Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №4

З дисципліни “Технології розроблення програмного забезпечення”

Тема: “ ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY»”

Виконав
студент групи ІА–22:
Прохоров О.Д.

Перевірив
Мягкий М. Ю.

Київ 2024

Зміст

Короткі теоретичні відомості: 3
Крок 1. Код шаблону Стратегія..... 4
Крок 2. Основний класс системи. 5
Крок 3. Результати програми..... 6
Висновки. 6

Тема: ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY».

Мета: Навчитися використанню одного з зазначених шаблонів проектування.

Хід роботи:

Короткі теоретичні відомості:

Будь-який патерн проектування, використовуваний при розробці інформаційних систем, являє собою формалізований опис, який часто зустрічається в завданнях проектування, вдале рішення даної задачі, а також рекомендації по застосуванню цього рішення в різних ситуаціях. Крім того, патерн проектування обов'язково має загальновживане найменування. Правильно сформульований патерн проектування дозволяє, відшукавши одного разу вдале рішення, користуватися ним знову і знову.

Застосування шаблонів проектування не гарантує, що розроблена архітектура буде кристально чистою і зручною з точки зору програмування. Однак в потрібних місцях застосування шаблонів дозволить досягти наступних вигод:

• Зменшення трудовитрат і часу на побудову архітектури;

• Надання проектованій системі необхідних якостей (гнучкість, адаптованість, ін.);

• Зменшити накладні витрати на подальшу підтримку системи;

• Та інші.

Шаблон «Strategy» (Стратегія) дозволяє змінювати деякий алгоритм поведінки об'єкта іншим алгоритмом, що досягає ту ж мету іншим способом. Прикладом можуть служити алгоритми сортування: кожен алгоритм має власну реалізацію і визначений в окремому класі; вони можуть бути взаємозамінними в об'єкті, який їх використовує. Даний шаблон дуже зручний у випадках, коли існують різні «політики» обробки даних.

Крок 1. Код шаблону Стратегія.

```java
import com.example.cps_project_console.model.Goods;
import com.example.cps_project_console.model.User;

import java.util.List;

import java.util.ArrayList;
import java.util.Comparator;

1 usage
public class CostBasedDistributionStrategy implements DistributionStrategy {
    1 usage
    @Override
    public void distribute(List<Goods> goods, List<User> users) {

        List<Goods> mutableGoodsList = new ArrayList<>(goods);
        mutableGoodsList.sort(Comparator.comparingDouble(Goods::getActPrice).reversed());

        users.forEach(user -> user.setShoppingLists(new ArrayList<>()));

        double[] userCosts = new double[users.size()];

        for (Goods good : goods) {
            int minCostIndex = 0;
            for (int i = 1; i < userCosts.length; i++) {
                if (userCosts[i] < userCosts[minCostIndex]) {
                    minCostIndex = i;
                }
            }

            users.get(minCostIndex).getShoppingLists().add(good);
            userCosts[minCostIndex] += good.getActPrice();
        }
    }
}
```

CostBasedDistributionStrategy.java

```java
import com.example.cps_project_console.model.Goods;
import com.example.cps_project_console.model.User;

import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;

2 usages
public class EvenDistributionStrategy implements DistributionStrategy {
    1 usage
    @Override
    public void distribute(List<Goods> goodsList, List<User> users) {

        List<Goods> mutableGoodsList = new ArrayList<>(goodsList);
        mutableGoodsList.sort(Comparator.comparingDouble(Goods::getActPrice).reversed());

        if (users.isEmpty() || goodsList.isEmpty()) {
            System.out.println("Cannot distribute: either users or goods are empty.");
            return;
        }

        int userCount = users.size();
        int index = 0;

        for (Goods goods : goodsList) {
            User assignedUser = users.get(index % userCount);
            assignedUser.addGoodToShoppingList(goods);
            System.out.println("Assigned " + goods.getName() + " to " + assignedUser.getName());
            index++;
        }
    }
}
```

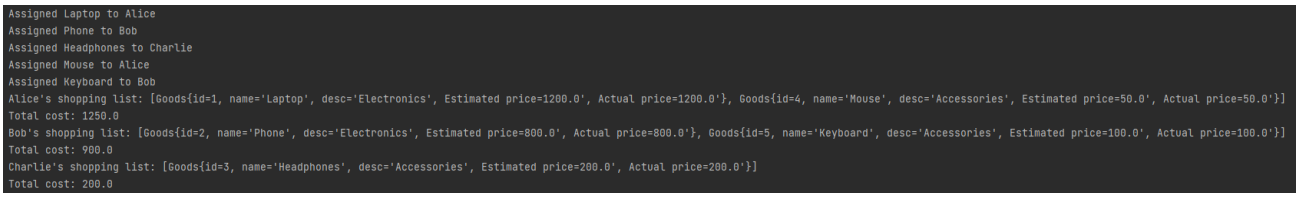
EvenDistributionStrategy.java

Крок 2. Основний класс системи.

```java
@SpringBootApplication
public class CpsProjectConsoleApplication {
	public static void main(String[] args) {
		SpringApplication.run(CpsProjectConsoleApplication.class, args);
		List<Goods> goodsList = List.of(
				new Goods( id: 1L,  name: "Laptop",  description: "Electronics",  estPrice: 1200.0,  actPrice: 1200.0),
				new Goods( id: 2L,  name: "Phone",  description: "Electronics",  estPrice: 800.0,  actPrice: 800.0),
				new Goods( id: 3L,  name: "Headphones",  description: "Accessories",  estPrice: 200.0,  actPrice: 200.0),
				new Goods( id: 4L,  name: "Mouse",  description: "Accessories",  estPrice: 50.0,  actPrice: 50.0),
				new Goods( id: 5L,  name: "Keyboard",  description: "Accessories",  estPrice: 100.0,  actPrice: 100.0)
		);

		List<User> users = List.of(
				new User( name: "Alice"),
				new User( name: "Bob"),
				new User( name: "Charlie")
		);

		DistributionStrategy strategy = new EvenDistributionStrategy();
		GroupPurchase groupPurchase = new GroupPurchase(strategy);

		groupPurchase.executeDistribution(goodsList, users);

		users.forEach(user -> {
			System.out.println(user.getName() + "'s shopping list: " + user.getShoppingLists());
			System.out.println("Total cost: " + user.getShoppingLists().stream()
					.mapToDouble(Goods::getActPrice).sum());
		});
	}
}
```

CpsProjectConsoleApplication.java

Крок 3. Результати програми



Results

Висновки.

В результаті цієї лабораторної роботи було виконано дії з створення коду, що відповідає шаблону проектування Стратегія та базового коду проекта.