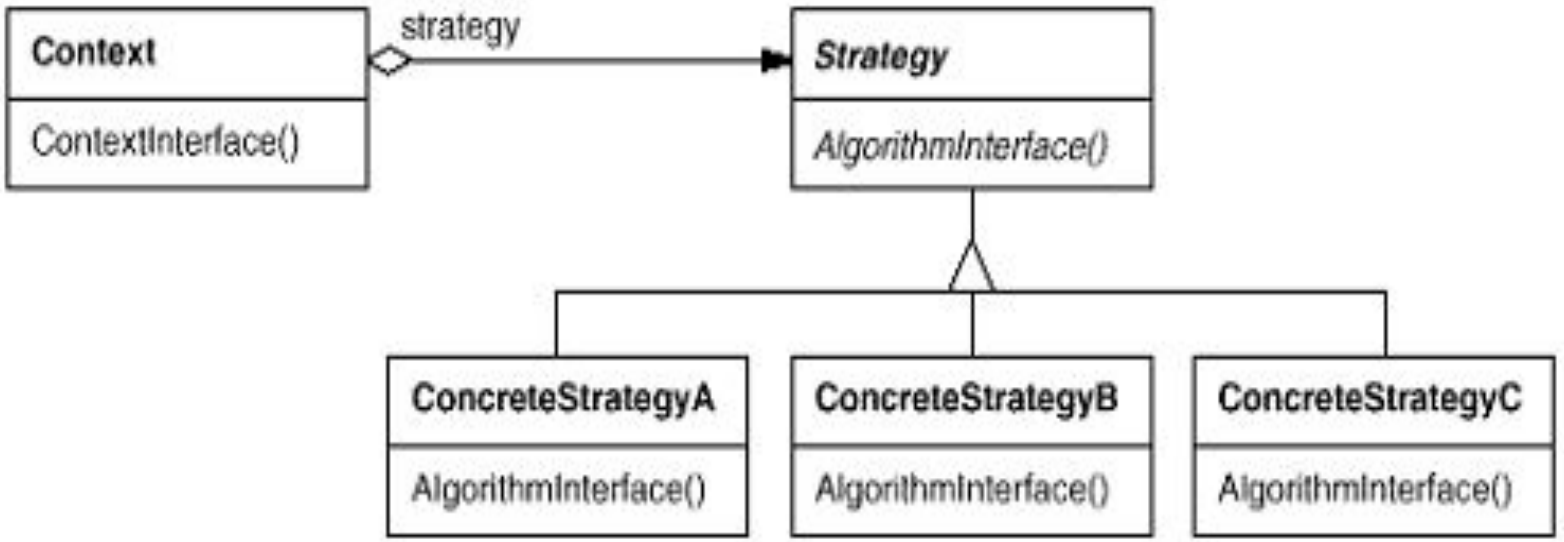
Strategy Pattern

Intent

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

- The Strategy pattern suggests keeping the implementation of each of the algorithms in a separate class.
- Each such algorithm encapsulated in a separate class is referred to as a strategy.
- An object that uses a Strategy object is often referred to as a **context object**.

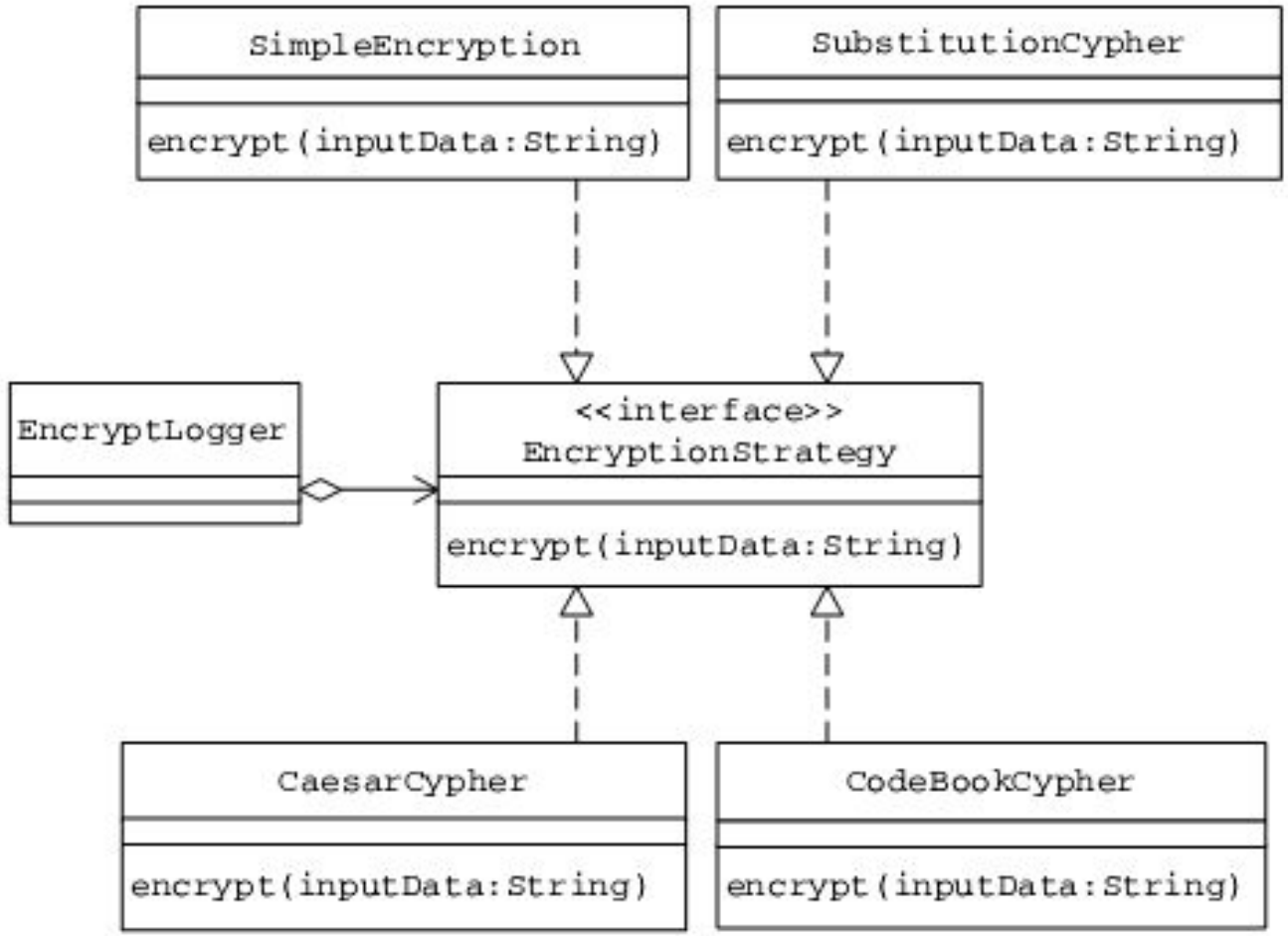
Structure


Context
ContextInterface()
strategy
Strategy
AlgorithmInterface()
ConcreteStrategyA
AlgorithmInterface()
ConcreteStrategyB
AlgorithmInterface()
ConcreteStrategyC
AlgorithmInterface()

Difference Between State and Strategy

Different types of possible behavior of an object are implemented in the form of a group of separate objects.	Similar to the State pattern, specific behaviors are modeled in the form of separate classes.
The choice of a State object is dependent on the state of the Context object.	The choice of a strategy object is based on the application need.

Example (ref Partha Kuchana book)



Example in Java (**Strategy**)

```
public interface SortInterface {
    public void sort(double[] list);
}
```

Example in Java (**ConcreteStrategy**)

```
public class QuickSort implements SortInterface {
    public void sort(double[] a) {
    quicksort(a, 0, a.length - 1);
    }
    private void quicksort(double[] a, int left, int right) {
    if (right <= left) return;
    int i = partition(a, left, right);
    quicksort(a, left, i-1);
    quicksort(a, i+1, right);
    }
    private int partition(double[] a, int left, int right) {
    int i = left;
    int j = right;
    while (true) {
    while (a[i]< a[right])
    i++;
    while (less(a[right], a[--j]))
    if (j == left) break;
    if (i >= j) break;
    exch(a, i, j);
    }
```

Example in Java (**ConcreteStrategy**)

```
exch(a, i, right);
return i;
}

private boolean less(double x, double y) {
return (x < y);
}

private void exch(double[] a, int i, int j) {
double swap = a[i];
a[i] = a[j];
a[j] = swap;
}
}
```

Example in Java (**ConcreteStrategy**)

```
public class BubbleSort implements SortInterface
    {
    public void sort(double[] list) {
    double temp;
    for(int i = 0; i < list.length; i++) {
    for(int j = 0; j < list.length - i; j++) {
    if(list[i] < list[j]) {
    temp = list[i];
    list[i] = list[j];
    list[j] = temp;
    }
    }
    }
    }
    }
```

Example in Java (**Context**)

```
public class SortingContext {
    private SortInterface sorter = null;

    public void sortDouble(double[] list) {
    sorter.sort(list);
    }

    public SortInterface getSorter() {
    return sorter;
    }

    public void setSorter(SortInterface sorter) {
    this.sorter = sorter;
    }
    }
```

Example in Java (**Client**)

```
public class SortingClient {
    public class SortingClient {
    public static void main(String[] args) {
    double[] list =
    {1,2.4,7.9,3.2,1.2,0.2,10.2,22.5,19.6,14,12,16,
    17};
    SortingContext context = new
    SortingContext();
    context.setSorter(new BubbleSort());
    context.sortDouble(list);
    for(int i =0; i< list.length; i++) {
    System.out.println(list[i]);
    }
    }
    }
```

Another Example in Java

```
public interface Strategy {
    boolean checkTemperature(int temperatureInF);
}
public class HikeStrategy implements Strategy {
    boolean checkTemperature(int temperatureInF) {
     if ((temperatureInF >= 50) && (temperatureInF <= 90)) {
          return true; }
     else {
          return false; }
    }
}
```

Another Example in Java

```
public class SkiStrategy implements Strategy {
    boolean checkTemperature(int temperatureInF) {
     if (temperatureInF <= 32) {
          return true; }
     else {
          return false; }
    }
}
```

Another Example in Java

```
public class Context {
    int temperatureInF;
    Strategy strategy;
    public Context(int temperatureInF, Strategy strategy) {
    this.temperatureInF = temperatureInF;
     this.strategy = strategy; }
    public void setStrategy(Strategy strategy) {
     this.strategy = strategy; }
    public int getTemperatureInF() {
     return temperatureInF; }
    public boolean getResult() {
     return strategy.checkTemperature(temperatureInF); }
}
```

Another Example in Java

```
public class Demo {
    public static void main(String[] args) {
    int temperatureInF = 60;
    Strategy skiStrategy = new SkiStrategy();
    Context context = new Context(temperatureInF, skiStrategy);
    System.out.println("Is the temperature (" +
    context.getTemperatureInF() + "F) good for skiing? " +
    context.getResult());
    Strategy hikeStrategy = new HikeStrategy();
    context.setStrategy(hikeStrategy);
    System.out.println("Is the temperature (" +
    context.getTemperatureInF() + "F) good for hiking? " +
    context.getResult());
} }
```

Another Example in Java

Output:

Is the temperature (60F) good for skiing? false

Is the temperature (60F) good for hiking? true