**ЛР 4. Алгоритмы. Шаблоны Singletone, Command**

**Цель**: реализовать алгоритмы обработки коллекции объектов; освоить приемы, позволяющие отделить объект обработки от метода обработки, применив соответствующие шаблоны проектирования программ.

**2  Индивидуальное задание**

Используя созданные ранее классы и шаблон проектирования Command, разработать класс Menu как расширяемый контейнер команд, реализовать обработку данных коллекции и отдельных элементов (масштабирование, интерполяция, нормализация, сортировка, поиск и т.д.).

Реализовать возможность отмены (undo) операций (команд).

Продемонстрировать понятие "макрокоманда".

При разработке приложения использовать шаблон Singletone.

Обеспечить диалоговый интерфейс с пользователем.

Разработать класс для тестирования функциональности приложения.

Использовать комментарии для автоматической генерации документации средствами javadoc.

**3  Пример проекта**

**3.1  Разработка программы**

Реализуем классы, структура которых соответствует схеме п.2.1.2.

Разработаем класс MainTest для проведения теста класса ChangeItemCommand. Реализуем методы:

testExecute() – для проверки метода ChangeItemCommand.execute().

testChangeConsoleCommand() – для проверки основной функциональности класса ChangeConsoleCommand.

В процессе разработки необходимо обеспечить прохождение всех тестов.

**3.1.1  Используемые средства ООП**

Поведенческий шаблон Command (Action, Transaction) обеспечивает обработку команды в виде объекта. Применяется, когда необходимо отделить источник запроса от объекта, отвечающего на запрос; позволяет выполнить поддержку таких операций, как отмена, ведение журнала, операций с транзакциями.

Макрокоманда – это коллекция объектов класса Command.

Коллекция MacroCommand содержит список подкоманд. Когда вызывается метод выполнения макрокоманды, коллекция переадресует вызов этого метода всем своим подкомандам.

Производящий шаблон Singleton обеспечивает наличие в системе только одного экземпляра заданного класса, позволяя другим классам получать к нему доступ.

Применяется, если нужен объект, доступ к которому можно осуществить из любой точки приложения, но чтобы он создавался только один раз. Т.е. к этому объекту должны иметь доступ все элементы приложения, но работать они должны с одним и тем же экземпляром.

**3.1.2  Иерархия и структура классов**

Структура классов и схема их отношений приведена на рис.1.

**3.1.3  Описание программы**

При разработке класса Application использовался шаблон Singleton.

При реализации шаблона Command использовали:

         интерфейс команды (задачи) Command, обеспечивающий выполнение команды методом execute();

         интерфейс консольной команды ConsoleCommand, расширяющий Command методом, возвращающим горячую клавишу команды getKey();

         команда Change item – класс ChangeItemCommand, реализующий Command;

         консольная команда Change item – класс ChangeConsoleCommand, расширяющий  ChangeItemCommand и реализующий ConsoleCommand;

         консольная команда Generate – класс GenerateConsoleCommand, реализующий ConsoleCommand;

         консольная команда Restore – класс RestoreConsoleCommand, реализующий ConsoleCommand;

         Консольная команда Save – класс SaveConsoleCommand, реализующий ConsoleCommand;

         Консольная команда View – класс ViewConsoleCommand, реализующий ConsoleCommand;

При написании исходного кода используем стиль комментариев документации javadoc.



Рис.1. Схема классов и их отношений

Структура проекта:

|  |  |
| --- | --- |
| Папка srcUntitled-1 | Папка testUntitled-2 |

Выполним генерацию документации:



После проверки работоспособности готовой программы, создадим исполняемый JAR файл ex04.jar

**3.2  Текст программы**

**3.2.1  Main.java**

**package** ex04;

/\*\* Вычисление и отображение

 \* результатов; cодержит реализацию

 \* статического метода main()

 \* **@author** xone

 \* **@version** 4.0

 \* **@see** Main#main

 \*/

**public** **class** Main {

/\*\* Выполняется при запуске программы;

 \* вызывает метод {@linkplain Application#run()}

 \* **@param** args параметры запуска программы

 \*/

**public** **static** **void** main(String[] args) {

       Application app = Application.*getInstance*();

       app.run();

}

}

**3.2.2  Application.java**

**package** ex04;

**import** ex02.View;

**import** ex03.ViewableTable;

/\*\* Формирует и отображает

 \* меню; реализует шаблон

 \* Singleton

 \* **@author** xone

 \* **@version** 1.0

 \*/

**public** **class** Application {

/\*\* Ссылка на экземпляр класса Application; шаблон Singleton

 \* **@see** Application

 \*/

**private** **static** Application *instance* = **new** Application();

/\*\* Закрытый конструктор; шаблон Singleton

 \* **@see** Application

 \*/

**private** Application() {}

/\*\* Возвращает ссылку на экземпляр класса Application;

 \* шаблон Singleton

 \* **@see** Application

 \*/

**public** **static** Application getInstance() {

       **return** *instance*;

}

/\*\* Объект, реализующий интерфейс {@linkplain View};

 \* обслуживает коллекцию объектов {@linkplain ex01.Item2d};

 \* инициализируется с помощью Factory Method

 \*/

**private** View view = **new** ViewableTable().getView();

/\*\* Объект класса {@linkplain Menu};

 \* макрокоманда (шаблон Command)

 \*/

**private** Menu menu = **new** Menu();

/\*\* Обработка команд пользователя

 \* **@see** Application

 \*/

**public** **void** run() {

       menu.add(**new** ViewConsoleCommand(view));

       menu.add(**new** GenerateConsoleCommand(view));

       menu.add(**new** ChangeConsoleCommand(view));

       menu.add(**new** SaveConsoleCommand(view));

       menu.add(**new** RestoreConsoleCommand(view));

       menu.execute();

}

}

**3.2.3  ChangeConsoleCommand.java**

**package** ex04;

**import** ex01.Item2d;

**import** ex02.View;

**import** ex02.ViewResult;

/\*\* Консольная команда

 \* Change item;

 \* шаблон Command

 \* **@author** xone

 \* **@version** 1.0

 \*/

**public** **class** ChangeConsoleCommand

**extends** ChangeItemCommand

**implements** ConsoleCommand {

/\*\* Объект, реализующий интерфейс {@linkplain View};

 \* обслуживает коллекцию объектов {@linkplain ex01.Item2d}

 \*/

**private** View view;

/\*\* Возвращает поле {@linkplain ChangeConsoleCommand#view}

 \* **@return** значение {@linkplain ChangeConsoleCommand#view}

 \*/

**public** View getView() {

       **return** view;

}

/\*\* Устанавливает поле {@linkplain ChangeConsoleCommand#view}

 \* **@param** view значение для {@linkplain ChangeConsoleCommand#view}

 \* **@return** новое значение {@linkplain ChangeConsoleCommand#view}

 \*/

**public** View setView(View view) {

       **return** **this**.view = view;

}

/\*\* Инициализирует поле {@linkplain ChangeConsoleCommand#view}

 \* **@param** view объект, реализующий интерфейс {@linkplain View}

 \*/

**public** ChangeConsoleCommand(View view) {

       **this**.view = view;

}

@Override

**public** **char** getKey() {

       **return** 'c';

}

@Override

**public** String toString() {

       **return** "'c'hange";

}

@Override

**public** **void** execute() {

       System.*out*.println("Change item: scale factor " + setOffset(Math.*random*() \* 100.0));

       **for** (Item2d item : ((ViewResult)view).getItems()) {

              **super**.setItem(item);

              **super**.execute();

       }

       view.viewShow();

}

}

**3.2.4  ChangeItemCommand.java**

**package** ex04;

**import** ex01.Item2d;

/\*\* Команда

 \* Change item;

 \* шаблон Command

 \* **@author** xone

 \* **@version** 1.0

 \*/

**public** **class** ChangeItemCommand **implements** Command {

/\*\* Обрабатываемый объект; шаблон Command \*/

**private** Item2d item;

/\*\* Параметр команды; шаблон Command \*/

**private** **double** offset;

/\*\* Устанавливаент поле {@linkplain ChangeItemCommand#item}

 \* **@param** item значение для {@linkplain ChangeItemCommand#item}

 \* **@return** новое значение {@linkplain ChangeItemCommand#item}

 \*/

**public** Item2d setItem(Item2d item) {

       **return** **this**.item = item;

}

/\*\* Возвращает поле {@linkplain ChangeItemCommand#item}

 \* **@return** значение {@linkplain ChangeItemCommand#item}

 \*/

**public** Item2d getItem() {

       **return** item;

}

/\*\* Устанавливаент поле {@linkplain ChangeItemCommand#offset}

 \* **@param** offset значение для {@linkplain ChangeItemCommand#offset}

 \* **@return** новое значение {@linkplain ChangeItemCommand#offset}

 \*/

**public** **double** setOffset(**double** offset) {

       **return** **this**.offset = offset;

}

/\*\* Возвращает поле {@linkplain ChangeItemCommand#offset}

 \* **@return** значение {@linkplain ChangeItemCommand#offset}

 \*/

**public** **double** getOffset() {

       **return** offset;

}

@Override

**public** **void** execute() {

       item.setY(item.getY() \* offset);

}

}

**3.2.5  Command.java**

**package** ex04;

/\*\* Интерфейс команды

 \* или задачи;

 \* шаблоны: Command,

 \* Worker Thread

 \* **@author** xone

 \* **@version** 1.0

 \*/

**public** **interface** Command {

/\*\* Выполнение команды; шаблоны: Command, Worker Thread \*/

**public** **void** execute();

}

**3.2.6  ConsoleCommand.java**

**package** ex04;

/\*\* Интерфейс

 \* консольной команды;

 \* шаблон Command

 \* **@author** xone

 \* **@version** 1.0

 \*/

**public** **interface** ConsoleCommand **extends** Command {

/\*\* Горячая клавиша команды;

 \* шаблон Command

 \* **@return** символ горячей клавиши

 \*/

**public** **char** getKey();

}

**3.2.7  GenerateConsoleCommand.java**

**package** ex04;

**import** ex02.View;

/\*\* Консольная команда

 \* Generate;

 \* шаблон Command

 \* **@author** xone

 \* **@version** 1.0

 \*/

**public** **class** GenerateConsoleCommand **implements** ConsoleCommand {

/\*\* Объект, реализующий интерфейс {@linkplain View};

 \* обслуживает коллекцию объектов {@linkplain ex01.Item2d}

 \*/

**private** View view;

/\*\* Инициализирует поле {@linkplain GenerateConsoleCommand#view}

 \* **@param** view объект, реализующий интерфейс {@linkplain View}

 \*/

**public** GenerateConsoleCommand(View view) {

       **this**.view = view;

}

@Override

**public** **char** getKey() {

       **return** 'g';

}

@Override

**public** String toString() {

       **return** "'g'enerate";

}

@Override

**public** **void** execute() {

       System.*out*.println("Random generation.");

       view.viewInit();

       view.viewShow();

}

}

**3.2.8  Menu.java**

**package** ex04;

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.util.ArrayList;

**import** java.util.List;

/\*\* Макрокоманда

 \* (шаблон Command);

 \* Коллекция объектов

 \* класса ConsoleCommand

 \* **@see** ConsoleCommand

 \*/

**public** **class** Menu **implements** Command {

/\*\* Коллекция консольных команд;

 \* **@see** ConsoleCommand

 \*/

**private** List<ConsoleCommand> menu = **new** ArrayList<ConsoleCommand>();

/\*\* Добавляет новую команду в коллекцию

 \* **@param** command реализует {@linkplain ConsoleCommand}

 \* **@return** command

 \*/

**public** ConsoleCommand add(ConsoleCommand command) {

       menu.add(command);

       **return** command;

}

@Override

**public** String toString() {

       String s = "Enter command...\n";

       **for** (ConsoleCommand c : menu) {

              s += c + ", ";

       }

       s += "'q'uit: ";

       **return** s;

}

@Override

**public** **void** execute() {

       String s = **null**;

       BufferedReader in = **new** BufferedReader(**new** InputStreamReader(System.*in*));

       menu: **while** (**true**) {

              **do** {

                     System.*out*.print(**this**);

                     **try** {

                           s = in.readLine();

                     } **catch** (IOException e) {

                            System.*err*.println("Error: " + e);

                            System.*exit*(0);

                     }

              } **while** (s.length() != 1);

              **char** key = s.charAt(0);

              **if** (key == 'q') {

                     System.*out*.println("Exit.");

                     **break** menu;

              }

              **for** (ConsoleCommand c : menu) {

                     **if** (s.charAt(0) == c.getKey()) {

                            c.execute();

                            **continue** menu;

                     }

              }

              System.*out*.println("Wrong command.");

              **continue** menu;

       }

}

}

**3.2.9  RestoreConsoleCommand.java**

**package** ex04;

**import** ex02.View;

/\*\* Консольная команда

 \* Restore;

 \* шаблон Command

 \* **@author** xone

 \* **@version** 1.0

 \*/

**public** **class** RestoreConsoleCommand **implements** ConsoleCommand {

/\*\* Объект, реализующий интерфейс {@linkplain View};

 \* обслуживает коллекцию объектов {@linkplain ex01.Item2d}

 \*/

**private** View view;

/\*\* Инициализирует поле {@linkplain RestoreConsoleCommand#view}

 \* **@param** view объект, реализующий интерфейс {@linkplain View}

 \*/

**public** RestoreConsoleCommand(View view) {

       **this**.view = view;

}

@Override

**public** **char** getKey() {

       **return** 'r';

}

@Override

**public** String toString() {

       **return** "'r'estore";

}

@Override

**public** **void** execute() {

       System.*out*.println("Restore last saved.");

       **try** {

              view.viewRestore();

       } **catch** (Exception e) {

              System.*err*.println("Serialization error: " + e);

       }

       view.viewShow();

}

}

**3.2.10  SaveConsoleCommand.java**

**package** ex04;

**import** java.io.IOException;

**import** ex02.View;

/\*\* Консольная команда

 \* Save;

 \* шаблон Command

 \* **@author** xone

 \* **@version** 1.0

 \*/

**public** **class** SaveConsoleCommand **implements** ConsoleCommand {

/\*\* Объект, реализующий интерфейс {@linkplain View};

 \* обслуживает коллекцию объектов {@linkplain ex01.Item2d}

 \*/

**private** View view;

/\*\* Инициализирует поле {@linkplain SaveConsoleCommand#view}

 \* **@param** view объект, реализующий интерфейс {@linkplain View}

 \*/

**public** SaveConsoleCommand(View view) {

       **this**.view = view;

}

@Override

**public** **char** getKey() {

       **return** 's';

}

@Override

**public** String toString() {

       **return** "'s'ave";

}

@Override

**public** **void** execute() {

       System.*out*.println("Save current.");

       **try** {

              view.viewSave();

       } **catch** (IOException e) {

              System.*err*.println("Serialization error: " + e);

       }

       view.viewShow();

}

}

**3.2.11  ViewConsoleCommand.java**

**package** ex04;

**import** ex02.View;

/\*\* Консольная команда

 \* View;

 \* шаблон Command

 \* **@author** xone

 \* **@version** 1.0

 \*/

**public** **class** ViewConsoleCommand **implements** ConsoleCommand {

/\*\* Объект, реализующий интерфейс {@linkplain View};

 \* обслуживает коллекцию объектов {@linkplain ex01.Item2d}

 \*/

**private** View view;

/\*\* Инициализирует поле {@linkplain SaveConsoleCommand#view}

 \* **@param** view объект, реализующий интерфейс {@linkplain View}

 \*/

**public** ViewConsoleCommand(View view) {

       **this**.view = view;

}

@Override

**public** **char** getKey() {

       **return** 'v';

}

@Override

**public** String toString() {

       **return** "'v'iew";

}

@Override

**public** **void** execute() {

       System.*out*.println("View current.");

       view.viewShow();

}

}

**3.2.12  MainTest.java**

**package** ex04;

**import** **static** org.junit.Assert.\*;

**import** org.junit.Test;

**import** ex01.Item2d;

**import** ex02.ViewResult;

/\*\* Тестирование класса

 \* ChangeItemCommand

 \* **@author** xone

 \* **@version** 4.0

 \* **@see** ChangeItemCommand

 \*/

**public** **class** MainTest {

/\*\* Проверка метода {@linkplain ChangeItemCommand#execute()} \*/

@Test

**public** **void** testExecute() {

       ChangeItemCommand cmd = **new** ChangeItemCommand();

       cmd.setItem(**new** Item2d());

       **double** x, y, offset;

       **for** (**int** ctr = 0; ctr < 1000; ctr++) {

              cmd.getItem().setXY(x = Math.*random*() \* 100.0, y = Math.*random*() \* 100.0);

              cmd.setOffset(offset = Math.*random*() \* 100.0);

              cmd.execute();

              *assertEquals*(x, cmd.getItem().getX(), .1e-10);

              *assertEquals*(y \* offset, cmd.getItem().getY(), .1e-10);

       }

}

/\*\* Проверка класса {@linkplain ChangeConsoleCommand} \*/

@Test

**public** **void** testChangeConsoleCommand() {

       ChangeConsoleCommand cmd = **new** ChangeConsoleCommand(**new** ViewResult());

       cmd.getView().viewInit();

       cmd.execute();

       *assertEquals*("'c'hange", cmd.toString());

       *assertEquals*('c', cmd.getKey());

}

}

**3.3  Результаты тестирования**

Выполним ex04.MainTest как JUnit Test



Выполним запуск программы из командной строки:

java -jar ex04.jar

В результате выполнения получим:



**4  Заключение**

Разработали программу решения задачи индивидуального задания. Результаты тестирования подтверждают корректность используемых алгоритмов.

Для решения задачи применялись шаблоны проектирования Command, Singleton и Factory Method. Использовались некоторые классы предыдущей лабораторной работы.

Продемонстрирована возможность разделения объектов и методов обработки на примере реализации алгоритмов обработки коллекции объектов.

Для тестирования программы использовались средства JUnit.