Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Емельянов Сергей Геннадьевич
Должность: ректор
Дата подписания: 18.02.2023 15:05:22
Уникальный программный ключ:
9ba7d3e34c012eba476ffd2d064cf2781953be730df2374d16f3c0ca53b0b1c6

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 «АБСТРАКТНАЯ ФАБРИКА»

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 «Абстрактная фабрика»: 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. – 26 с.: ил. 34.

Изложена последовательность действий по разработке и применению паттерна проектирования «**Абстрактная фабрика**» 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. 1,3. Уч.-изд. л. 1,2. Тираж 50 экз. Заказ 491 . 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 для использования паттерна проектирования «Абстрактная фабрика» в проектировании 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.

Порождающие паттерны (Creational) — это паттерны, которые абстрагируют процесс порождения классов и объектов. Они позволяют сделать систему независимой от способа создания, композиции и представления объектов. Шаблон, порождающий классы, использует наследование, чтобы изменять порождаемый класс, а шаблон, порождающий объекты, делегирует порождение другому объекту.

Среди них выделяются следующие:

- Абстрактная фабрика (Abstract Factory),
- Строитель (Builder),
- Фабричный метод (Factory Method),
- Прототип (Prototype),
- Одиночка (Singleton).

Абстрактная фабрика (Abstract factory) — предоставляет интерфейс для создания объектов, конкретные классы которых неизвестны.

Строитель (Builder) — представляет класс, который представляет собой интерфейс для создания сложного объекта. Отделяет конструирование объекта от представления, позволяя использовать один процесс конструирования для различных представлений.

Фабричный метод (Factory method) — определяет интерфейс для создания объекта, но оставляет подклассам решение о том, какой класс порождать.

Прототип (Prototype) — определяет интерфейс создания объекта через клонирование другого объекта вместо создания через конструктор.

Одиночка (Singleton) — гарантирует, что некоторый класс может иметь только один экземпляр.

«Абстрактная фабрика» является порождающим шаблоном проектирования, который предоставляет интерфейс для создания

семейств взаимосвязанных или взаимозависимых объектов без указания их конкретных классов.

Преимущества от применения паттернов проектирования заключаются в следующем:

- Паттерны позволяют суммировать опыт экспертов и сделать его доступным рядовым разработчикам.
- Имена паттернов образуют своего рода словарь, который позволяет разработчикам лучше понимать друг друга.
- Если в документации системы указано, какие паттерны в ней используются, это позволяет читателю быстрее понять систему.
- Паттерны упрощают реструктуризацию системы независимо от того, использовались ли паттерны при ее проектировании.
- Паттерн дает название проблеме и определяет способы решения многих проблем за счет готового набора абстракций.
- Использование шаблонов проектирования аналогично использованию готовых библиотек кода.
- Правильное использование шаблонов помогает разработчикам определить нужный вектор развития и уйти от многих проблем, которые могут возникнуть в процессе разработки.
- Паттерны проектирования не зависят от языка программирования.

Правильно выбранные паттерны проектирования позволяют сделать программную систему более гибкой, ее легче поддерживать и модифицировать, а код такой системы в большей степени соответствует концепции повторного использования.

Проблемы, которые порождают шаблоны проектирования:

- потеря гибкости проектирования и разработки системы;
- использование шаблонов усложняет систему;
- слепое следование определенному шаблону и повсеместное его использование может породить много архитектурных и логических проблем.

3 Порядок выполнения лабораторной работы

1. В интегрированной среде разработки Eclipse необходимо создать Java-проект AbstractFactory для разработки паттерна проектирования «Абстрактная фабрика» (рисунки 1–2).

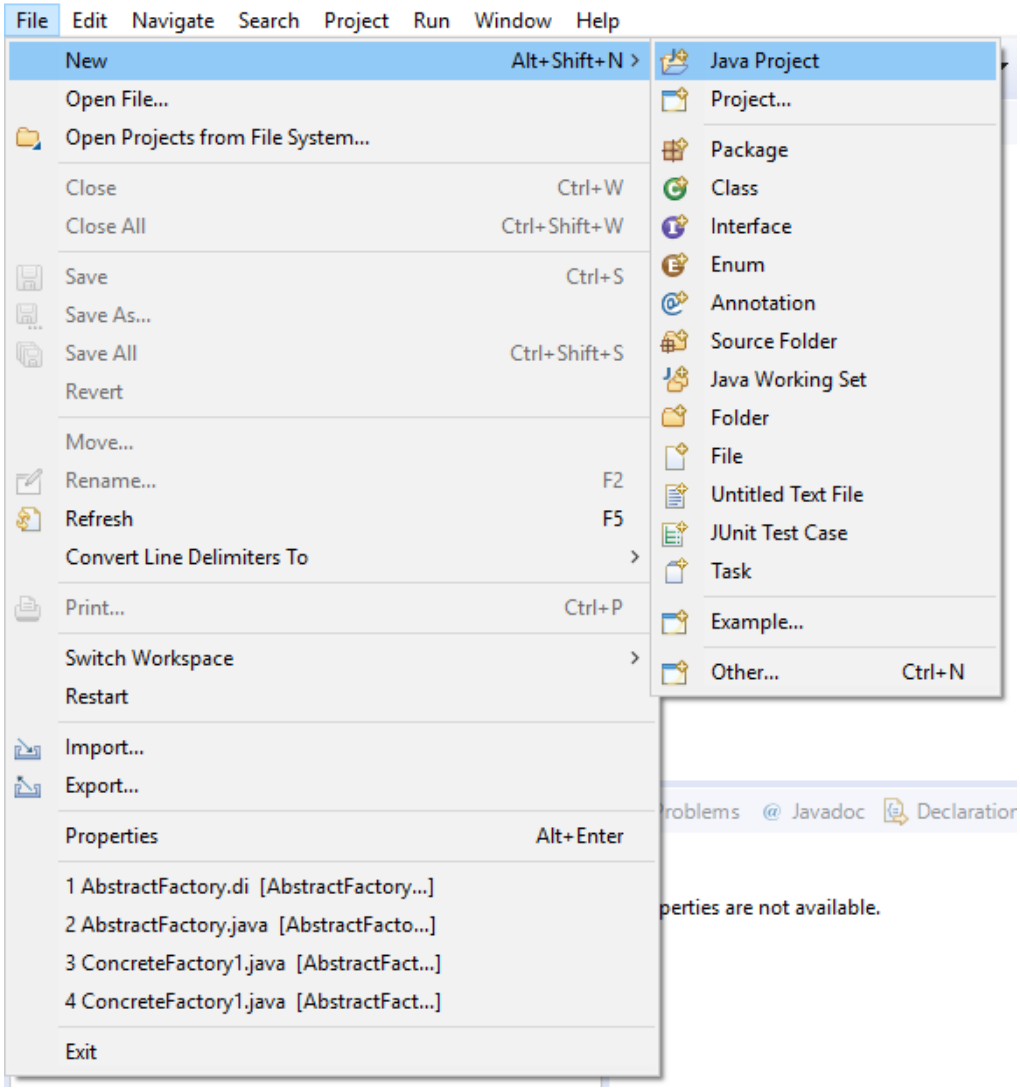

Рисунок 1

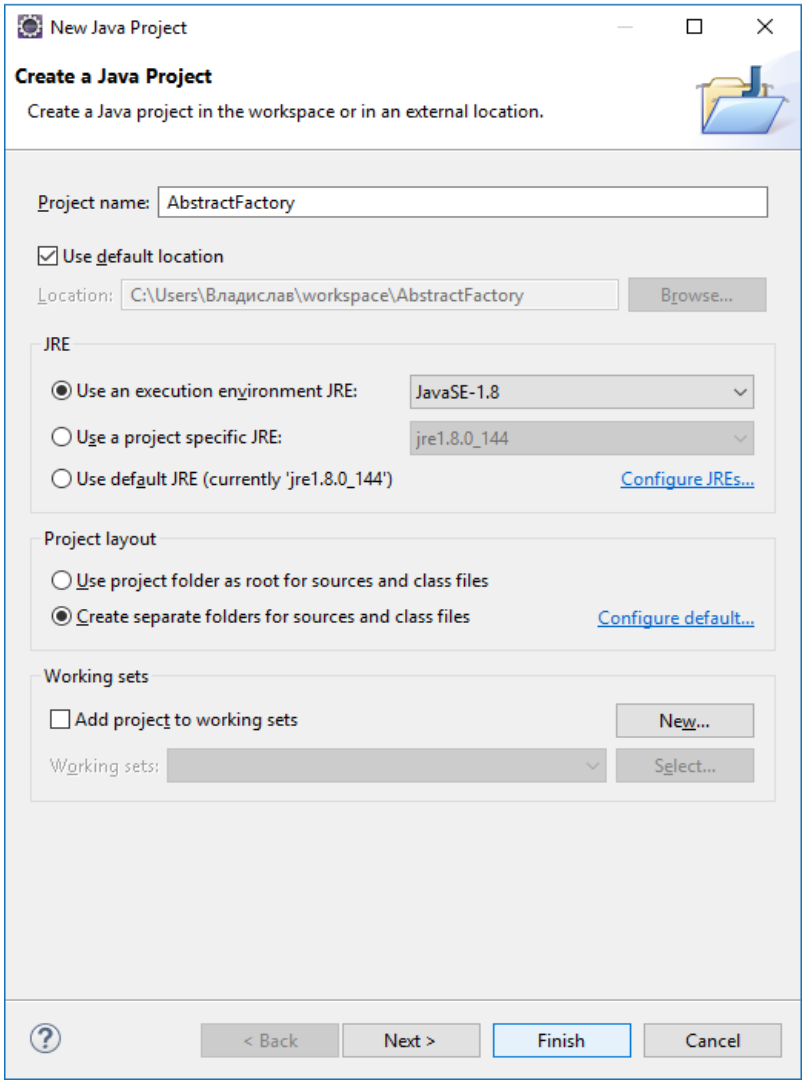

Рисунок 2

2. В проекте необходимо создать папку Model для проектирования диаграммы классов (рисунки 3–4).

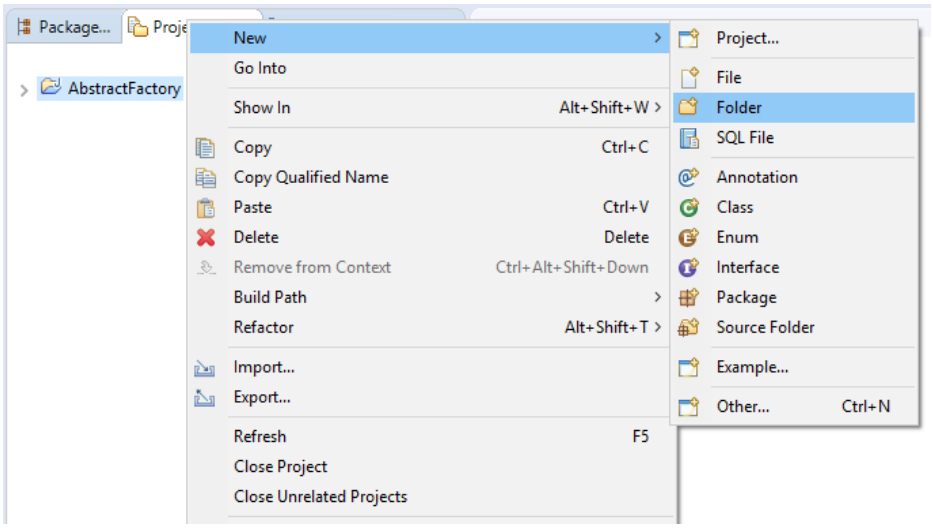

Рисунок 3

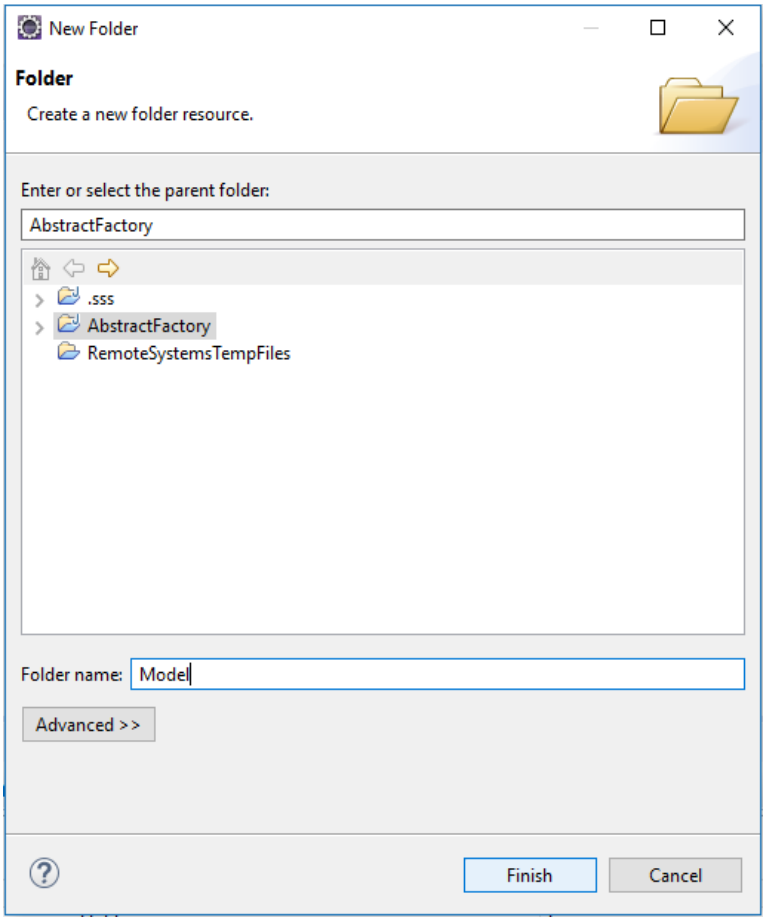

Рисунок 4

3. В созданной папке необходимо добавить файлы для работы с инструментом проектирования UML-диаграмм Papyrus (рисунки 5–10).

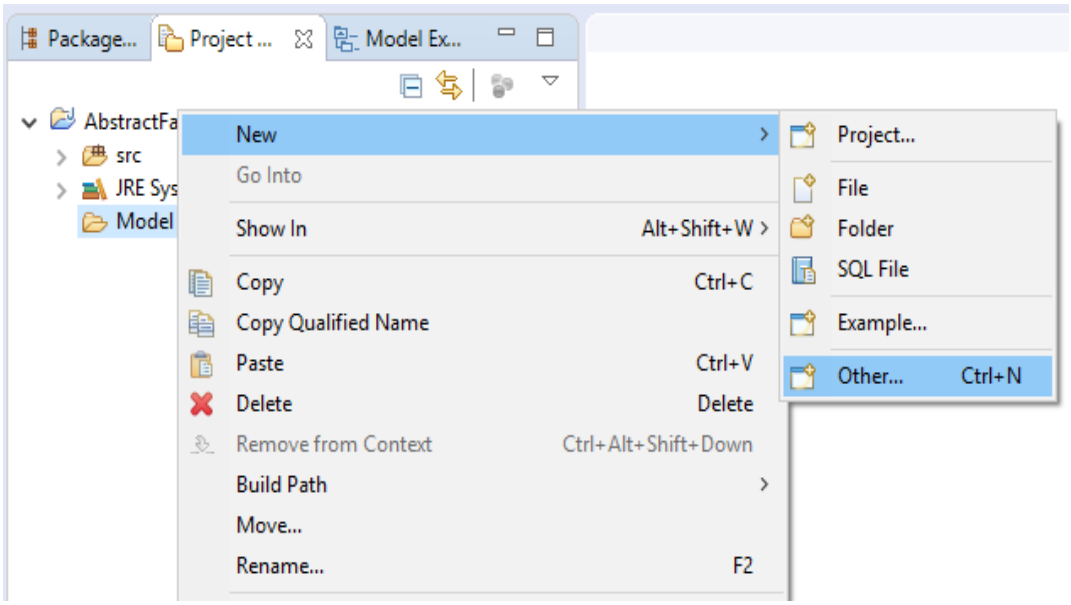

Рисунок 5

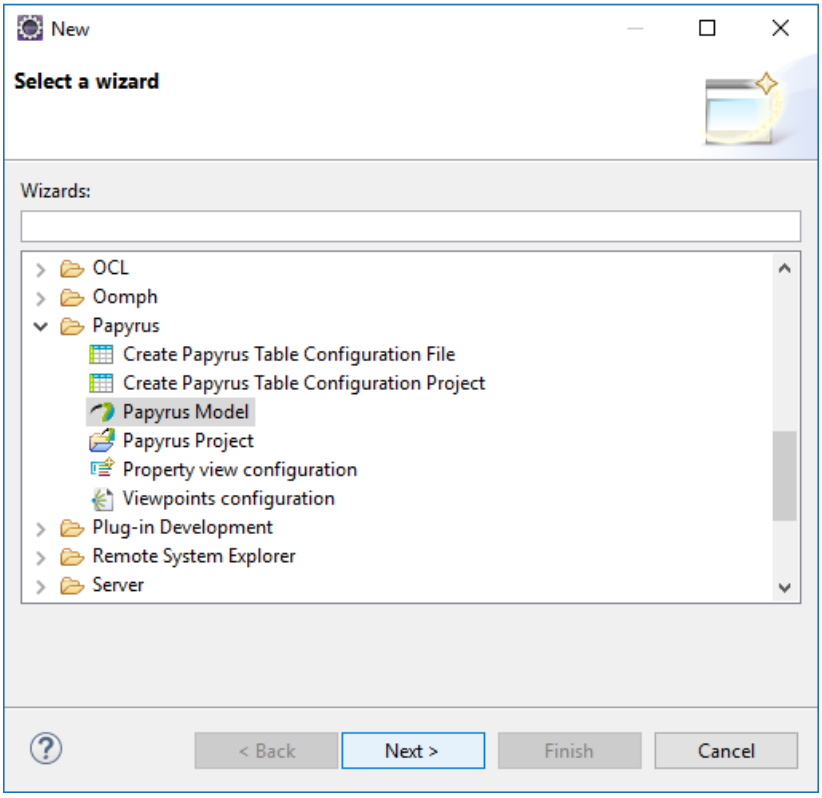

Рисунок 6

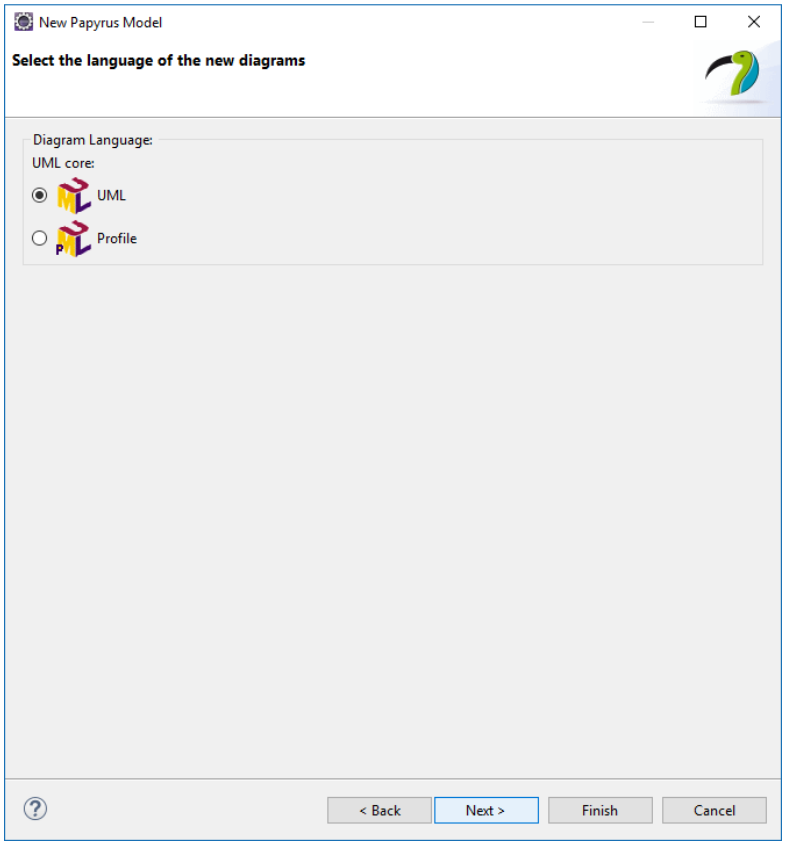

Рисунок 7

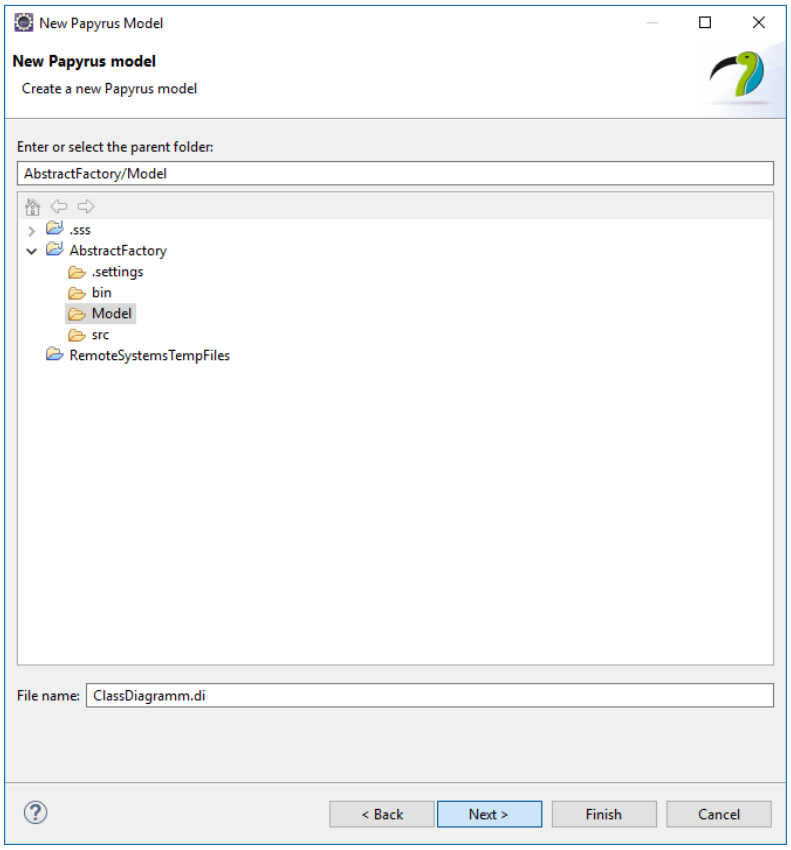

Рисунок 8

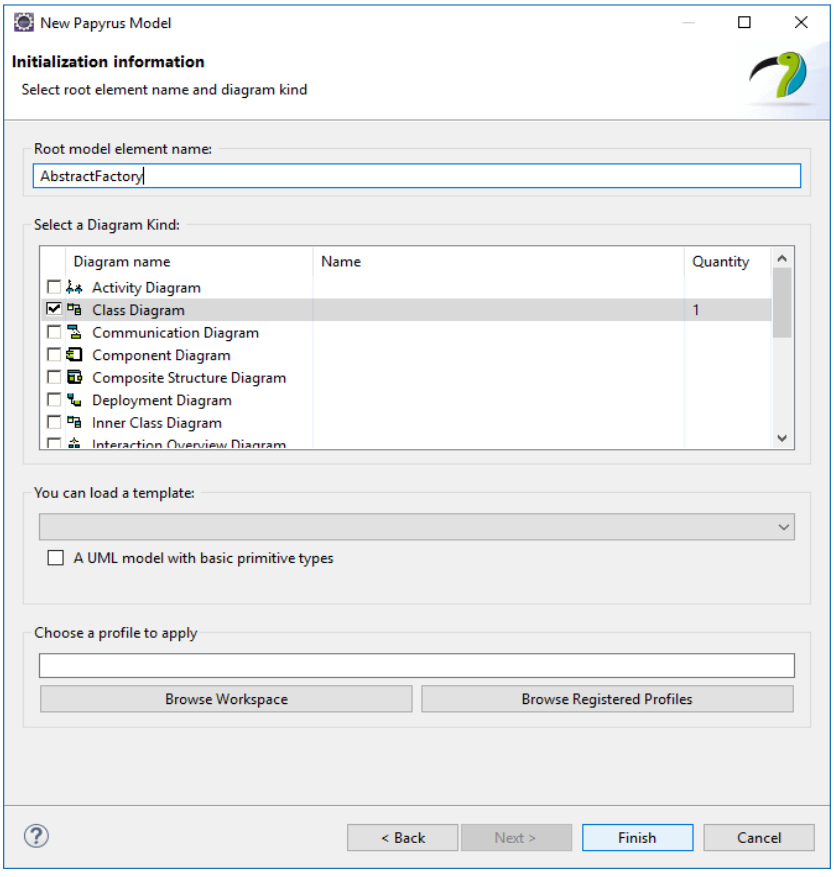

Рисунок 9

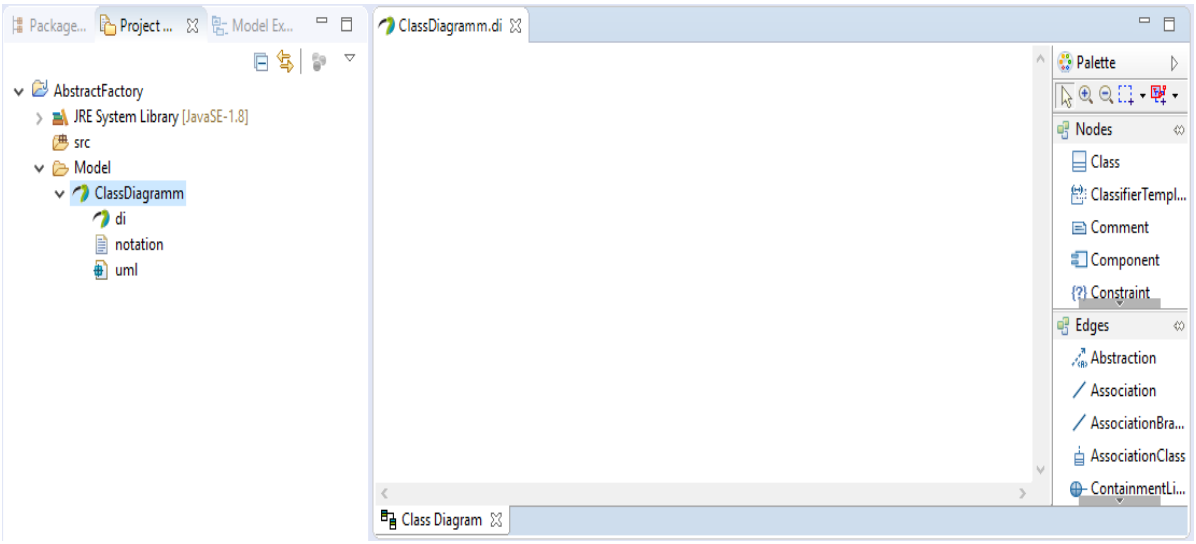

Рисунок 10

4. Необходимо разработать в проектировщике UML-диаграмм Papyrus диаграмму классов паттерна проектирования «Абстрактная фабрика» для конкретной задачи (рисунок 12) по схеме, представленной на рисунке 11.

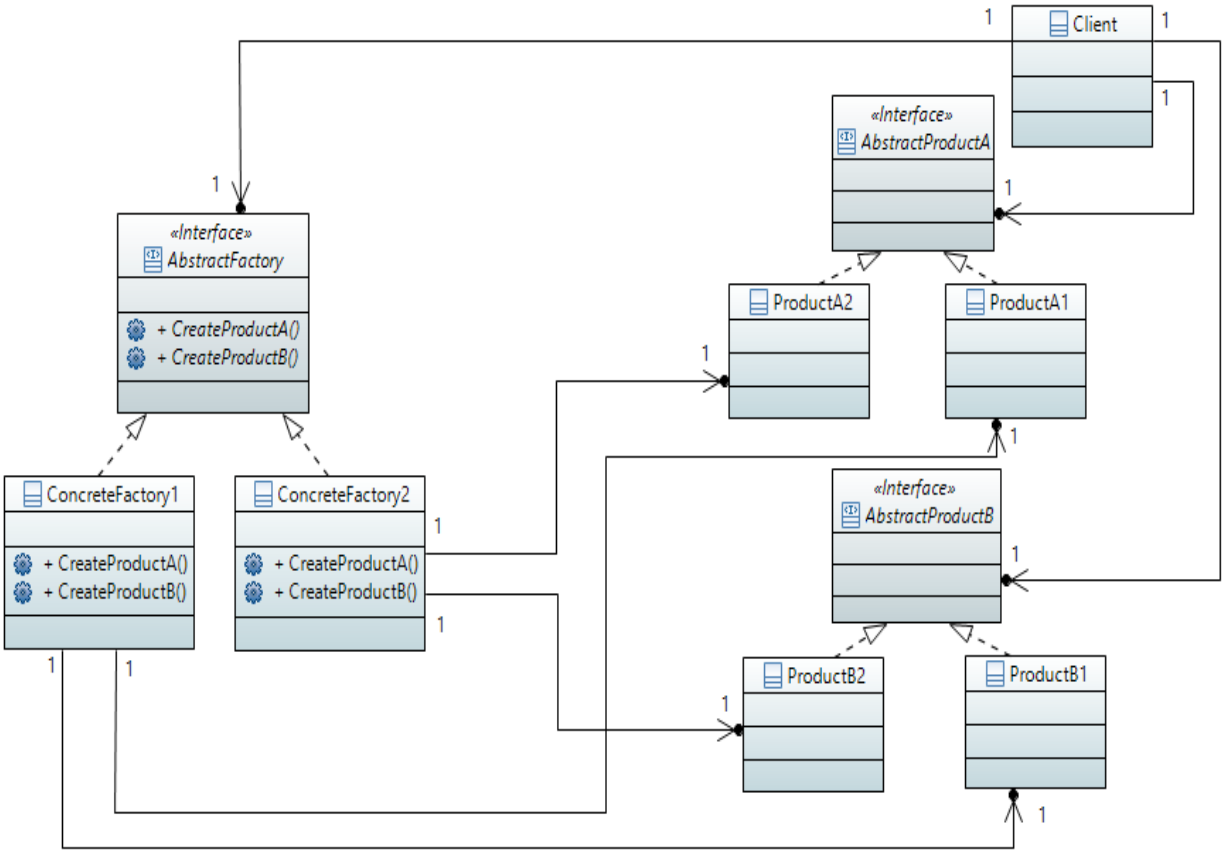

Рисунок 11

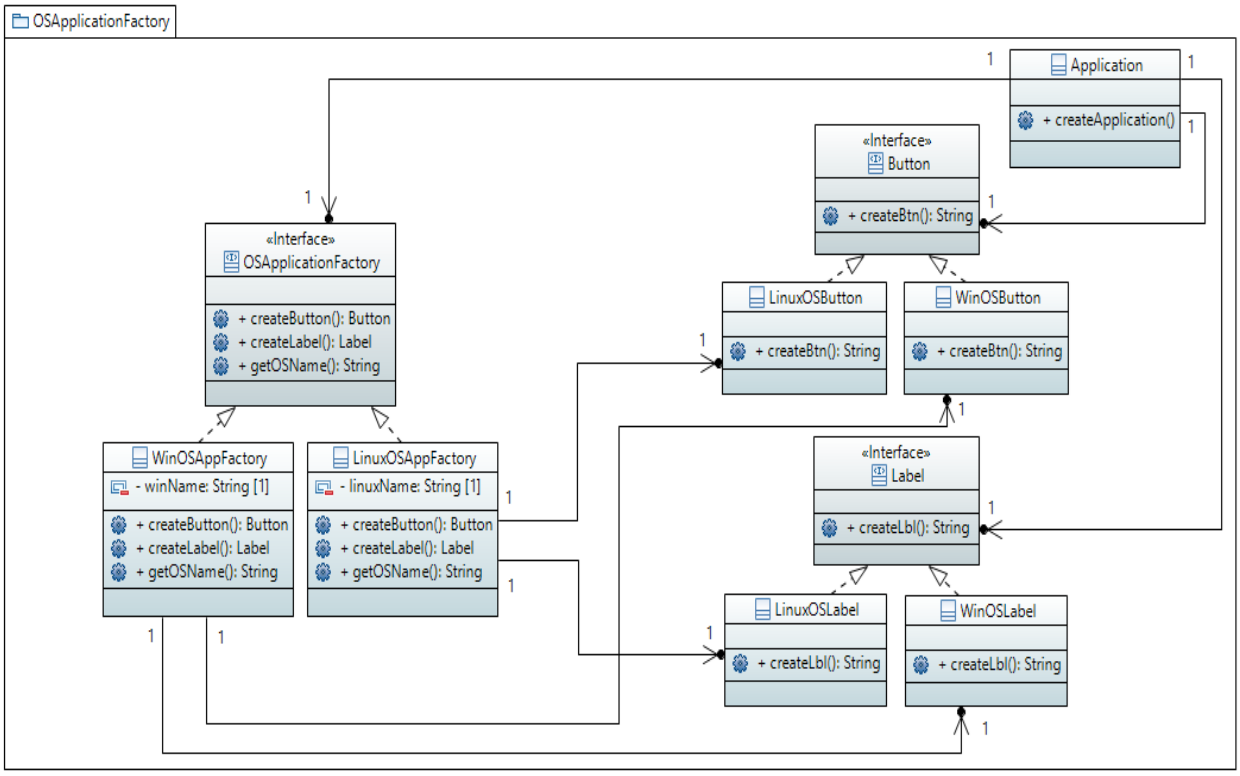

Рисунок 12

5. Полученную диаграмму классов необходимо сгенерировать в Java-код. Сгенерированные файлы объектов по диаграммам классов можно увидеть в папке src проекта (рисунок 13).

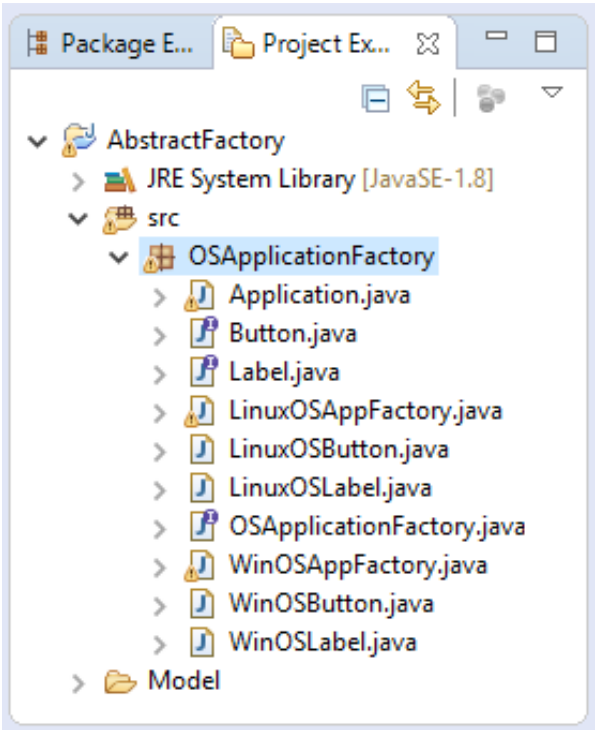

Рисунок 13

6. Необходима корректировка сгенерированного кода классов LinuxOSButton и WinOSButton (рисунки 14–15), являющихся наследниками интерфейса Button (рисунок 16). Интерфейс Button играет роль первого продукта, производимого конкретной фабрикой. Скорректированный код представлен на рисунках 17–18.

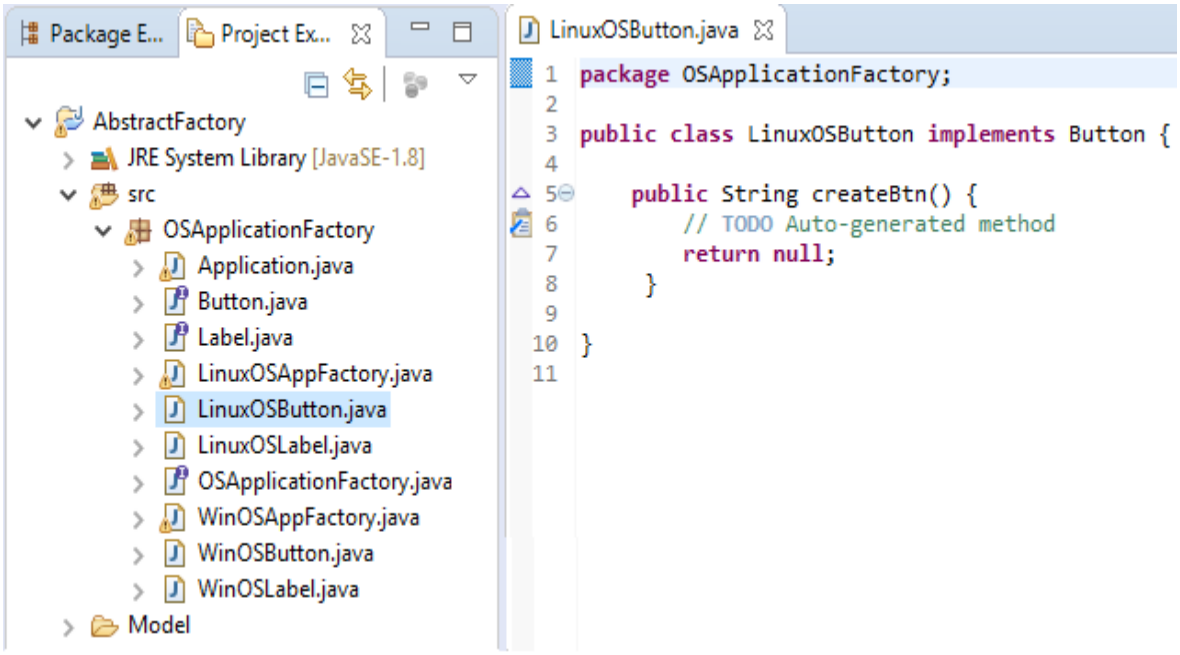

Рисунок 14

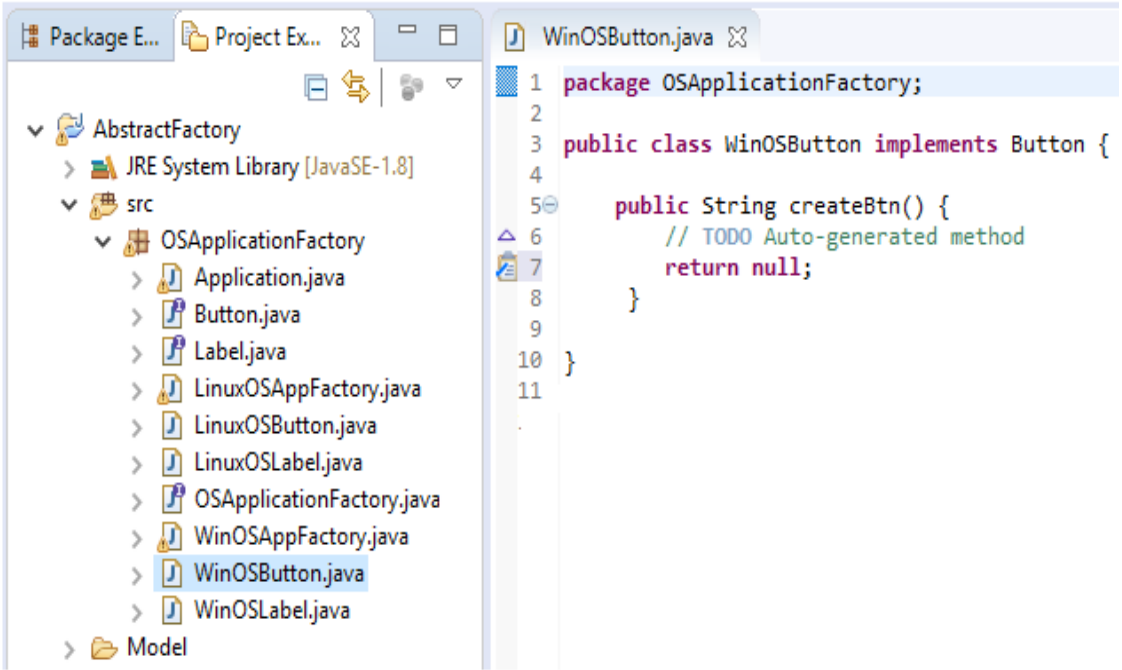

Рисунок 15

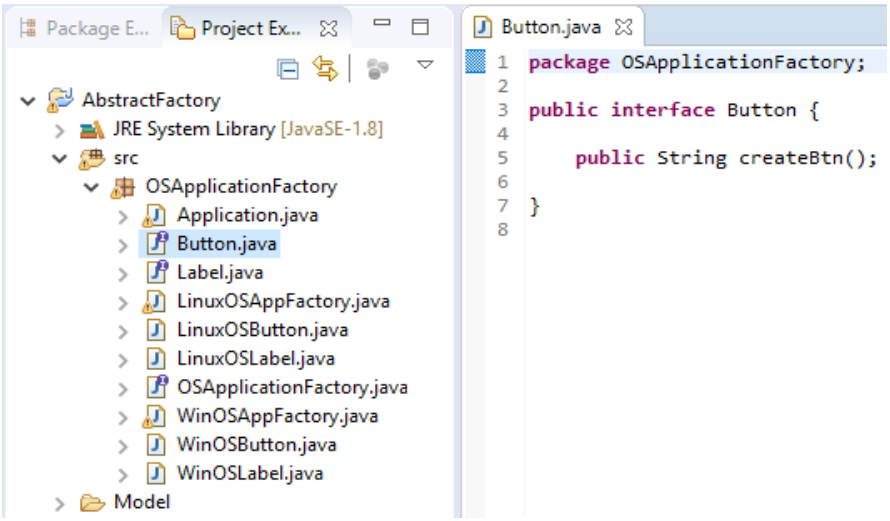

Рисунок 16

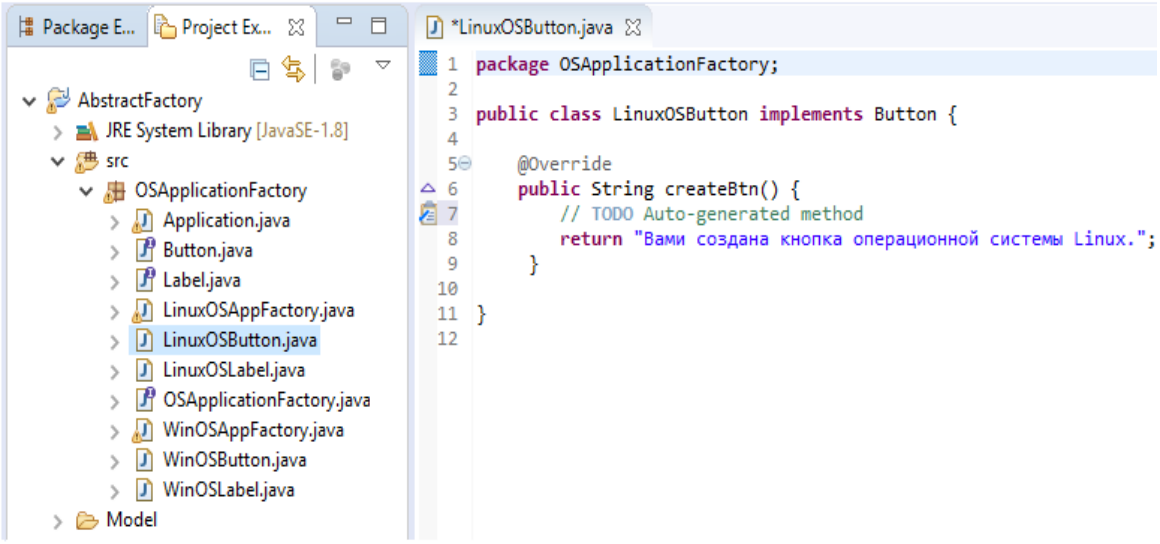

Рисунок 17

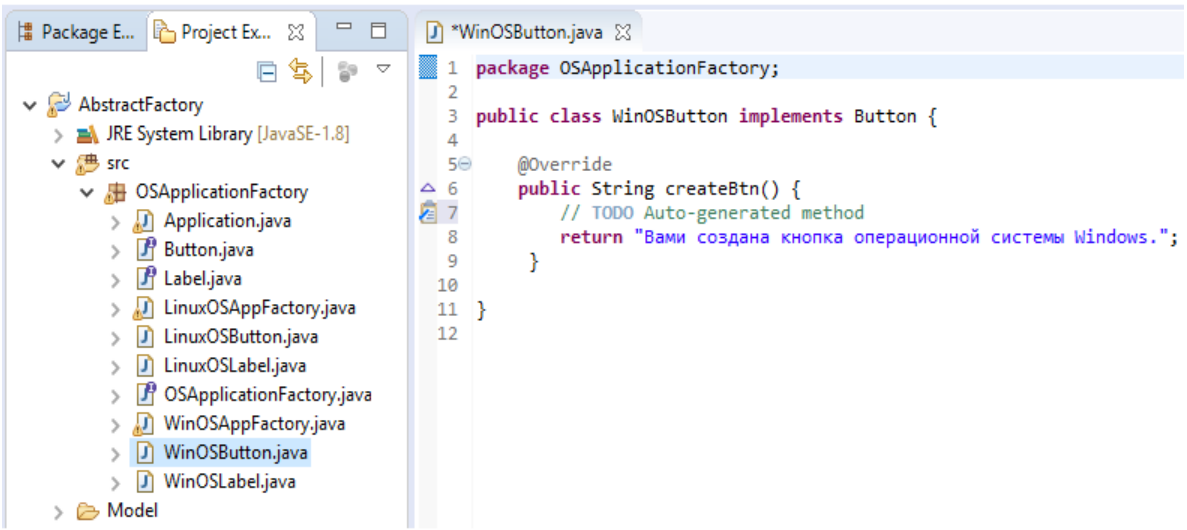

Рисунок 18

7. Необходима корректировка сгенерированного кода классов LinuxOSLabel и WinOSLabel (рисунки 19–20), являющихся наследниками интерфейса Label (рисунок 21). Интерфейс Label играет роль второго продукта, производимого конкретной фабрикой. Скорректированный код представлен на рисунках 22–23.

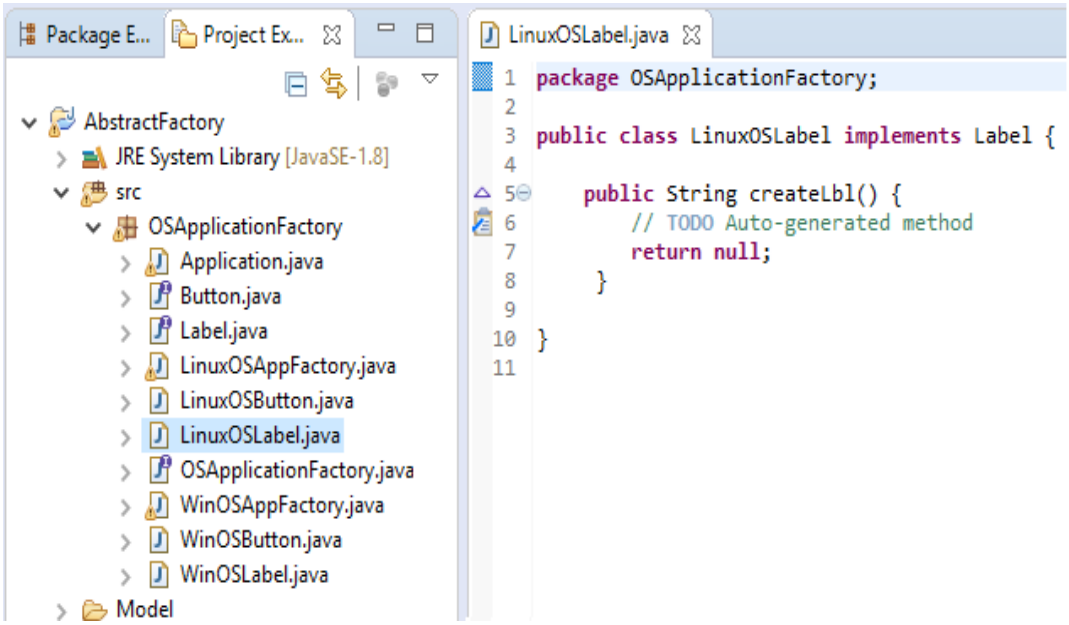

Рисунок 19

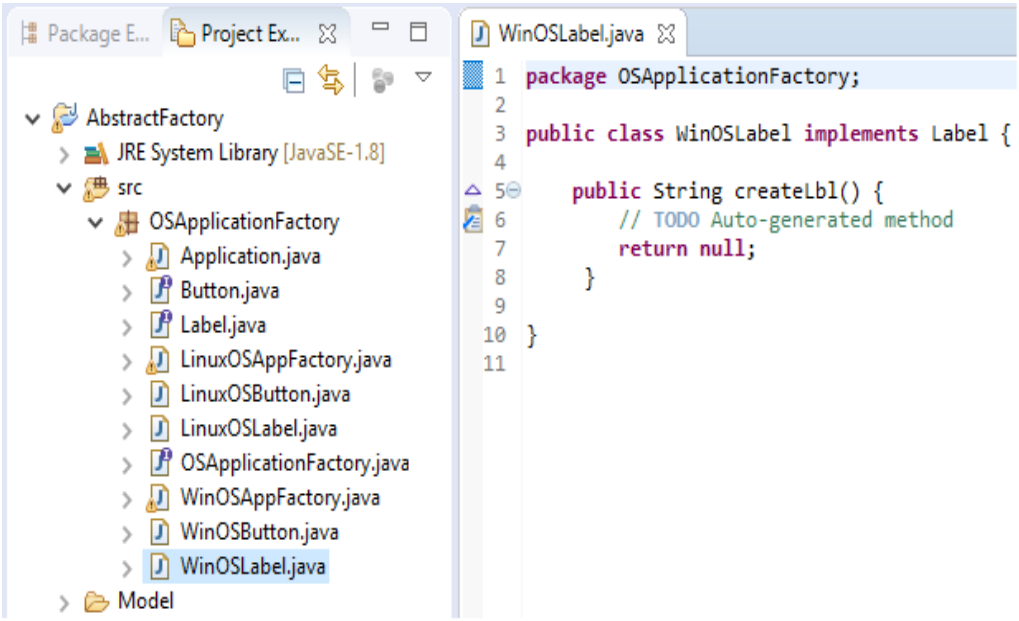

Рисунок 20

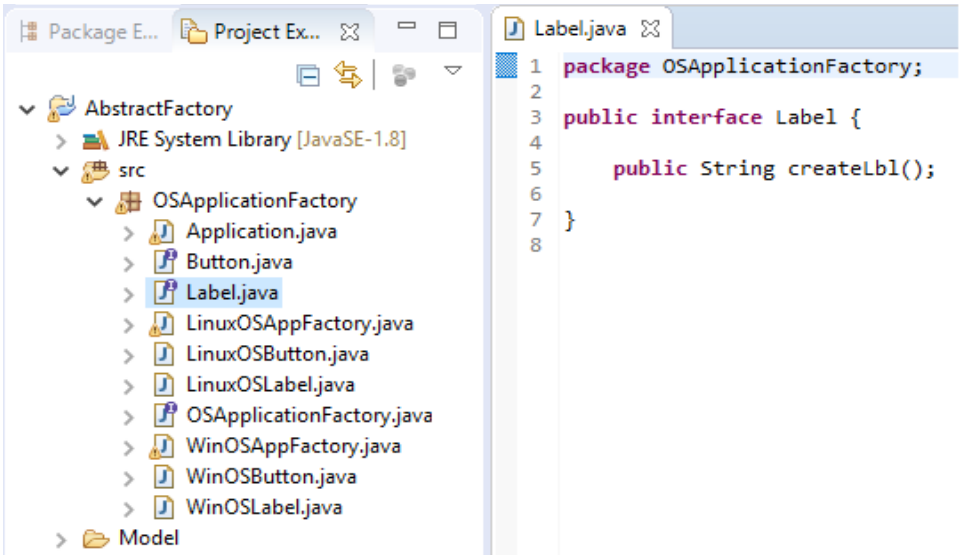

Рисунок 21

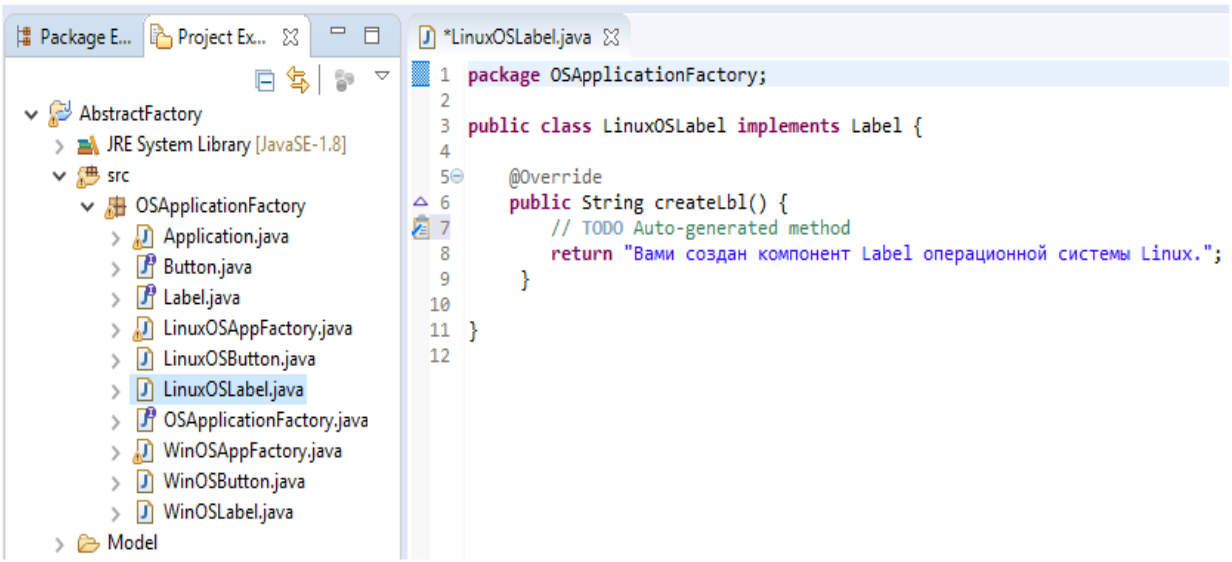

Рисунок 22

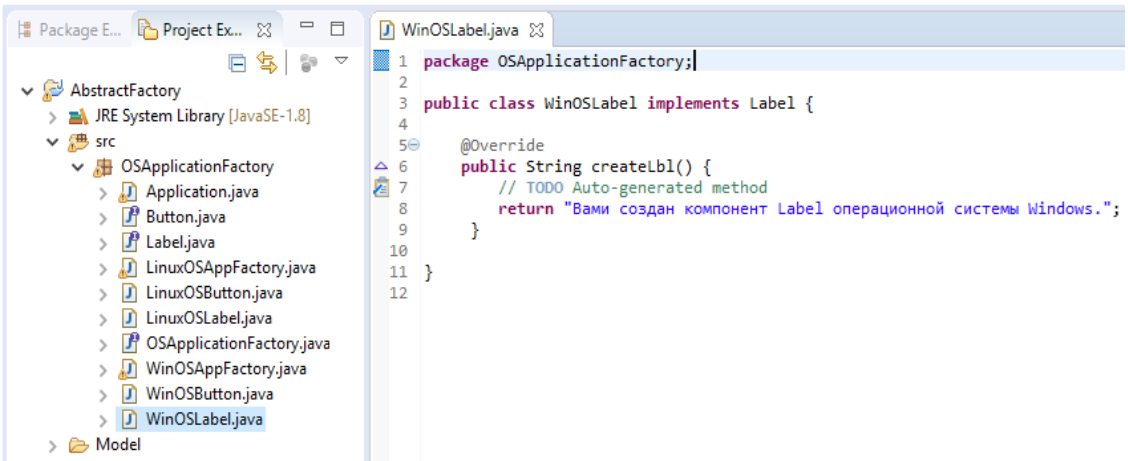

Рисунок 23

8. Необходима корректировка сгенерированного кода классов LinuxOSAppFactory и WinOSAppFactory (рисунки 24–25), являющихся наследниками интерфейса OSApplicationFactory (рисунок 26). Интерфейс OSApplicationFactory реализуется всеми конкретными фабриками и состоит из методов создания продуктов Button и Label. Скорректированный код представлен на рисунках 27–28.

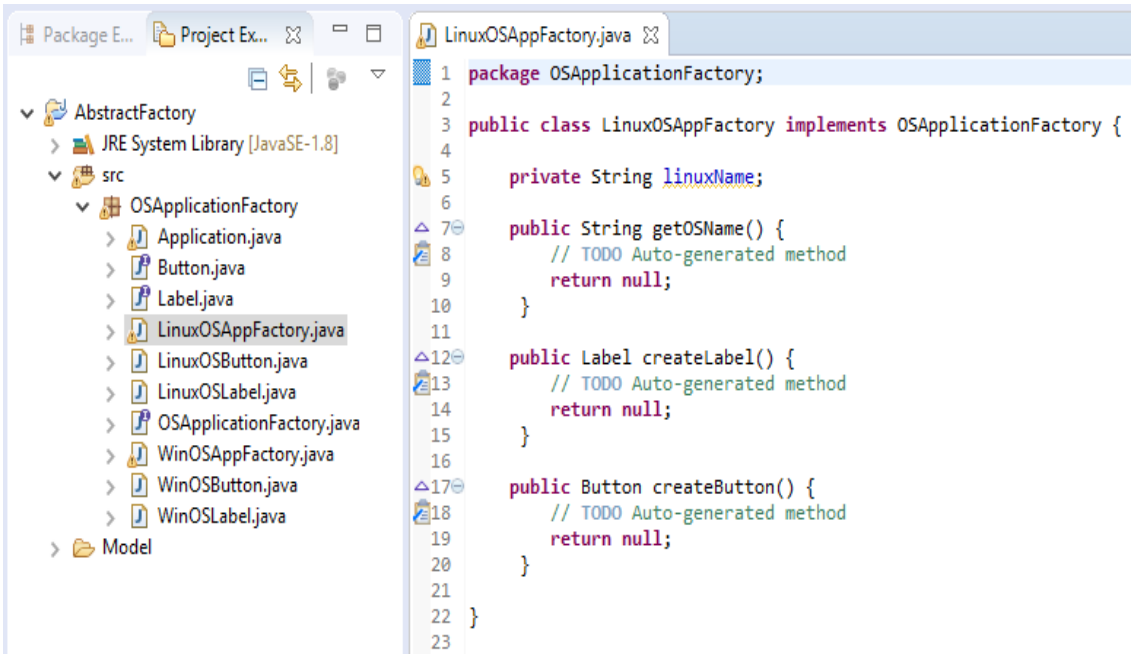

Рисунок 24

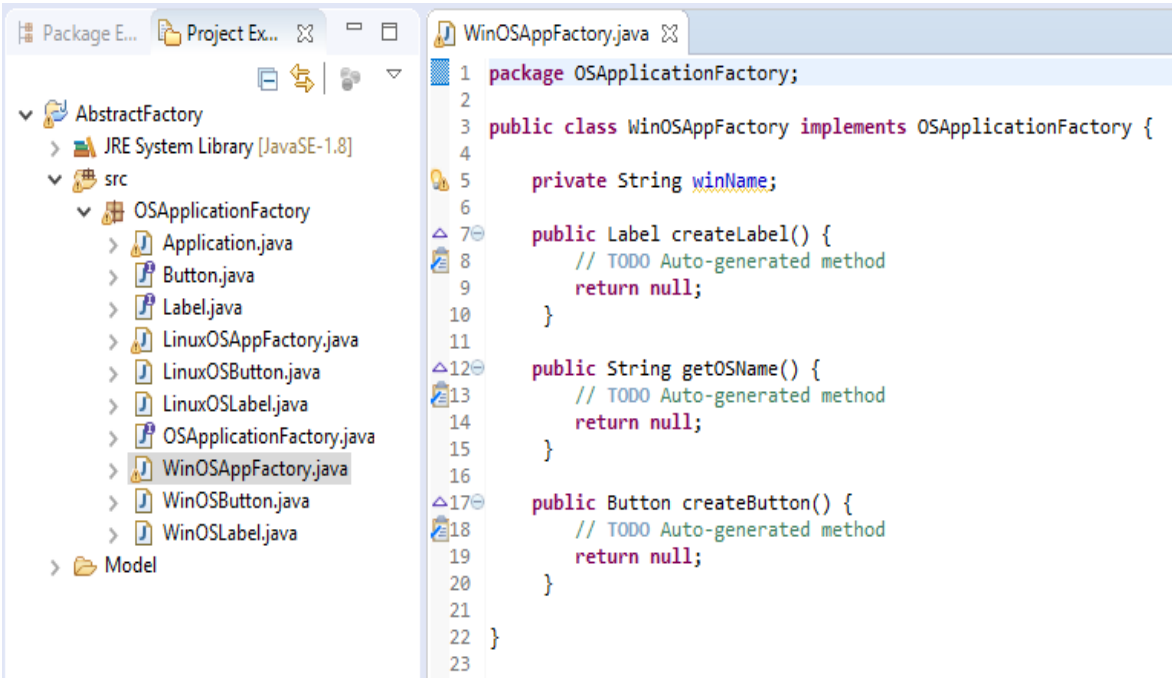

Рисунок 25

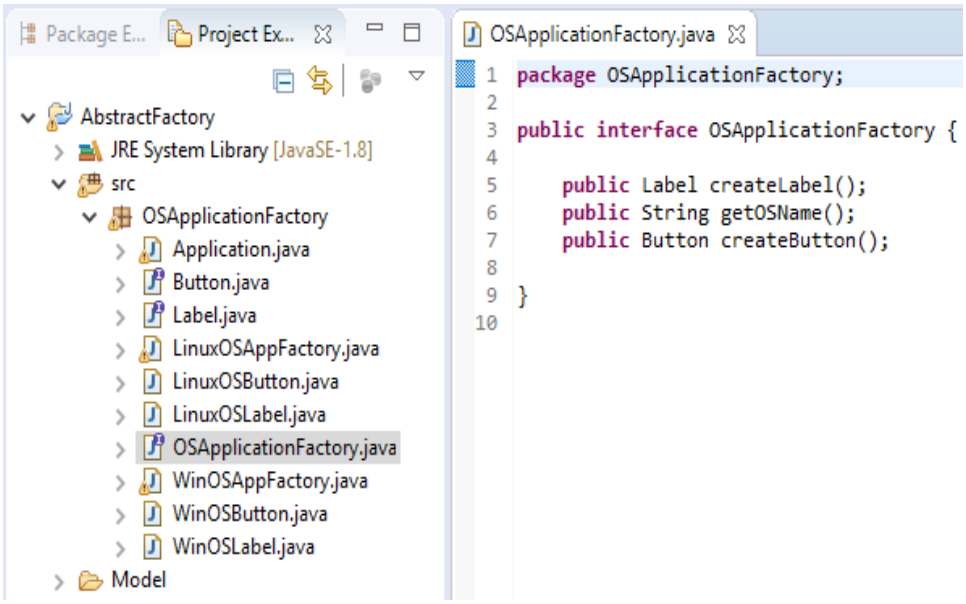

Рисунок 26

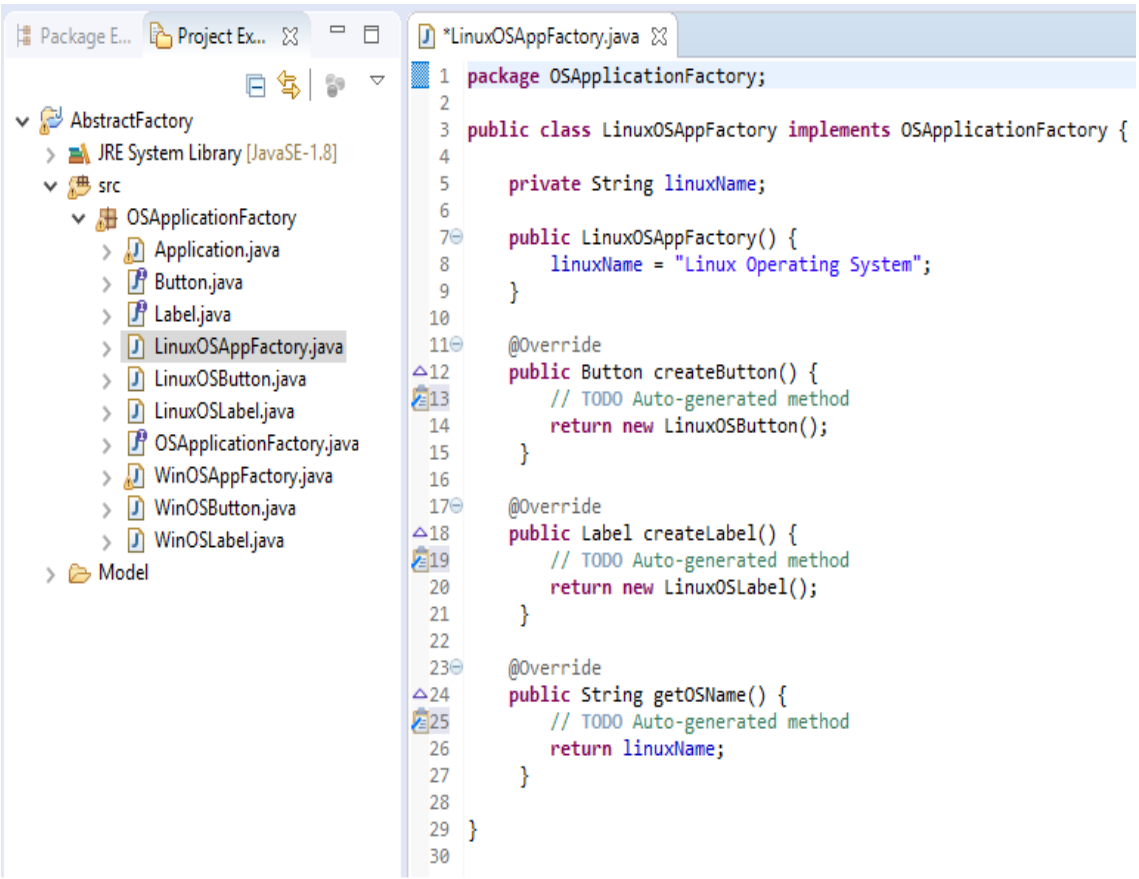

Рисунок 27

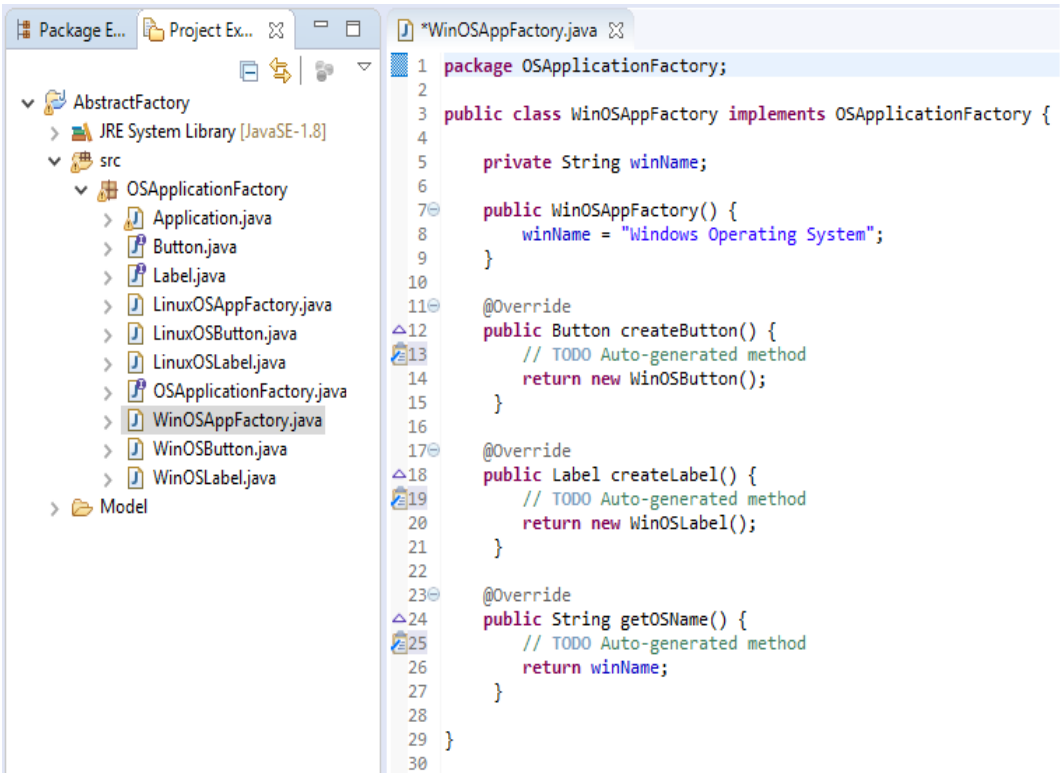

Рисунок 28

9. Необходима корректировка сгенерированного кода класса Application (рисунок 29). Данный класс написан для абстрактной фабрики OSApplicationFactory, а затем во время выполнения связывается с одной из конкретных фабрик LinuxOSAppFactory и WinOSAppFactory. Скорректированный код представлен на рисунке 30.

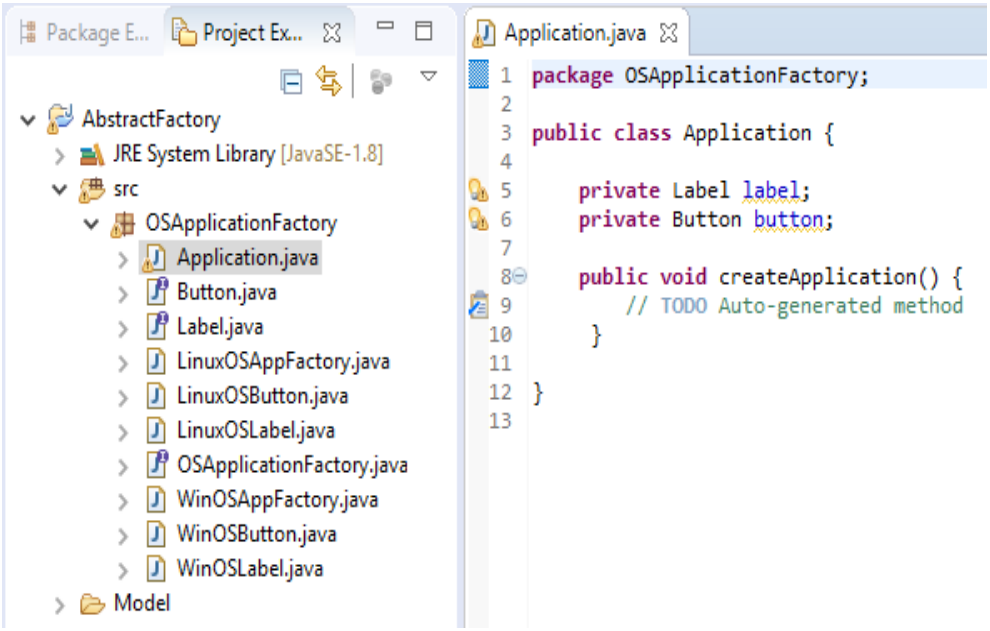

Рисунок 29

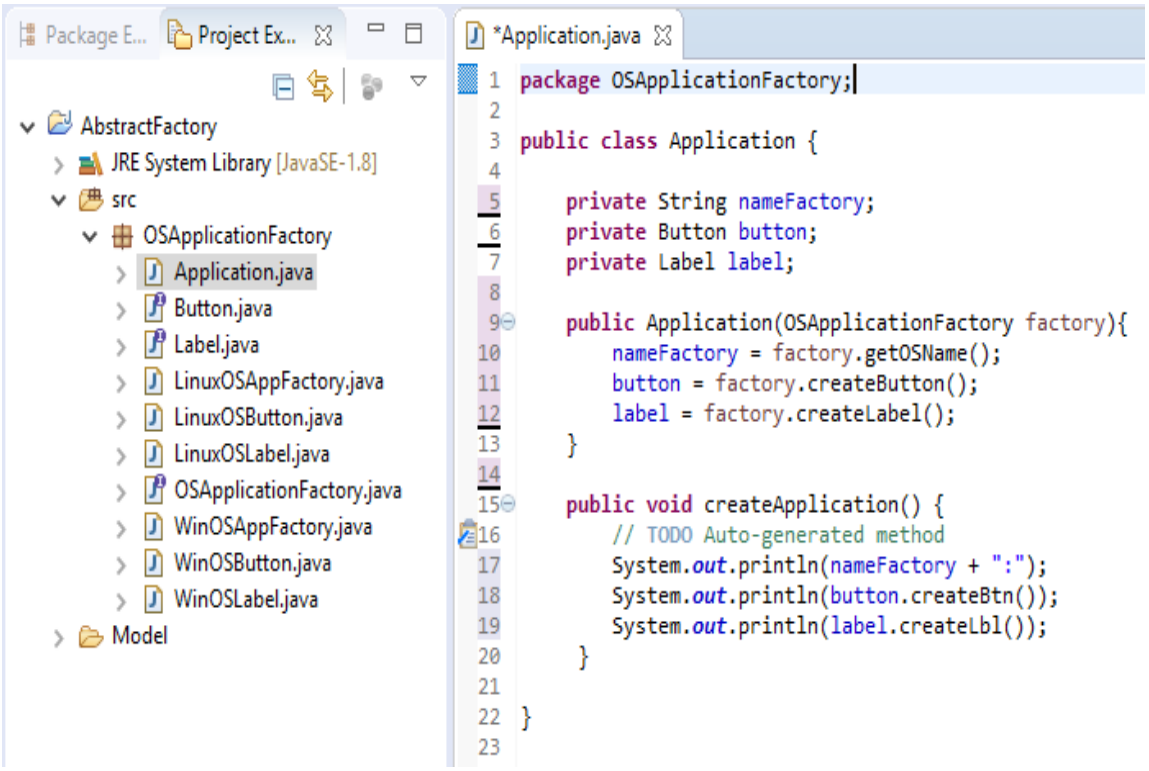

Рисунок 30

10. В пакете проекта OSApplicationFactory реализовать класс TestAbstractFactory для тестирования паттерна проектирования «Абстрактная фабрика» (рисунки 31–33).

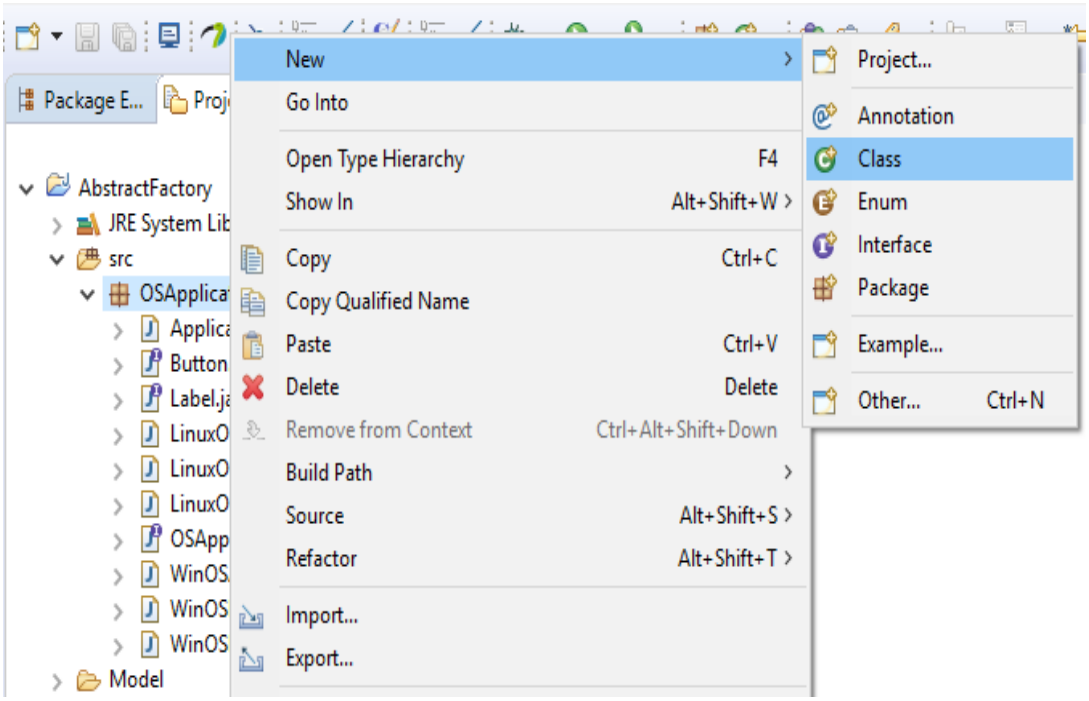

Рисунок 31

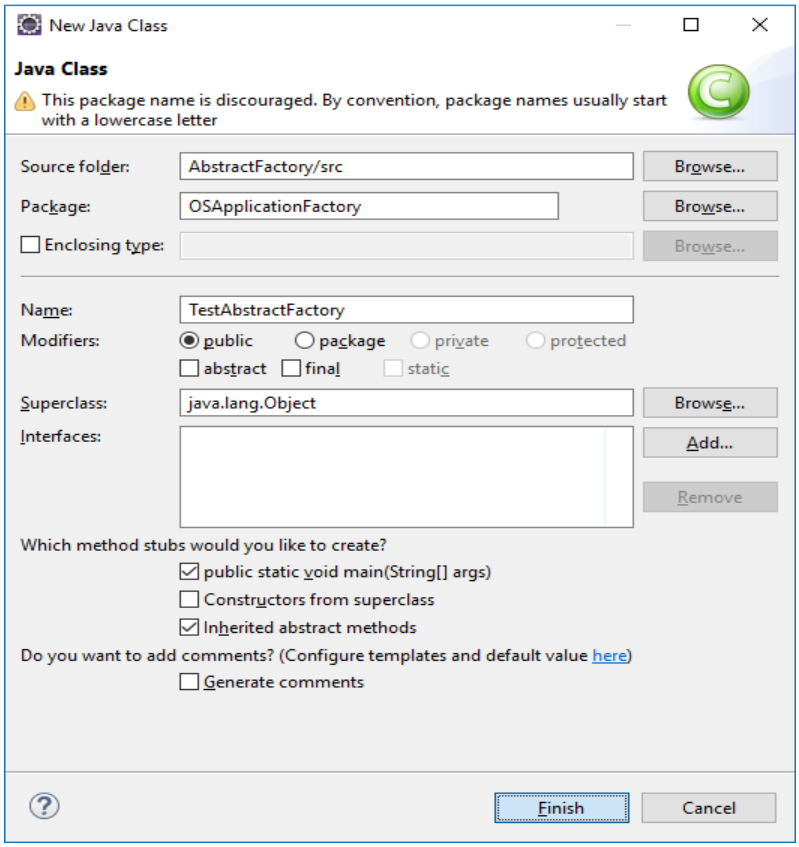

Рисунок 32

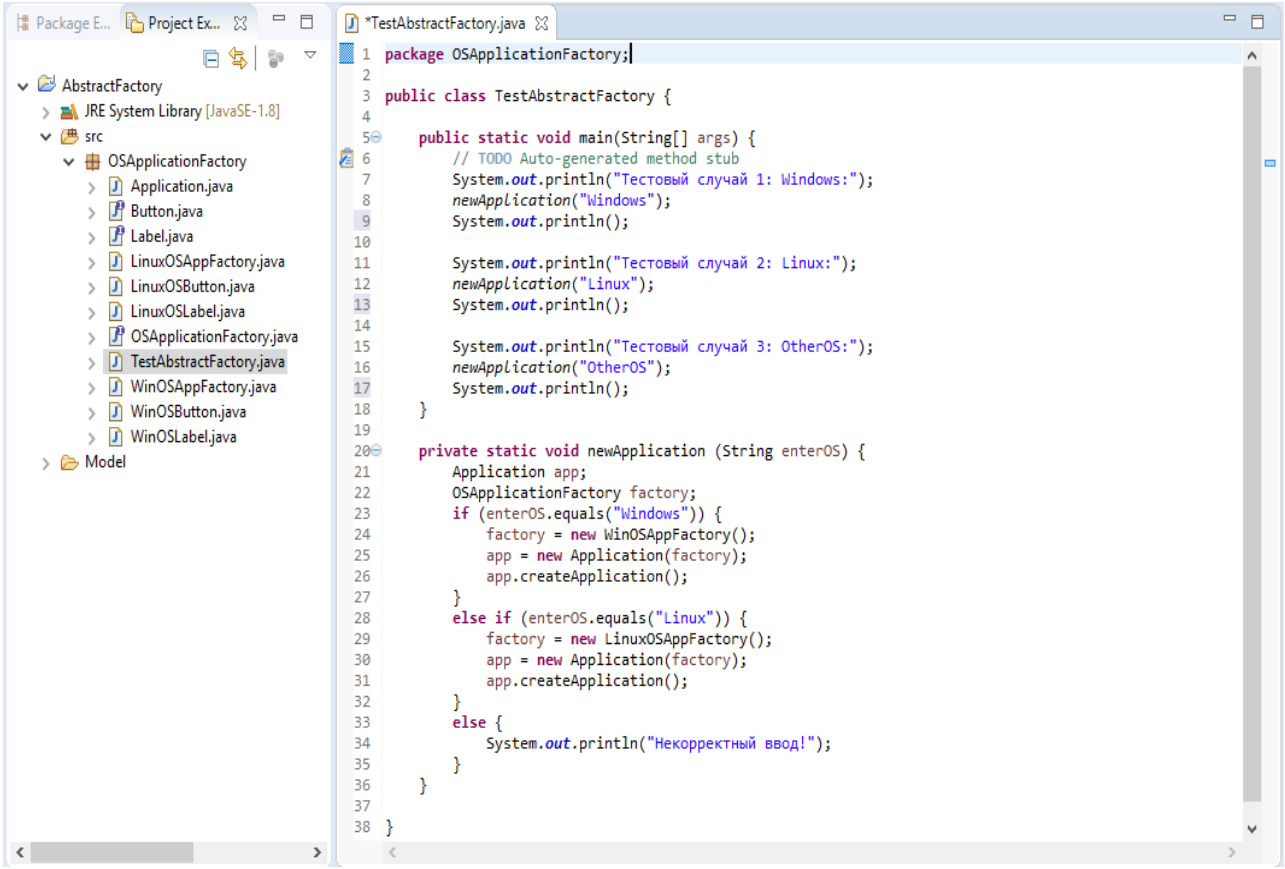

Рисунок 33

11. Для компиляции класса TestAbstractFactory нажать кнопку Run , чтобы запустить тестирование программы. В окне Console можно увидеть результаты тестирования (рисунок 34).

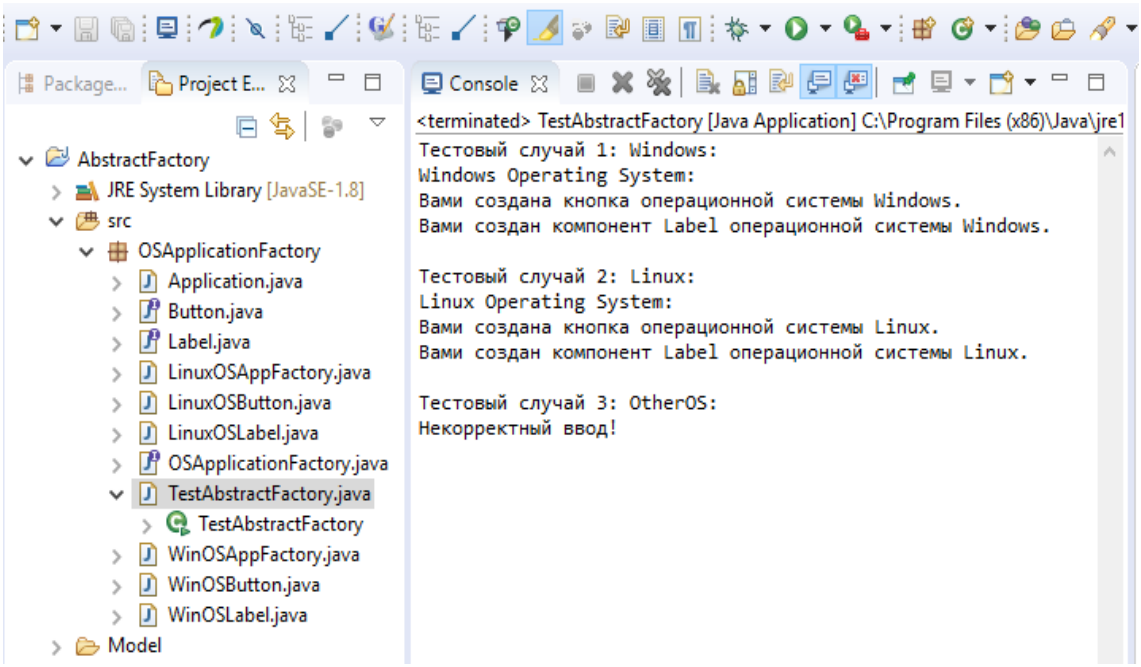

Рисунок 34

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 «Абстрактная фабрика» для индивидуального задания;
- результаты тестирования программы.

5 Вопросы к защите лабораторной работы

1. Что такое паттерн проектирования?
2. Для чего предназначен паттерн проектирования «Абстрактная фабрика»?
3. К какому типу паттернов проектирования относится «Абстрактная фабрика»?

4. Какие классы/интерфейсы являются участниками «Абстрактная фабрика»?

5. Назовите родственные для паттерна «Абстрактная фабрика» паттерны проектирования.

6. Какие методы использует интерфейс AbstractFactory в данном паттерне проектирования?

7. В чем отличие шаблона проектирования «Абстрактная фабрика» от шаблона «Фабричный Метод»?

8. Какие достоинства и недостатки имеет паттерн «Абстрактная фабрика»?

9. В каких случаях применяется данный шаблон проектирования?

6 Индивидуальные задания

1. Автомобилестроительная компания Ford производит легковой автомобиль Ford Focus и микроавтобус Ford Transit, а автомобилестроительная компания Mercedes-Benz производит легковой автомобиль Mercedes-Benz C-klasse и микроавтобус Mercedes-Benz Sprinter. Использовав паттерн проектирования «Абстрактная фабрика», реализовать программный продукт, демонстрирующий произведенные автомобили в каждой из перечисленных компаний.

2. На мебельной фабрике производят комплекты, каждый из которых состоит из кресла, дивана и стола, в трех разных стилях: Викторианском, Ампир и Модерн. Использовав паттерн проектирования «Абстрактная фабрика», реализовать программный продукт, демонстрирующий произведенный на фабрике комплект мебели в каждом из перечисленных стилях.

3. У каждого государства есть символика в виде флага и гимна и есть столица. Использовав паттерн проектирования «Абстрактная фабрика», реализовать программный продукт, демонстрирующий символику и столицу для каждого из следующих государств: Российская Федерация, Республика Беларусь.

4. В операционной системе Linux используется офисный пакет LibreOffice, содержащий текстовый процессор LibreOffice Writer и табличный редактор LibreOffice Calc, а в операционной системе macOS используется офисный пакет iWork, содержащий текстовый процессор iWork Pages и табличный редактор iWork Numbers. Использовав паттерн проектирования «Абстрактная фабрика», реализовать программный

продукт, демонстрирующий набор офисных приложений для каждой из перечисленных операционных систем.

5. В стратегической игре для каждой империи используют три типа персонажей: император, воин и крестьянин. Использовав паттерн проектирования «Абстрактная фабрика», реализовать программный продукт, демонстрирующий персонажей для каждой из следующих империй: Римская империя, Британская империя.

6. Для приготовления пиццы «Маргарита» используют в качестве начинки томаты и сыр моцарелла, а для приготовления гавайской пиццы – ананас и сыр пармезан. Использовав паттерн проектирования «Абстрактная фабрика», реализовать программный продукт, демонстрирующий основные ингредиенты начинки для каждой из перечисленной пиццы.

7. Интерфейс пользователя одного проекта состоит из следующих компонентов: текст интерфейса, изображения и справка пользователя. Использовав паттерн проектирования «Абстрактная фабрика», реализовать программный продукт, демонстрирующий интерфейс пользователя для каждого из следующих языков пользователя: русский, французский.

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.