Лабораторная работа №6

Лабораторная работа представлена в трех вариантах. Два – оконные приложения, swing и JavaFX, для них есть методические указания. Третий вариант – серверное приложение, пользователь при этом будет обращаться к приложению через браузер. Методичка для этого не написана (самостоятельная работа с доп баллами), ключевые слова для поиска в интернете: JavaEE, Wildfly, java beans (EJB), MVC, сервлет, jsp. Можете этим не ограничиваться :) Авторизация пользователя (через куки) не требуется, но приветствуется.

Задание на лабораторную работу – SWING.

Разработать графическое приложение, позволяющее работать с табулированными функциями: создавать их, редактировать, сохранять в файлы и считывать из файлов, а также создавать табулированные аналоги функций, загружаемых в программу в виде байт-кода классов.

Перед началом работы над собственной программой ознакомьтесь с предлагаемой реализацией (jar-файл программы, а также пример загружаемого класса предоставляются вместе с заданием).

При разработке окон приложения настоятельно рекомендуется использование визуальных средств и мастеров среды разработки, предназначенных для создания Swing-приложений.

Использование в качестве имён переменных неинформативных идентификаторов вида `jPanel5` (в таком виде их любят создавать среды разработки) запрещено.

Все новые классы этого задания следует создавать в отдельном пакете, название которого выберите самостоятельно.

За написание веб-приложения можно получить до 5 дополнительных баллов!

Задание 1 *(1 балл)*

Реализуйте класс вспомогательного окна, в котором вводятся параметры табулирования функции. Пример такого окна показан на рисунке 1. Для создания класса окна удобнее всего воспользоваться мастером среды разработки.

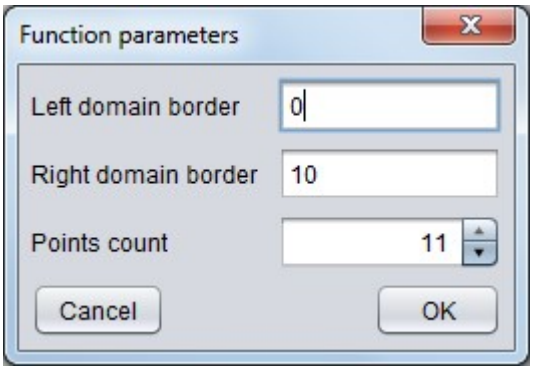

Рисунок 1 – Примерный вид окна параметров табулирования функции

Класс данного диалогового окна должен наследовать от класса `JDialog`.

Окно должно быть модальным, т.е. оно должно блокировать работу пользователя с родительским приложением до тех пор, пока пользователь это окно не закроет. Для этого свойство окна `modal` должно быть изменено на `true`.

Для простоты данное окно не должно позволять изменять свой размер. Для этого свойство окна `resizable` должно быть изменено на `false`.

Поскольку далее в работе приложения потребуется один и только один экземпляр этого окна, измените значение свойства `defaultCloseOperation` на `HIDE` (т.е. по нажатию на кнопку закрытия окна оно должно просто скрываться).

Расположите на форме следующие компоненты:

- два поля редактирования для левой и правой границы области определения табулированной функции (компоненты типа `JTextField`);
- изменяемое числовое поле для редактирования количества точек табулированной функции (компонент типа `JSpinner`);
- текстовые пояснения для редактируемых полей (компоненты типа `JLabel`);
- кнопки отмены действия и подтверждения действия (компоненты типа `JButton`).

Для текстовых полей редактирования задайте значения по умолчанию.

Поведение изменяемого числового поля определяется значением его свойства `model`. Это значение является объектом, описывающим параметры работы поля. Установите эти параметры таким образом, чтобы допустимыми являлись только целые числа, минимальное возможное значение было равно двум, а шаг изменения был равен единице.

Для удобства дальнейшей работы определите в классе две целочисленных «константы» `OK` и `CANCEL` с различными значениями. Также объявите целочисленное поле, в котором будет храниться статус окна после его закрытия, т.е. то, какую из кнопок нажал пользователь.

Напишите обработчик события нажатия на кнопку подтверждения операции, выполняющий следующие действия:

- проверка корректности введённых числовых значений,
- сокрытие окна,
- присвоение полю статуса окна значения `OK`.

Введённое в текстовое поле значение хранится в его свойстве `text` типа `String`. Для преобразованию к типу `double` будет удобно воспользоваться методом `Double.parseDouble()`, выбрасывающим исключение `NumberFormatException` в том случае, если строка содержит не число.

Если пользователь ввёл некорректные данные, нужно вывести ему сообщение об ошибке. Это разумно сделать во всплывающем диалоговом окне, которое можно вывести с путём вызова метода `JOptionPane.showMessageDialog()`. Изучите параметры этого метода и другие статические методы этого класса.

Видимость окна определяется его свойством `visible`, поэтому, чтобы скрыть окно, свойству следует установить значение `false`.

Напишите обработчик события нажатия на кнопку отмены операции, выполняющий следующие действия:

- сокрытие окна,
- присвоение полю статуса окна значения `CANCEL`.

Поскольку кроме кнопки отмены операции пользователь может просто нажать на закрывающую окно кнопку, требуется написать обработчик события `windowClosing`, возникающего при попытке закрытия окна. В этом обработчике нужно только присвоить полю статуса окна значение `CANCEL`, а скрыто окно будет автоматически из-за указанного до этого значения свойства `defaultCloseOperation`.

Добавьте в класс метод `showDialog()`, который и будет использоваться для вывода окна диалога на экран. Метод должен делать окно видимым и возвращать статус окна после завершения работы с ним.

Также добавьте методы `getLeftDomainBorder()`, `getRightDomainBorder()` и `getPointsCount()`, возвращающие, соответственно, введённые значения левой и правой границ области определения и количество точек.

Задание 2 *(1 балл)*

Описать класс документа с табулированной функцией.

Объект этого класса должен описывать открытый в программе документ, т.е. содержать ссылку на объект табулированной функции, текущее имя файла документа, а также булевский флаг, показывающий то, изменялся ли документ с момента последнего сохранения. Значения последних двух полей должны соответствующим образом изменяться в ходе работы методов класса.

Реализуйте в классе следующие методы, соответствующие действиям с документом, которые может выполнить пользователь:

- метод `newFunction(double leftX, double rightX, int pointsCount)` должен заменять объект табулированной функции на новый с указанными параметрами;
- Метод `saveFunctionAs(String filenName)` должен сохранять текущую табулированную функцию в формате JSON. Для работы с ним следует подключить библиотеку `org.json.simple.JSONObject` (для этого ее потребуется скачать, например здесь: http://www.java2s.com/Code/Jar/j/Downloadjsonsimple11jar.htm, а пример работы с JSON рассмотрен здесь https://howtodoinjava.com/java/library/json-simple-read-write-json-examples). Для подключения библиотеки в NetBeans следует щелкнуть правой клавишей мыши по проекту в дереве проекта, выбрать свойства, далее в разделе Libraries – Classpath нажать “+”, выбрать “Add JAR/Folder” и указать файл библиотеки, извлеченный из архива.
- метод `loadFunction(String fileName)` должен загружать из файла с указанным именем табулированную функцию в формате JSON. Для разбора JSON формата предлагается использовать класс `org.json.simple.parser.JSONParser` (разбор JSON описан здесь: https://hr-vector.com/java/json-parser)
- метод `saveFunction()` должен сохранять текущую табулированную функцию в текстовый файл с текущим именем в формате JSON;
- метод `tabulateFunction(Function function, double leftX, double rightX, int pointsCount)` должен заменять объект табулированной функции на новый, полученный путём табулирования указанной функции с указанными параметрами.

При написании этих методов воспользуйтесь методами класса `TabulatedFunctions`.

Определите в классе два доступных только для чтения булевских свойства `modified` и `fileNameAssigned`, показывающих, соответственно, изменялся ли документ с последнего сохранения и назначено ли для документа имя файла.

Чтобы предоставить возможность работы с хранящейся табулированной функцией, а также обеспечить отслеживание изменений, вместо предоставления прямого доступа к объекту функции поступим следующим образом. Пусть класс документа сам реализует интерфейс `TabulatedFunction` и содержит реализации всех методов этого интерфейса. Но вместо действительной реализации этих методов (такой, какая была в классах `ArrayTabulatedFunction` и `LinkedListTabulatedFunction`) в них просто будут вызываться методы объекта табулированной функции, хранящейся в документе. Т.е. объект

документа, вместо того, чтобы самому, например, вычислять значение функции, будет заставлять это делать хранящийся в документе объект табулированной функции.

Таким образом, требуется реализовать все методы из интерфейса `TabulatedFunction`, а также переопределить методы из класса `Object` так, чтобы для реального выполнения действия вызывались методы объекта хранящейся в документе табулированной функции. Исключениями будут являться методы `equals()` и `clone()` (подумайте, почему это так, и реализуйте методы корректно). В методах, которые меняют функцию, нужно ещё и изменять значение поля, показывающего, что документ изменялся.

Задание 3 *(1 балл)*

Для работы компонента типа `JTable` (а именно он будет использоваться для вывода и редактирования табулированной функции) необходима так называемая модель таблицы, т.е. специальный объект, хранящий данные таблицы и описывающий то, как с этими данными таблица может работать. Классы таких объектов должны реализовывать интерфейс `TableModel`. Однако часто оказывается проще унаследовать свой класс модели от класса `DefaultTableModel` и изменить только те особенности поведения, которые нужно.

Опишите свой класс модели для таблицы, содержащий конструктор и ряд описанных ниже методов.

Конструктор этого класса должен принимать и сохранять в полях объекта два параметра. Первый из них должен иметь тип `TabulatedFunction` и являться ссылкой на объект табулированной функции, данные из которой выводятся и редактируются в таблице. Второй должен иметь тип `Component` и являться ссылкой на родительский компонент, относительно которого должны выводиться сообщения об ошибках.

Метод `int getRowCount()` должен возвращать количество строк в таблице. Очевидно, что оно совпадает с количеством точек в табулированной функции.

Метод `int getColumnCount()` должен возвращать количество столбцов в таблице.

Метод `String getColumnName(int index)` должен возвращать заголовок столбца таблицы по его номеру.

Метод `Class getColumnClass(int index)` должен возвращать ссылку на рефлексивное описание типа объектов, которые выводятся в указанном столбце таблицы. Поскольку в таблицу будут выводиться числа, возвращать этот метод должен значение `Double.class`.

Метод `boolean isCellEditable(int rowIndex, int columnIndex)` должен возвращать логическое значение, показывающее, можно ли редактировать значение в ячейке с указанными координатами.

Метод `Object getValueAt(int rowIndex, int columnIndex)` должен возвращать объект, значение которого выводится в ячейку с указанными координатами. Здесь этими значениями являются абсциссы и ординаты точек табулированной функции.

Метод `void setValueAt(Object value, int rowIndex, int columnIndex)` должен изменять значение в табулированной функции. Первый его параметр – новое значение, второй и третий – координаты ячейки таблицы, в которой произошло изменение значения. В случае если значение некорректно, следует вывести на экран сообщение об ошибке с помощью метода `JOptionPane.showMessage()` (здесь пригодится второй параметр конструктора класса модели таблицы).

Задание 4 *(1 балл)*

Реализуйте класс основного окна программы. Пример такого окна показан на рисунке 2. Для создания класса окна удобнее всего воспользоваться мастером среды разработки.

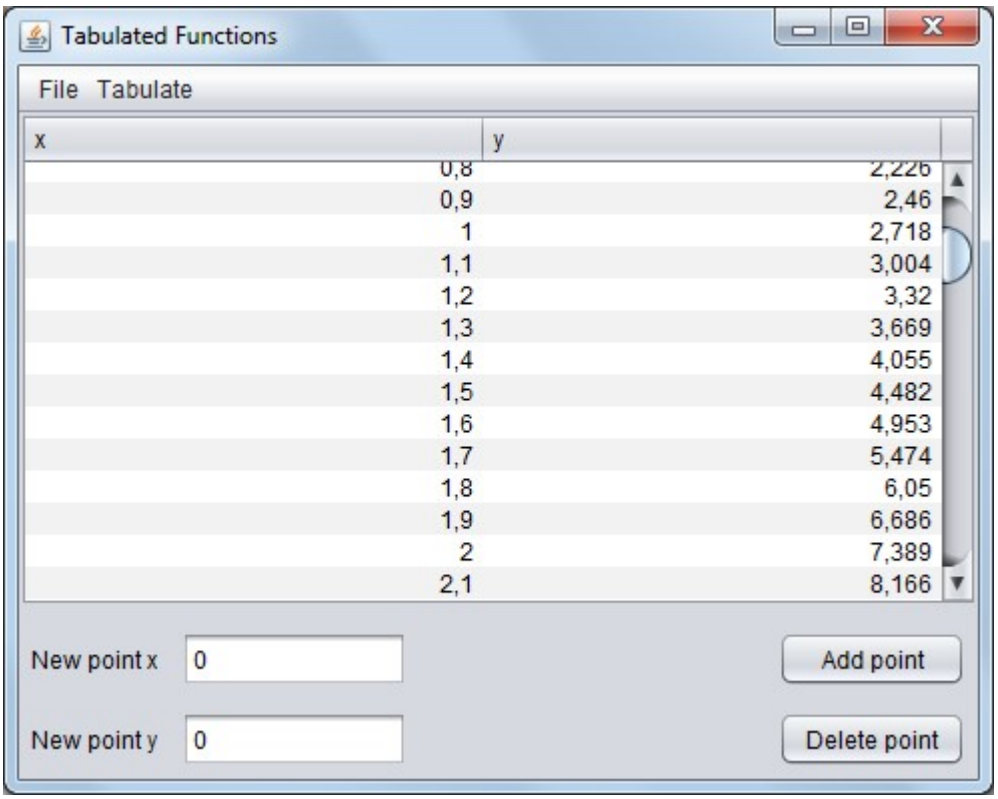

Рисунок 2 – Примерный вид основного окна программы

Класс данного диалогового окна должен наследовать от класса `JFrame`.

Данное окно должно позволять изменять свой размер, поэтому убедитесь, что свойство окна `resizable` имеет значение `true`. В случае изменения размеров окна находящиеся внутри него компоненты должны корректно изменять свои размеры и положение (при реализации с помощью визуальных средств среды разработки этого можно легко добиться, правильно «пристыковав» компоненты друг к другу и к границам окна).

Поскольку далее потребуется контролировать процесс закрытия окна и приложения, измените значение свойства `defaultCloseOperation` на `DO_NOTHING` (т.е. по нажатию на кнопку закрытия окна автоматически нечего происходить не будет).

Расположите на форме следующие компоненты:

- основное меню окна (компонент типа `JMenuBar`);
- элементы основного меню окна (компоненты типа `JMenu`) для работы с файлами табулированных функций и для операции табуляции;
- элементы первого меню (компоненты типа `JMenuItem`) для создания нового документа, открытия документа из файла, сохранения документа, сохранения документа под новым именем, выхода из программы;
- элемент второго меню (компоненты типа `JMenuItem`) для загрузки и табулирования функции;
- область с полосами прокрутки (компонент типа `JScrollPane`) и находящуюся в нём таблицу (компонент типа `JTable`) (при использовании визуальных средств среды разработки при добавлении на форму таблицы обрамляющая её область прокрутки будет добавлена автоматически);
- два поля редактирования координат добавляемой точки (компоненты типа `JTextField`);
- текстовые пояснения для редактируемых полей (компоненты типа `JLabel`);
- кнопки добавления новой точки и удаления точки (компоненты типа `JButton`).

Для текстовых полей редактирования задайте значения по умолчанию.

Одной из особенностей таблиц `JTable` является то, что сама таблица имеет такой размер, чтобы в ней размещались все её элементы. Поэтому, чтобы таблицы не занимали большие площади в основном окне, их обычно помещают внутрь компонента типа `JScrollPane`. Этот компонент позволяет находящейся внутри таблице иметь любые размеры и предоставляет полосы прокрутки для перемещения, если это необходимо.

Для работы приложения в классе также потребуются следующие поля:

- ссылка на объект вспомогательного окна для ввода параметров (его класс написан вами в рамках выполнения задания 1);
- ссылка на документ табулированной функции (его класс написан вами в рамках выполнения задания 2);
- ссылка на объект типа `JFileChooser` (объекты этого типа позволяют открывать диалоги выбора файлов), который потребуется в методах открытия и сохранения файлов.

Все эти поля должны быть проинициализированы объектами в конструкторе класса. У объекта документа также следует вызвать метод создания новой табулированной функции, чтобы при открытии программы пользователь мог сразу начать работу.

Сделайте так, чтобы при инициализации компонентов в таблицу в качестве модели передавался объект класса, разработанного вами при выполнении задания 3. В качестве первого параметра конструктора следует передать ссылку на документ табулированной функции, а в качестве второго параметра – ссылку на само основное окно.

Создайте обработчик события выбора элемента меню создания нового документа (для этого также можно воспользоваться средствами среды разработки). Обработчик должен вызывать метод `showDialog()` вспомогательного окна выбора параметров, и если его результат соответствует константе `OK` из класса вспомогательного окна, выполнить следующие действия:

- вызвать метод `newFunction()` у документа, передав в качестве параметров введённые во вспомогательном окне значения;
- вызвать метод `revalidate()` у объекта таблицы (этот метод подстраивает размеры таблицы под новые значения);
- вызвать метод `repaint()` у объекта таблицы (этот даёт команду на перерисовку таблицы).

Изучите описание и возможности класса `JFileChooser`.

Создайте обработчик события выбора элемента меню сохранения документа в новый файл. Обработчик должен выводить диалог выбора файла для сохранения, и в случае подтверждающего сохранение действия пользователя должен вызываться метод сохранения в новый файл документа табулированной функции. Если в ходе сохранения возникли проблемы, следует вывести окно с сообщением.

Создайте обработчик события выбора элемента меню сохранения документа текущий файл. Обработчик должен проверить значение свойства `fileNameAssigned` документа табулированной функции и, если имя назначено, вызвать её метод сохранения, а если не назначено, вызвать обработчик события выбора элемента меню сохранения документа в новый файл. Аналогично, при возникновении проблем в ходе записи должно выводиться окно с сообщением.

Создайте обработчик события выбора элемента меню открытия документа из файла. Обработчик должен выводить диалог выбора файла для открытия, и в случае подтверждающего открытие действия пользователя должен вызываться метод считывания из файла документа табулированной функции. Если в ходе считывания возникли проблемы, следует вывести окно с сообщением. После считывания также следует вызвать методы перестроения и перерисовки таблицы.

Создайте обработчик события выбора элемента меню выхода из программы. Обработчик должен проверять значение свойства `modified` документа табулированной функции. Если

документ изменялся с последнего сохранения, нужно вывести окно с вопросом о том, действительно ли пользователь хочет завершить работу приложения или нет. Для этого можно воспользоваться методом `JOptionPane.showConfirmDialog()`. Если документ не изменялся или пользователь подтвердил выход из программы, нужно завершить её работу. Для этого вызовите метод `dispose()` вашего объекта окна (он закроет и освободит ресурсы окна, после чего приложение тоже закроется, т.к. это было его основное окно).

Создайте обработчик события `windowClosing` попытки закрытия окна программы. Поведение программы должно быть аналогично случаю выбора элемента меню выхода из программы.

Создайте обработчик события нажатия на кнопку удаления точки из табулированной функции. Удаляться должна точка, строка которой выбрана пользователем в таблице. Номер выбранной строки можно узнать с помощью метода `getSelectedRow()` таблицы. Если точку удалить нельзя, должно выводиться окно с сообщением об этом. После удаления точки следует вызвать методы перестроения и перерисовки таблицы.

Создайте обработчик события нажатия на кнопку добавления точки в табулированную функцию. Координаты добавляемой точки должны браться из текстовых полей редактирования. Если точку нельзя добавить или если пользователь ввёл некорректные значения, должно выводиться окно с сообщением об этом. После добавления точки следует вызвать методы перестроения и перерисовки таблицы.

Создайте точку входа программы, выводящую на экран основное окно программы.

Задание 5 *(1 балл)*

Реализуйте возможность построения табулированной функции путём табулирования обычной функции, указываемой пользователем в виде байт-кода класса функции.

Для этого напишите класс загрузчика классов из указанного файла (он должен расширять класс `ClassLoader`).

В классе основного окна программы добавьте поле для хранения ссылки на объект загрузчика классов, проинициализируйте это поле в конструкторе.

Создайте обработчик события выбора элемента меню для загрузки и табулирования функции. Обработчик должен выводить диалог выбора файла, и в случае подтверждающего открытие действия пользователя должен выводиться диалог выбора параметров табулирования функции (вспомогательное окно из задания 1). Если пользователь и здесь подтвердил параметры, необходимо выполнить следующие действия:

- с помощью загрузчика классов считать класс;
- создать объект этого класса;
- вызвать метод табулирования функции у объекта документа, передав туда в качестве параметров созданный объект функции и введённые во вспомогательном окне значения;
- перестроить и перерисовать таблицу.

В случае возникновения ошибок (если файл невозможно считать, если в файле находится не байт-код, если класс в файле не реализует интерфейс `Function` и т.д. и т.п.) должны выводиться окна с сообщениями об этих ошибках.