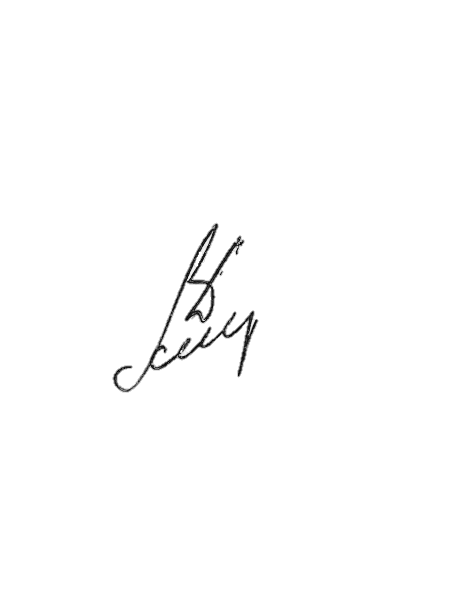
**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Национальный исследовательский ядерный университет «МИФИ»**

**ОТЧЕТ**

по лабораторной работе №3

по дисциплине: «Теория и технология программирования»

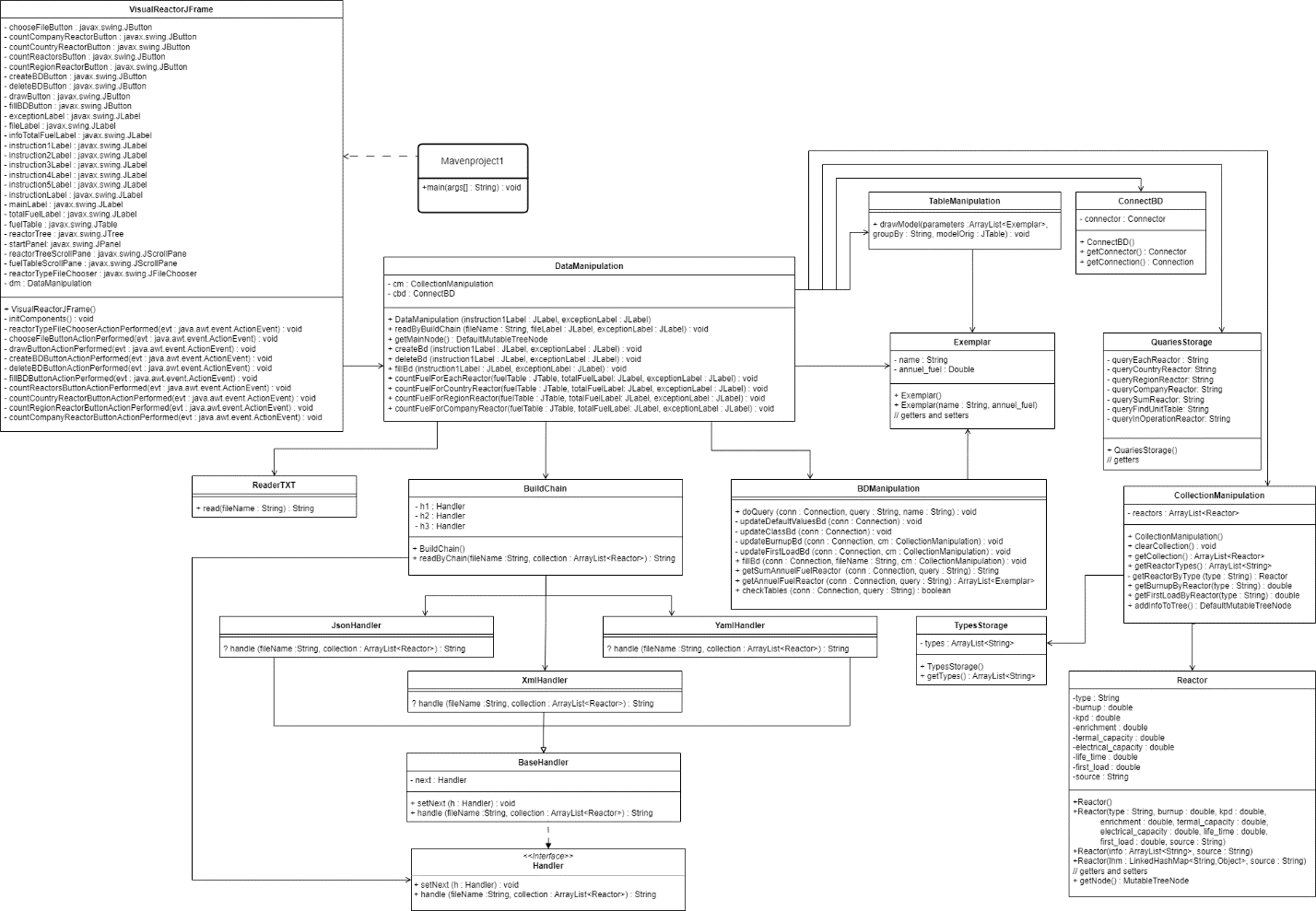


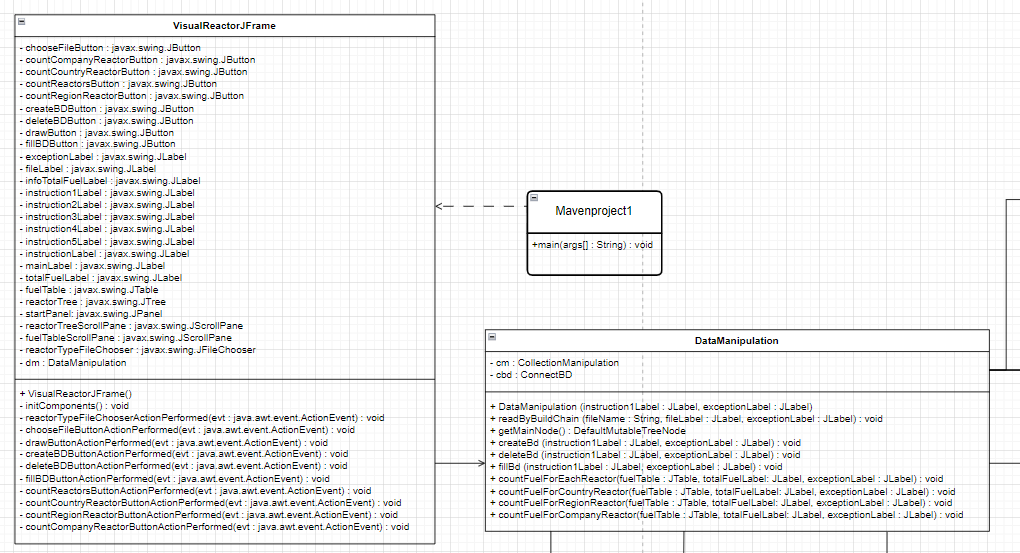
Выполнил: *студент группы Б20-901*  Кондрашкина К.Д.

Проверил: Смирнов Д.С.

Москва 2023 г

1. Описание проекта:

Рис.1. Схема всего проекта

Рис.2. Фрагмент схемы, состоящий из классов Mavenproject1, VisualReactorJFrame, Datamanipulation

Класс Mavenproject1 содержит один метод main(args[] : String), который автоматически вызывается при запуске программы, в данном методе создается объект VisualReactorJFrame.

В классе VisualReactorJFrameсодержатся визуальные объекты (jButton, jTree, jTable, jLabel, jFileChooser, jScrollPane, jPanel), объект класса DataManipulation. Этот класс отвечает за создание визуальной составляющей проекта и вызов методов класса DataManipulation. Метод reactorTypeFileChooserActionPerformed(java.awt.event.ActionEvent evt) вызывается при выборе файла в jFileChooser.

Метод chooseFileButtonActionPerformed(java.awt.event.ActionEvent evt) вызывается при нажатии на определенную кнопку и дает возможность пользователю выбрать файл с помощью jFileChooser (именно в этом методе он становится видимым).

Метод drawButtonActionPerformed(java.awt.event.ActionEvent evt) вызывается при нажатии на определенную кнопку и дает возможность пользователю увидеть считанную информацию о реакторах в виде дерева (jTree).

Методы createBDButtonActionPerformed, deleteBDButtonActionPerformed , fillBDButtonActionPerformed , countReactorsButtonActionPerformed , countCountryReactorButtonActionPerformed , countRegionReactorButtonActionPerformed , countCompanyReactorButtonActionPerformed вызывают методы класса DataManipulation (createBD, deleteBD, fillBd, countFuelForEachReactor, countFuelForCountryReactor, countFuelForRegionReactor, countFuelForCompanyReactor соответственно).

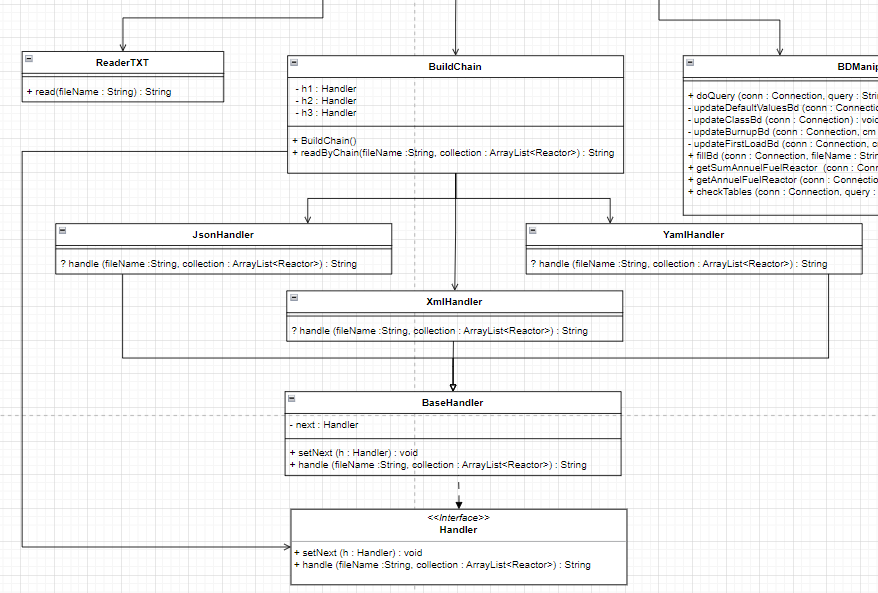
В классе DataManipulation осуществляется реализация перечисленных выше методов, реализуется взаимосвязь различных классов проекта, хранятся объекты классов CollectionManipulation и ConnectBD.

Рис.3. Фрагмент схемы, состоящий из классов ReaderTXT, BuildChain, JsonHandler, XmlHandler, YamlHandler, BaseHandler и интерфейса Handler

В классе ReaderTXT определен метод, осуществляющий считывание из файла формата txt.

В интерфейсе Handler объявлены методы setNext, handle. Класс BaseHandler реализует объявленные в интерфейсе методы (метод handle должен пытаться прочитать файл, при неудачной попытке прочтения передать следующему обработчику). Конкретные JsonHandler, XmlHandler, YamlHandler переопределяют метод handle, возвращают строку, которая выводится в fileLabel. Класс BuildChain формирует определенную цепочку из Handler, в соответствии с которой будет проходить попытка прочитать файл.

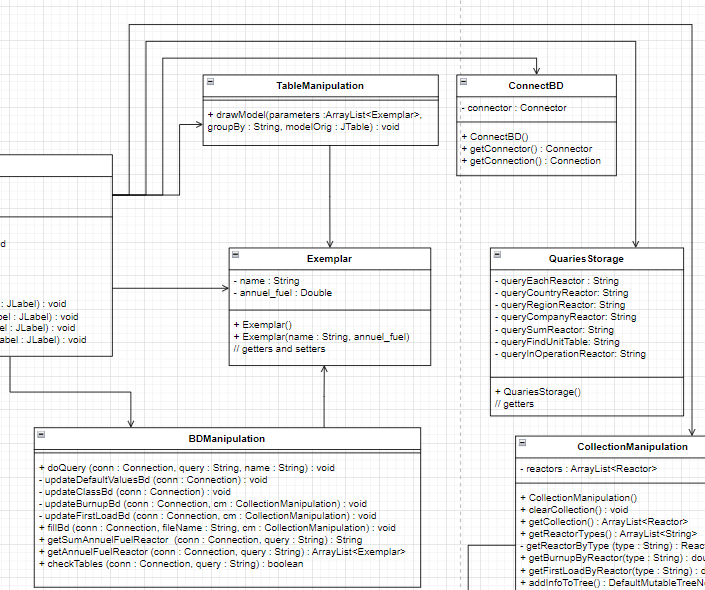
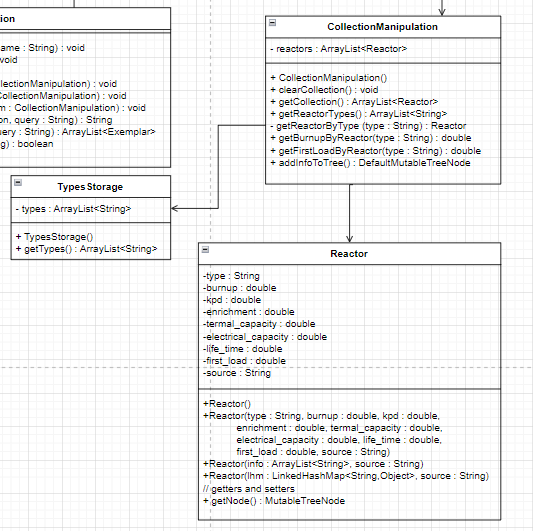


Рис.4. Фрагмент схемы, состоящий из классов BDManipulation, Exemplar, TableManipulation, ConnectBD, QuariesStorage

В классе ConnectBD задаюся конкретные параметры подсоединения к базе данных, расположенной на сервере, а также есть метод, которые позволяют получить объект типа Connection. В классе QuariesStorage хранятся запросы к базе данных, которые необходимы, чтобы вычислить потребляемое реактором топливо и объем потребляемого топлива в различных группировках. Класс Exemplar необходим для того, чтобы одинаковым образом взаимодействовать с результатами запросов (например реактор и потребляемое им топливо и регион и потребляемое им топливо). Класс TableManipulation определяет метод, отрисовывающий таблицу по имеющимся данным. Класс BDManipulation отвечает за непосредственное подсоединение к базе данных и получение данных в результате выполнения запросов.

Рис.5. Фрагмент схемы, состоящий из классов CollectionManipulation, Reactor, TypesStorage

Класс CollectionManipulation отвечает за хранение данных о реакторах (в виде коллекции объектов класса Reactor), считанных из файла ReactorType, а также взаимодействие с данной коллекцией (например получение данных о первой загрузке и выгорания для определенного типа реакторов). В классе Reactor находятся параметры, которые должны быть у реакторов (type – тип реактора, burnup – выгорание топлива, kpd – кпд, enrichment – обогащение, termal\_capacity – тепловая мощность, electrical\_capacity - электрическая мощность, life\_time – время эксплуатации, first\_load – первая загрузка уранового продукта, source – тип файла, из которого считаны данные). В этом классе находятся несколько конструкторов, т.к. при считывании данных из разных файлов получаем данные в различном виде. Также в данном классе есть геттеры/сеттеры для каждого из параметров и метод getNode(), который формирует структуру узла в дереве (jTree) для реактора, а затем возвращает узел для добавления к общему дереву. В классе TypesStorage хранятся названия возможных типов реакторов. Они необходимы, чтобы обновить данные о типах реакторов, считанные из файла excel, на необходимый класс реакторов.

2. Визуальная составляющая.

В результате мы получаем:

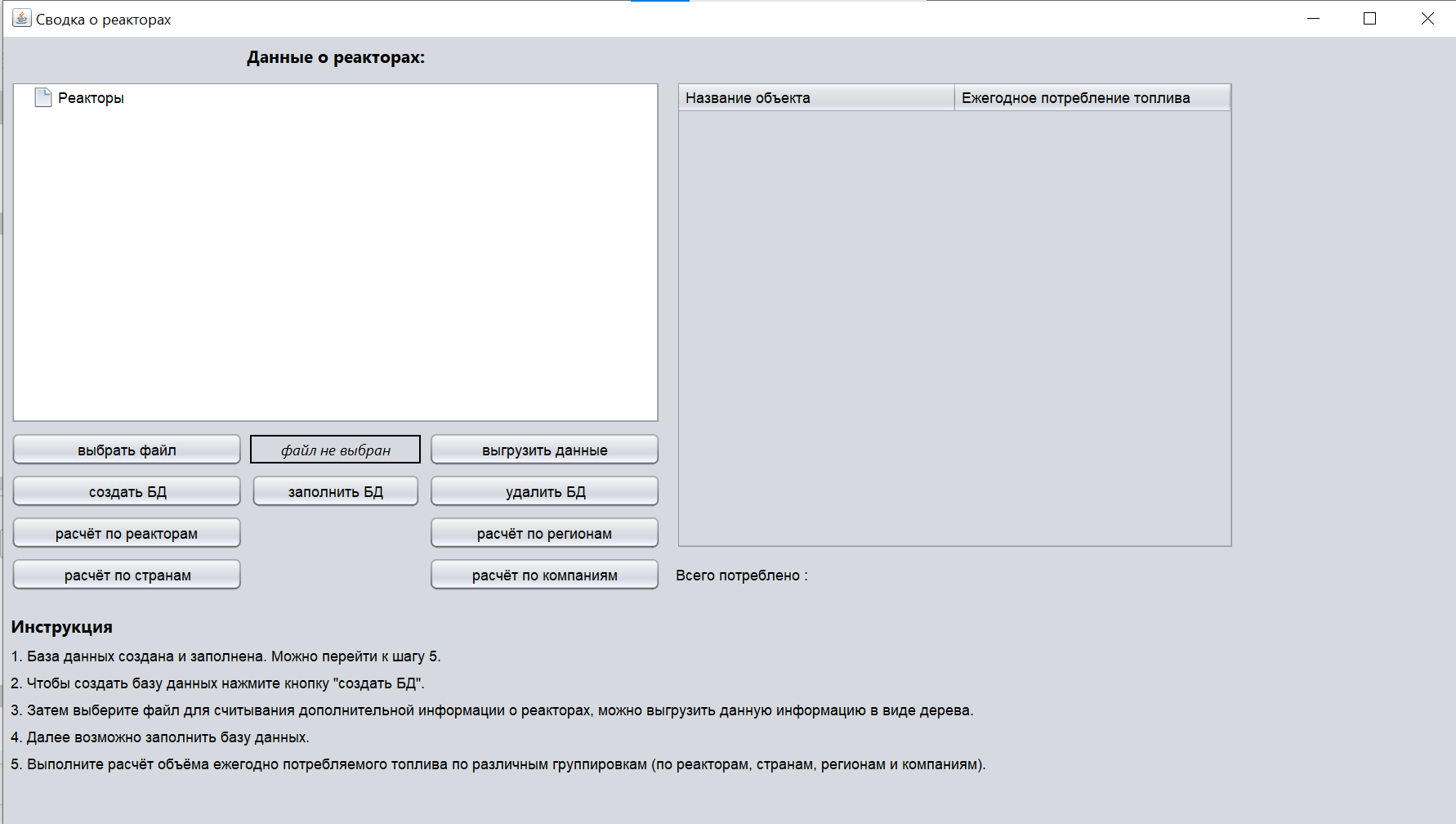


Рис. 6. Начальное окно

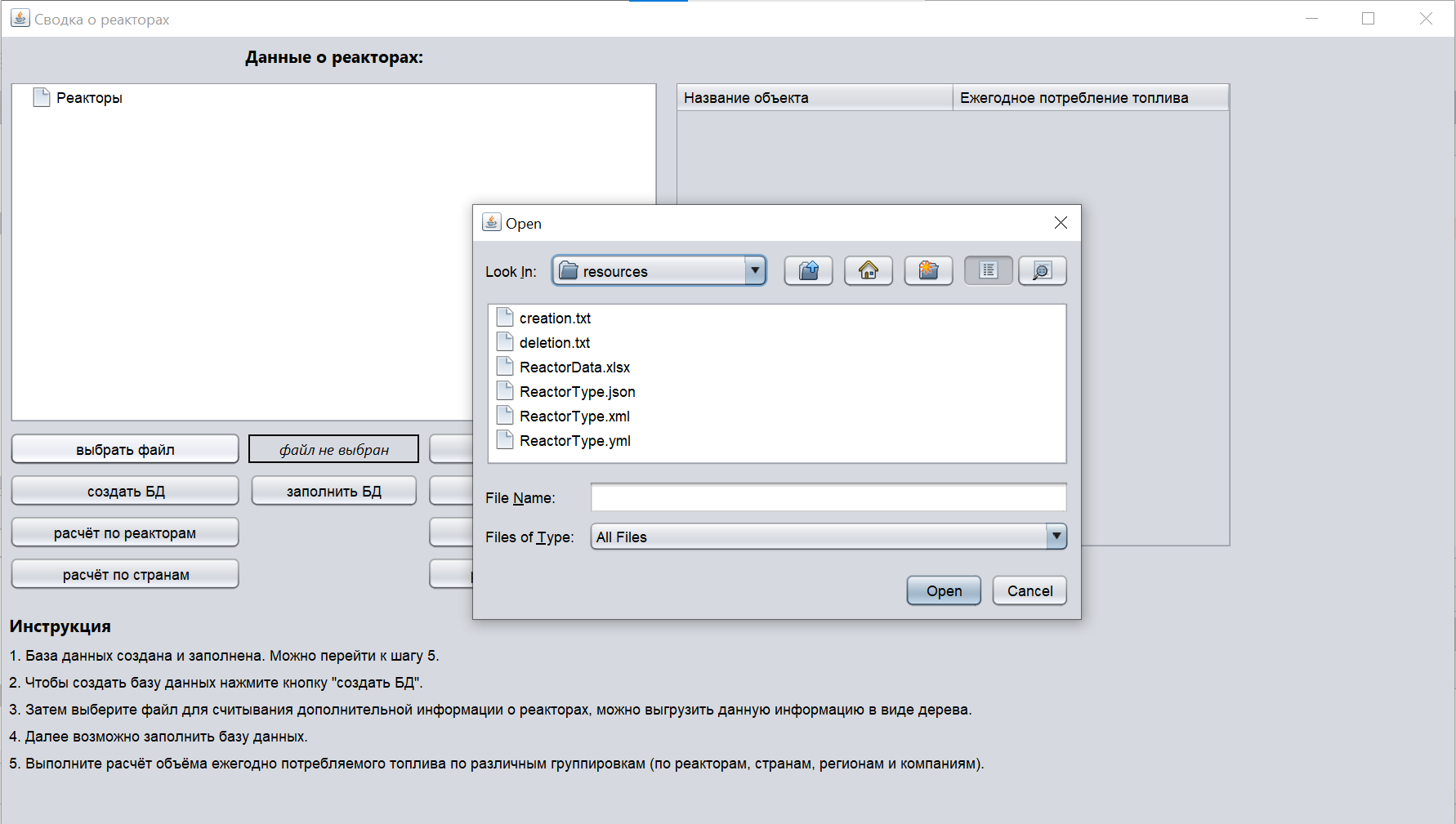


Рис. 7. jFileChooser, появившийся после нажатия кнопки «выбрать файл»

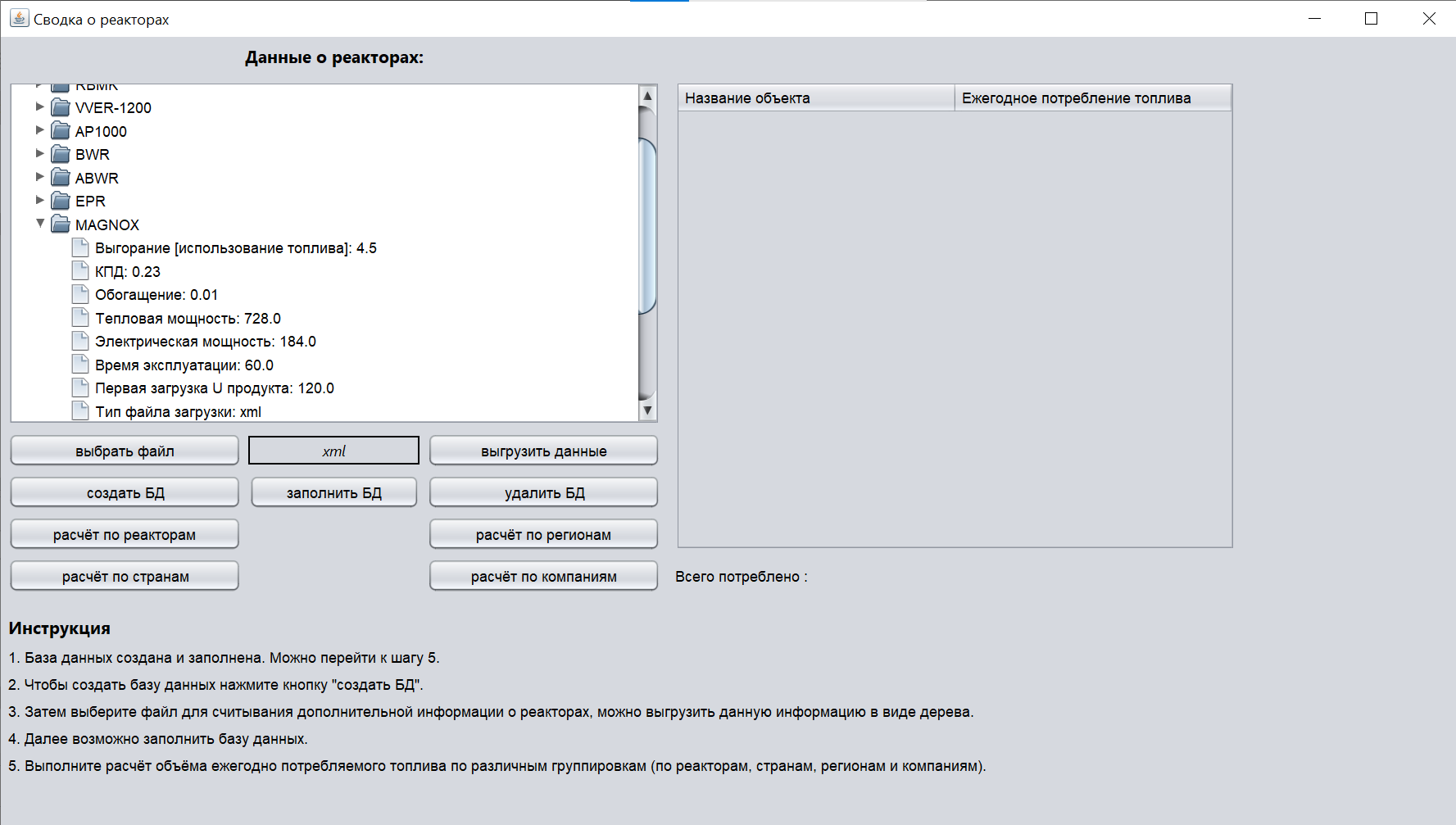


Рис. 8. Сводка о реакторах при выборе файла с расширением «xml»

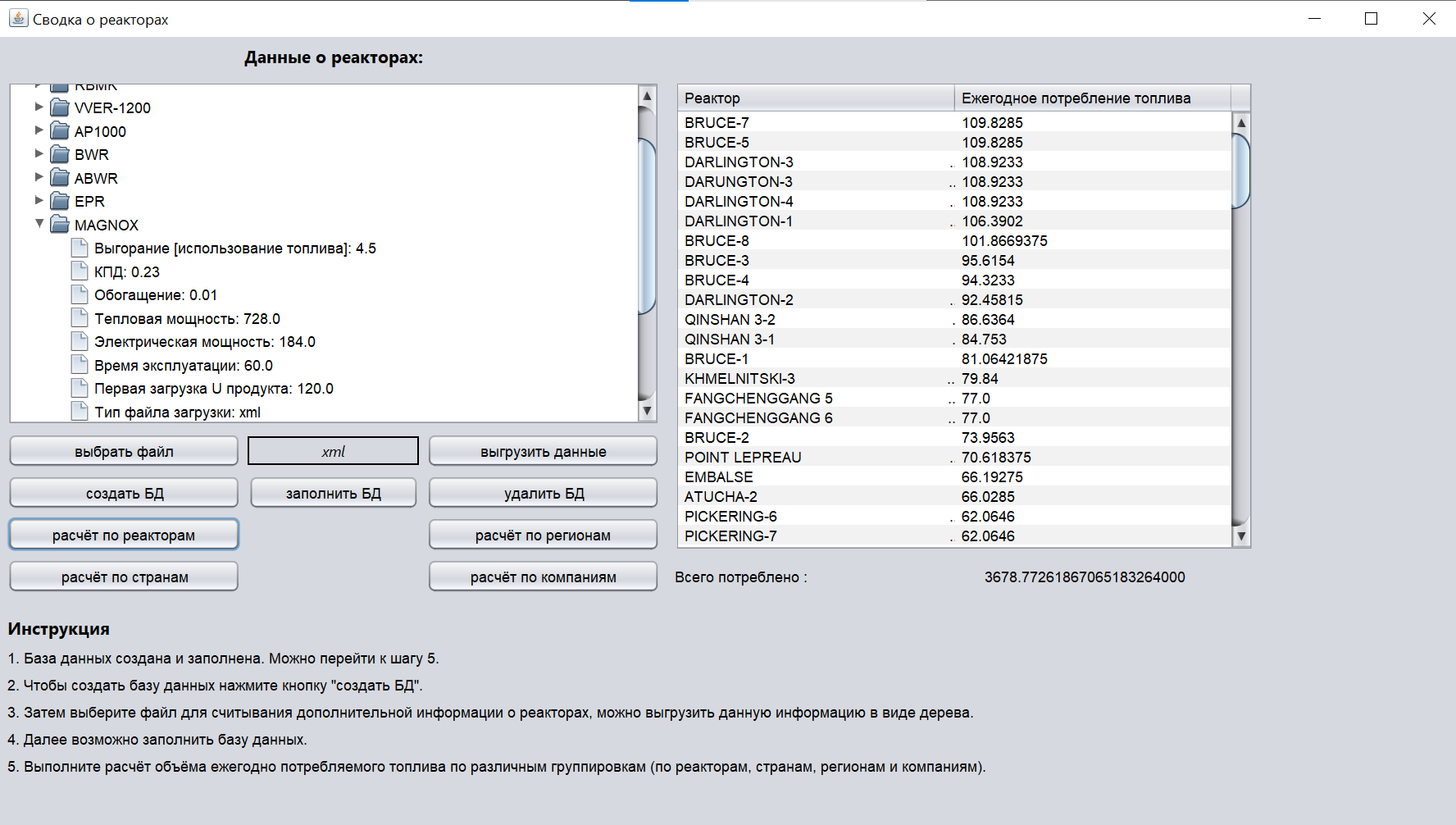


Рис. 9. Расчёт объёма потребляемого за год топлива каждым реактором

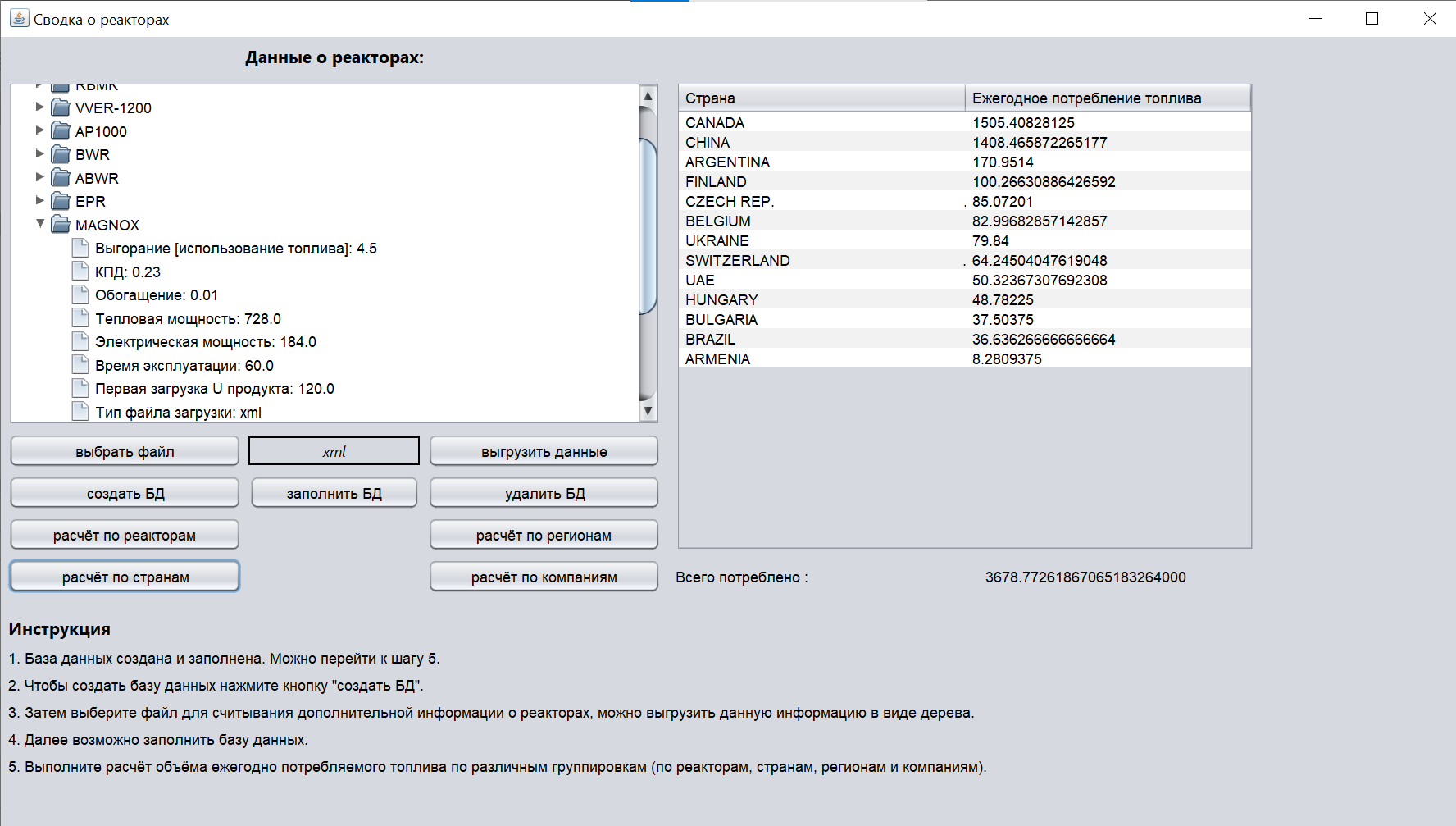


Рис. 10. Расчёт объёма потребляемого за год топлива в зависимости от страны

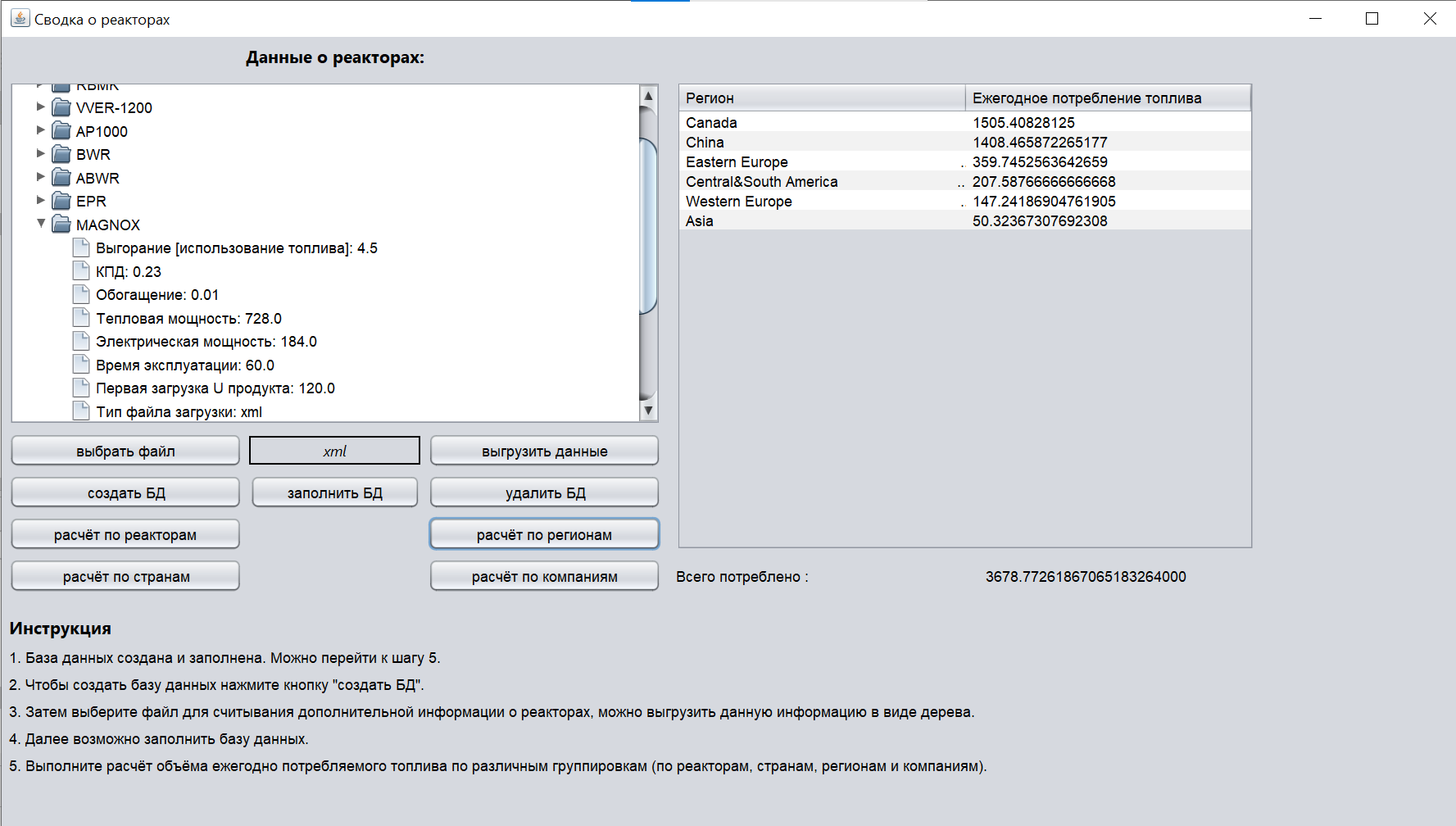


Рис. 11. Расчёт объёма потребляемого за год топлива в зависимости от региона

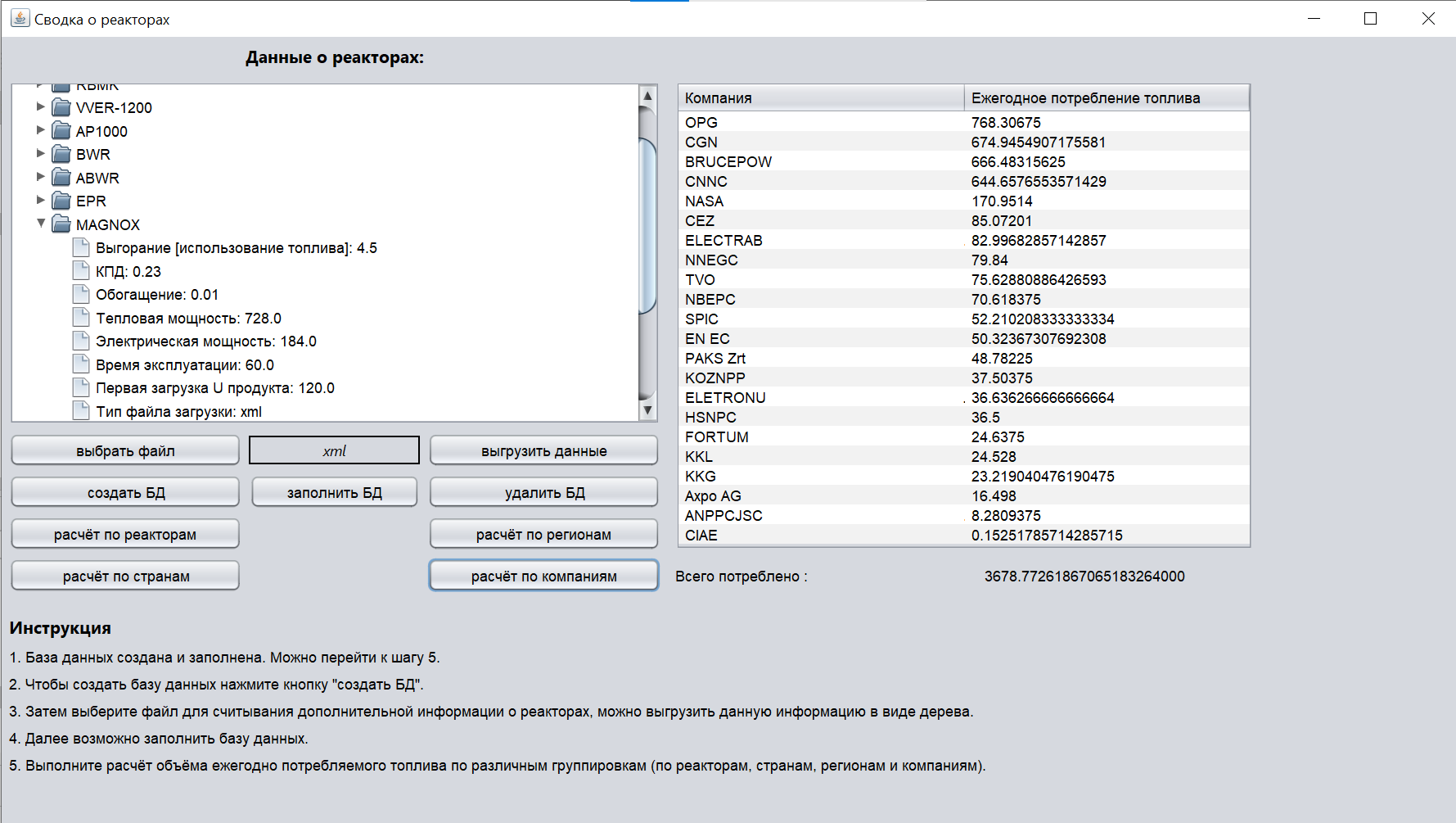


Рис. 12. Расчёт объёма потребляемого за год топлива в зависимости от компании

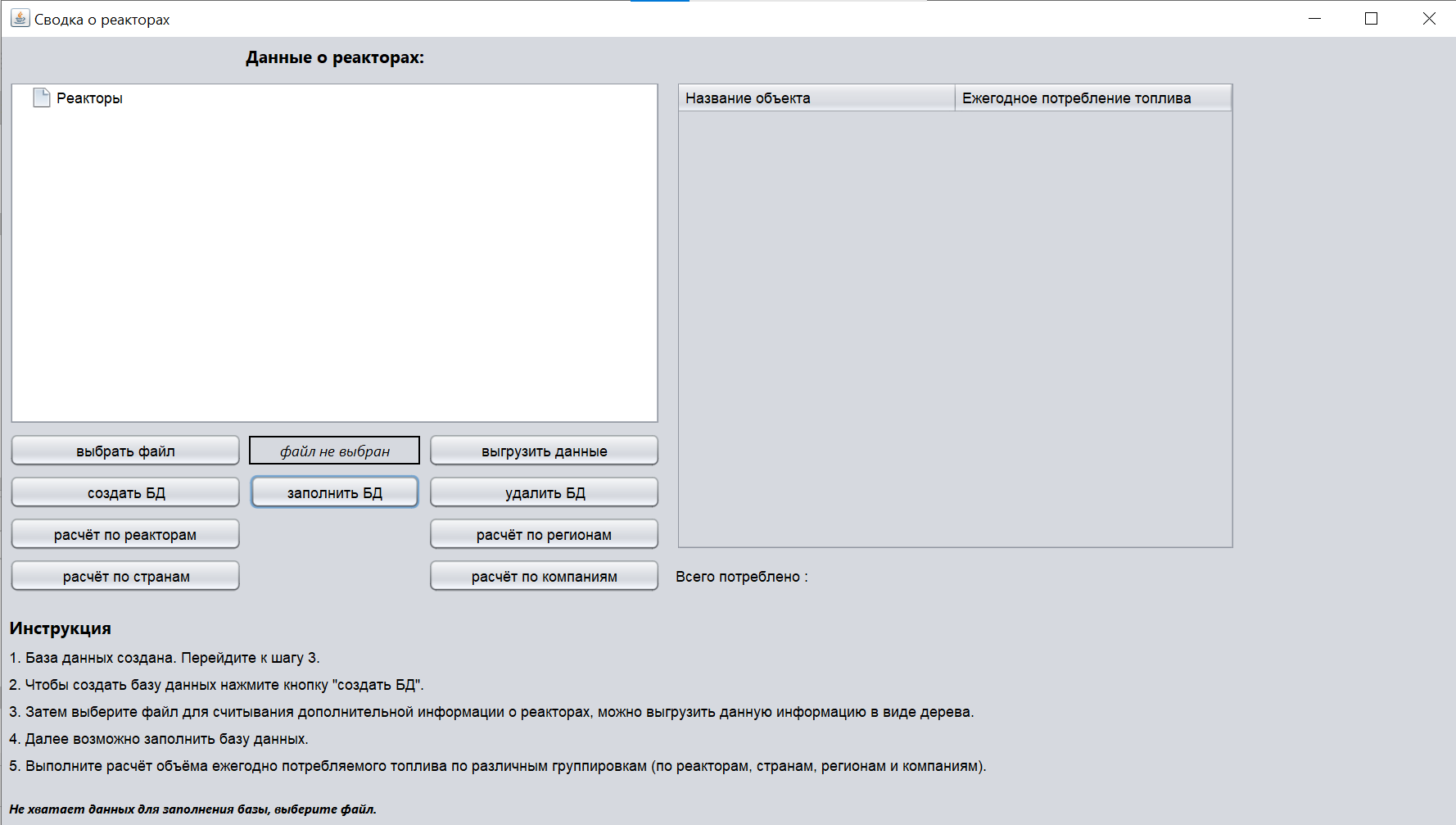


Рис. 13. Демонстрация сообщения об ошибке при попытке заполнить базу данных без считывания файла ReactorType

3. Файлы, с запросами на создание и удаление базы данных

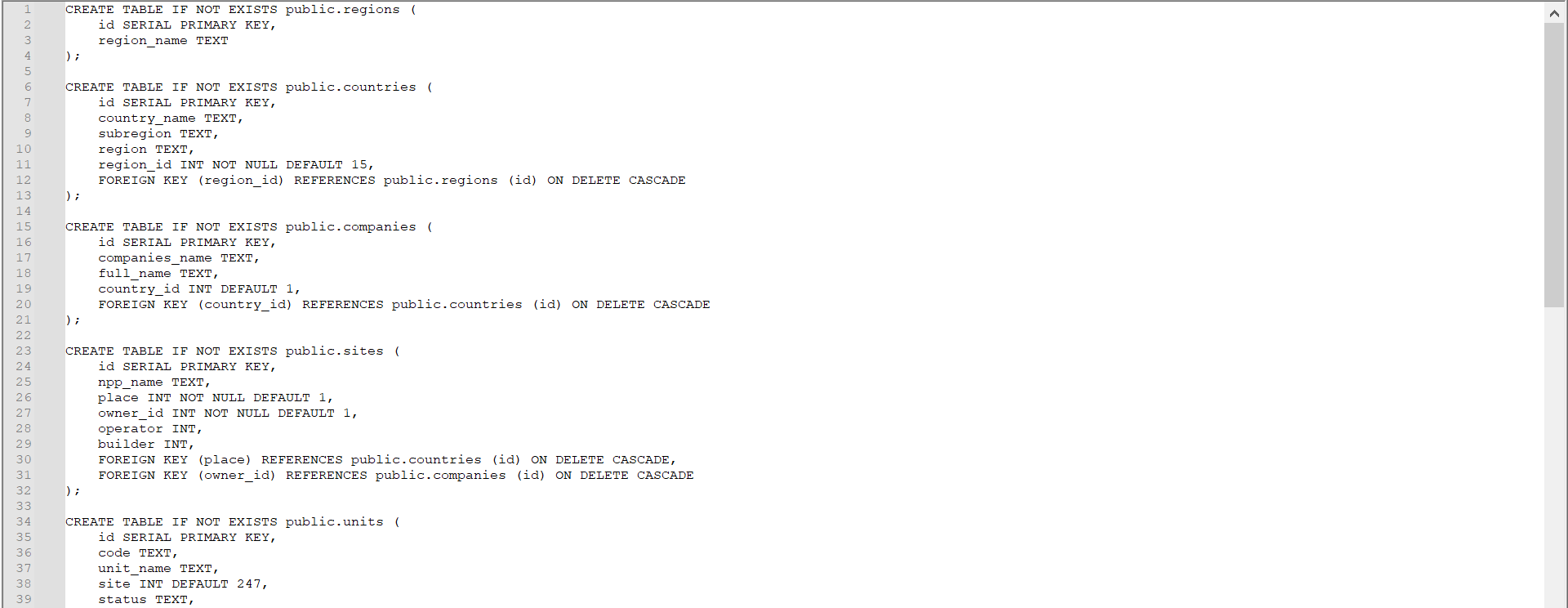


Рис. 14 Фрагмент текста, для создания базы данных

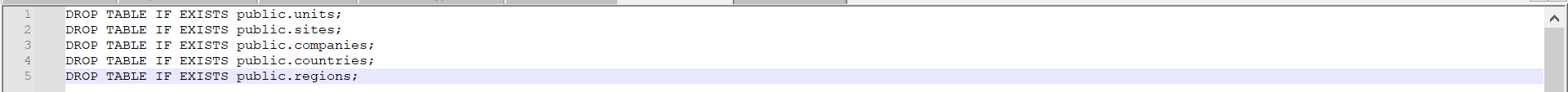


Рис. 15 Текст, для удаления базы данных