# Strategy パターン

[Java]

・アルゴリズムをカプセル化して、それらを交換可能にする。
・アルゴリズムを、利用するクライアントから独立に変更することができるようになる

```
package strategy;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;

/**
 * Strategy パターン
 * アルゴリズムをカプセル化して、それらを交換可能にする。
 * アルゴリズムを、利用するクライアントから独立に変更することができるようになる
 *
 * Strategy クラス
 *   サポートするすべてのアルゴリズムに共通のインタフェースを宣言
 *    ・ Sorter
 *
 * ConcreteStrategy クラス
 *   Strategy クラスのインタフェースを利用して、アルゴリズムを実装
 *    ・ MargeSorter
 *    ・ InsertionSorter
 *    ・ BubbleSorter
 *
 * Context クラス
 *   ConcreteStrategy オブジェクトを備えている
 *   Strategy のオブジェクトに対する参照を保持
 *    ・ SortContext
 *
 */
public class Strategy {
    public static void main(String[] args) {
        String fileName = "c:\\work\\sort\\test.txt";
        String outName  = "c:\\work\\sort\\test.sort.txt";
        BufferedReader reader = null;
        PrintWriter    writer = null;
        String line = null;
        long st = 0;
        long et = 0;
        try {
            List<String> buf = new ArrayList<String>();
            reader = new BufferedReader(new FileReader(fileName));
            while ((line = reader.readLine()) != null) {
                buf.add(line);
            }
            String[] ary = new String[buf.size()];
            buf.toArray(ary);

            st = System.currentTimeMillis();

            // アルゴリズムの交換
            SortContext srt = new SortContext(new MargeSorter());
            // SortContext srt = new SortContext(new InsertionSorter());
            // SortContext srt = new SortContext(new BubbleSorter());

            ary = srt.sort(ary);
            et = System.currentTimeMillis();
            System.out.format("%d msec.", et - st);

            writer = new PrintWriter(outName);
            for (String s : ary) {
                writer.println(s);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                reader.close();
                writer.close();
            } catch (IOException e) {}
        }
```

```
    }
}

class SortContext {
    private Sorter sorter;
    public SortContext(Sorter sorter) {
        this.sorter = sorter;
    }
    public String[] sort(String[] text) {
        return sorter.sort(text);
    }
}
/**
 *
 */
abstract class Sorter {
    public abstract String[] sort(String[] text);
}
/**
 * マージソート
 */
class MargeSorter extends Sorter {
    @Override
    public String[] sort(String[] text) {
        if (text.length > 1) {
            int m = text.length /2;
            int n = text.length - m;
            String[] a1 = new String[m];
            String[] a2 = new String[n];
            System.arraycopy(text,0,a1,0,m);
            System.arraycopy(text,m,a2,0,n);
            a1 = sort(a1);
            a2 = sort(a2);
            text = marge(text,a1, a2);
        }
        return text;
    }
    private String[] marge(String[] text, String[] a1, String[] a2) {
        int i = 0;
        int j = 0;
        int k = 0;
        while (i < a1.length && j < a2.length ) {
            if (a1[i].compareTo(a2[j]) < 0 ) {
                text[k++] = a1[i++];
            } else {
                text[k++] = a2[j++];
            }
        }
        while (i < a1.length ) {
            text[k++] = a1[i++];
        }
        while (j < a2.length ) {
            text[k++] = a2[j++];
        }
        return text;
    }
}
/**
 * 挿入ソート
 */
class InsertionSorter extends Sorter {
    @Override
    public String[] sort(String[] text) {
        String line;
        for (int i=1; i<text.length; i++) {
            for (int j=0; j<i; j++) {
                if (text[i].compareTo(text[j]) < 0) {
                    line = text[i];
                    for (int k=i; k>j; k--) {
                        text[k] = text[k-1];
                    }
                    text[j] = line;
                    break;
                }
            }
        }
        return text;
    }
}
/**
 * バブルソート
 */
```

```
class BubbleSorter extends Sorter {
    @Override
    public String[] sort(String[] text) {
        String line;
        for (int i=0; i<text.length; i++) {
            for (int j=i; j<text.length; j++) {
                if (text[j].compareTo(text[i]) < 0) {
                    line = text[i];
                    text[i] = text[j];
                    text[j] = line;
                }
            }
        }
        return text;
    }
}
```