are going to create a *Strategy* interface defining an action and concrete strategy classes implementing the *Strategy* interface. *Context* is a class which uses a Strategy.

*StrategyPatternDemo*, our demo class, will use *Context* and strategy objects to demonstrate change in Context behaviour based on strategy it deploys or uses.

1. **Strategy**

defines an interface common to all supported algorithms. Context uses this interface to call the algorithm defined by a ConcreteStrategy.

1. **ConcreteStrategy**

each concrete strategy implements an algorithm.

1. **Context**

contains a reference to a strategy object. may define an interface that lets strategy accessing its data.

The Context objects contains a reference to the ConcreteStrategy that should be used. When an operation is required then the algorithm is run from the strategy object. The Context is not aware of the strategy implementation. If necessary, addition objects can be defined to pass data from context object to strategy.

The context object receives requests from the client and delegates them to the strategy object. Usually the ConcreteStartegy is created by the client and passed to the context. From this point the clients interacts only with the context.

