Лабораторная работа №7

Задание на лабораторную работу

Лабораторная работа является продолжением лабораторных работ. Требуется создать графический интерфейс для пользователя, с помощью которого он сможет пользоваться функционалом, написанным ранее. Студенты самостоятельно выбирают технологии для разработки интерфейса, а также сами распределяют между собой задачи (примерно поровну) и решают, когда делать «коммиты». Визуальное размещение компонент в окнах интерфейса в том числе определяется студентами. При этом интерфейс должен быть дружественным по отношении к пользователю и не вызывать желания закрыть программу и больше никогда ей не пользоваться. В работе может встретиться очень много однотипных задач и подзадач – чтобы избежать дублирования кода, рекомендуется создавать новые методы и/или классы. Также настоятельно рекомендуется пользоваться таким изобретением человечества, как сеть Интернет, поскольку задачи создания графического интерфейса являются типовыми и где-то уже наверняка были реализованы.

Задание 1

Все классы, относящиеся исключительно к графической части работы, должны располагаться в пакете `ui` рядом с пакетом functions.

Реализовать веб-интерфейс, позволяющее создавать объекты `TabulatedFunction` из массивов `x` и `y`. Пользователь должен вводить в текстовое поле количество точек у функции, нажимать на кнопку, после чего появляется таблица с двумя столбцами, в которые пользователь должен будет вводить табулированные значения `x` и `y`. Можно сделать так, что таблица уже видна изначально, но в ней нет строк до нажатия на кнопку. Обдумать возможность нажатия на кнопку ещё раз, когда значения в таблице уже введены: что будет происходить? После того, как пользователь ввёл все значения и нажал на кнопку «Создать», с помощью фабрики должна создаваться табулированная функция, а выпадающее окно должно закрываться. Выбрать любую фабрику самостоятельно – во втором задании это поведение изменится.

Здесь и далее во всех заданиях любое исключение не должно обрабатываться путём вывода информации в консоль – пользователь веб приложения её не увидит. Вместо этого при возникновении исключения должно появляться выпадающее окно в модальном режиме (т.е. не позволяющее работать с вкладкой, пока его не закроют) и выводить читаемую информацию об исключении. Для этого нужно продумать, какие исключения могут произойти и корректно обрабатывать каждое из них. Например, если в текстовое поле размера таблицы введено отрицательное число – это одна ошибка, а если там текст – то другая, и обрабатываться они должны по-разному. Поскольку в программе будет присутствовать не одна вкладка, можно создать специальный класс, который будет обрабатывать все исключения, и передавать объекты исключений ему.

Реализовать другое веб-интерфейс, позволяющее создавать объекты `TabulatedFunction` из другой функции `MathFunction`, т.е. используя второй конструктор. Для этого пользователь должен выбирать функцию из выпадающего списка (в списке должны присутствовать все простые функции без аргументов конструктора, написанные в программе и не являющиеся табулированными, такие как `SqrFunction`, `IdentityFunction` и т.д.), указывать количество точек разбиения и интервал разбиения. Пользователь при выборе функции из выпадающего списка должен видеть её

локализованное название (т.е., например, «Квадратичная функция», «Тождественная функция» и т.п.), при этом названия должны быть упорядочены по алфавиту. А для того, чтобы по названию можно было определить, какую функцию выбрал пользователь, потребуется обращаться к отображению `Map`, ставящему в соответствии названию функции объект `MathFunction` (пример: ключ «Квадратичная функция», значение `new SqrFunction()`). После того, как пользователь заполнил все значения и нажал на кнопку «Создать», с помощью фабрики должна создаваться табулированная функция, а окно должно закрываться. Если что-то введено некорректно, соответствующее исключения должно быть обработано, как описывалось выше. В интерфейс фабрики (и его реализации) потребуется добавить дополнительный метод, позволяющий создавать объекты табулированных функций вторым способом (по аналогии с созданием через массив).

Задание 2

Создать главную страничку интерфейса, с помощью которого пользователь будет открывать другие странички.

Создать страничку(меню) настроек, в котором можно будет выбрать, какую фабрику использовать для генерации новых табулированных функций. По умолчанию – реализация на основе массива, но пользователь может изменить это на связный список и обратно. Объект фабрики должен где-то храниться, а ссылка на него в дальнейшем должна передаваться в другие окна, которые могут создать табулированную функцию. В главном окне должна быть возможность открыть окно настроек.

Пользователь не должен иметь возможности открывать одинаковые окна. Для этого рекомендуется открывать новые окна в модальном режиме.

Создать окно, открываемое из главного окна, реализующее простейшие поэлементные операции над двумя табулированными функциями – сложение, вычитание, умножение и деление. Для реализации потребуется использовать объект класса `TabulatedFunctionOperationService`, содержащий эти методы (а он в свою очередь должен иметь ссылку на корректную фабрику). В окне должны присутствовать три основные области – для первого операнда-функции, для второго операнда-функции и для результата. Все три функции должны отображаться в виде таблицы (изначально пустой, если функции нет). При этом таблица должна уметь считывать значения из табулированной функции методами `getX()` и `getY()` и задавать значение методом `setY()`. Итератором пользоваться нельзя – он реализован так, что создаёт копии точек. Колонка `x` не должна быть редактируемой. Функция-результат не должна быть редактируемая вообще. Для каждой функции-операнда должно быть задано как минимум три кнопки – создать, загрузить и сохранить. Причём у пользователя должна быть возможность выбора, из чего создавать функцию – из массивов или из другой функции. Этого можно достичь разными способами, один из которых – вместо одной кнопки создания сделать две. После выбора должно открываться первое или второе окно, созданные при выполнении первого задания. Когда пользователь завершит создание функции и закроет окно, она должна появиться в соответствующей таблице.

При нажатии на кнопку сохранения должно открываться диалоговое окно, где пользователь выбирает, куда именно и с каким названием сохранять функцию – в этот файл следует записать сериализованный объект табулированной функции (использовать класс `FunctionsIO`). При нажатии на кнопку загрузки пользователь должен выбрать файл, из которого функция будет десериализована и записана в таблицу. Для

функции-результата должны быть кнопки сложения, вычитания, умножения и деления, а также кнопка сохранения.

Создать окно, открываемое из главного окна, реализующую операцию дифференцирования табулированной функции с помощью соответствующего дифференциального оператора (со ссылкой на выбранную в настройках фабрику). Окно должно содержать две основные области: слева – начальная функция, справа – результат дифференцирования. Функционал должен быть аналогичен предыдущему случаю.

Изучить весь графический интерфейс программы и подумать, каких элементов не хватает – добавить их.

Задание 3 *

Найти библиотеку для отображения функций, заданных таблично, в виде графиков. Добавить окно, позволяющее изучать и изменять табулированную функцию (с кнопками создания, загрузки и сохранения сериализованного объекта). При этом сама функция должна отображаться на графике. Добавить кнопку, позволяющую вычислить значение функции в произвольной точке `x` с помощью метода `apply()`.

В окнах, где допускается изменение значений уже созданной функции (т.е. окно элементарных операций, окно дифференцирования и окно графического изучения функции) добавить кнопку «Вставка», которая должна быть видна тогда и только тогда, когда соответствующая функция реализует интерфейс `Insertable`. Кнопка, очевидно, должна добавлять какое-то новое значение в выбранную функцию. Например, в окне элементарных операций может получиться так, что первая функция будет иметь `Insertable`-тип, а вторая – нет. Поэтому у первой должна отображаться эта кнопка, а у второй – нет.

В тех же окнах по аналогии добавить кнопку «Удалить» для интерфейса `Removable`.

Для окна, создающего табулированную функцию из другой простой функции, переписать добавление в выпадающий список всех возможных функций. Вместо этого программа должна сканировать пакет `functions` с помощью механизма рефлексии, чтобы самостоятельно находить нужные простые функции. Для реализации рекомендуется создать аннотацию, которой и следует снабдить желаемые простые функции. Аннотация должна содержать параметры локализованного названия и приоритета отображения в списке функций. Т.е. в выпадающем списке должны присутствовать локализованные названия, упорядоченные согласно приоритету отображения, и, если он одинаков, то по алфавиту.

Добавить возможность сохранять и загружать табулированную функцию в формате xml и/или json.

Добавить окно, в котором для выбранной функции, будет вычисляться определённый интеграл на всей области определения табулированной функции. Интеграл должен быть рассчитан параллельно с использованием ранее написанного функционала. Количество потоков выполнения указывает пользователь.

Добавить окно, позволяющее пользователю создать сложные функции `CompositeFunction` из простых или из других сложных, которые он уже создал ранее или создаёт прямо сейчас. Подумать, как должен будет в этом случае меняться графический интерфейс окна. После создания новой сложной функции она должна быть

добавлена в список функций, из которых можно создавать табулированную функцию (да, пользователь дополнительно должен придумать ей локализованное название и потребуется где-то хранить это название).

Создать jar-файл, который при запуске будет открывать главное окно приложения. Не добавлять его в репозиторий.