策略模式Strategy

波波老师~研发总监/资深架构师

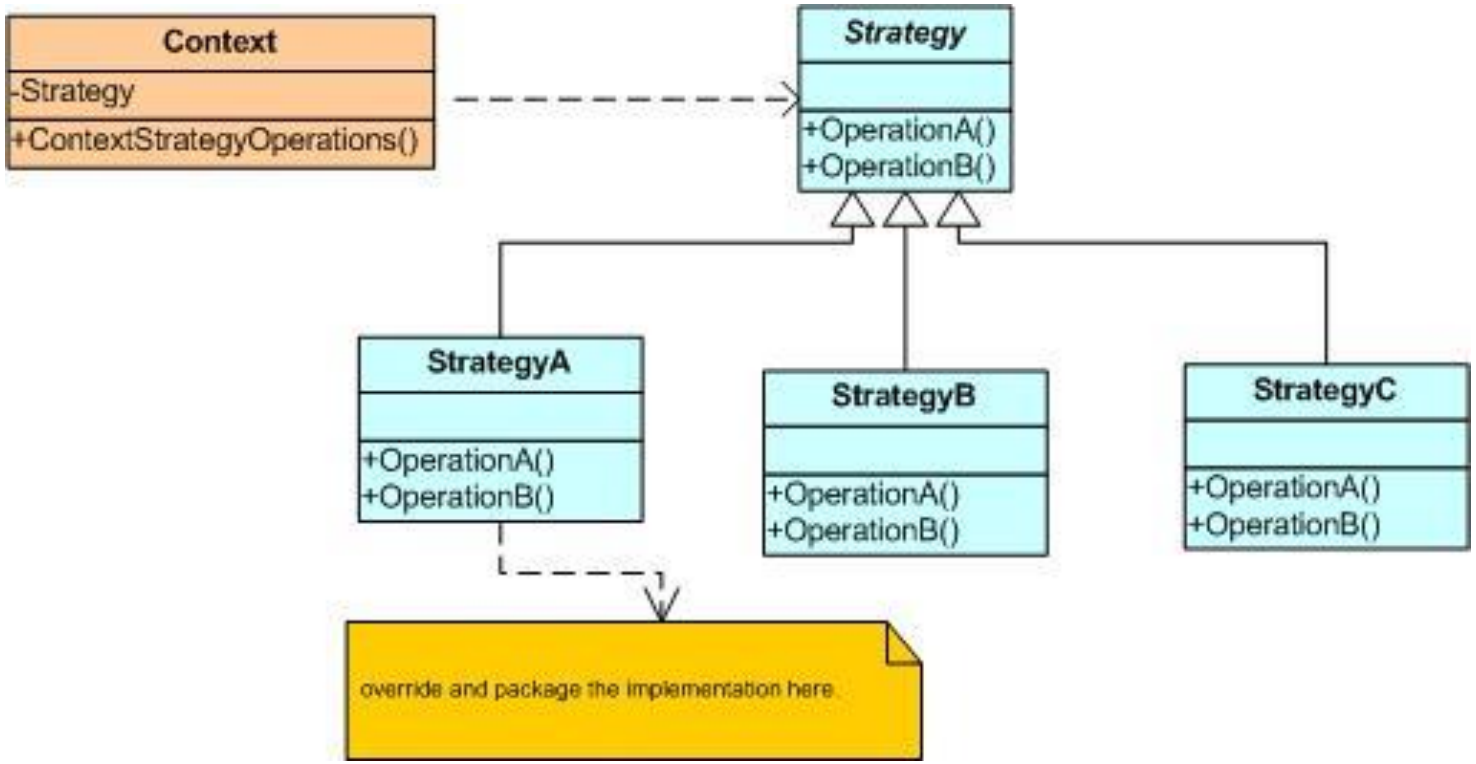
波波微课

spring2go.com

定义

- 策略模式定义**一族算法（a family of algorithms）**，将它们每个进行**封装(encapsulates)**，让它们变得**可替换(interchangeable)**
- 场景
  - 排序算法
  - 加密算法
  - 报表制作
  - 客户打折
  - 文件读取
- 原理：封装变化

关系图


Context
-Strategy
+ContextStrategyOperations()
Strategy
+OperationA()
+OperationB()
StrategyA
+OperationA()
+OperationB()
StrategyB
+OperationA()
+OperationB()
StrategyC
+OperationA()
+OperationB()
override and package the implementation here

案例~排序算法

V1代码~SortingType

```java
package io.spring2go.corespring.strategy;

public enum SortingType {
    MERGE_SORT, QUICK_SORT;
}
```

V1代码~SortingManagerV1

```
public class SortingManagerV1 {
    List list;

    public SortingManagerV1(List list) {
        this.list = list;
    }

    public void sortListBasedOnType(SortingType sortingType) {
        System.out.println("====================================");
        System.out.println("Sorting List based on Type");
        System.out.println("====================================");

        if (SortingType.MERGE_SORT == sortingType) {
            sortListUsingMergeSort();
        } else if (SortingType.QUICK_SORT == sortingType) {
            sortListUsingQuickSort();
        }
    }

    private void sortListUsingMergeSort() {
        System.out.println("Sorting List using merge sort");
    }

    private void sortListUsingQuickSort() {
        System.out.println("Sorting List using quick sort");
    }
}
```

V1代码~SortingMainV1

```java
import java.util.Arrays;
import java.util.List;

public class SortingMainV1 {
    public static void main(String[] args) {
        List list = Arrays.asList(new Integer[] { 44, 5, 3, 5, 5, 64, 3 });

        SortingManagerV1 sm = new SortingManagerV1(list);

        // Sorting using merge sort
        sm.sortListBasedOnType(SortingType.MERGE_SORT);

        System.out.println();

        // Sorting using quick sort
        sm.sortListBasedOnType(SortingType.QUICK_SORT);

    }

}
```

V1代码~添加HEAP_SORT

```
package io.spring2go.corespring.strategy;

public enum SortingType {
    MERGE_SORT, QUICK_SORT, HEAP_SORT;
}
```

V1代码~SortingManagerV1修改

```
public void sortListBasedOnType(SortingType sortingType) {
    System.out.println("====================================");
    System.out.println("Sorting List based on Type");
    System.out.println("====================================");

    if (SortingType.MERGE_SORT == sortingType) {
        sortListUsingMergeSort();
    } else if (SortingType.QUICK_SORT == sortingType) {
        sortListUsingQuickSort();
    } else if (SortingType.HEAP_SORT == sortingType) {
        sortListUsingHeapSort();
    }
}

private void sortListUsingMergeSort() {
    System.out.println("Sorting List using merge sort");
}

private void sortListUsingQuickSort() {
    System.out.println("Sorting List using quick sort");
}

private void sortListUsingHeapSort() {
    System.out.println("Sorting List using heap sort");
}
```

违反开放封闭原则

- 软件实体应该对扩展开放，对修改封闭

V2代码~SortingStrategy接口

```java
package io.spring2go.corespring.strategy;

import java.util.List;

// Strategy
public interface SortingStrategy {
    void sort(List<Integer> list);
}
```

V2代码~MergeSortingStrategy

```java
package io.spring2go.corespring.strategy;

import java.util.List;

// ConcreteStrategy
public class MergeSortStrategy implements SortingStrategy {

    @Override
    public void sort(List<Integer> list) {
        System.out.println("Sorting List using merge sort");
    }

}
```

V2代码~QuickSortStrategy

```java
package io.spring2go.corespring.strategy;

import java.util.List;

// ConcreteStrategy
public class QuickSortStrategy implements SortingStrategy {

    @Override
    public void sort(List<Integer> list) {
        System.out.println("Sorting List using quick sort");
    }

}
```

V2代码~SortingManagerV2

```
public class SortingManagerV2 {
    SortingStrategy sortingStrategy;
    List<Integer> list;

    public SortingManagerV2(List<Integer> list, SortingStrategy sortingStrategy) {
        super();
        this.list = list;
        this.sortingStrategy = sortingStrategy;
    }

    public void sortList() {
        System.out.println("====================================");
        System.out.println("Sorting List based on Type");
        System.out.println("====================================");

        sortingStrategy.sort(list);
    }

    public SortingStrategy getSortingStrategy() {
        return sortingStrategy;
    }

    public void setSortingStrategy(SortingStrategy sortingStrategy) {
        this.sortingStrategy = sortingStrategy;
    }
```

V2代码~SortingMainV2

```java
package io.spring2go.corespring.strategy;

import java.util.Arrays;
import java.util.List;

public class SortingMainV2 {

    public static void main(String[] args) {
        List<Integer> list = Arrays.asList(new Integer[] { 44, 5, 3, 5, 5, 64, 3 });
        MergeSortStrategy mergeSortStrategy = new MergeSortStrategy();
        SortingManagerV2 sm = new SortingManagerV2(list, mergeSortStrategy);
        sm.sortList();

        System.out.println();

        QuickSortStrategy quickSort = new QuickSortStrategy();
        sm.setSortingStrategy(quickSort);
        sm.sortList();
    }

}
```

V2代码~增加HeapSortStrategy

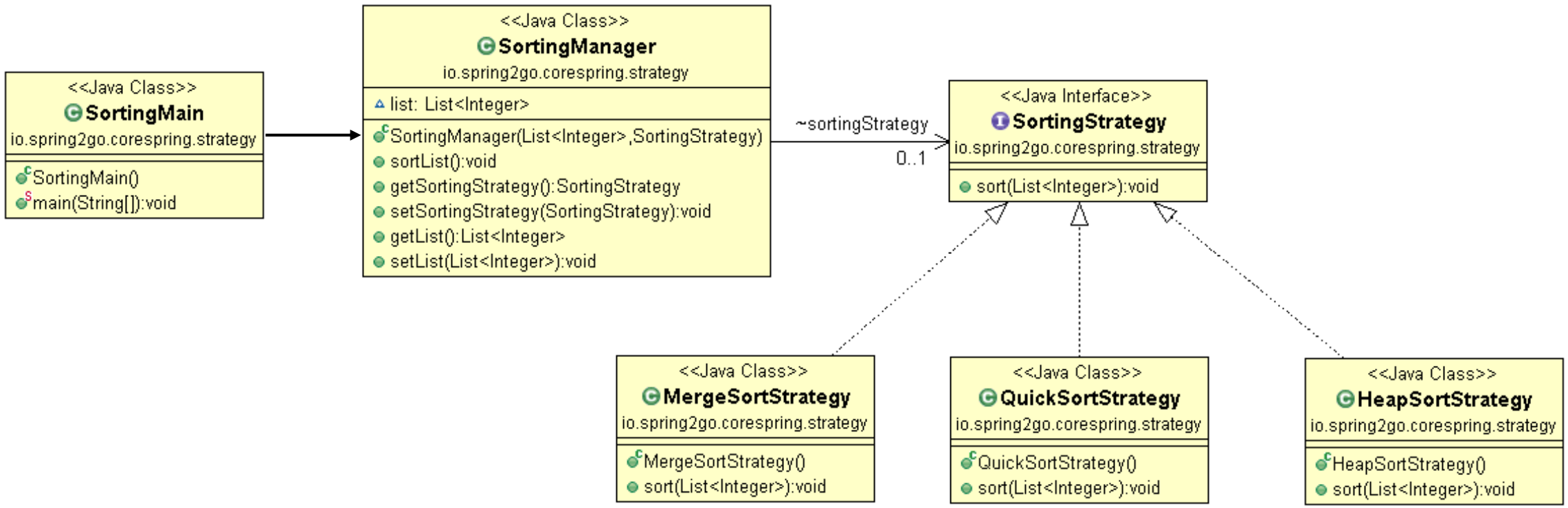
```java
package io.spring2go.corespring.strategy;

import java.util.List;

public class HeapSortStrategy implements SortingStrategy {

    @Override
    public void sort(List<Integer> list) {
        System.out.println("Sorting using heap sort");
    }

}
```

V2代码~SortingMainV2

```java
import java.util.Arrays;
import java.util.List;

public class SortingMainV2 {

    public static void main(String[] args) {
        List<Integer> list = Arrays.asList(new Integer[] { 44, 5, 3, 5, 5, 64, 3 });
        MergeSortStrategy mergeSortStrategy = new MergeSortStrategy();
        SortingManagerV2 sm = new SortingManagerV2(list, mergeSortStrategy);
        sm.sortList();

        System.out.println();

        QuickSortStrategy quickSort = new QuickSortStrategy();
        sm.setSortingStrategy(quickSort);
        sm.sortList();

        System.out.println();

        HeapSortStrategy heapSort = new HeapSortStrategy();
        sm.setSortingStrategy(heapSort);
        sm.sortList();
    }

}
```

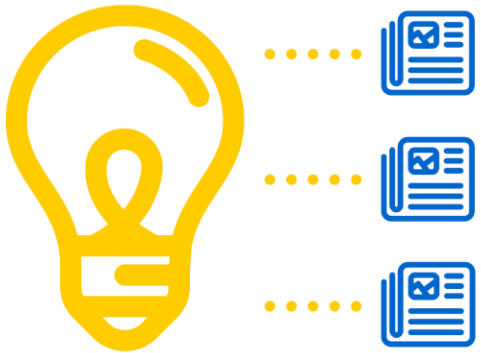
```
<terminated> SortingMain [Java Application] C
==================================
Sorting List based on Type
==================================
Sorting List using merge sort

==================================
Sorting List based on Type
==================================
Sorting List using quick sort

==================================
Sorting List based on Type
==================================
Sorting using heap sort
```

案例UML


<<Java Class>>
SortingMain
io.spring2go.corespring.strategy
SortingMain()
main(String[]):void
<<Java Class>>
SortingManager
io.spring2go.corespring.strategy
list: List<Integer>
SortingManager(List<Integer>,SortingStrategy)
sortList():void
getSortingStrategy():SortingStrategy
setSortingStrategy(SortingStrategy):void
getList():List<Integer>
setList(List<Integer>):void
~sortingStrategy
0..1
<<Java Interface>>
SortingStrategy
io.spring2go.corespring.strategy
sort(List<Integer>):void
<<Java Class>>
MergeSortStrategy
io.spring2go.corespring.strategy
MergeSortStrategy()
sort(List<Integer>):void
<<Java Class>>
QuickSortStrategy
io.spring2go.corespring.strategy
QuickSortStrategy()
sort(List<Integer>):void
<<Java Class>>
HeapSortStrategy
io.spring2go.corespring.strategy
HeapSortStrategy()
sort(List<Integer>):void

应用

- Java Collection.sort() & Comparator interface



课后练习

- 使用工厂模式封装具体策略的创建逻辑
- 策略模式和模板方法模式的差异？

参考

- Strategy design pattern in java
  - https://java2blog.com/strategy-design-pattern-java/
- Strategy Design Pattern (Case Study)
  - https://www.codeproject.com/Articles/39136/Strategy-Design-Pattern-Case-Study

代码

- https://github.com/spring2go/core-spring-patterns

波波微课

spring2go.com

Strategy Design Pattern