МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«14» 02 2018 г.

ПАТТЕРН ПРОЕКТИРОВАНИЯ «СОСТОЯНИЕ»

Методические указания по выполнению лабораторной работы по дисциплине "Методология программной инженерии" для студентов направления подготовки магистров 09.04.04 "Программная инженерия"

Курск 2018

УДК 004.65

Составители: Т.М. Белова, В.Г. Белов

Рецензент
Кандидат технических наук, доцент кафедры программной инженерии ЮЗГУ И.Н. Ефремова

Паттерн проектирования «Состояние»: методические указания по выполнению лабораторной работы по дисциплине "Методология программной инженерии" для студентов направления подготовки магистров 09.04.04 "Программная инженерия"/ Юго-Зап. гос. ун-т; сост.: Т.М. Белова, В.Г. Белов, – Курск, 2018. – 16 с.: ил. 15.

Изложена последовательность действий по разработке и применению паттерна проектирования «Состояние» при работе в интегрированной среде разработки Eclipse.

Материал предназначен для студентов направления подготовки магистров 09.04.04 «Программная инженерия», а также будет полезен студентам всех направлений подготовки, изучающим технологии разработки программных продуктов с использованием паттернов.

Текст печатается в авторской редакции.

Подписано в печать 14.02.18. Формат 60x84 1/16.
Усл. печ. л. 0,7. Уч.-изд. л. 0,6. Тираж 50 экз. Заказ 489 . Бесплатно.
Юго-Западный государственный университет
305040, Курск, ул.50 лет Октября, 94.

Содержание

1 Цель лабораторной работы

Целью лабораторной работы является приобретение знаний, умений и навыков использования возможностей паттерна проектирования «Состояние» в разработке информационно-вычислительных систем.

2 Основные понятия

При реализации проектов по разработке программных систем и моделированию бизнес-процессов встречаются ситуации, когда решение проблем в различных проектах имеют сходные структурные черты. Попытки выявить похожие схемы или структуры в рамках объектно-ориентированного анализа и проектирования привели к появлению понятия паттерна, которое из абстрактной категории превратилось в непременный атрибут современных CASE-средств.

Паттерны различаются степенью детализации и уровнем абстракции. Предлагается следующая общая классификация паттернов по категориям их применения:

- архитектурные паттерны,
- паттерны проектирования,
- паттерны анализа,
- паттерны тестирования,
- паттерны реализации.

Архитектурные паттерны (Architectural patterns) - множество предварительно определенных подсистем со спецификацией их ответственности, правил и базовых принципов установления отношений между ними.

Архитектурные паттерны предназначены для спецификации фундаментальных схем структуризации программных систем. Наиболее известными паттернами этой категории являются паттерны GRASP (General Responsibility Assignment Software Pattern). Эти паттерны относятся к уровню системы и подсистем, но не к уровню классов. Как правило, формулируются в обобщенной форме, используют обычную терминологию и не зависят от области приложения. Паттерны этой категории систематизировал и описал К. Ларман.

Паттерны проектирования (Design patterns) - специальные схемы для уточнения структуры подсистем или компонентов программной системы и отношений между ними.

Паттерны проектирования описывают общую структуру взаимодействия элементов программной системы, которые реализуют

исходную проблему проектирования в конкретном контексте. Наиболее известными паттернами этой категории являются паттерны GoF (Gang of Four), названные в честь Э. Гаммы, Р. Хелма, Р. Джонсона и Дж. Влиссидеса, которые систематизировали их и представили общее описание. Паттерны GoF включают в себя 23 паттерна. Эти паттерны не зависят от языка реализации, но их реализация зависит от области приложения.

Паттерны анализа (Analysis patterns) - специальные схемы для представления общей организации процесса моделирования.

Паттерны анализа относятся к одной или нескольким предметным областям и описываются в терминах предметной области. Наиболее известными паттернами этой группы являются паттерны бизнес-моделирования ARIS (Architecture of Integrated Information Systems), которые характеризуют абстрактный уровень представления бизнес-процессов. Паттерны анализа конкретизируются в типовых моделях с целью выполнения аналитических оценок или имитационного моделирования бизнес-процессов.

Паттерны тестирования (Test patterns) - специальные схемы для представления общей организации процесса тестирования программных систем.

К этой категории паттернов относятся такие паттерны, как тестирование черного ящика, белого ящика, отдельных классов, системы. Паттерны этой категории систематизировал и описал М. Гранд. Некоторые из них реализованы в инструментальных средствах, наиболее известными из которых является IBM Test Studio. В связи с этим паттерны тестирования иногда называют стратегиями или схемами тестирования.

Паттерны реализации (Implementation patterns) - совокупность компонентов и других элементов реализации, используемых в структуре модели при написании программного кода.

Эта категория паттернов делится на следующие подкатегории: паттерны организации программного кода, паттерны оптимизации программного кода, паттерны устойчивости кода, паттерны разработки графического интерфейса пользователя и др. Паттерны этой категории описаны в работах М. Гранда, К. Бека, Дж. Тидвелла и др. Некоторые из них реализованы в популярных интегрированных средах программирования в форме шаблонов создаваемых проектов. В этом случае выбор шаблона программного приложения позволяет получить некоторую заготовку программного кода.

Шаблон проектирования или паттерн (англ. design pattern) в разработке программного обеспечения — повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

Технически, паттерны (шаблоны) проектирования - это абстрактные примеры правильного использования небольшого числа комбинаций простейших техник объектно-ориентированного программирования. Паттерны проектирования - это простые примеры, показывающие правильные способы организации взаимодействий между классами или объектами.

Паттерн "Состояние" (State) относится к группе, называемой поведенческой (Behavioral). Паттерны из этой группы определяют алгоритмы и взаимодействие между классами и объектами, то есть их поведение, увеличивая таким образом их гибкость.

В группу включены следующие паттерны:

- Цепочка обязанностей (Chain of responsibility),
- Команда (Command),
- Интерпретатор (Interpreter),
- Итератор (Iterator),
- Посредник (Mediator),
- Хранитель (Memento),
- Наблюдатель (Observer),
- Состояние (State),
- Стратегия (Strategy),
- Шаблонный метод (Template method),
- Посетитель (Visitor).

• Паттерн "Состояние " (State) - используется в тех случаях, когда во время выполнения программы объект должен менять свое поведение в зависимости от своего состояния.

3 Порядок выполнения лабораторной работы

1. Открыть Eclipse и создать новый Java Project с именем State (рисунок 1).

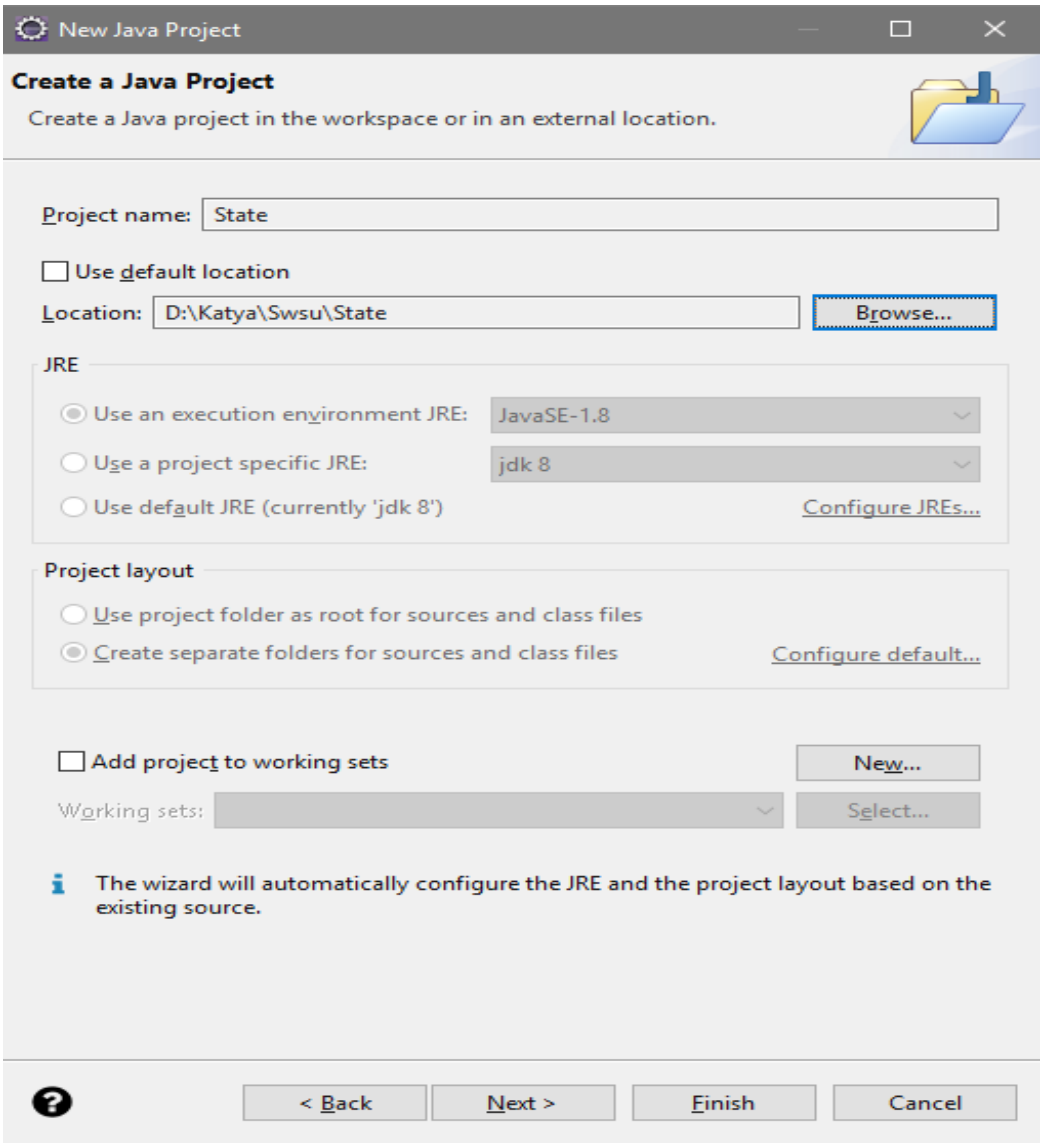

Рисунок 1

2. Схема задачи представлена ниже в виде UML-диаграммы, которая будет постепенно наполняться содержанием. Схема создается в этом проекте путем добавления PapyrusModel (рисунок 2).

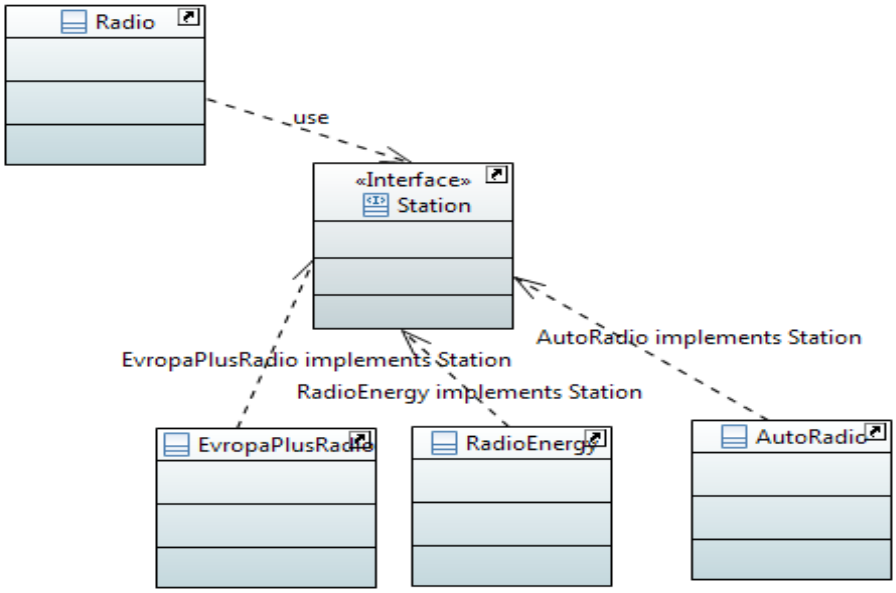

Рисунок 2

3. По этой схеме нужно сгенерировать Java-код. Автоматически создаются заготовки для будущих классов и интерфейса.
4. Добавить в интерфейс Station метод play (рисунок 3).

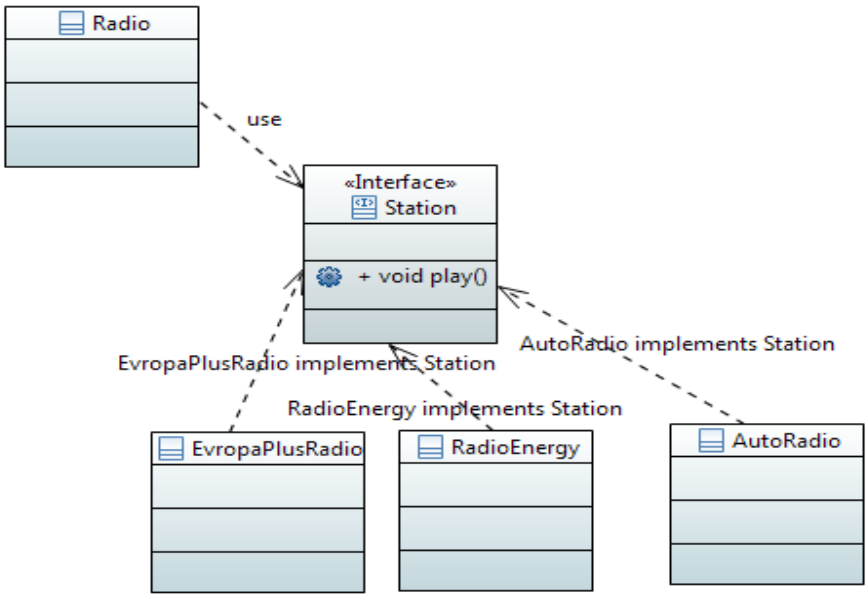

Рисунок 3

Интерфейс будет иметь следующий вид (рисунок 4):

Station.java *model.di

```
1  package States;
2
3  //State
4  public interface Station {
5
6      void play();
7  }
8
```

Рисунок 4

5. В классе AutoRadio реализовать метод play, который будет выводить результат проигрывания радиостанции (рисунок 5).

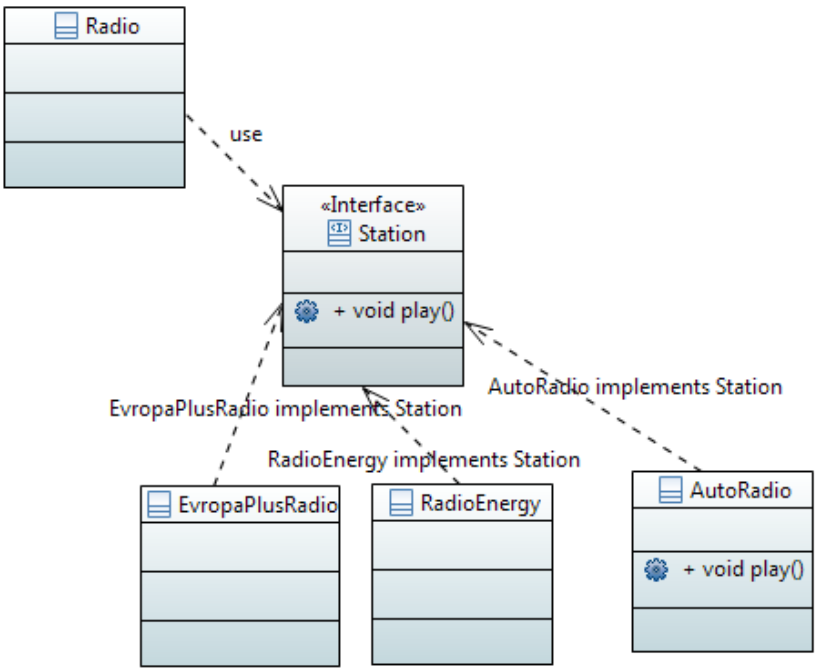

Рисунок 5

Класс будет иметь следующий вид (рисунок 6):

Station.java *model.di AutoRadio.java

```
1  package States;
2
3  public class AutoRadio implements Station{
4
5      public void play() {System.out.println("AutoRadio plays...");}
6  }
7
```

Рисунок 6

6. В классе RadioEnergy реализовать метод play, который будет выводить результат проигрывания радиостанции (рисунок 7).

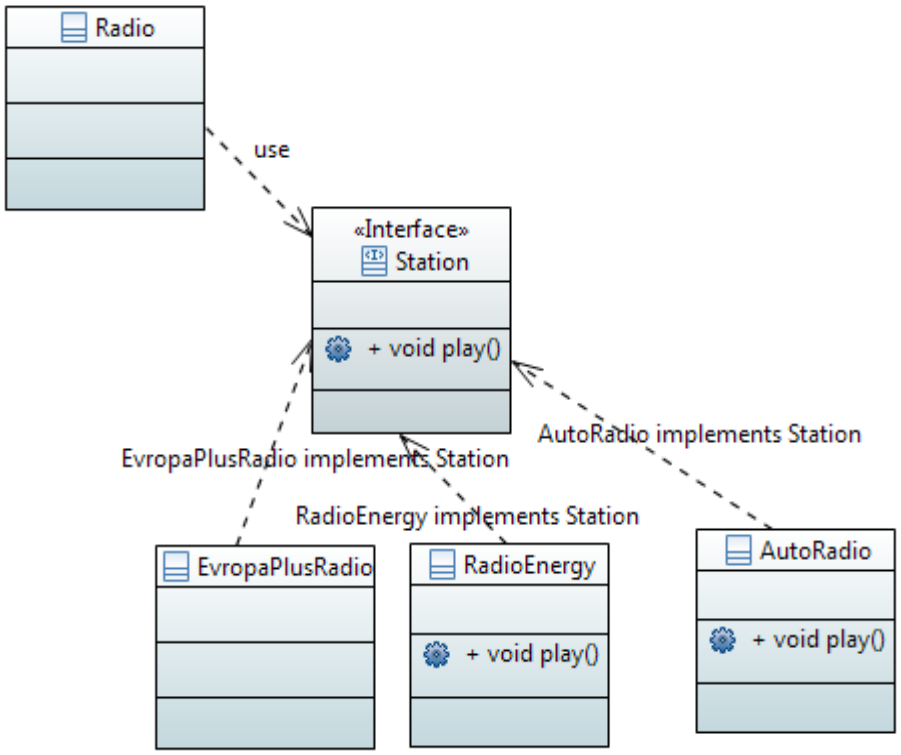

Рисунок 7

Класс будет иметь следующий вид (рисунок 8):

Station.java *model.di AutoRadio.java RadioEnergy.java

```
1  package States;
2
3  public class RadioEnergy implements Station{
4
5      public void play() {System.out.println("RadioEnergy plays...");}
6  }
7
```

Рисунок 8

7. В классе EvropaPlusRadio реализовать метод play, который будет выводить результат проигрывания радиостанции (рисунок 9).

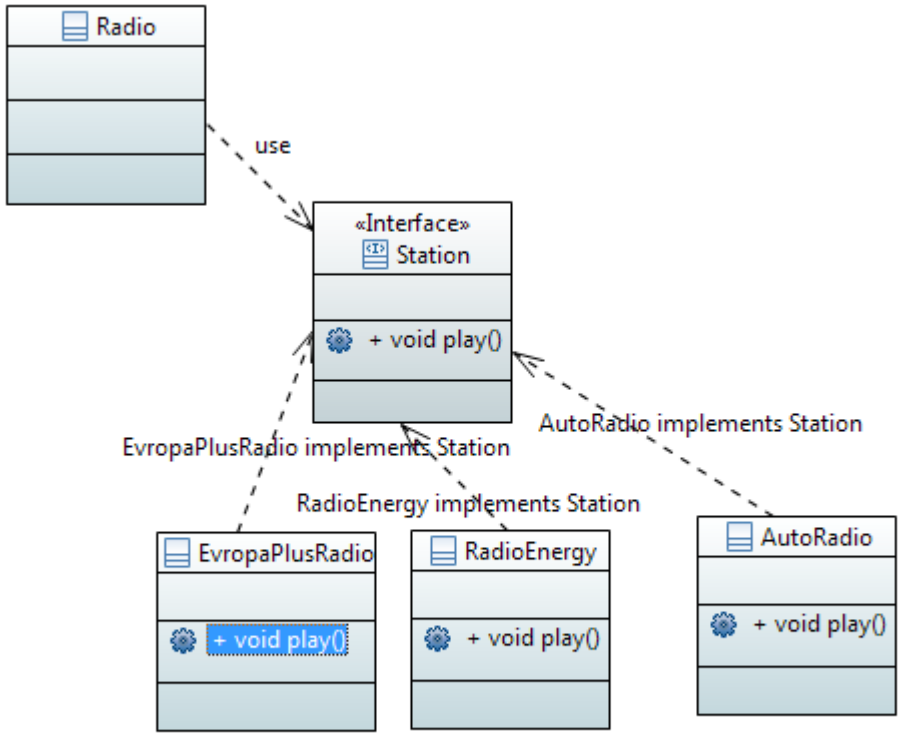

Рисунок 9

Класс будет иметь следующий вид (рисунок 10):

Station.java | *model.di | AutoRadio.java | RadioEnergy.java | EvropaPlusRadio.java

```
1  package States;
2
3  public class EvropaPlusRadio implements Station{
4
5      public void play() {System.out.println("EvropaPlusRadio plays...");}
6  }
7
```

Рисунок 10

8. В класс Radio (рисунок 11), использующий интерфейс Station, следует добавить методы setStation (устанавливает конкретное состояние) и nextStation (реализует установленное состояние).

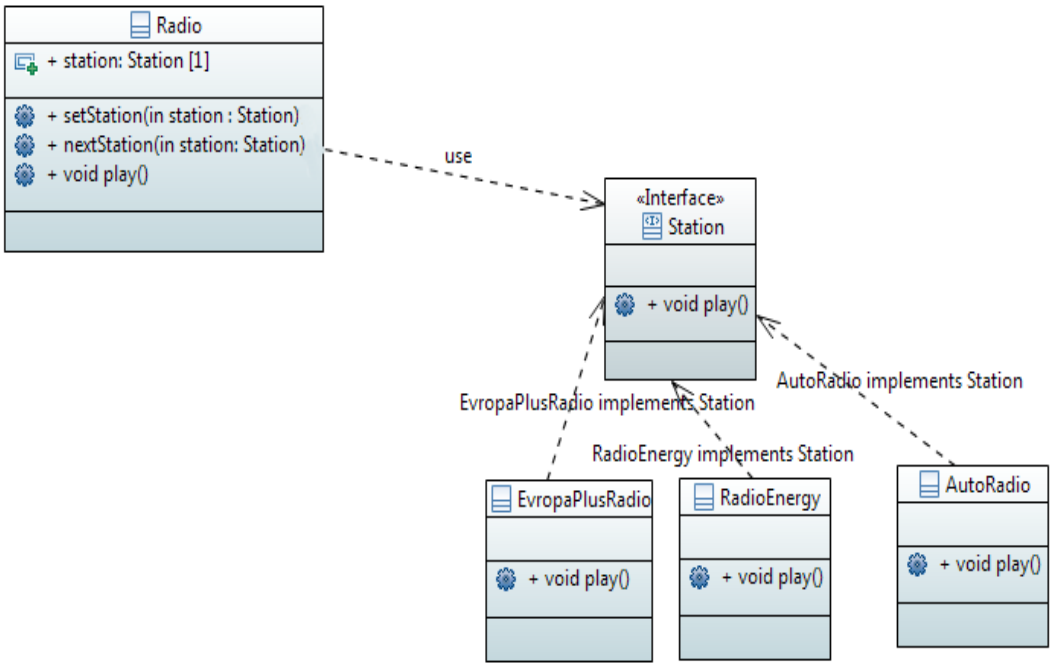

Рисунок 11

Класс будет иметь следующий вид (рисунок 12):

Station.java | *model.di | AutoRadio.java | RadioEnergy.java | EvropaPlusRadio.java | Radio.java

```
1  package States;
2
3  //Context
4  public class Radio {
5      Station station;
6      void setStation(Station st) {station = st;}
7      void nextStation() {
8          if (station instanceof RadioEnergy) {
9              setStation (new EvropaPlusRadio ());
10         }
11         else if (station instanceof EvropaPlusRadio) {
12             setStation (new AutoRadio ());
13         }
14         else if (station instanceof AutoRadio) {
15             setStation (new RadioEnergy ());
16         }
17
18     }
19     void play() {
20         station.play();
21     }
22 }
23
```

Рисунок 12

9. Нужно добавить класс ClassMain и поместить в него метод main, в котором следует написать тестовое приложение для паттерна «Состояние» (рисунок13).

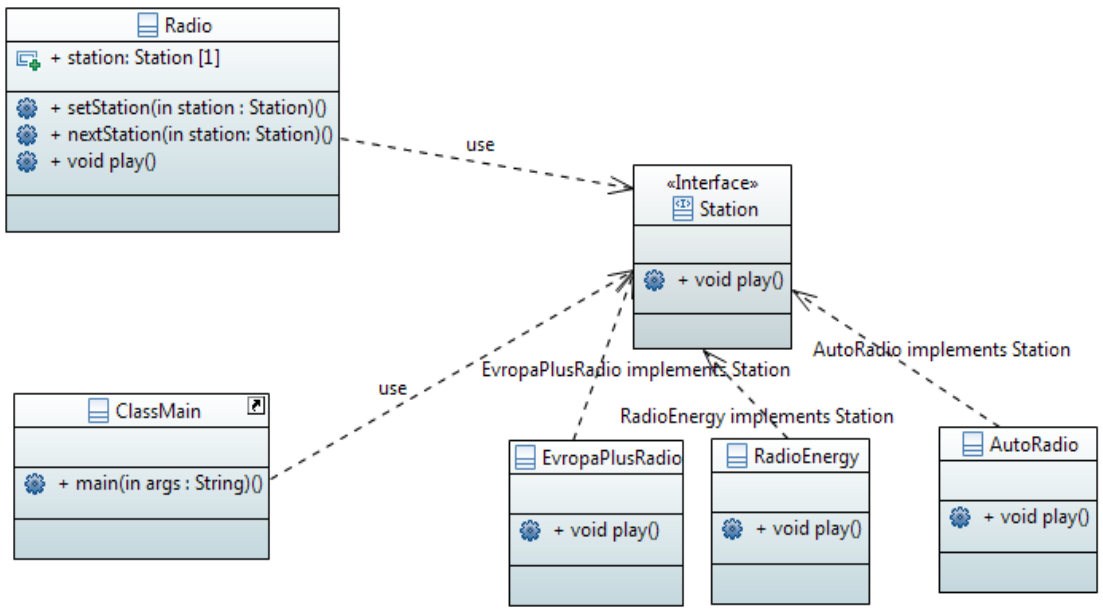

Рисунок 13

Класс тестового приложения будет иметь следующий вид (рисунок 14):

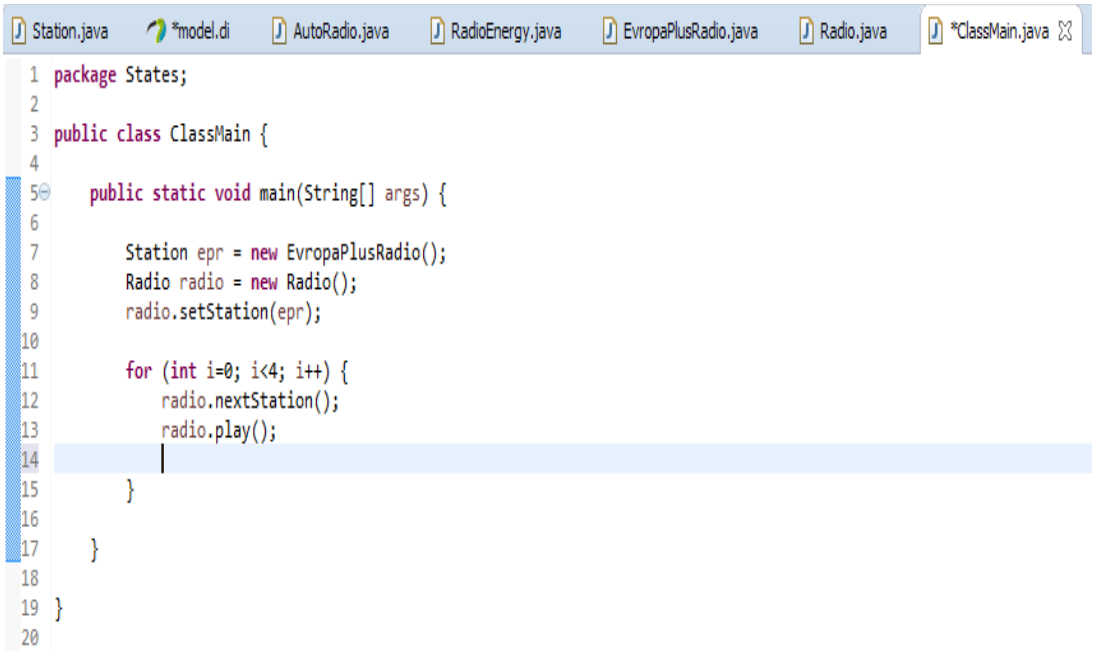

Рисунок 14

10. Результат выполнения тестового приложения (рисунок 15).

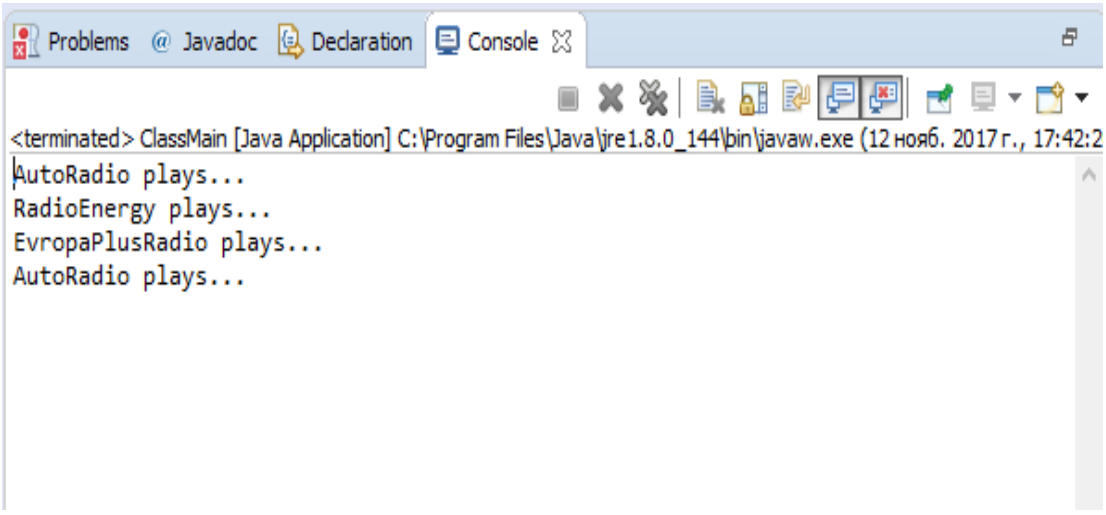

Рисунок 15

4 Содержание отчета по лабораторной работе

Отчет по лабораторной работе включает:

- титульный лист;
- условие задания;
- диаграммы классов для решения задачи;
- диаграммы последовательности для решения задачи;
- текст программы реализации паттерна проектирования «Состояние» для индивидуального задания;
- результаты тестирования программы.

5 Вопросы к защите лабораторной работы

1.Что такое паттерн проектирования?

2. Для чего предназначен паттерн проектирования «Состояние»?

3. К какому типу паттернов проектирования относится «Состояние»?

4. Какие классы/интерфейсы являются участниками «Состояние»?

5. Для чего необходим метод setStation в классе Radio?

6. Назовите родственные для «Состояния» паттерны проектирования.

7. Выделите достоинства и недостатки этого паттерна проектирования.

6 Индивидуальные задания

1. Реализовать программный продукт, который является примером конечного автомата с двумя состояниями и двумя событиями. При разработке использовать поведенческий паттерн проектирования «Состояние».
2. Реализовать программный продукт на тему «Ауидоплеер». Основной класс плеера меняет своё поведение в зависимости от того, в каком состоянии находится проигрывание. При разработке использовать поведенческий паттерн проектирования «Состояние».
3. Реализовать программный продукт на тему «Светофор». При разработке использовать поведенческий паттерн проектирования «Состояние».
4. Реализовать программный продукт на тему «Вода». Предусмотреть случай нахождения воды в трех состояниях. При разработке использовать поведенческий паттерн проектирования «Состояние».
5. Реализовать программный продукт на тему «Телевизор». Предусмотреть случай нахождения телевизора в трех состояния: «Выключен», «Включен», «Переключение канала». При разработке использовать поведенческий паттерн проектирования «Состояние».
6. Реализовать программный продукт на тему «Движение автомобиля». Предусмотреть случай нахождения автомобиля в трех состояниях. При разработке использовать поведенческий паттерн проектирования «Состояние».
7. Реализовать программный продукт на тему «Калькулятор». При разработке использовать поведенческий паттерн проектирования «Состояние».

Список использованных источников

1 Ларман, К. Применение UML 2.0 и шаблонов проектирования/ К. Ларман. - М.: Издательский дом «Вильямс», 2013. -736 с.

2 Османи, Э. Паттерны для масштабируемых JavaScript-приложений/ Э. Османи. - М.: Техносфера, 2015. -188 с.

3 Фримен Э. Паттерны проектирования/ Э. Фримен, Э. Фримен, К. Сьерра, Б. Бейтс. – СПб.: Питер, 2016. -653 с.