УВО «Университет управления «ТИСБИ»
Факультет Информационных Технологий

Пояснительная записка к курсовой работе по объектно-ориентированному программированию на тему: Разработка объектной программы для задачи «Центр тестирования»

Выполнил(а):
Студент(ка) группы П-711
Хайруллин Б. М.
Руководитель:
Канд. технических наук Козин А.Н.

г. Казань 2019

Оглавление

1. Постановка задачи

Разработать объектную программу для хранения и обработки данных о пользователях центра тестирования. Программа должна поддерживать список пользователей с указанием их фамилии и числа пройденных тестов. Для каждого пользователя создается свой список пройденных тестов с указанием названия теста и результата его прохождения по 100-бальной шкале.

Разработка включает в себя определение необходимых объектов и описание их в виде классов, программную реализацию методов добавления и удаления пользователей и тестов с подсчетом среднего результата для каждого пользователя, всестороннее тестирование методов с помощью консольного (при разработке) и оконного (в окончательном варианте) приложения.

Для объединения пользователей используется структура данных в виде адресного разомкнутого неупорядоченного однонаправленного списка без заголовка. Для объединения тестов используется очередь на основе массива со сдвигом элементов.

Разработка выполняется с учётом следующих требований:

- имена классов, свойств и методов должны носить содержательный смысл и соответствовать информационной задаче
- обязательное соблюдение принципа инкапсуляции – использование в классах только закрытых свойств и реализация необходимого набора методов доступа
- наличие двух методов для сохранения всей объектной структуры во внешнем файле с обратной загрузкой, при этом стандартные механизмы сериализации разрешается использовать только как дополнение к самостоятельно реализованным методам
- тестовое оконное приложение должно обладать удобным пользовательским интерфейсом с контролем вводимых данных и отображением текущего состояния объектной структуры с помощью списковых или табличных компонентов

- стандартные контейнеры/коллекции (включая обобщенные классы) разрешается использовать только как дополнение к самостоятельно разработанным классам
- в качестве языка разработки разрешается использовать Java, C#, C++, Object/Free Pascal и соответствующие инструменты быстрой разработки приложений.

2. Описание используемых структур данных с алгоритмами выполнения основных операций

О́чередь на основе динамического массива — абстрактный тип данных с дисциплиной доступа к элементам «первый пришёл — первый вышел» (FIFO, англ. first in, first out). Добавление элемента (принято обозначать словом enqueue — поставить в очередь) возможно лишь в конец очереди, выборка — только из начала очереди (что принято называть словом dequeue — убрать из очереди), при этом выбранный элемент из очереди удаляется.

Динамический массив – это массив, размер которого может изменяться во время исполнения программы. Возможность изменения размера отличает динамический массив от статического, размер которого задаётся на момент компиляции программы. Для изменения размера динамического массива язык программирования, поддерживающий такие массивы, должен предоставлять встроенную функцию или оператор. Динамические массивы дают возможность более гибкой работы с данными, так как позволяют не прогнозировать хранимые объёмы данных, а регулировать размер массива в соответствии с реально необходимыми объёмами.

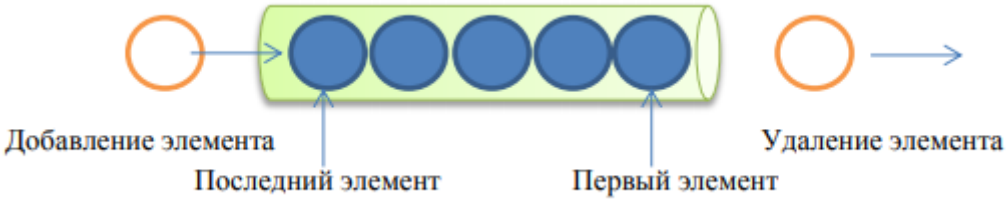
Также иногда к динамическим относят массивы переменной длины, размер которых не фиксируется при компиляции, а задаётся при создании или инициализации массива во время исполнения программы. От «настоящих» динамических массивов они отличаются тем, что для них не предоставляются средства автоматического изменения размера с сохранением содержимого, так что при необходимости программист должен реализовать такие средства самостоятельно.

Алгоритм добавления элемента в очередь на основе массива:

1. Проеверка на возможность добавления элемента.
2. Добавить элемент в массив.
3. Увеличить размер массива, если массив заполнен.
4. Если есть счётчик, увеличить счётчик на 1.

Алгоритм удаления элемента из очереди на основе массива:

1. Проверить наличие первого элемента в массиве (Перейти к пункту 2. Если нет - закончить удаление как неудачное)
2. Переместить все значения в ячейках на одну ячейку к началу массива, начиная со второй ячейки.
3. Если есть счётчик, понизить значение счётчика на 1.



Однонаправленный список - это структура данных, состоящая из элементов одного типа, связанных между собой последовательно посредством указателей. Каждый элемент списка имеет указатель на следующий элемент. Последний элемент списка указывает на NULL. Элемент, на который нет указателя, является первым (головным) элементом списка. Здесь ссылка в каждом узле указывает на следующий узел в списке. В односвязном списке можно передвигаться только в сторону конца списка. Узнать адрес предыдущего элемента, опираясь на содержимое текущего узла, невозможно.

Алгоритм добавления элемента в однонаправленный список:

1. Проверка на возможность добавления.
2. Создание добавляемого узла и заполнение его поля данных.
3. Если список пуст, то создать указатель на только что созданный узел. Закончить процедуру добавления как удачную.
4. Если список не пуст, следует переустановка указателя узла, предшествующего добавляемому, на добавляемый узел.

5. Установка указателя добавляемого узла на следующий узел (тот, на который указывал предшествующий узел). Закончить процедуру как удачную.

Алгоритм удаления элемента из однонаправленного списка:

1. Проверка на возможность удаления удаляемого элемента. В случае наличия таковой возможности перейти к пункту 2.
2. Поиск удаляемого элемента в списке.
3. Если удаляемый элемент оказался первым элементом, то изменить указатель, ссылающийся на первый элемент на указатель, ссылающийся на второй элемент.
4. Если удаляемый элемент оказался последний элементом, то изменить указатель, ссылающийся на этот элемент у предешествующего элемента на null.
5. Если удаляемый элемент оказался не первым и не последним элементом, то изменить указатель, ссылающийся на этот элемент у предшествующего элемента на элемент, следующий после удаляемого элемента.
6. Освобождение памяти удаляемого элемента.

Краткие сведения об объектном подходе

Объектно-ориентированное программирование (ООП) — парадигма программирования, в которой основными концепциями являются понятия объектов и классов (либо, в менее известном варианте языков с прототипированием — прототипов).

Класс это тип, описывающий устройство объектов. Понятие «класс» подразумевает некоторое поведение и способ представления. Понятие «объект» подразумевает нечто, что обладает определённым поведением и способом представления. Говорят, что объект - это экземпляр класса. Класс можно сравнить с чертежом, согласно которому создаются объекты. Обычно классы разрабатывают таким образом, чтобы их объекты соответствовали объектам предметной области.

Объектное и объектно-ориентированное программирование (ООП) возникло в результате развития идеологии процедурного программирования, где данные и подпрограммы (процедуры, функции) их обработки формально не связаны. Кроме того, в современном объектно-ориентированном программировании часто большое значение имеют понятия события (так называемое событийно-ориентированное программирование) и компонента (компонентное программирование).

Первым языком программирования, в котором были предложены принципы объектной ориентированности, была Симула. В момент своего появления (в 1967 году), этот язык программирования предложил поистине революционные идеи: объекты, классы, виртуал современниками как нечто грандиозное. Тем не менее, большинство концепций были развиты Аланом Кэйем и Дэном Ингаллсом в языке объектно-ориентированным языком программирования.

В настоящее время количество прикладных языков программирования (список языков), реализующих объектно-ориентированную парадигму, является наибольшим по отношению к другим парадигмам. В области системного программирования до сих пор применяется парадигма процедурного программирования, и общепринятым языком программирования является язык C. Хотя при взаимодействии системного и прикладного уровней операционных систем заметное влияние стали оказывать языки объектно-ориентированного программирования. Например, одной из наиболее распространенных библиотек

мульти платформенного программирования является объектно-ориентированная библиотека.

Структура данных «класс», представляющая собой объектный тип данных, внешне похожа на типы данных процедурно-ориентированных языков, такие как структура в языке Си или запись в Паскале или QuickBasic. При этом элементы такой структуры (члены класса) могут сами быть не только данными, но и методами (то есть процедурами или функциями). Такое объединение называется инкапсуляцией.

Наличие инкапсуляции достаточно для объектности языка программирования, но ещё не означает его объектной ориентированности — для этого требуется наличие наследования.

Но даже наличие инкапсуляции и наследования не делает язык программирования в полной мере объектным с точки зрения ООП.

Основные преимущества ООП проявляются только в том случае, когда в языке программирования реализован полиморфизм.

ООП имеет уже более чем сорокалетнюю историю, но, несмотря на это, до сих пор не существует чёткого общепринятого определения данной технологии. Основные принципы, заложенные в первые объектные языки и системы,

подверглись существенному изменению (или искажению) и дополнению при многочисленных реализациях последующего времени.

Кроме того, примерно с середины 1980-х годов термин «объектно-ориентированный» стал модным, в результате с ним произошло то же самое, что несколько раньше с термином «структурный» (ставшим модным после распространения технологии структурного программирования), его стали искусственно «прикреплять» к любым новым разработкам, чтобы обеспечить им привлекательность. Бьёрн Страуструп в 1988 году писал, что обоснование «объектной ориентированности» чего-либо, в большинстве случаев, сводится к силлогизму: «X — это хорошо. Объектная ориентированность - это хорошо. Следовательно, X является объектно-ориентированным».

Появление в ООП отдельного понятия класса закономерно вытекает из желания иметь множество объектов со сходным поведением. Класс в ООП — это в чистом виде абстрактный тип данных, создаваемый программистом. С

этой точки зрения объекты являются значениями данного абстрактного типа, а определение класса задаёт внутреннюю структуру значений и набор операций, которые над этими значениями могут быть выполнены. Желательность иерархии классов (а значит, наследования) вытекает из требований к повторному использованию кода — если несколько классов имеют сходное поведение, нет смысла дублировать их описание, лучше выделить общую часть в общий родительский класс, а в описании самих этих классов оставить только различающиеся элементы.

Необходимость совместного использования объектов разных классов, способных обрабатывать однотипные сообщения, требует поддержки полиморфизма — возможности записывать разные объекты в переменные одного и того же типа.

В таких условиях объект, отправляя сообщение, может не знать в точности, к какому классу относится адресат, и одни и те же сообщения, отправленные переменным одного типа, содержащим объекты разных классов, вызовут различную реакцию.

Отдельного пояснения требует понятие обмена сообщениями. Первоначально (например, в том же Smalltalk) взаимодействие объектов представлялось как «настоящий» обмен сообщениями, то есть пересылка от одного объекта другому специального объекта-сообщения. Такая модель является чрезвычайно общей. Она прекрасно подходит, например, для описания параллельных вычислений с помощью активных объектов, каждый из которых имеет собственный поток исполнения и работает одновременно с прочими.

Такие объекты могут вести себя как отдельные, абсолютно автономные вычислительные единицы. Посылка сообщений естественным образом решает вопрос обработки сообщений объектами, присвоенными полиморфным переменным — независимо от того, как объявляется переменная, сообщение обрабатывает код класса, к которому относится присвоенный переменной объект.

Однако общность механизма обмена сообщениями имеет и другую сторону — «полноценная» передача сообщений требует дополнительных накладных расходов, что не всегда удобно. Поэтому в большинстве ныне существующих объектно-ориентированных языков программирования используется концепция «отправка сообщения как вызов метода» — объекты

имеют доступные извне методы, вызовами которых и обеспечивается взаимодействие объектов. Данный подход реализован в огромном количестве языков программирования, в том числе C++, Object Pascal, Oberon-2.

В настоящий момент именно он является наиболее распространенным в объектно-ориентированных языках

Концепция виртуальных методов, поддерживаемая этими и другими современными языками, появилась как средство обеспечить выполнение нужных методов при использовании полиморфных переменных, то есть, по сути, как попытка расширить возможности вызова методов для реализации части функциональности, обеспечиваемой механизмом обработки сообщений.

ООП ориентировано на разработку крупных программных комплексов, разрабатываемых командой программистов (возможно, достаточно большой).

Проектирование системы в целом, создание отдельных компонент и их объединение в конечный продукт при этом часто выполняется разными людьми, и нет ни одного специалиста, который знал бы о проекте всё.

Объектно-ориентированное проектирование основывается на описании структуры и поведения проектируемой системы, то есть, фактически, в ответе на два основных вопроса:

Из каких частей состоит система.

В чём состоит ответственность каждой из частей.

Выделение частей производится таким образом, чтобы каждая имела минимальный по объёму и точно определённый набор выполняемых функций (обязанностей), и при этом взаимодействовала с другими частями как можно меньше.

Дальнейшее уточнение приводит к выделению более мелких фрагментов описания. По мере детализациописания и определения ответственности выявляются данные, которые необходимо хранить, наличиеблизких по поведению агентов, которые становятся кандидатами на реализацию в виде классов с общимипредками. После выделения компонентов и определения интерфейсов между ними реализация каждогокомпонента может проводиться практически независимо от остальных (разумеется, при соблюдении соответствующей технологической дисциплины).

Большое значение имеет правильное построение иерархии классов. Одна из известных проблем больших систем, построенных по ООП-технологии — так называемая проблема хрупкости базового класса. Она состоит в том, что на поздних этапах разработки, когда иерархия классов построена и на её основе разработано большое количество кода, оказывается трудно или даже невозможно внести какие-либо изменения в код базовых классов иерархии (от которых порождены все или многие работающие в системе классы). Даже если вносимые изменения не затронут интерфейс базового класса, изменение его поведения может непредсказуемым образом отразиться на классах-потомках. В случае крупной системы разработчик базового класса не просто не в состоянии предугадать последствия изменений, он даже не знает о том, как именно базовый класс используется и от каких особенностей его поведения зависит корректность работы классов-потомков.

4. Формализованное описание разработанных классов

Всего четыре класса.

Класс Test

```
package com.akaleaf.course;
/** Класс Тестов */
public class Test {
    private String name;
    private int    result;
    public Test(String name, int result) {}
    public void setName(String name) {}
    public void setResult(int result) {}
    public String getName() {}
    public int getResult() {}
}
```

Класс User

```
package com.akaleaf.course;
import java.util.Arrays;
/** Класс пользователей */
```

```
public class User {
    private String lastName;
    private User next;
    private Test tests[];
    private int testsTop = -1;
    public User() {}
    /**
     * Конструктор объектов User.
     *
     * @param lastName    Используется для задания имени для нового объекта.
     * @param testsLength Используется для задания размера массива, в котором хранятся тесты.
     */
    public User(String lastName, int testsLength) {}
    public void setLastName(String lastName) {}
    /**
     * Ищет test в массиве tests по совпадению свойства name и
     * присваивает свойству result объекта test
     * новое значение result.
     *
     * @param name   Используется для нахождения test
     * @param result Новое значение для свойства result у test
     */
    public void setTestResult(String name, int result) {}
    /**
     * Ищет test в массиве tests по совпадению свойства name и
     * присваивает свойству name объекта test
     * новое значение newName.
     *
     * @param oldName Используется для нахождения test
     * @param newName Новое значение для свойства name у test
     */
    public void setTestName(String oldName, String newName) {}
    /**
     * Удаляет первый(т.к. очередь) test из массива tests
```

```
     *
     * @return результат удаления: true - успешно
     *                             false - неудачно.
     */
    public boolean removeTest() {}
    /**
     * Удаляет test с совпадающим lastName из массива tests
     *
     * @param testName Удаляемый тест
     * @return результат удаления: true - успешно
     *                             false - неудачно.
     */
    public boolean removeTest(String testName) {}
    /**
     * Добавляет новый test в конец массива tests
     *
     * @param name   Используется для задания свойства name новому test
     * @param result Используется для задания свойства result новому test
     */
    public void addTest(String name, int result) {}
    public boolean isTestAlreadyExists(String testName) {}
    public void setNext(User newNext) {}
    public String getLastName() {}
    public int getTestsTop() {}
    public Test[] getTests() {}
    /**
     * Используется для получения среднего значения среди всех result объектов test
     *
     * @return среднее значение среди всех result объектов test
     */
    public String getAverageResult() {}
}
```

Класс TestingCenter

```
package com.akaleaf.course;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;
/** Центр тестирования */
public class TestingCenter {
    private User head;
    public User getHead() {}
    /**
     * Добавляет нового пользователя в конец списка или после пользователя с
     * соответствующим lastName, если таковой задан.
     *
     * @param newUser добавляемый пользователь в список
     * @param lastNameAfterWhich lastName пользователя, после которого необходимо
     *                           добавить newUser
     * @return результат выполнения работы: true - успешно
     *                                      false - не успешно
     * @see User
     */
    public boolean addUser(User newUser, String lastNameAfterWhich) {}
    /**
     * Метод определяет есть ли уже в системе пользователь с таким именем.
     *
     * @param lastName имя пользователя
     * @return если пользователь с именем lastName есть - true,
     *         если пользователя с именем lastName нет - false.
     */
    public boolean hasAlreadyExists(String lastName) {}
    /**
     * Удаляет пользователя removableUser из списка
     *
     * @param lastName Используется для нахождения пользователя
     *                 с соответствующим lastName
     * @return результат выполнения работы: true - успешно
```

```
 *                              false - не успешно.
 * @see User
 */
public boolean removeUser(String lastName) {}
/**
 * Служит для получения количества пользователей в системе
 *
 * @return количество пользователей в системе
 */
public int getUsersCount() {}
/**
 * Создаёт или перезаписывает уже созданный файл с именем <fileName>.course,
 * в котором записывается текущее состояние системы в формате, напоминающем
 * XML. Создание(или перезапись) файла ведётся в папку проекта.
 *
 * Более подробно о формате содержимого файла см. документацию loadFromFile()
 *
 * @param fileName используется для имени файла,
 *                 конечный вариант файла выглядит как <fileName>.course
 */
public void saveToFile(String fileName) {}
/**
 * Метод загружает ранее сохранённую или вручную созданную систему в существующую систему.
 *
 * Система из файла может быть соединена с существующей системой или
 * система из файла перезапишет существующую систему.
 *
 * Файл, из которого считывается ранее сохранённая система должен быть формата ".course"
 * и иметь строгую структуру, напоминающая структуру файлов XML.
 *
 * Метод работает проверкой тегов в файле:
 *
 * Теги 0 уровня: testingCenter;
 * Теги 1 уровня: user;
```

```
 * Теги 2 уровня: lastName, testsCount, testsTop, tests;
 * Теги 3 уровня: test;
 * Теги 4 уровня: name, result.
 *
 * Тег testingCenter - открывающий тег не обязателен, закрывающий тег
 *   обязателен - используется для определения конца сканирования(записи).
 * Тег user используется для определения нового пользователя.
 * Теги lastName, testsCount, testsTop, tests, относятся к тегу user и могут быть
 *   только внутри тега user. Используются для опредления строк со свойствами
 *   пользователя.
 * Тег test относится к тегу tests. Используется для определения добавления
 *   нового теста к пользователю.
 * Теги name, result относятся к тегу test. Используются для определения строк
 *   со свойствами теста.
 *
 * Файл сканируется только до первого закрывающего тега testingCenter, все дальнейшие
 * строки файла игнорируются и будут преданы забвению.
 *
 * При дальнейшем разборе кода нижеописанного метода рекомендуется держать
 * под рукой <fileName>.course или какой-либо иной файл .course формата.
 *
 * @param fName    имя файла без приписки формата,
 *                 в конечном варианте имя файла
 *                 будет <fileName>.course
 * @param merge    Объединить уже с существующей
 *                 системой или нет
 *                 (true - объединить)
 *                 (false - не объединять, создаётся
 *                 новая система, старая просто отбрасывается)
 */
public void loadFromFile(String fName, boolean merge) {}
/**
 * Находит пользователя посредством сравнения lastName
 *
```

```
     * @param lastName Использвуется для нахождения пользователя с соответсвующим lastName
     * @return пользователя с соответствующим lastName или null в случае ненахождения
     *         пользователя с соотвутствующим lastName
     * @see User
     */
    public User findMe(String lastName) {}
    /**
     * Находит пользователя посредством сравнения lastName и возвращает его порядковый номер начинающийся
с 1
     *
     * @param index Какой по счёту
     * @return пользователя с соответствующим lastName или null в случае ненахождения
     *         пользователя с соотвутствующим lastName
     * @see User
     */
    public User findMe(int index) {}
    /**
     * Метод отрезает кусок конца строки соответствующий cut, если таковой имеется, от cutting.
     * К примеру cutMe("guy.course", ".course") вернёт "guy".
     *
     * В случае если cut в конце строки cutting не был найден, возвращается неизменённый cutting.
     * К примеру cutMe("guy.course", ".txt") вернёт "guy.course".
     *
     * @param cutting Строка, от которой нужно отрезать cut
     * @param cut Строка, которую нужно отрезать с конца cutting
     * @return cutting
     */
    public String cutMe(String cutting, String cut) {}
}
```

Класс TestingCenter

```
package com.akaleaf.course;
import java.io.*;
import java.util.Arrays;
```

```
import java.util.Scanner;
/** Центр тестирования */
public class TestingCenter {
    private User head;
    public User getHead() {}
    /**
     * Добавляет нового пользователя в конец списка или после пользователя с
     * соответствующим lastName, если таковой задан.
     *
     * @param newUser добавляемый пользователь в список
     * @param lastNameAfterWhich lastName пользователя, после которого необходимо
     *                           добавить newUser
     * @return результат выполнения работы: true - успешно
     *                                      false - не успешно
     * @see User
     */
    public boolean addUser(User newUser, String lastNameAfterWhich) {}
    /**
     * Метод определяет есть ли уже в системе пользователь с таким именем.
     *
     * @param lastName имя пользователя
     * @return если пользователь с именем lastName есть - true,
     *         если пользователя с именем lastName нет - false.
     */
    public boolean hasAlreadyExists(String lastName) {}
    /**
     * Удаляет пользователя removableUser из списка
     *
     * @param lastName Используется для нахождения пользователя
     *                 с соответствующим lastName
     * @return результат выполнения работы: true - успешно
     *                                      false - не успешно.
     * @see User
     */
```

```
public boolean removeUser(String lastName) {}
/**
 * Служит для получения количества пользователей в системе
 *
 * @return количество пользователей в системе
 */
public int getUsersCount() {}
/**
 * Создаёт или перезаписывает уже созданный файл с именем <fileName>.course,
 * в котором записывается текущее состояние системы в формате, напоминающем
 * XML. Создание(или перезапись) файла ведётся в папку проекта.
 *
 * Более подробно о формате содержимого файла см. документацию loadFromFile()
 *
 * @param fileName используется для имени файла,
 *                 конечный вариант файла выглядит как <fileName>.course
 */
public void saveToFile(String fileName) {}
/**
 * Метод загружает ранее сохранённую или вручную созданную систему в существующую систему.
 *
 * Система из файла может быть соединена с существующей системой или
 * система из файла перезапишет существующую систему.
 *
 * Файл, из которого считывается ранее сохранённая система должен быть формата ".course"
 * и иметь строгую структуру, напоминающая структуру файлов XML.
 *
 * Метод работает проверкой тегов в файле:
 *
 * Теги 0 уровня: testingCenter;
 * Теги 1 уровня: user;
 * Теги 2 уровня: lastName, testsCount, testsTop, tests;
 * Теги 3 уровня: test;
 * Теги 4 уровня: name, result.
```

```
*
* Тег testingCenter - открывающий тег не обязателен, закрывающий тег
*   обязателен - используется для определения конца сканирования(записи).
* Тег user используется для определения нового пользователя.
* Теги lastName, testsCount, testsTop, tests, относятся к тегу user и могут быть
*   только внутри тега user. Используются для опредления строк со свойствами
*   пользователя.
* Тег test относится к тегу tests. Используется для определения добавления
*   нового теста к пользователю.
* Теги name, result относятся к тегу test. Используются для определения строк
*   со свойствами теста.
*
* Файл сканируется только до первого закрывающего тега testingCenter, все дальнейшие
* строки файла игнорируются и будут преданы забвению.
*
* При дальнейшем разборе кода нижеописанного метода рекомендуется держать
* под рукой <fileName>.course или какой-либо иной файл .course формата.
*
* @param fName    имя файла без приписки формата,
*                 в конечном варианте имя файла
*                 будет <fileName>.course
* @param merge    Объединить уже с существующей
*                 системой или нет
*                 (true - объединить)
*                 (false - не объединять, создаётся
*                 новая система, старая просто отбрасывается)
*/
public void loadFromFile(String fName, boolean merge) {}
/**
 * Находит пользователя посредством сравнения lastName
 *
 * @param lastName Использвуется для нахождения пользователя с соответсвующим lastName
 * @return пользователя с соответствующим lastName или null в случае ненахождения
 *         пользователя с соотвутствующим lastName
```

```
     * @see User
     */
    public User findMe(String lastName) {}
    /**
     * Находит пользователя посредством сравнения lastName и возвращает его порядковый номер начинающийся
с 1
     *
     * @param index Какой по счёту
     * @return пользователя с соответствующим lastName или null в случае ненахождения
     *         пользователя с соотвутствующим lastName
     * @see User
     */
    public User findMe(int index) {}
    /**
     * Метод отрезает кусок конца строки соответствующий cut, если таковой имеется, от cutting.
     * К примеру cutMe("guy.course", ".course") вернёт "guy".
     *
     * В случае если cut в конце строки cutting не был найден, возвращается неизменённый cutting.
     * К примеру cutMe("guy.course", ".txt") вернёт "guy.course".
     *
     * @param cutting Строка, от которой нужно отрезать cut
     * @param cut Строка, которую нужно отрезать с конца cutting
     * @return cutting
     */
    public String cutMe(String cutting, String cut) {}
}
```

Класс TestingCenterGUI

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.akaleaf.course;
```

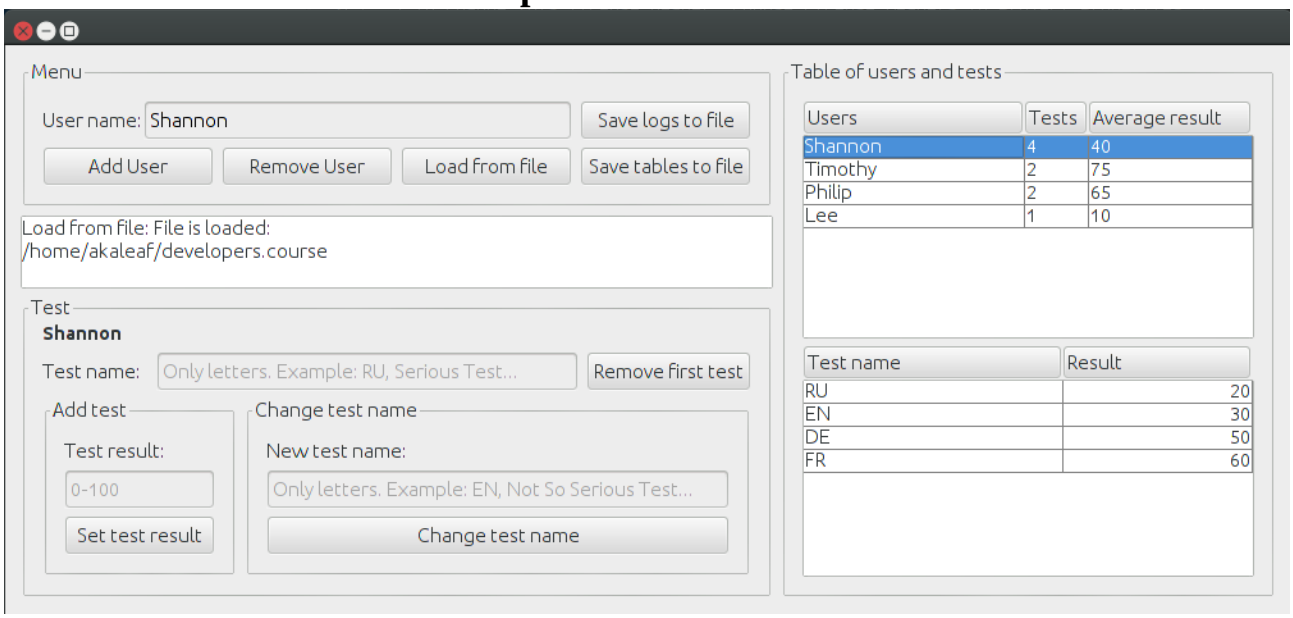
```
/**
 *
 * @author akaleaf
 */
import java.awt.Color;
import java.awt.TextField;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.*;
import javax.swing.table.*;
import javax.swing.text.AttributeSet;
import javax.swing.text.BadLocationException;
import javax.swing.text.DocumentFilter;
public class TestingCenterGUI extends javax.swing.JFrame {
    public TestingCenter testingCenter = new TestingCenter();
    /**
     * Creates new form TestingCenterUI
     */
    public TestingCenterGUI() {}
    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    private void initComponents() {}
    private void removeUserButtonActionPerformed(java.awt.event.ActionEvent evt) {}
    private void addUserButtonActionPerformed(java.awt.event.ActionEvent evt) {}
    private void updateTestsTable() {}
    private void updateUsersTable() {}
    private void addTestButtonActionPerformed(java.awt.event.ActionEvent evt) {}
    private void removeTestButtonActionPerformed(java.awt.event.ActionEvent evt) {}
    private void saveTablesToFileButtonActionPerformed(java.awt.event.ActionEvent evt) {}
    private void loadFromFileButtonActionPerformed(java.awt.event.ActionEvent evt) {}
```

```
private void changeTestNameButtonActionPerformed(java.awt.event.ActionEvent evt) {}
private void userNameTextFieldKeyPressed(java.awt.event.KeyEvent evt) {}
public boolean isOnlyAlphabetic(String string) {}
public boolean isOnlyNumeric(String string) {}
/**
 *
 * Выбор пользователя через таблицу пользователей левой кнопкой мыши
 *
 * @param evt
 */
private void usersTableMouseClicked(java.awt.event.MouseEvent evt) {}
private void saveLogsToFileButtonActionPerformed(java.awt.event.ActionEvent evt) {}
private void testsTableMouseClicked(java.awt.event.MouseEvent evt) {}
private void testNameTextFieldKeyPressed(java.awt.event.KeyEvent evt) {}
private void testResultTextFieldKeyPressed(java.awt.event.KeyEvent evt) {}
private void newTestNameTextFieldKeyPressed(java.awt.event.KeyEvent evt) {}
private void testResultTextFieldFocusGained(java.awt.event.FocusEvent evt) {}
private void testResultTextFieldFocusLost(java.awt.event.FocusEvent evt) {}
private void userNameTextFieldFocusGained(java.awt.event.FocusEvent evt) {}
private void userNameTextFieldFocusLost(java.awt.event.FocusEvent evt) {}
private void testNameTextFieldFocusGained(java.awt.event.FocusEvent evt) {}
private void testNameTextFieldFocusLost(java.awt.event.FocusEvent evt) {}
private void newTestNameTextFieldFocusGained(java.awt.event.FocusEvent evt) {}
private void newTestNameTextFieldFocusLost(java.awt.event.FocusEvent evt) {}
/**
 * @param args the command line arguments
 */
public static void main(String args[]) {}
// Variables declaration - do not modify
private javax.swing.JButton addTestButton;
private javax.swing.JPanel addTestPanel;
private javax.swing.JButton addUserButton;
private javax.swing.JButton changeTestNameButton;
private javax.swing.JPanel changeTestNamePanel;
```

```
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JButton loadFromFileButton;
    private javax.swing.JTextArea logs;
    private javax.swing.JPanel menuPanel;
    private javax.swing.JLabel newNameLabel;
    private javax.swing.JLabel newTestNameLabel;
    private javax.swing.JTextField newTestNameTextField;
    private javax.swing.JButton removeTestButton;
    private javax.swing.JButton removeUserButton;
    private javax.swing.JButton saveLogsToFileButton;
    private javax.swing.JButton saveTablesToFileButton;
    private javax.swing.JLabel selectedUserLabel;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JTextField testNameTextField;
    private javax.swing.JPanel testPanel;
    private javax.swing.JLabel testResultLabel;
    private javax.swing.JTextField testResultTextField;
    private javax.swing.JTable testsTable;
    private javax.swing.JLabel userNameLabel;
    private javax.swing.JTextField userNameTextField;
    private javax.swing.JTable usersTable;
    // End of variables declaration
}
```

Конец списка формализованного описания разработанных классов

5. Описание демонстрационного модуля с характеристикой использованных стандартных компонентов и списком реализованных обработчиков



Внешний вид программы

Слева направо, сверху вниз: добавление пользователя(Add User), удаление пользователя(Remove User), загрузка системы в отдельном файле(Load from file), сохранение системы в отдельном файле(Save tables to file), удаление первого теста(Remove first test), добавление теста или изменение результата уже существующего теста(Set test result), изменение названия существующего теста(Change test name). Также на форме дополнительно присутствует поле для ведения записей действий пользователя и функция сохранения записей действий пользователя в отдельный файл(Save logs to file).

Первая из двух таблиц имеет три столбца: первый столбец предназначен для отображения имени пользователя, второй столбец – количества тестов, третий столбец – среднего результата среди всех тестов пользователя. Вторая таблица имеет два столбца: первый столбец предназначен для отображения названия теста, второй – результата.

Добавление пользователя

Для добавления пользователя необходимо ввести его имя в поле “User name” и нажать кнопку “Add User”.

Примечание: в имени пользователя допустимы только пробел и буквы латиницы, кириллицы.

Удаление пользователя

Для удаления пользователя необходимо ввести его имя в поле “User name” и нажать кнопку “Remove User”.

Сохранение системы в файл

Для сохранения системы в отдельный файл:

1) Нажмите кнопку “Save tables to file”. Появится окно.
2) В открывшемся окне выберите папку для сохранения файла, если необходимо
3) Введите имя файла.
4) Нажмите кнопку “Save”

Загрузка системы из файла

Для загрузки системы из файла:

1) Нажмите кнопку “Load from file”. Появится окно.
2) В открывшемся окне выберите папку из которой нужно выбрать файл, если необходимо.
3) Выберите файл, нажав на него один раз.
4) Нажмите кнопку “Load”.

Удаление первого теста

Для удаления первого теста вам нужно ввести имя пользователя, у которого нужно удалить тест в поле “User name” и нажать кнопку “Remove first test”.

Добавление теста

Для того чтобы добавить тест:

1) Если пользователь, которому нужно добавить тест, не выбран, то выберите его, нажав на его имя в таблице пользователей.
2) Введите в поле “Test name” название теста, который нужно добавить

Примечание: название теста недопустимо, если такое название теста уже есть у пользователя

3) Введите в поле “Test result” результат теста по 100-бальной шкале.
4) Нажмите кнопку “Set test result”.
5) В таблице тестов должен появится новый тест.

Изменение результата теста

Для того чтобы изменить результат теста:

1) Если пользователь, которому нужно изменить результат теста не выбран, то выберите его, нажав на его имя в таблице пользователей.
2) Если тест, которому нужно изменить результат не выбран, то выберите его, нажав на его название в таблице тестов. Или введите его название вручную в поле “Test name”.
3) Введите новое значение результата теста по 100-бальной шкале.
4) Нажмите кнопку “Set test result”.
5) В таблице тестов результат теста должен измениться на число введённое в поле “Test name”.

Изменение названия теста

Для того чтобы изменить название теста:

1) Если пользователь, которому нужно изменить название теста не выбран, то выберите его, нажав на его имя в таблице пользователей.
2) Если тест, которому нужно изменить название не выбран, то выберите его, нажав на его название в таблице тестов. Или введите его название вручную в поле “Test name”.
3) Введите новое название в поле “New test name”.
4) Нажмите кнопку “Change test name”.
5) В таблице тестов название теста должно измениться на название, введённое в поле “New test name”.

Описание структуры проекта в соответствии с использованным инструментом разработки

При написании прорграммы были использованы такие программы, как IntelliJ IDEA, NetBeans 8.2, gedit, JVM, OpenJDK 1.8.2_222.

Проект содержит 5 необходимых файлов: TestingCenterGUI.java, TestingCenterGUI.form, TestingCenter.java, User.java, Test.java. Первые два файла отвечают за графический интерфейс. Проект компилируется в один исполняемый на JVM файл TestingCenter.jar.

7. Список литературы

1. Вирт Н.. Алгоритмы и структуры данных. – М.:ДМК, 2017 г.
2. Вирт Н. Алгоритмы и структуры обработки данных [Электронный ресурс]/Профобразование, 2017 г.
3. Буч Г. и др. Объектно-ориентированный анализ и проектирование с примерами применения. – М .:Вильямс, 2017г.
4. Васильев А.Н. Java. Объектно-ориентированное программирование. Учебное пособие. Стандарт 3-го поколения. – СПб.:2012 г.
5. Кнут Д. Искусство программирования, том 1. Основные алгоритмы. – М.: Вильямс, 2014 г.
6. Кормен Т. Алгоритмы. Вводный курс. – М.: Вильямс, 2016 г.
7. Медведев Д.М. Структуры и алгоритмы обработки данных в системах автоматизации и управления [Электронный ресурс]: учебное пособие/ Ай Пи Эр Медиа, 2018 г.
8. Курапова Е.В. Структуры и алгоритмы обработки данных [Электронный ресурс]: лабораторный практикум/ Сибирский государственный университет телекоммуникаций и информатики, 2015 г.
9. Орлов С.А. Теория и практика языков программирования. Учебник для вузов. Стандарт третьего поколения. СПб.:Питер, 2017г.
10. Стивенс Р. Алгоритмы. Теория и практическое применение. - М.: Издательство «Э», 2016 г.
11. www.tisbi.ru – Электронная библиотека Университета управления«ТИСБИ
12. www.iprbookshop.ru – Электронно – библиотечная система «Университетская библиотека онлайн»

8. Полный листинг программы

Класс Test

```
package com.akaleaf.course;
/** Класс Тестов */
public class Test {
    private String name;
    private int    result;
    public Test(String name, int result) {
        this.name = name;
        this.result = result;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setResult(int result) {
        this.result = result;
    }
    public String getName() {
        return name;
    }
    public int getResult() {
        return result;
    }
}
```

Класс User

```
package com.akaleaf.course;
import java.util.Arrays;
/** Класс пользователей */
public class User {
    private String lastName;
    private User next;
    private Test tests[];
```

```
private int testsTop = -1;
public User() {}
/**
 * Конструктор объектов User.
 *
 * @param lastName    Используется для задания имени для нового объекта.
 * @param testsLength Используется для задания размера массива, в котором хранятся тесты.
 */
public User(String lastName, int testsLength) {
    this.lastName = lastName;
    tests = new Test[testsLength];
}
public void setLastName(String lastName) {
    this.lastName = lastName;
}
/**
 * Ищет test в массиве tests по совпадению свойства name и
 * присваивает свойству result объекта test
 * новое значение result.
 *
 * @param name   Используется для нахождения test
 * @param result Новое значение для свойства result у test
 */
public void setTestResult(String name, int result) {
    int i = 0;
    while (!(tests[i].getName().equals(name))) {
        i++;
    }
    tests[i].setResult(result);
}
/**
 * Ищет test в массиве tests по совпадению свойства name и
 * присваивает свойству name объекта test
 * новое значение newName.
```

```
     *
     * @param oldName Используется для нахождения test
     * @param newName Новое значение для свойства name у test
     */
    public void setTestName(String oldName, String newName) {
        int i = 0;
        while (i == tests.length && !(tests[i].getName().equals(oldName))) {
            i++;
        }
        for (i = 0; i != tests.length; i++) {
            if (tests[i] != null && tests[i].getName().equals(oldName))  {
                tests[i].setName(newName);
                break;
            }
        }
    }
    /**
     * Удаляет первый(т.к. очередь) test из массива tests
     *
     * @return результат удаления: true - успешно
     *                             false - неудачно.
     */
    public boolean removeTest() {
        if (tests[0] != null) {
            for (int index = 0; index != testsTop + 1; index++) {
                tests[index] = tests[index + 1];
            }
            testsTop--;
            return true;
        } else {
            return false;
        }
    }
    /**
```

```
 * Удаляет test с совпадающим lastName из массива tests
 *
 * @param testName Удаляемый тест
 * @return результат удаления: true - успешно
 *                             false - неудачно.
 */
public boolean removeTest(String testName) {
    if (isTestAlreadyExists(testName)) {
        int iterator = 0;
        while (tests[iterator] != null && !tests[iterator].getName().equals(testName)) {
            iterator++;
        }
        for (iterator = iterator; iterator != testsTop + 1; iterator++) {
            tests[iterator] = tests[iterator + 1];
        }
        testsTop--;
        return true;
    } else {
        return false;
    }
}
/**
 * Добавляет новый test в конец массива tests
 *
 * @param name   Используется для задания свойства name новому test
 * @param result Используется для задания свойства result новому test
 */
public void addTest(String name, int result) {
    if (!isTestAlreadyExists(name)) {
        testsTop++;
        tests[testsTop] = new Test(name, result);
    }
    /*
    Увеличение размера массива, если остаётся мало места.
```

```
     */
    if (testsTop == tests.length - 1) {
        tests = Arrays.copyOf(tests, tests.length * 2);
    }
}
public boolean isTestAlreadyExists(String testName) {
    for (int i = 0; i != testsTop + 1; i++) {
        if (tests[i].getName().equals(testName)) {
            return true;
        }
    }
    return false;
}
public User getNext() {
    return next;
}
public void setNext(User newNext) {
    this.next = newNext;
}
public String getLastName() {
    return lastName;
}
public int getTestsTop() {
    return testsTop;
}
public Test[] getTests() {
    return tests;
}
/**
 * Используется для получения среднего значения среди всех result объектов test
 *
 * @return среднее значение среди всех result объектов test
 */
public String getAverageResult() {
```

```
        if (testsTop != -1) {
            int value = 0;
            for (int index = 0; index != (testsTop + 1); index++) {
                value += tests[index].getResult();
            }
            return Integer.toString(value / (testsTop + 1));
        } else {
            return null;
        }
    }
}
```

Класс TestingCenter

```
package com.akaleaf.course;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;
/** Центр тестирования */
public class TestingCenter {
    private User head;
    public User getHead() {
        return head;
    }
    /**
     * Добавляет нового пользователя в конец списка или после пользователя с
     * соответствующим lastName, если таковой задан.
     *
     * @param newUser добавляемый пользователь в список
     * @param lastNameAfterWhich lastName пользователя, после которого необходимо
     *                           добавить newUser
     * @return результат выполнения работы: true - успешно
     *                                      false - не успешно
     * @see User
     */
```

```
public boolean addUser(User newUser, String lastNameAfterWhich) {
    if (head == null) {
        head = newUser;
        return true;
    } else {
        if (!hasAlreadyExists(newUser.getLastName())) {
            User helpUser = head;
            if (lastNameAfterWhich.equals("")) {
                while (helpUser.getNext() != null) {
                    helpUser = helpUser.getNext();
                }
                helpUser.setNext(newUser);
                return true;
            } else {
                User workUser = findMe(lastNameAfterWhich);
                if (workUser != null) {
                    newUser.setNext(workUser.getNext());
                    workUser.setNext(newUser);
                    return true;
                }
                return false;
            }
        } else {
            return false;
        }
    }
}
/**
 * Метод определяет есть ли уже в системе пользователь с таким именем.
 *
 * @param lastName имя пользователя
 * @return если пользователь с именем lastName есть - true,
 *         если пользователя с именем lastName нет - false.
 */
```

```
public boolean hasAlreadyExists(String lastName) {
    return (findMe(lastName) != null);
}
/**
 * Удаляет пользователя removableUser из списка
 *
 * @param lastName Используется для нахождения пользователя
 *                 с соответствующим lastName
 * @return результат выполнения работы: true - успешно
 *                                      false - не успешно.
 * @see User
 */
public boolean removeUser(String lastName) {
    User helpUser = head;
    if (findMe(lastName) != null) {
        if (lastName.equals(head.getLastName())) {
            head = head.getNext();
            return true;
        }
        while ((helpUser.getNext() != null) && !(helpUser.getNext().getLastName().equals(lastName))) {
            helpUser = helpUser.getNext();
        }
        if (helpUser.getNext().getNext() != null) {
            helpUser.setNext(helpUser.getNext().getNext());
            return true;
        } else {
            helpUser.setNext(null);
            return true;
        }
    } else {
        return false;
    }
}
/**
```

```
 * Служит для получения количества пользователей в системе
 *
 * @return количество пользователей в системе
 */
public int getUsersCount() {
    User helpUser = head;
    int usersCount = 0;
    while (helpUser != null) {
        helpUser = helpUser.getNext();
        usersCount++;
    }
    return usersCount;
}
/**
 * Создаёт или перезаписывает уже созданный файл с именем <fileName>.course,
 * в котором записывается текущее состояние системы в формате, напоминающем
 * XML. Создание(или перезапись) файла ведётся в папку проекта.
 *
 * Более подробно о формате содержимого файла см. документацию loadFromFile()
 *
 * @param fileName используется для имени файла,
 *                 конечный вариант файла выглядит как <fileName>.course
 */
public void saveToFile(String fileName) {
    String fName = fileName + ".course";
    try {
        FileWriter file = new FileWriter(fName);
        String indent = "    ";
        User helpUser = head;
        file.write("<testingCenter>");
        file.append("\n");
        while (helpUser != null) {
            file.write(indent + "<user>");
            file.append("\n");
```

```
file.write(indent + indent + "<lastName>");
file.append("\n");
file.write(indent + indent + indent + helpUser.getLastName());
file.append("\n");
file.write(indent + indent + "</lastName>");
file.append("\n");
file.write(indent + indent + "<testsTop>");
file.append("\n");
file.write(indent + indent + indent + helpUser.getTestsTop());
file.append("\n");
file.write(indent + indent + "</testsTop>");
file.append("\n");
if (helpUser.getTestsTop() != -1) {
    file.write(indent + indent + "<tests>");
    file.append("\n");
}
for (int i = 0; i != helpUser.getTestsTop() + 1; i++) {
    file.write(indent + indent + indent + "<test>");
    file.append("\n");
    file.write(indent + indent + indent + indent + "<name>");
    file.append("\n");
    file.write(indent + indent + indent + indent + indent + helpUser.getTests()[i].getName());
    file.append("\n");
    file.write(indent + indent + indent + indent + "</name>");
    file.append("\n");
    file.write(indent + indent + indent + indent + "<result>");
    file.append("\n");
    file.write(indent + indent + indent + indent + indent + helpUser.getTests()[i].getResult());
    file.append("\n");
    file.write(indent + indent + indent + indent + "</result>");
    file.append("\n");
    file.write(indent + indent + indent + "</test>");
    file.append("\n");
}
```

```
            if (helpUser.getTestsTop() != -1) {
                file.write(indent + indent + "</tests>");
                file.append("\n");
            }
            file.write(indent + "</user>");
            file.append("\n");
            helpUser = helpUser.getNext();
        }
        file.write("</testingCenter>");
        file.flush();
        file.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
/**
 * Метод загружает ранее сохранённую или вручную созданную систему в существующую систему.
 *
 * Система из файла может быть соединена с существующей системой или
 * система из файла перезапишет существующую систему.
 *
 * Файл, из которого считывается ранее сохранённая система должен быть формата ".course"
 * и иметь строгую структуру, напоминающая структуру файлов XML.
 *
 * Метод работает проверкой тегов в файле:
 *
 * Теги 0 уровня: testingCenter;
 * Теги 1 уровня: user;
 * Теги 2 уровня: lastName, testsCount, testsTop, tests;
 * Теги 3 уровня: test;
 * Теги 4 уровня: name, result.
 *
 * Тег testingCenter - открывающий тег не обязателен, закрывающий тег
 *   обязателен - используется для определения конца сканирования(записи).
```

```
* Тег user используется для определения нового пользователя.
* Теги lastName, testsCount, testsTop, tests, относятся к тегу user и могут быть
*   только внутри тега user. Используются для опредления строк со свойствами
*   пользователя.
* Тег test относится к тегу tests. Используется для определения добавления
*   нового теста к пользователю.
* Теги name, result относятся к тегу test. Используются для определения строк
*   со свойствами теста.
*
* Файл сканируется только до первого закрывающего тега testingCenter, все дальнейшие
* строки файла игнорируются и будут преданы забвению.
*
* При дальнейшем разборе кода нижеописанного метода рекомендуется держать
* под рукой <fileName>.course или какой-либо иной файл .course формата.
*
* @param fName    имя файла без приписки формата,
*                 в конечном варианте имя файла
*                 будет <fileName>.course
* @param merge    Объединить уже с существующей
*                 системой или нет
*                 (true - объединить)
*                 (false - не объединять, создаётся
*                 новая система, старая просто отбрасывается)
*/
public void loadFromFile(String fName, boolean merge) {
    /*
    Отброс старой системы в случае merge == false.
     */
    if (!merge) {
        head = null;
    }
    /*
    Создание объекта для работы с файлом.
     */
```

```
try {
    FileReader file = new FileReader(fName);
    FileReader file1 = new FileReader(fName);
    /*
    Следующие 2 объявления и цикл используются для счёта количества значащих строк в файле.
    Далее информация передаётся при инициализации служебного массива fileArray.
     */
    Scanner scanForCount = new Scanner(file1);
    int countOfStrokes = 1;
    while (scanForCount.hasNextLine() && !(scanForCount.nextLine().equals("</testingCenter>"))) {
        countOfStrokes++;
    }
    Scanner scan = new Scanner(file);
    int index = 0;
    /*
    Служебный массив - используется для более удобной работы с содержимым файла.
     */
    String fileArray[] = new String[countOfStrokes];
    /*
    Считывание строк файла в служебный массив fileArray.
     */
    while (scan.hasNextLine()) {
        /*
        При первой встрече </testingCenter> - остановка сканирования файла.
         */
        fileArray[index] = scan.nextLine();
        if ((index != 0) && (fileArray[index - 1].equals("</testingCenter>"))) {
            break;
        }
        index++;
    }
    /*
    Сокращение служебного массива до значащих элементов:
    убираются null элементы массива, начиная с конца до появления первого не null элемента.
```

```
 */
fileArray = Arrays.copyOf(fileArray, index);
int i = 0;
String indent = "    ";
/*
Цикл прохода по каждому элементу служебного массива (элементы - ранее строки в файле).
 */
while (fileArray.length != i) {
    /*
    Если в служебном массиве встречается тег 1 уровня user.
     */
    if (fileArray[i].equals(indent + "<user>")) {
        String lastName = "";
        int testsCount = 0;
        int i1 = i;
        /*
        Создание и добавление нового пользователя в систему.
         */
        User newUser = new User(lastName, 10);
        addUser(newUser, "");
        /*
        Просмотр содержимого <user>..</user>.
         */
        while (!(fileArray[i1].equals(indent + "</user>"))) {
            /*
            При встрече тега lastName.
             */
            if (fileArray[i1].equals(indent + indent + "<lastName>")) {
                int ind = 12;
                String newLastName = "";
                while (ind != fileArray[i1 + 1].length()) {
                    newLastName += fileArray[i1 + 1].charAt(ind);
                    ind++;
                }
```

```
    /*
    Изменение свойства lastName ранее добавленного пользователя на
    содержимое <lastName>..</lastName>.
     */
    findMe(lastName).setLastName(newLastName);
    lastName = newLastName;
}
/*
При встрече тега tests.
 */
if (fileArray[i1].equals(indent + indent + "<tests>")) {
    int i2 = i1;
    /*
    Просмотр содержимого <tests>..</tests>.
     */
    while (!(fileArray[i2].equals(indent + indent + "</tests>"))) {
        /*
        При встрече тега test.
         */
        if (fileArray[i2].equals(indent + indent + indent + "<test>")) {
            int i3 = i2;
            /*
            Добавление теста к ранее добавленному пользователю.
             */
            findMe(lastName).addTest("", 0);
            String testName = "";
            /*
            Просмотр содержимого <test>..</tests>.
             */
            while (!(fileArray[i3].equals(indent + indent + indent + "</test>"))) {
                /*
                При встрече тега name.
                 */
                if (fileArray[i3].equals(indent + indent + indent + indent + "<name>")) {
```

```
                int ind = 20;
                String newName = "";
                while (ind != fileArray[i3 + 1].length()) {
                    newName += fileArray[i3 + 1].charAt(ind);
                    ind++;
                }
                /*
                Изменение свойства name ранее добавленного теста на
                содержимое <name>..</name>.
                 */
                findMe(lastName).setTestName("", newName);
                testName = newName;
            }
            /*
            При встрече тега result.
             */
            if (fileArray[i3].equals(indent + indent + indent + indent + "<result>")) {
                int ind = 20;
                String newResult = "";
                while (ind != fileArray[i3 + 1].length()) {
                    newResult += fileArray[i3 + 1].charAt(ind);
                    ind++;
                }
                /*
                Изменение свойства result ранее добавленного теста на
                содержимое <result>..</result>.
                 */
                findMe(lastName).setTestResult(testName, Integer.valueOf(newResult));
            }
            i3++;
        }
    }
    i2++;
}
```

```
                }
                i1++;
            }
        }
        i++;
    }
    /*
    Приехали.
    Одинокий catch.
     */
  } catch (IOException e) {
    e.printStackTrace();
  }
}
/**
 * Находит пользователя посредством сравнения lastName
 *
 * @param lastName Использвуется для нахождения пользователя с соответсвующим lastName
 * @return пользователя с соответствующим lastName или null в случае ненахождения
 *         пользователя с соотвутствующим lastName
 * @see User
 */
public User findMe(String lastName) {
  /*
  Вспомогательная переменная
   */
  User helpUser = head;
  if (helpUser != null) {
    while ((helpUser.getNext() != null) && !(helpUser.getLastName().equals(lastName))) {
      helpUser = helpUser.getNext();
    }
    if (helpUser.getLastName().equals(lastName)) {
      return helpUser;
    } else {
```

```
            return null;
        }
    } else {
        return null;
    }
}
/**
 * Находит пользователя посредством сравнения lastName и возвращает его порядковый номер начинающийся с 1
 *
 * @param index Какой по счёту
 * @return пользователя с соответствующим lastName или null в случае ненахождения
 *         пользователя с соотвутствующим lastName
 * @see User
 */
public User findMe(int index) {
    int iter = 1;
    User helpUser = head;
    while (iter != index) {
        helpUser = helpUser.getNext();
        iter++;
    }
    return helpUser;
}
/**
 * Метод отрезает кусок конца строки соответствующий cut, если таковой имеется, от cutting.
 * К примеру cutMe("guy.course", ".course") вернёт "guy".
 *
 * В случае если cut в конце строки cutting не был найден, возвращается неизменённый cutting.
 * К примеру cutMe("guy.course", ".txt") вернёт "guy.course".
 *
 * @param cutting Строка, от которой нужно отрезать cut
 * @param cut Строка, которую нужно отрезать с конца cutting
 * @return cutting
```

```
     */
    public String cutMe(String cutting, String cut) {
        if (cutting.length() > cut.length() + 1) {
            String cuttable = "";
            for (int i = cutting.length() - cut.length(); i != cutting.length(); i++) {
                cuttable += cutting.charAt(i);
            }
            String newFileName = "";
            if (cuttable.equals(cut)) {
                for (int i = 0; i != cutting.length() - cut.length(); i++) {
                    newFileName += cutting.charAt(i);
                }
                cutting = newFileName;
            }
        }
        return cutting;
    }
}
```

Класс TestingCenterGUI

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.akaleaf.course;
/**
 *
 * @author akaleaf
 */
import java.awt.Color;
import java.awt.TextField;
import java.io.File;
import java.io.FileWriter;
```

```
import java.io.IOException;
import javax.swing.*;
import javax.swing.table.*;
import javax.swing.text.AttributeSet;
import javax.swing.text.BadLocationException;
import javax.swing.text.DocumentFilter;
public class TestingCenterGUI extends javax.swing.JFrame {
    public TestingCenter testingCenter = new TestingCenter();
    /**
     * Creates new form TestingCenterUI
     */
    public TestingCenterGUI() {
        initComponents();
        setLocationRelativeTo(null);
    }
    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        tablePanel = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        testsTable = new javax.swing.JTable();
        jScrollPane2 = new javax.swing.JScrollPane();
        usersTable = new javax.swing.JTable();
        menuPanel = new javax.swing.JPanel();
        addUserButton = new javax.swing.JButton();
        removeUserButton = new javax.swing.JButton();
        userNameTextField = new javax.swing.JTextField();
        userNameLabel = new javax.swing.JLabel();
        saveTablesToFileButton = new javax.swing.JButton();
```

```
loadFromFileButton = new javax.swing.JButton();
saveLogsToFileButton = new javax.swing.JButton();
jScrollPane3 = new javax.swing.JScrollPane();
logs = new javax.swing.JTextArea();
testPanel = new javax.swing.JPanel();
addTestPanel = new javax.swing.JPanel();
addTestButton = new javax.swing.JButton();
testResultLabel = new javax.swing.JLabel();
testResultTextField = new javax.swing.JTextField();
newTestNameLabel = new javax.swing.JLabel();
testNameTextField = new javax.swing.JTextField();
removeTestButton = new javax.swing.JButton();
changeTestNamePanel = new javax.swing.JPanel();
newNameLabel = new javax.swing.JLabel();
newTestNameTextField = new javax.swing.JTextField();
changeTestNameButton = new javax.swing.JButton();
selectedUserLabel = new javax.swing.JLabel();
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
tablePanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Table of users and tests"));
testsTable.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
    },
    new String [] {
        "Test name", "Result"
    }
) {
    Class[] types = new Class [] {
        java.lang.Object.class, java.lang.Integer.class
    };
    boolean[] canEdit = new boolean [] {
        false, false
    };
    public Class getColumnClass(int columnIndex) {
        return types [columnIndex];
```

```
        }
        public boolean isCellEditable(int rowIndex, int columnIndex) {
            return canEdit [columnIndex];
        }
    });
    testsTable.setColumnSelectionAllowed(true);
    testsTable.getTableHeader().setReorderingAllowed(false);
    testsTable.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            testsTableMouseClicked(evt);
        }
    });
    testsTable.addInputMethodListener(new java.awt.event.InputMethodListener() {
        public void inputMethodTextChanged(java.awt.event.InputMethodEvent evt) {
            testsTableInputMethodTextChanged(evt);
        }
        public void caretPositionChanged(java.awt.event.InputMethodEvent evt) {
        }
    });
    jScrollPane1.setViewportView(testsTable);

testsTable.getColumnModel().getSelectionModel().setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELE
CTION);
    if (testsTable.getColumnModel().getColumnCount() > 0) {
        testsTable.getColumnModel().getColumn(1).setPreferredWidth(20);
    }
    usersTable.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {
        },
        new String [] {
            "Users", "Tests", "Average result"
        }
    ) {
        boolean[] canEdit = new boolean [] {
```

```
            false, false, false
        };
        public boolean isCellEditable(int rowIndex, int columnIndex) {
            return canEdit [columnIndex];
        }
    });
    usersTable.getTableHeader().setReorderingAllowed(false);
    usersTable.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            usersTableMouseClicked(evt);
        }
    });
    jScrollPane2.setViewportView(usersTable);
    if (usersTable.getColumnModel().getColumnCount() > 0) {
        usersTable.getColumnModel().getColumn(1).setMinWidth(50);
        usersTable.getColumnModel().getColumn(1).setPreferredWidth(50);
        usersTable.getColumnModel().getColumn(1).setMaxWidth(50);
        usersTable.getColumnModel().getColumn(2).setPreferredWidth(30);
    }
    javax.swing.GroupLayout tablePanelLayout = new javax.swing.GroupLayout(tablePanel);
    tablePanel.setLayout(tablePanelLayout);
    tablePanelLayout.setHorizontalGroup(
        tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(tablePanelLayout.createSequentialGroup()
            .addContainerGap()
            .addGroup(tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 353,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 353,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );
    tablePanelLayout.setVerticalGroup(
        tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, tablePanelLayout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 185,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)
            .addContainerGap())
    );
    menuPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Menu"));
    addUserButton.setText("Add User");
    addUserButton.setName(""); // NOI18N
    addUserButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            addUserButtonActionPerformed(evt);
        }
    });
    removeUserButton.setText("Remove User");
    removeUserButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            removeUserButtonActionPerformed(evt);
        }
    });
    userNameTextField.addFocusListener(new java.awt.event.FocusAdapter() {
        public void focusGained(java.awt.event.FocusEvent evt) {
            userNameTextFieldFocusGained(evt);
        }
        public void focusLost(java.awt.event.FocusEvent evt) {
            userNameTextFieldFocusLost(evt);
        }
    });
    userNameTextField.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            userNameTextFieldActionPerformed(evt);
        }
```

```
});
userNameTextField.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        userNameTextFieldKeyPressed(evt);
    }
});
userNameLabel.setText("User name:");
saveTablesToFileButton.setText("Save tables to file");
saveTablesToFileButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        saveTablesToFileButtonActionPerformed(evt);
    }
});
loadFromFileButton.setText("Load from file");
loadFromFileButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        loadFromFileButtonActionPerformed(evt);
    }
});
saveLogsToFileButton.setText("Save logs to file");
saveLogsToFileButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        saveLogsToFileButtonActionPerformed(evt);
    }
});
javax.swing.GroupLayout menuPanelLayout = new javax.swing.GroupLayout(menuPanel);
menuPanel.setLayout(menuPanelLayout);
menuPanelLayout.setHorizontalGroup(
    menuPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(menuPanelLayout.createSequentialGroup()
        .addContainerGap()
        .addGroup(menuPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(menuPanelLayout.createSequentialGroup()
                .addComponent(addUserButton)
```

```
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(removeUserButton, javax.swing.GroupLayout.PREFERRED_SIZE, 97,
Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(loadFromFileButton)
                .addGap(0, 0, Short.MAX_VALUE))
            .addGroup(menuPanelLayout.createSequentialGroup()
                .addComponent(userNameLabel)
                .addGap(1, 1, 1)
                .addComponent(userNameTextField)))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(menuPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(saveLogsToFileButton, javax.swing.GroupLayout.PREFERRED_SIZE, 134,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(saveTablesToFileButton, javax.swing.GroupLayout.PREFERRED_SIZE, 134,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(12, 12, 12))
    );
    menuPanelLayout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new java.awt.Component[]
{addUserButton, loadFromFileButton, removeUserButton, saveTablesToFileButton});
    menuPanelLayout.setVerticalGroup(
      menuPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(menuPanelLayout.createSequentialGroup()
        .addContainerGap()
        .addGroup(menuPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
          .addComponent(userNameLabel)
          .addComponent(userNameTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
          .addComponent(saveLogsToFileButton))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addGroup(menuPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
          .addComponent(removeUserButton)
          .addComponent(addUserButton)
          .addComponent(saveTablesToFileButton)
          .addComponent(loadFromFileButton))
```

```
        .addContainerGap())
);
logs.setEditable(false);
logs.setColumns(20);
jScrollPane3.setViewportView(logs);
testPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Test"));
addTestPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Add test"));
addTestButton.setText("Set test result");
addTestButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        addTestButtonActionPerformed(evt);
    }
});
testResultLabel.setText("Test result:");
testResultTextField.setForeground(javax.swing.UIManager.getDefaults().getColor("TextField.shadow"));
testResultTextField.setText("0-100");
testResultTextField.addFocusListener(new java.awt.event.FocusAdapter() {
    public void focusGained(java.awt.event.FocusEvent evt) {
        testResultTextFieldFocusGained(evt);
    }
    public void focusLost(java.awt.event.FocusEvent evt) {
        testResultTextFieldFocusLost(evt);
    }
});
testResultTextField.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        testResultTextFieldKeyPressed(evt);
    }
});
javax.swing.GroupLayout addTestPanelLayout = new javax.swing.GroupLayout(addTestPanel);
addTestPanel.setLayout(addTestPanelLayout);
addTestPanelLayout.setHorizontalGroup(
    addTestPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(addTestPanelLayout.createSequentialGroup()
```

```
            .addContainerGap()
            .addGroup(addTestPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
               .addGroup(addTestPanelLayout.createSequentialGroup()
                  .addComponent(testResultLabel)
                  .addGap(0, 0, Short.MAX_VALUE))
               .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
addTestPanelLayout.createSequentialGroup()
                  .addGroup(addTestPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                     .addComponent(addTestButton, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 118, Short.MAX_VALUE)
                     .addComponent(testResultTextField))
                  .addContainerGap())))
      );
      addTestPanelLayout.setVerticalGroup(
         addTestPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
         .addGroup(addTestPanelLayout.createSequentialGroup()
            .addContainerGap()
            .addComponent(testResultLabel)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(testResultTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(addTestButton)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
      );
      newTestNameLabel.setText("Test name:");
      testNameTextField.setForeground(javax.swing.UIManager.getDefaults().getColor("TextField.shadow"));
      testNameTextField.setText("Only letters. Example: RU, Serious Test...");
      testNameTextField.addFocusListener(new java.awt.event.FocusAdapter() {
         public void focusGained(java.awt.event.FocusEvent evt) {
            testNameTextFieldFocusGained(evt);
         }
         public void focusLost(java.awt.event.FocusEvent evt) {
            testNameTextFieldFocusLost(evt);
         }
```

```
});
testNameTextField.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        testNameTextFieldActionPerformed(evt);
    }
});
testNameTextField.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        testNameTextFieldKeyPressed(evt);
    }
});
removeTestButton.setText("Remove first test");
removeTestButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        removeTestButtonActionPerformed(evt);
    }
});
changeTestNamePanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Change test name"));
newNameLabel.setText("New test name:");
newTestNameTextField.setForeground(javax.swing.UIManager.getDefaults().getColor("TextField.shadow"));
newTestNameTextField.setText("Only letters. Example: EN, Not So Serious Test...");
newTestNameTextField.addFocusListener(new java.awt.event.FocusAdapter() {
    public void focusGained(java.awt.event.FocusEvent evt) {
        newTestNameTextFieldFocusGained(evt);
    }
    public void focusLost(java.awt.event.FocusEvent evt) {
        newTestNameTextFieldFocusLost(evt);
    }
});
newTestNameTextField.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        newTestNameTextFieldActionPerformed(evt);
    }
});
```

```
        newTestNameTextField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyPressed(java.awt.event.KeyEvent evt) {
                newTestNameTextFieldKeyPressed(evt);
            }
        });
        changeTestNameButton.setText("Change test name");
        changeTestNameButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                changeTestNameButtonActionPerformed(evt);
            }
        });
        javax.swing.GroupLayout changeTestNamePanelLayout = new javax.swing.GroupLayout(changeTestNamePanel);
        changeTestNamePanel.setLayout(changeTestNamePanelLayout);
        changeTestNamePanelLayout.setHorizontalGroup(
            changeTestNamePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(changeTestNamePanelLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(changeTestNamePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(changeTestNamePanelLayout.createSequentialGroup()
                        .addComponent(newNameLabel)
                        .addGap(0, 0, Short.MAX_VALUE))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
changeTestNamePanelLayout.createSequentialGroup()

.addGroup(changeTestNamePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(changeTestNameButton, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(newTestNameTextField))
                        .addContainerGap())))
        );
        changeTestNamePanelLayout.setVerticalGroup(
            changeTestNamePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
changeTestNamePanelLayout.createSequentialGroup()
```

```
            .addContainerGap()
            .addComponent(newNameLabel)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(newTestNameTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(changeTestNameButton)
            .addContainerGap())
        );
        selectedUserLabel.setFont(new java.awt.Font("Ubuntu", 1, 15)); // NOI18N
        selectedUserLabel.setText("<User is not selected>");
        javax.swing.GroupLayout testPanelLayout = new javax.swing.GroupLayout(testPanel);
        testPanel.setLayout(testPanelLayout);
        testPanelLayout.setHorizontalGroup(
            testPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(testPanelLayout.createSequentialGroup()
                .addContainerGap()
                .addGroup(testPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addGroup(testPanelLayout.createSequentialGroup()
                        .addGroup(testPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(selectedUserLabel)
                            .addGroup(testPanelLayout.createSequentialGroup()
                                .addComponent(newTestNameLabel)
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(testNameTextField)))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(removeTestButton))
                    .addGroup(testPanelLayout.createSequentialGroup()
                        .addComponent(addTestPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(changeTestNamePanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                .addContainerGap())
```

```
        );
        testPanelLayout.setVerticalGroup(
            testPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(testPanelLayout.createSequentialGroup()
                .addComponent(selectedUserLabel)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(testPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(testNameTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(removeTestButton)
                    .addComponent(newTestNameLabel))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(testPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(testPanelLayout.createSequentialGroup()
                        .addComponent(addTestPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(0, 0, Short.MAX_VALUE))
                    .addComponent(changeTestNamePanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addContainerGap())
        );
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                    .addComponent(menuPanel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(testPanel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jScrollPane3, javax.swing.GroupLayout.Alignment.LEADING))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
            .addComponent(tablePanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(tablePanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(menuPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(testPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
        pack();
    }// </editor-fold>
    private void removeUserButtonActionPerformed(java.awt.event.ActionEvent evt) {
        if (testingCenter.removeUser(userNameTextField.getText())) {
            logs.append("Remove User: User is removed: " + userNameTextField.getText() + "\n");
            updateUsersTable();
            updateTestsTable();
        };
    }
    private void addUserButtonActionPerformed(java.awt.event.ActionEvent evt) {
        if (!isOnlyAlphabetic(userNameTextField.getText())) {
            JOptionPane.showMessageDialog(null, "Only letters are allowed");
            return;
```

```
    }
    User newUser = new User(userNameTextField.getText(), 8);
    if (!userNameTextField.getText().equals("") && testingCenter.addUser(newUser, "")) {
        updateUsersTable();
        logs.append("Add User: User is added: " + userNameTextField.getText() + "\n");
    } else {
        logs.append("Add User: User is not added: " + userNameTextField.getText() + "\n");
    }
    selectedUserLabel.setText(userNameTextField.getText());
    updateTestsTable();
}
private void updateTestsTable() {
    if (selectedUserLabel.getText().equals("<User is not selected>"))
        return;
    User user = testingCenter.findMe(selectedUserLabel.getText());
    DefaultTableModel testsTableModel = (DefaultTableModel)testsTable.getModel();
    testsTableModel.setRowCount(0);
    for (int iter = 0; iter != user.getTestsTop() + 1; iter++) {
        testsTableModel.addRow(new Object[]{user.getTests()[iter].getName()});
        testsTableModel.setValueAt(user.getTests()[iter].getResult(), iter, 1);
    }
}
private void updateUsersTable() {
    User user;
    DefaultTableModel usersTableModel = (DefaultTableModel)usersTable.getModel();
    usersTableModel.setRowCount(0);
    for (int i = 0; i != testingCenter.getUsersCount(); i++) {
        user = testingCenter.findMe(i + 1);
        usersTableModel.addRow(new Object[]{user.getLastName()});
        usersTableModel.setValueAt(user.getTestsTop() + 1, i, 1);
        usersTableModel.setValueAt(user.getAverageResult(), i, 2);
    }
}
private void addTestButtonActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    if (selectedUserLabel.getText().equals("<User is not selected>")) {
        logs.append("Add test: Test is not added: User is not selected\n");
        return;
    }
    if (!isOnlyAlphabetic(testNameTextField.getText())) {
        JOptionPane.showMessageDialog(null, "Only letters are allowed in test name");
        return;
    }
    if (!isOnlyNumeric(testResultTextField.getText())) {
        JOptionPane.showMessageDialog(null, "Only numbers are allowed in test result");
        return;
    }
    int testResult = Integer.valueOf(testResultTextField.getText());
    if (testResult < 0 || testResult > 100) {
        logs.append("Add test: Test is not added: Out of range[0-100]: " + testResult + "\n");
        return;
    }
    String testName = testNameTextField.getText();
    User selectedUser = testingCenter.findMe(selectedUserLabel.getText());
    if (selectedUser.isTestAlreadyExists(testName)) {
        selectedUser.setTestResult(testName, testResult);
        logs.append("Add test: Test is changed: " + testName + "\n");
    } else {
        selectedUser.addTest(testName, testResult);
        logs.append("Add test: Test is added: " + testName + "\n");
    }
    updateTestsTable();
    updateUsersTable();
}
private void removeTestButtonActionPerformed(java.awt.event.ActionEvent evt) {
    if (testingCenter.findMe(selectedUserLabel.getText()).removeTest())
        logs.append("Remove test: Test is removed: " + "\n");
    else logs.append("Remove test: Test is not removed: " + "\n");
    updateUsersTable();
```

```
        updateTestsTable();
    }
    private void saveTablesToFileButtonActionPerformed(java.awt.event.ActionEvent evt) {
        JFileChooser fc = new JFileChooser();
        fc.showDialog(saveTablesToFileButton, "Save");
        String path = fc.getSelectedFile().toString();
        testingCenter.saveToFile(path);
        logs.append("Save tables to file: File is saved:\n");
        logs.append(path + "\n");
    }
    private void loadFromFileButtonActionPerformed(java.awt.event.ActionEvent evt) {
        JFileChooser fc = new JFileChooser();
        fc.setAcceptAllFileFilterUsed(false);
        fc.showDialog(loadFromFileButton, "Load");
        String path = fc.getSelectedFile().toString();
        testingCenter.loadFromFile(path, false);
        updateTestsTable();
        updateUsersTable();
        if (testingCenter.findMe(1) != null) {
            selectedUserLabel.setText(testingCenter.findMe(1).getLastName());
            updateTestsTable();
        }
        logs.append("Load from file: File is loaded:\n");
        logs.append(path + "\n");
    }
    private void changeTestNameButtonActionPerformed(java.awt.event.ActionEvent evt) {
        if (!isOnlyAlphabetic(newTestNameTextField.getText())) {
            JOptionPane.showMessageDialog(null, "Only letters is allowed");
            return;
        }
        logs.append("Change test name: Test is changed: " + testNameTextField.getText() + " -> " +
newTestNameTextField.getText() + "\n");
        testingCenter.findMe(selectedUserLabel.getText()).setTestName(testNameTextField.getText(),
newTestNameTextField.getText());
```

```
    updateTestsTable();
}
private void userNameTextFieldKeyPressed(java.awt.event.KeyEvent evt) {
    char character = evt.getKeyChar();
    if(Character.isLetter(character) || Character.isWhitespace(character) || Character.isISOControl(character)) {
        userNameTextField.setEditable(true);
    } else {
        userNameTextField.setEditable(false);
    }
}
public boolean isOnlyAlphabetic(String string) {
    if (string.matches("[a-яA-Яa-zA-Z ]+"))
        return true;
    return false;
}
public boolean isOnlyNumeric(String string) {
    if (string.matches("[0-9 ]+"))
        return true;
    return false;
}
/**
 *
 * Выбор пользователя через таблицу пользователей левой кнопкой мыши
 *
 * @param evt
 */
private void usersTableMouseClicked(java.awt.event.MouseEvent evt) {
    userNameTextField.setForeground(Color.BLACK);
    selectedUserLabel.setText((String)usersTable.getValueAt(usersTable.getSelectedRow(), 0));
    userNameTextField.setText((String)usersTable.getValueAt(usersTable.getSelectedRow(), 0));
    updateTestsTable();
}
private void saveLogsToFileButtonActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fc = new JFileChooser();
```

```
    fc.showDialog(saveTablesToFileButton, "Save");
    String path = fc.getSelectedFile().toString();
    logs.append("Save logs to file: File is saved:\n");
    logs.append(path + "\n");
    try {
        FileWriter file = new FileWriter(path);
        file.append(logs.getText());
        file.flush();
        file.close();
        } catch (IOException e) {
        e.printStackTrace();
    }
}
private void testsTableMouseClicked(java.awt.event.MouseEvent evt) {
    testNameTextField.setForeground(Color.BLACK);
    testNameTextField.setText((String)testsTable.getValueAt(testsTable.getSelectedRow(), 0));
}
private void testNameTextFieldKeyPressed(java.awt.event.KeyEvent evt) {
    char character = evt.getKeyChar();
    if(Character.isLetter(character) || Character.isWhitespace(character) || Character.isISOControl(character)) {
        testNameTextField.setEditable(true);
    } else {
        testNameTextField.setEditable(false);
    }
}
private void testResultTextFieldKeyPressed(java.awt.event.KeyEvent evt) {
    char character = evt.getKeyChar();
    if(Character.isDigit(character) || Character.isISOControl(character)) {
        testResultTextField.setEditable(true);
    } else {
        testResultTextField.setEditable(false);
    }
}
private void newTestNameTextFieldKeyPressed(java.awt.event.KeyEvent evt) {
```

```
    char character = evt.getKeyChar();
    if(Character.isLetter(character) || Character.isWhitespace(character) || Character.isISOControl(character)) {
        newTestNameTextField.setEditable(true);
    } else {
        newTestNameTextField.setEditable(false);
    }
}
private void testResultTextFieldFocusGained(java.awt.event.FocusEvent evt) {
    if (testResultTextField.getText().equals("0-100")) {
        testResultTextField.setText("");
        testResultTextField.setForeground(Color.BLACK);
    }
}
private void testResultTextFieldFocusLost(java.awt.event.FocusEvent evt) {
    if (testResultTextField.getText().equals("")) {
        testResultTextField.setText("0-100");
        testResultTextField.setForeground(Color.GRAY);
    }
}
private void userNameTextFieldFocusGained(java.awt.event.FocusEvent evt) {
    if (userNameTextField.getText().equals("Only letters. Example: Иван, John, Sam...")) {
        userNameTextField.setText("");
        userNameTextField.setForeground(Color.BLACK);
    }
}
private void userNameTextFieldFocusLost(java.awt.event.FocusEvent evt) {
    if (userNameTextField.getText().equals("")) {
        userNameTextField.setText("Only letters. Example: Иван, John, Sam...");
        userNameTextField.setForeground(Color.GRAY);
    }
}
private void testNameTextFieldFocusGained(java.awt.event.FocusEvent evt) {
    if (testNameTextField.getText().equals("Only letters. Example: RU, Serious Test...")) {
        testNameTextField.setText("");
```

```
        testNameTextField.setForeground(Color.BLACK);
    }
}
private void testNameTextFieldFocusLost(java.awt.event.FocusEvent evt) {
    if (testNameTextField.getText().equals("")) {
        testNameTextField.setText("Only letters. Example: RU, Serious Test...");
        testNameTextField.setForeground(Color.GRAY);
    }
}
private void newTestNameTextFieldFocusGained(java.awt.event.FocusEvent evt) {
    if (newTestNameTextField.getText().equals("Only letters. Example: EN, Not So Serious Test...")) {
        newTestNameTextField.setText("");
        newTestNameTextField.setForeground(Color.BLACK);
    }
}
private void newTestNameTextFieldFocusLost(java.awt.event.FocusEvent evt) {
    if (newTestNameTextField.getText().equals("")) {
        newTestNameTextField.setText("Only letters. Example: EN, Not So Serious Test...");
        newTestNameTextField.setForeground(Color.GRAY);
    }
}
/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("GTK+".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
```

```
                break;
            }
            if ("Windows classic".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(TestingCenterGUI.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(TestingCenterGUI.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(TestingCenterGUI.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(TestingCenterGUI.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    }
    //</editor-fold>
    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new TestingCenterGUI().setVisible(true);
        }
    });
}
// Variables declaration - do not modify
private javax.swing.JButton addTestButton;
private javax.swing.JPanel addTestPanel;
private javax.swing.JButton addUserButton;
private javax.swing.JButton changeTestNameButton;
private javax.swing.JPanel changeTestNamePanel;
private javax.swing.JScrollPane jScrollPane1;
```

```
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JButton loadFromFileButton;
    private javax.swing.JTextArea logs;
    private javax.swing.JPanel menuPanel;
    private javax.swing.JLabel newNameLabel;
    private javax.swing.JLabel newTestNameLabel;
    private javax.swing.JTextField newTestNameTextField;
    private javax.swing.JButton removeTestButton;
    private javax.swing.JButton removeUserButton;
    private javax.swing.JButton saveLogsToFileButton;
    private javax.swing.JButton saveTablesToFileButton;
    private javax.swing.JLabel selectedUserLabel;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JTextField testNameTextField;
    private javax.swing.JPanel testPanel;
    private javax.swing.JLabel testResultLabel;
    private javax.swing.JTextField testResultTextField;
    private javax.swing.JTable testsTable;
    private javax.swing.JLabel userNameLabel;
    private javax.swing.JTextField userNameTextField;
    private javax.swing.JTable usersTable;
    // End of variables declaration
}
```