МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» им.В.И.УЛЬЯНОВА (ЛЕНИНА)

Кафедра вычислительной техники

Отчет по курсовой работе

по дисциплине «Объектно-ориентированное программирование»

|  |  |
| --- | --- |
| Студент гр. 9307 | \_\_\_\_\_\_\_\_\_\_\_\_\_ Брызгалова Е.А. |
|  |  |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_ Павловский М.Г. |

Санкт-Петербург

2021

Оглавление

[Техническое задание 3](#_Toc74257880)

[Введение 3](#_Toc74257881)

[Основание для разработки 3](#_Toc74257882)

[Требования к программе 3](#_Toc74257883)

[Требования к функциональным характеристикам 3](#_Toc74257884)

[Требования к организации и форме представления выходных данных 3](#_Toc74257885)

[Требования к организации и форме представления входных данных 3](#_Toc74257886)

[Требования к надежности 3](#_Toc74257887)

[Условия эксплуатации 4](#_Toc74257888)

[Требование к информационной и программной совместимости 4](#_Toc74257889)

[Требования к программной документации 4](#_Toc74257890)

[Стадии и этапы разработки 5](#_Toc74257891)

[Порядок контроля и приемки 5](#_Toc74257892)

[Проектирование ПК 6](#_Toc74257893)

[Описание вариантов использования ПК 6](#_Toc74257894)

[Создание прототипа интерфейса пользователя 8](#_Toc74257895)

[Разработка объектной модели ПК 13](#_Toc74257896)

[Построение диаграммы программных классов 15](#_Toc74257897)

[Описание поведения ПК 17](#_Toc74257898)

[Построение диаграммы действий 20](#_Toc74257899)

[Руководство оператора 22](#_Toc74257900)

[Назначение программы 22](#_Toc74257901)

[Описание задачи 22](#_Toc74257902)

[Входные и выходные данные 23](#_Toc74257903)

[Выполнение программы 23](#_Toc74257904)

[Запуск программы 23](#_Toc74257905)

[Выполнение основных функций 24](#_Toc74257906)

[Заключение 30](#_Toc74257907)

[Исходный код 31](#_Toc74257908)

[Data 31](#_Toc74257909)

[Window 56](#_Toc74257910)

[Main 70](#_Toc74257911)

[Password 70](#_Toc74257912)

[Xml\_file 71](#_Toc74257913)

[Report 73](#_Toc74257914)

# Техническое задание

## Введение

Разработать ПК для менеджера соревнований автогонщиков. В ПК должна храниться информация о командах, гонщиках, соревнованиях. Менеджер соревнований может добавлять, изменять и удалять эти сведения.

## Основание для разработки

Основанием для разработки ПК является курсовой проект по дисциплине «Объектно-ориентированное программирование».

# Требования к программе

## Требования к функциональным характеристикам

ПК должен обеспечивать выполнение следующих функций:

− просмотр, добавление, удаление и изменение в базы данных (БД);

− выдача справочной информации, хранимой в БД, по запросам ОЛ.

Ввод, просмотр, добавление, удаление и изменение в БД должны обеспечивать ведение и хранение следующих данных:

− сведений о командах гонщиков;

− данных о соревнованиях.

## Требования к организации и форме представления выходных данных

Выходные данные должны быть представлены в виде таблицы содержащий описание необходимых информационных объектов, выполненного посредствам представления его характеристик.

## Требования к организации и форме представления входных данных

Входная информация содержится в приходно-расходной документации. Ввод исходных данных должен осуществляется ОЛ в режиме диалога. Вводимые данные являются значениями характеристик (атрибутов) информационных объектов.

## Требования к надежности

ПК должен устойчиво функционировать при соблюдении гарантии устойчивого функционирования операционной системы и системы управления базой данных. Под устойчивой работой ПК понимается непрерывное функционирование программы в отсутствии критических сбоев, приводящих к аварийному завершению. Кроме того, должен быть обеспечен контроль входных данных на предмет соответствия предполагаемому типу.

## Условия эксплуатации

Выполнение ПК своих функций должно быть обеспечено для однопользовательского режима работы с монопольным доступом к базе данных.

## Требование к информационной и программной совместимости

Выходная и входная информация ПК должна быть удобна для визуального восприятия.

ПК должен быть выполнен на языке программирования высокого уровня Java и должен быть совместим с операционной системой Windows.

Обязательными требованиями при разработке кода ПК являются использование следующих конструкций языка Java:

− закрытые и открытые члены классов;

− наследование;

− конструкторы с параметрами;

− виртуальные функции;

− обработка исключительных ситуаций;

− динамическое создание объектов.

## Требования к программной документации

Программная документация (ПД) должна удовлетворять требованиям стандартов ЕСПД.

Документация должна быть представлена в следующем составе:

− описание процесса проектирования ПК;

− руководство оператора;

− исходные тексты ПК.

## Стадии и этапы разработки

− Разработка технического задания;

− Описание вариантов использования ПК;

− Создание прототипа интерфейса пользователя;

− Разработка объектной модели ПК;

− Построение диаграмм программных классов;

− Описание поведения ПК;

− Построение диаграмм действий;

## Порядок контроля и приемки

В процессе приема работы устанавливается соответствие ПК и прилагаемой документации требованиям, обозначенным в техническом задании.

# Проектирование ПК

## Описание вариантов использования ПК

Развернутое описание функциональных требований осуществляется на этапе проектирования комплекса. Для того чтобы детализировать требования, необходимо выделить процессы, происходящие в заданной предметной области. Описание таких процессов на UML выполняется в виде прецедентов (use case). Прецеденты являются сценарием или вариантом использования ПК при взаимодействии с внешней средой. Они являются продолжением описаний требований и функциональных спецификаций, указанных в техническом задании. Прецедент изображается в виде эллипса, в котором содержится имя прецедента. Название прецедента обязательно включает в себя глагол, выражающий суть выполняемой функции. С помощью прецедентов описывается функционирование ПК с точки зрения внешнего пользователя, который называется в UML актором (actor). Актор представляет собой любую внешнюю по отношению к моделируемой системе сущность (человек, программная система, устройство), которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач. Актор на диаграмме изображается пиктограммой в виде человечка, под которым указано его имя. Совокупность функций, реализуемых ПК, изображается в виде диаграммы (use case diagram). Для построения диаграммы необходимо определить акторы, прецеденты (функции) и взаимоотношение между акторами и прецедентами, и между прецедентами, если один прецедент расширяет или использует другой. В языке UML для вариантов использования и действующих лиц поддерживается несколько типов связей. Это связи коммуникации (communication), использования (uses) и расширения (extends).

Связь коммуникации — это связь между прецедентом и актором. На языке UML связь коммуникации изображают в виде стрелки. Направление стрелки показывает, кто инициирует коммуникацию. При задании коммуникации необходимо указать данные, которые вводит или получает пользователь. Кроме данных на концах стрелки можно указать кратности отношения, которые характеризуют количество взаимодействующих между собой акторов и прецедентов. На диаграммах прецедентов наиболее распространенными являются две формы записи кратности 1 и 1 .. \*. Первая форма записи означает, что один актор (прецедент) участвует во взаимодействии, а вторая форма записи, что один или несколько акторов (прецедентов) участвуют во взаимодействии.

Связь использования предполагает, что один прецедент всегда применяет функциональные возможности другого. С помощью таких связей структурируют прецеденты, показывая тем самым, какой прецедент является составной частью другого прецедента. Такой включаемый прецедент является абстрактным прецедентом в том смысле, что он не может исполняться независимо от других прецедентов, а лишь в их составе. Связь использования изображается с помощью стрелок и слова «uses» (использование). Направление стрелки указывает, какой прецедент используется для реализации функциональности другого прецедента.

Связь расширения задается в том случае, если необходимо показать родственные отношения между двумя прецедентами. Один из них является базовым, а другой его расширением. Базовый прецедент не зависит от расширяющих прецедентов и способен функционировать без них. С другой стороны, расширяющие прецеденты без базового прецедента функционировать не могут. Связи расширения изображают в виде стрелки со словом «extends» (расширение), которая имеет направление от базового прецедента к расширяемому.

Прецеденты необходимо ранжировать, чтобы в начальных циклах разработки реализовать наиболее приоритетные из них. Разбиение функциональности системы на отдельные прецеденты служит примерно той же цели, что и разбиение сложного алгоритма на подпрограммы. Основная стратегия должна заключаться в том, чтобы сначала сконцентрировать внимание на тех прецедентах, которые в значительной мере определяют базовую архитектуру ПК.

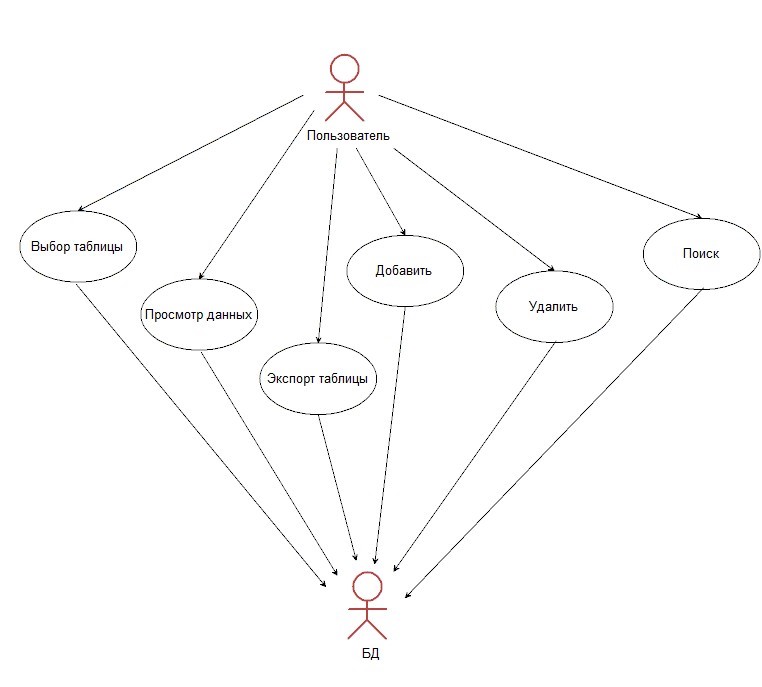


Рис. 1. Диаграмма прецедентов

Создание прототипа интерфейса пользователя

Описание прецедента выражает общую сущность процесса без детализации его реализации. Проектные решения, связанные с интерфейсом пользователя, при этом опускаются. Для разработки пользовательского интерфейса необходимо описать процесс в терминах реальных проектных решений, на основе конкретных технологий ввода-вывода информации. Когда речь идет об интерфейсе пользователя, прецеденты разбиваются на экранные формы, которые определяют содержимое диалоговых окон и описывают способы взаимодействия с конкретными устройствами. Для каждой экранной формы указываются поля ввода и перечень элементов управления, действия пользователя (нажать кнопку, выбрать пункт меню, ввести данные, нажать правую/левую кнопку мыши) и отклики системы (отобразить данные, вывести подсказку, переместить курсор). Такое описание интерфейса представляется в виде таблицы экранных форм.

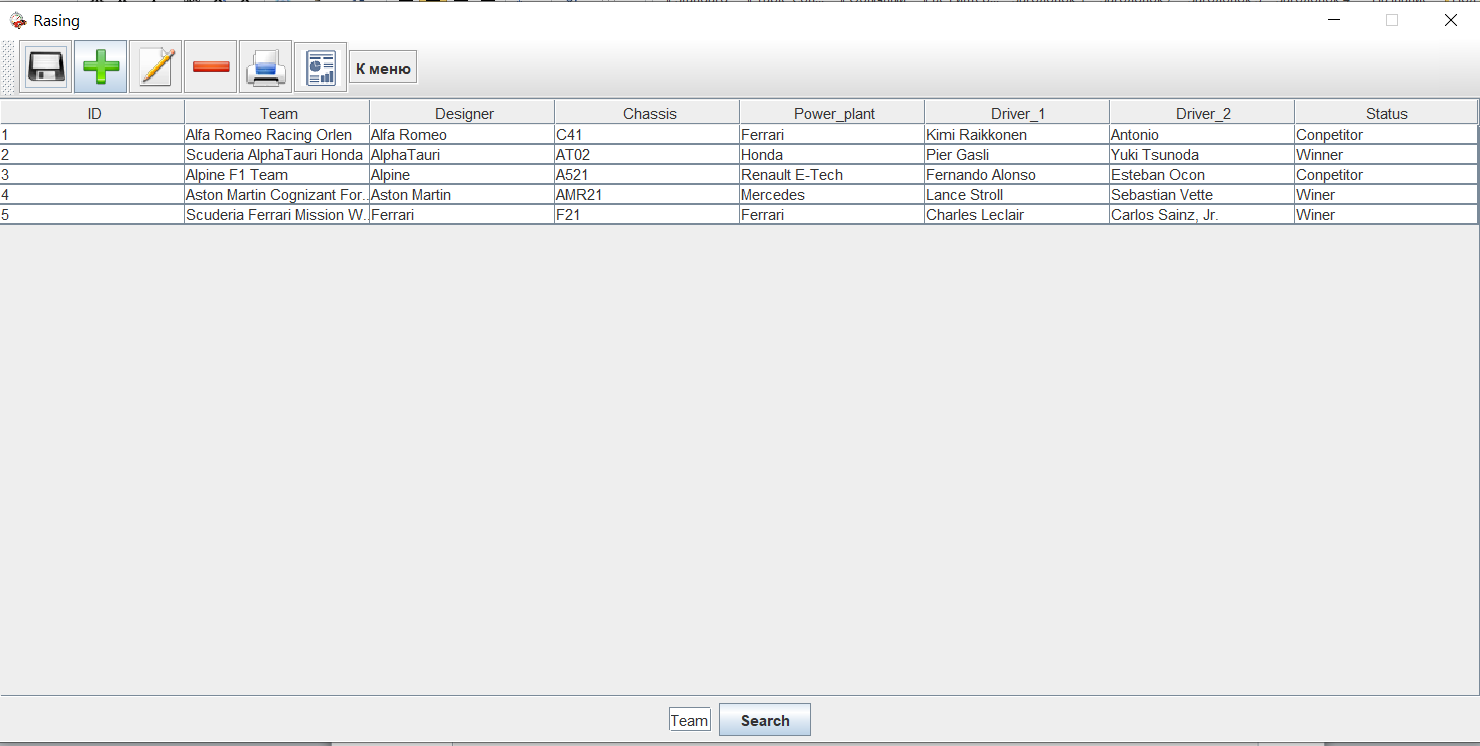


Рис. 2. Таблица команд

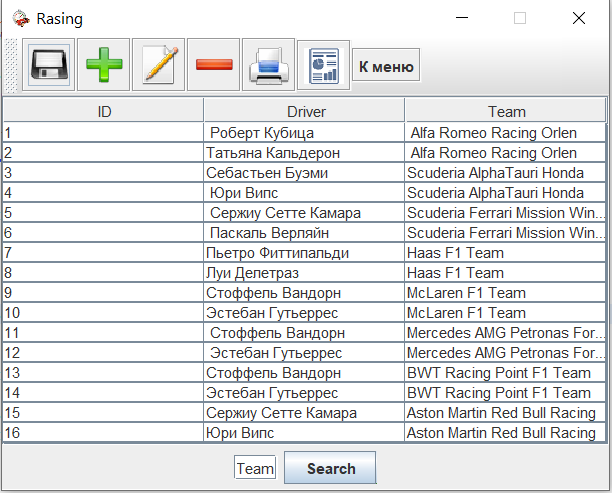


Рис. 3. Таблица резервных пилотов

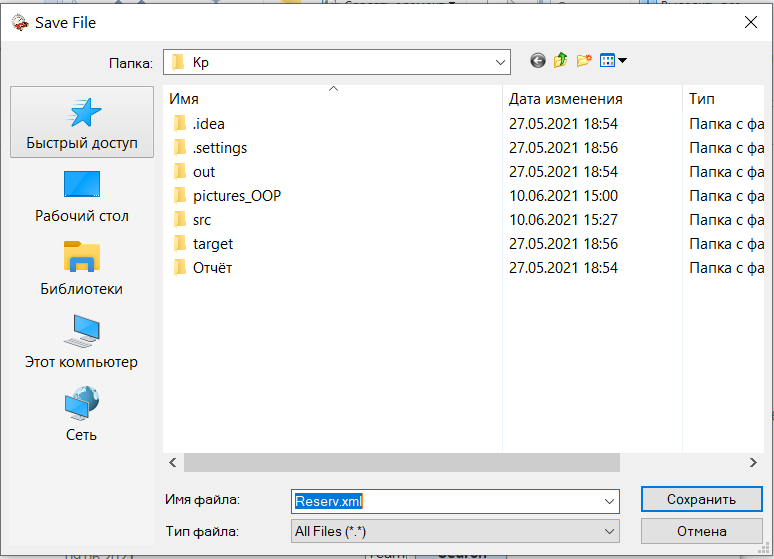


Рис. 4. Экранная форма экспорта таблицы резервных пилотов

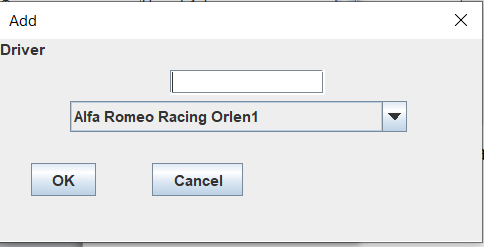


Рис. 5. Экранная форма добавления резервных пилотов

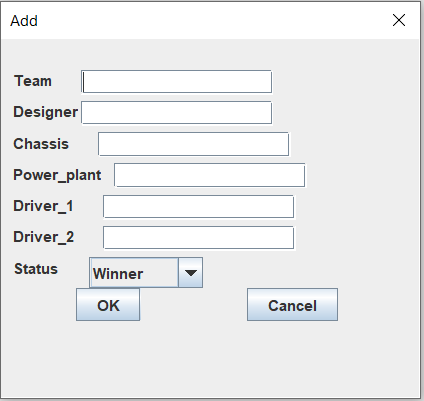


Рис. 6. Экранная форма добавления информации

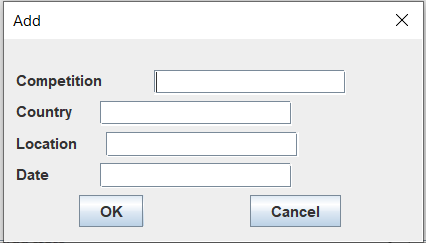


Рис.7 Экранная форма добавления соревнований

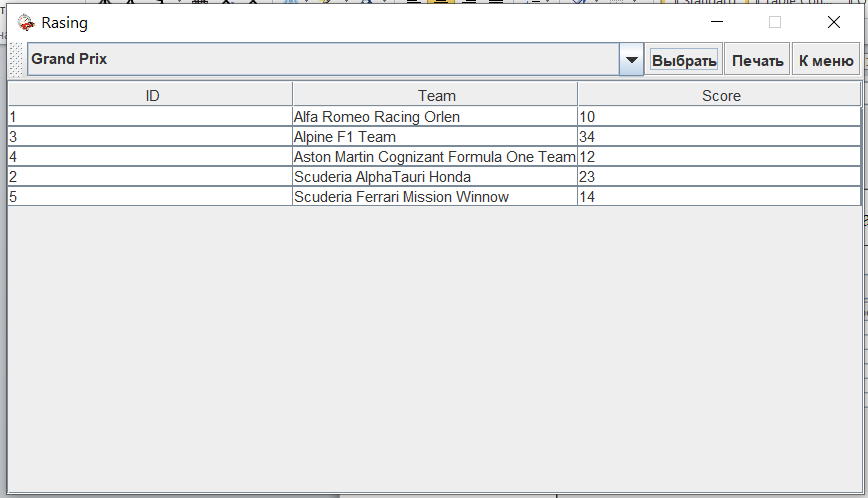


Рис.8 Таблица итогов соревнования

Таблица 1.

|  |  |
| --- | --- |
| **Экранная форма** | **Элементы управления** |
| Таблица команд | Поле поиска по командам  Кнопка «Экспорт таблицы»  Кнопка «Добавление строки»  Кнопка «Редактирования строки»  Кнопка «Удаление строки»  Кнопка «Поиск»  Кнопка «Печать»  Кнопка «Отчёт»  Кнопка «К меню» |
| Таблица резервных гонщиков | Поле поиска по команде  Кнопка «Экспорт таблицы»  Кнопка «Добавление строки»  Кнопка «Редактирования строки»  Кнопка «Удаление строки»  Кнопка «Поиск»  Кнопка «Печать»  Кнопка «Отчёт»  Кнопка «К меню» |
| Добавление команды | Поле «Команда» (Название)  Поле «Конструктор»  Поле «Шасси»  Поле «Силовая установка»  Поле «Первый пилот»  Поле «Второй пилот»  Комбобокс «Статус»  Кнопка «ОК»  Кнопка «Cancel» |
| Добавление резервного гонщика | Поле «Имя»  Комбобокс «Команда»  Кнопка «ОК»  Кнопка «Cancel» |
| Добавление соревнования | Поле «Соревнование»  Поле «Страна»  Поле «Локация»  Поле «Дата»  Кнопка «ОК»  Кнопка «Cancel» |
| Экспорт таблицы | Поле «Название файла»  Кнопка «ОК»  Кнопка «Отмена» |
| Таблица участников выбранного соревнования | Комбобокс «Выбор соревнования»  Кнопка «Печать»  Кнопка «Выбрать»  Кнопка «К меню» |

## Разработка объектной модели ПК

Объектная модель не описывает структуру ПК, она отображает основные понятия предметной области в виде совокупности типов объектов (сущностей). Сущности строятся путем выделения их из предметной области и анализа прецедентов. На диаграмме сущность обозначается прямоугольником, внутри которого записывается имя сущности, ее атрибуты и операции.

Атрибуты описывают свойства сущности. В объектную модель включаются те атрибуты, для которых определены соответствующие требования или для которых предполагается хранить определенную информацию. Атрибут характеризуется именем и типом. Для атрибута рекомендуется использовать простые типы данных (число, строка, дата, время и другие).

Описание операций помогает определить поведение объектов сущности. На этом этапе, прежде всего, определяется внутреннее поведение каждого объекта сущности, без учета взаимодействия с другими объектами предметной области. На диаграмме обычно указывается только имя операции, а ее подробное описание приводится в отдельной таблице. В таблице должно содержаться краткое описание назначения операции, ее имя и список входных и выходных параметров.

Ассоциация между сущностями отражает некоторое бинарное отношение между ними. Ассоциация обозначается проведенной между сущностями линией, с которой связывается определенное имя. Имя записывается в глагольной форме, и оно должно отражать семантический смысл отношения. Стрелка на линии указывает, в каком направлении нужно читать имя. На концах линии могут содержаться выражения, определяющие количественную связь между экземплярами сущности (кратность). Кратность определяет, сколько экземпляров одной сущности может быть ассоциировано с одним экземпляром другой сущности. Примеры кратностей:

0 ..\* - нуль или больше,

1 .. \* - один или больше,

1 – ровно один.

Необходимо устанавливать отношения ассоциации между двумя сущностями в том случае, если объект одной сущности должен знать об объекте другой. Прежде всего, следует включать в модель те ассоциации, которые отражают структурные отношения («содержит», «включает», «хранит» и т.д.), или те, которые должны сохраняться в течение некоторого времени.

\*

\*

\*

1

1

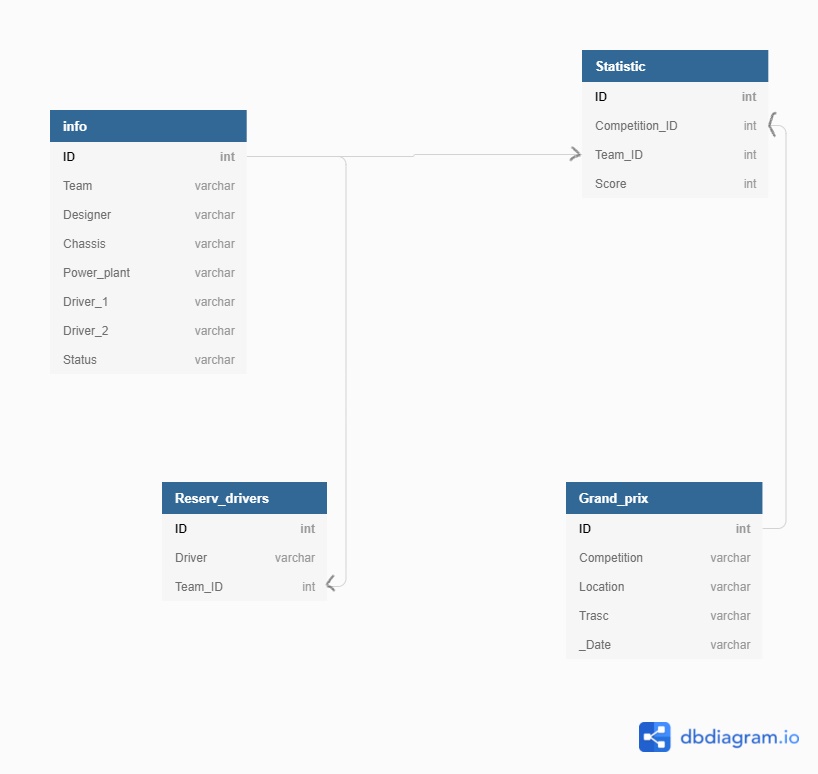


Рис. 9. Диаграмма сущностей.

## Построение диаграммы программных классов

Диаграмма классов (class diagram) иллюстрирует спецификации будущих программных классов и интерфейсов. Она строится на основе объектной модели. В описание класса указываются три раздела: имя класса, состав компонентов класса и методы класса. Графически класс изображается в виде прямоугольника. Имя программного класса может совпадать с именем сущности или быть другим. Но поскольку для записи идентификаторов переменных в языках программирования используют латинские буквы, то и имена программных классов и их атрибутов, как правило, записываются латинскими буквами. Атрибуты и операции класса перечисляются в горизонтальных отделениях этого прямоугольника. Атрибутам и методам классов должны быть присвоены права доступа. Права доступа помечаются специальными знаками:

+ - означает открытый (public) доступ;

— - означает скрытый (private) доступ;

# - означает наследуемый (protected) доступ.

При описании атрибутов после двоеточия указывается их тип, а при описании методов класса возвращаемое значение (для конструкторов возвращаемое значение не указывается).

В диаграмме классов могут вводиться дополнительно новые атрибуты, операции и связи или осуществляться конкретизация ассоциаций, указанных в объектной модели. На диаграмме классов могут быть три вида отношений: ассоциация, агрегация и наследование.

На диаграмме классов ассоциация имеет такое же обозначение, как и в объектной модели. На линиях ассоциации может присутствовать стрелка. Это стрелка видимости, которая показывает направление посылки запросов в ассоциации. Стрелка видимости также показывает, какой из классов содержит компоненты для реализации отношения ассоциации, иными словами, кто является инициатором посылки запроса к другому объекту. Ассоциация без стрелки является двунаправленной.

Агрегирование — это отношение между классами типа целое/часть. Агрегируемый класс в той или иной форме является частью агрегата. На практике это может быть реализовано по-разному. Например, объект класса-агрегата может хранить объект агрегируемого класса, или хранить ссылку на него. Агрегирование изображается на диаграмме полым ромбом на конце линии со стороны агрегирующего класса (агрегата). Если агрегируемый объект может быть создан только тогда, когда создан агрегат, а с уничтожением агрегата уничтожаются и все агрегируемые объекты, то такое агрегирование называется сильным и отображается в виде закрашенного ромба.

Наследование — это отношение типа общее-частное между классами. Его следует вводить в том случае, когда поведение и состояние различных классов имеют общие черты. Наследование связывает конкретные классы с общими или в терминологии языков программирования производные классы (подклассы) с базовыми классами (суперклассами). На диаграммах наследование изображается в виде стрелки с полым треугольником, идущей от производного класса к базовому. Если один производный класс наследует несколько базовых, то такое наследование называется множественным.

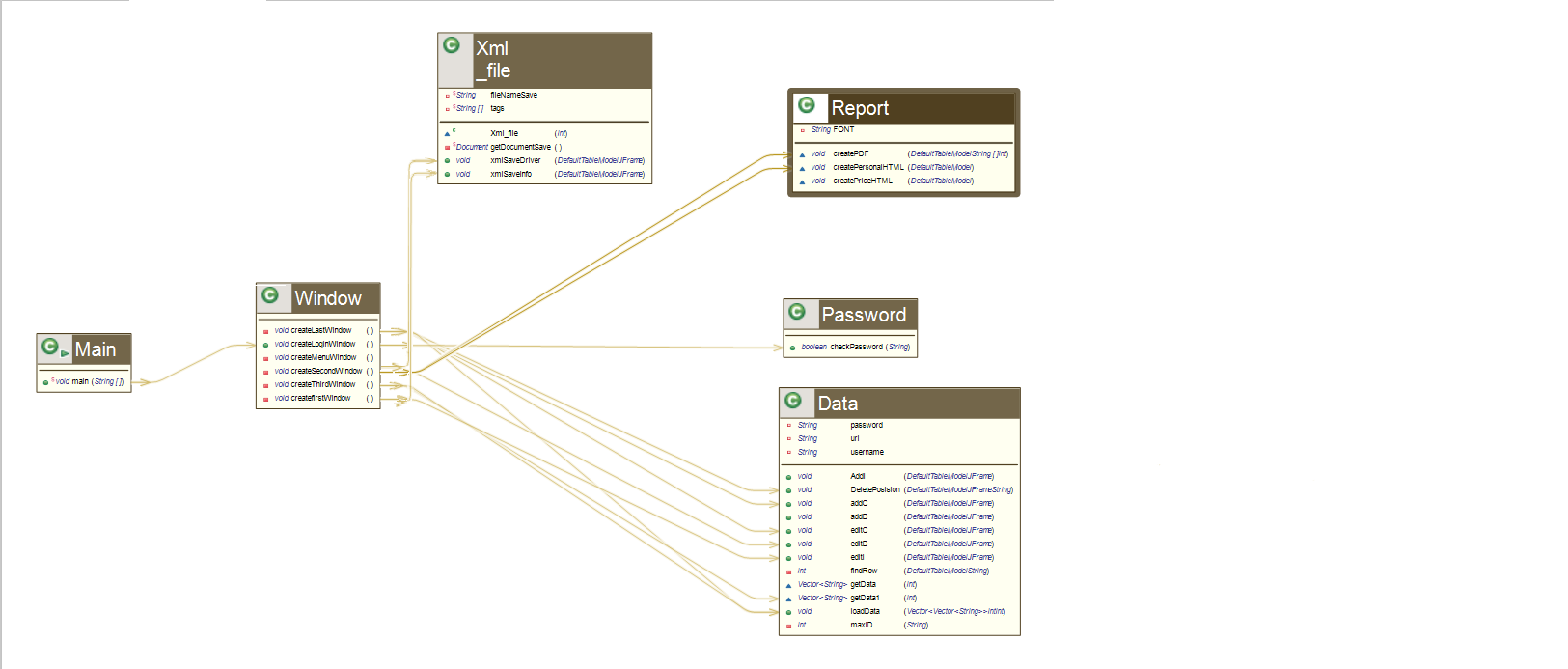


Рис. 10. Диаграмма классов.

## Описание поведения ПК

Поведение ПК представляет собой описание того, какие действия выполняет ПК, без определения механизма их реализации. Одной из составляющей такого описания является диаграмма последовательностей (seguence diagram). Диаграмма последовательностей является схемой, которая для определенного сценария прецедента показывает генерируемые пользователями и объектами события (запросы) на выполнение некоторой операции и их порядок. Диаграммы последовательности имеют две размерности: вертикальная представляет время, горизонтальная - различные объекты. Чтобы построить диаграмму последовательностей необходимо выполнить следующие действия:

1. Идентифицировать пользователей и объекты программных классов, участвующие в начальной стадии реализации сценария прецедента, и их изображения в виде прямоугольников расположить наверху в одну линию. Для каждого пользователя и объекта нарисовать вертикальную пунктирную линию, которая является линией их жизни. Внутри прямоугольника указываются подчеркнутое имя объекта и имя класса, к которому принадлежит объект.

2. Из объектной модели выбрать те операции, которые участвуют в реализации сценария. Если такие операции не были определены при построении диаграммы программных классов, то необходимо их описать и внести в модель.

3. На диаграмме последовательностей каждому запросу на выполнение операции должна соответствовать горизонтальная линия со стрелкой, начинающаяся от вертикальной линии того пользователя или объекта, который вызывает операцию, и заканчивающаяся на линии жизни того пользователя или объекта, который будет ее выполнять. Над стрелкой указывается номер операции, число итераций, имя операции и в скобках ее параметры. После описания операции может следовать комментарий, поясняющий смысл операции и начинающийся со знака "//".

Операция, которая реализует запрос, на линии жизни объекта обозначается прямоугольником. Порядок выполнения операций определяется ее номером, который указывается перед именем, и положением горизонтальной линии на диаграмме. Чем ниже горизонтальная линия, тем позже выполняется операция. В диаграммах последовательности принято применять вложенную систему нумерации, так как это позволяет отобразить их вложенность. Нумерация операций каждого уровня вложенности должна начинаться с 1.

На диаграмме последовательностей можно описать вызов операции по условию (конструкция if-else) и показать моменты создания и уничтожения объектов. Если объект создается или уничтожается на отрезке времени, представленном на диаграмме, то его линия жизни начинается и заканчивается в соответствующих точках, в противном случае линия жизни объекта проводится от начала до конца диаграммы. Символ объекта рисуется в начале его линии жизни; если объект создается не в начале диаграммы, то сообщение о создании объекта рисуется со стрелкой, проведенной к символу объекта. Если объект уничтожается не в конце диаграммы, то момент его уничтожения помечается большим крестиком "Х".



Рис. 11. Диаграмма последовательностей

## Построение диаграммы действий

Диаграмма действий (activity diagram) строится для сложных операций. Основным направлением использования диаграмм деятельности является визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются действия, а дугами — переходы от одного действия к другому. Она очень похожа на блок-схемы алгоритмов. Каждая диаграмма деятельности должна иметь единственное начальное и единственное конечное состояние. Диаграмму деятельности принято строить таким образом, чтобы действия следовали сверху вниз. Отличительной чертой диаграммы действий является то, что в ней можно отобразить параллельные процессы. Для этой цели используется специальный символ (линия синхронизации), который позволяет задать разделение и слияние потоков управления. При этом разделение имеет один входящий переход и несколько выходящих, а слияние, наоборот, имеет несколько входящих переходов и один выходящий.

В общем случае действия на диаграмме деятельности выполняются над теми или иными объектами. Эти объекты либо инициируют выполнение действий, либо определяют некоторый результат этих действий. При этом действия специфицируют вызовы, которые передаются от одного объекта графа деятельности к другому. Чтобы связать объекты с действиями, необходимо явно указать их на диаграмме деятельности. Для графического представления объектов, используются прямоугольник, в котором указывается подчеркнутое имя класса. Подчеркнутое имя означает, что на диаграмме задается объект, а не его класс. Далее после имени можно указать в прямых скобках значения атрибутов объекта после выполнения предшествующего действия. Такие прямоугольники объектов присоединяются к переходам отношением зависимости с помощью пунктирной линией со стрелкой.

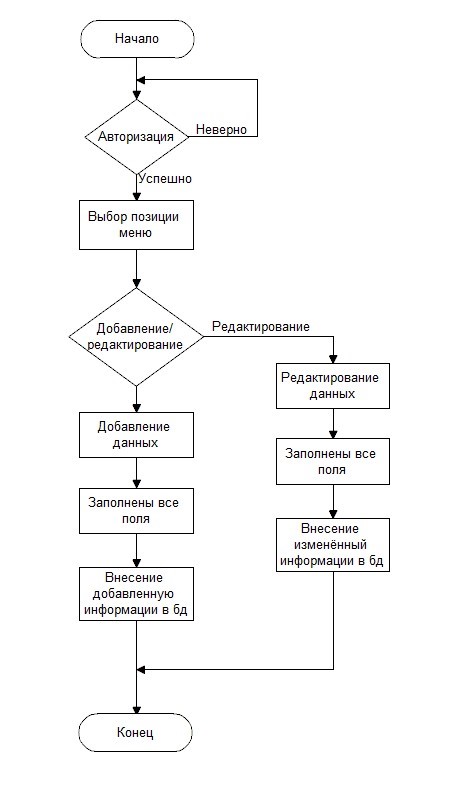


Рис. 12. Диаграмма действий.

# Руководство оператора

## Назначение программы

ПК «» должен входить в состав автоматизированной системы учета и администрирования информации, и предназначен для автоматизации деятельности ОЛ, ответственного за управление соревнованиями.

В рамках ПК ОЛ может:

− добавлять, править и удалять информацию о командах;

− добавлять, править и удалять информацию о гонщиках;

− добавлять, править и удалять информацию о соревнованиях;

− получать справочную информацию о участниках каждого соревнования;

Описание задачи

В ПК должны храниться сведения о команде, гонщиках, соревнованиях. Администратор может добавлять, изменять и удалять эти сведения.

Обязательными требованиями при разработке кода ПК являются использование следующих конструкций языка Java:

− закрытые и открытые члены классов;

− наследование;

− конструкторы с параметрами;

− абстрактные базовые классы;

− виртуальные функции;

− обработка исключительных ситуаций;

− динамическое создание объектов.

С целью выполнения поставленной задачи в процессе проектирования разработана общая модель ПК с выявлением основных объектов и связей между ними. На основании полученной модели разработаны программные классы.

Требования к коду ПК учтены при создании программных классов и непосредственном написании программы.

Входные и выходные данные

Выходные данные должны быть представлены в виде таблице содержащий описание необходимых информационных объектов, выполненного посредствам представления его характеристик.

Ввод исходных данных должен осуществляется ОЛ в режиме диалога. Вводимые данные являются значениями характеристик (атрибутов) информационных объектов. Вводимая информация может выбираться или набираться из списка предлагаемых значений.

# **Выполнение программы**

## Запуск программы

При запуске программы на экране появится окно для авторизации.

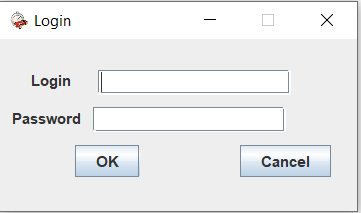


Рис. 13. Окно авторизации

При успешной авторизации появляется меню программы.

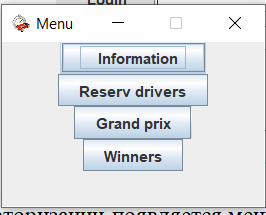


Рис. 14. Меню программы

## Выполнение основных функций

При выборе пункта «Information» автоматически загружаются данные из бд и появляется основная экранная форма.

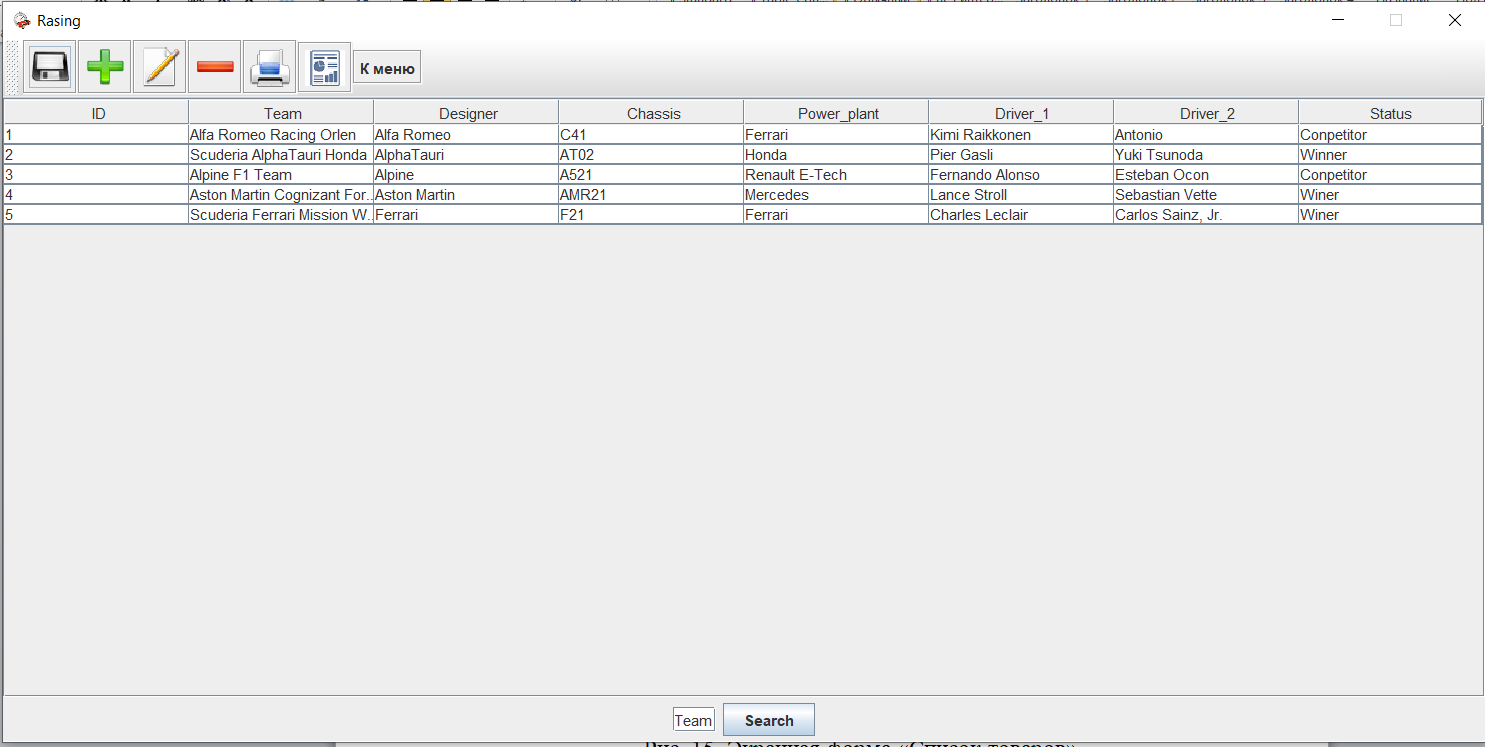


Рис. 15. Экранная форма «Список товаров»

При нажатии кнопки «Save» представленная на экране таблица будет экспортирована в формате xml.

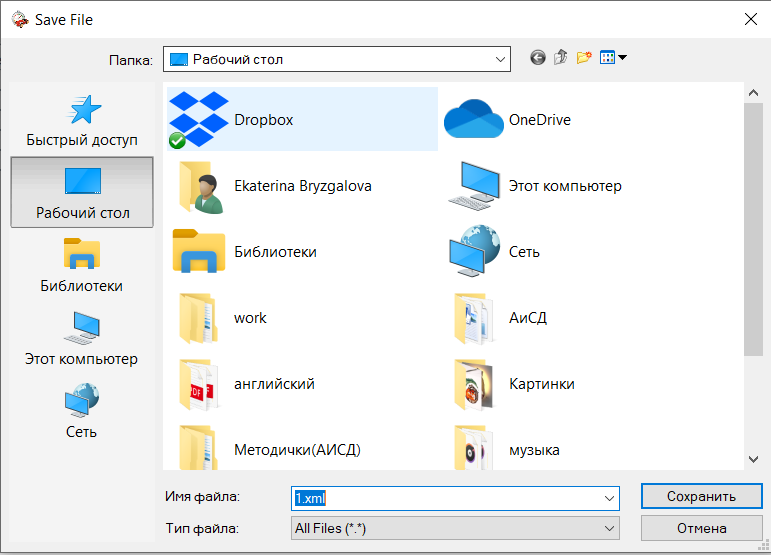


Рис. 16. Экспорт таблицы

При нажатии кнопки «Add» появляется экранная форма для ввода данных.

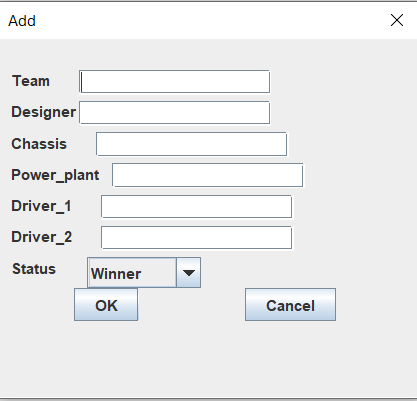


Рис. 17 Добавление команды

Если заполнены не все поля, то выводится соответствующие сообщение (рис.18).

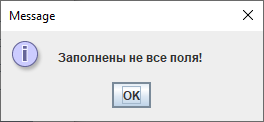


Рис. 18

При нажатии кнопки «Edit» появляется окно для ввода ID поля для редактирования (рис. 19). Затем выводится окно для редактирования данных (рис. 20). Если введён несуществующий ID, то выводится окно с ошибкой (рис. 21).

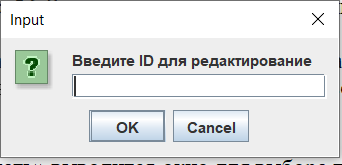


Рис. 19 Окно для ввода ID

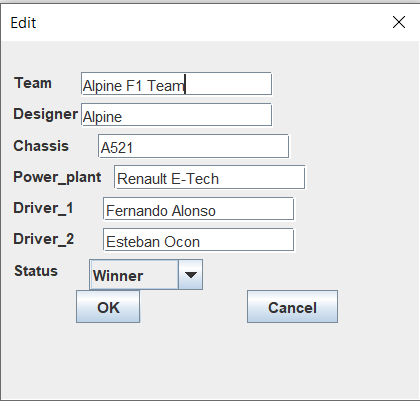


Рис. 20 Окно для редактирования данных

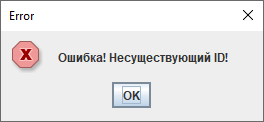


Рис. 21 Сообщение об ошибке

При нажатии кнопки «Delite» появляется окно для ввода ID поля (рис.19) для удаления позиции. Если введён несуществующий ID, то выводится окно с ошибкой (рис. 21).

При нажатии кнопки «Print» выводится окно для выбора параметров печати данных таблицы (рис. 22).

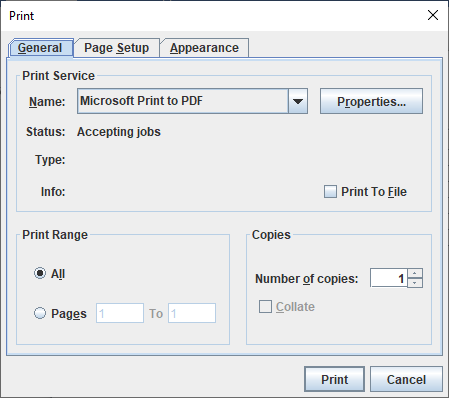


Рис. 22 Окно печати

При нажатии кнопки «Report» выводится окно для выбора формата (PDF/ HTML) формирования отчёта (рис. 23).

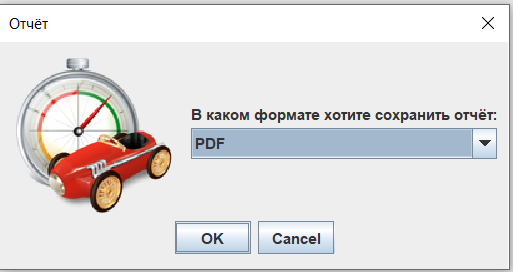


Рис. 23 Окно отчёта

При выборе пункта «Reserv drivers» автоматически загружаются данные из БД и появляется основная экранная форма.

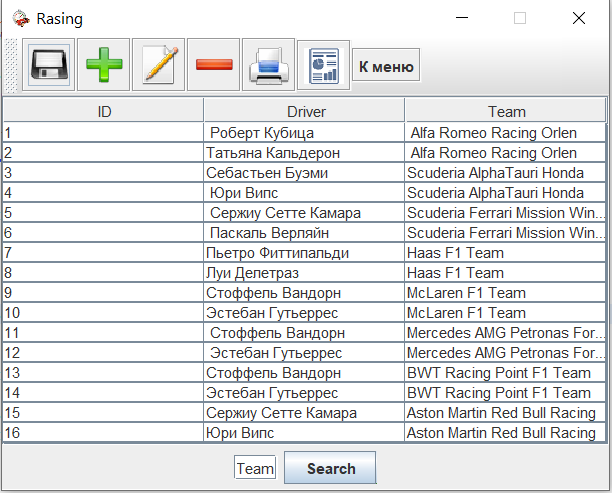


Рис. 24. Экранная форма «Reserv drivers»

При нажатии кнопки «Save» представленная на экране таблица будет экспортирована в формате xml (рис. 16).

При нажатии кнопки «добавить» появляется экранная форма для ввода данных.

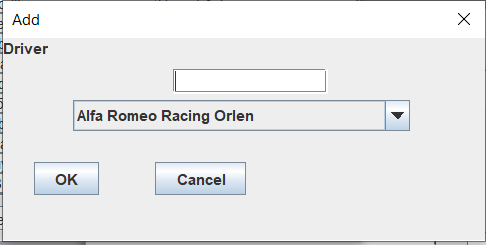


Рис. 25 Добавление пилота

Если заполнены не все поля, то выводится соответствующие сообщение (рис.18).

При нажатии кнопки «Edit» появляется окно для ввода ID поля для редактирования (рис. 19). Затем выводится окно для редактирования данных (рис. 26). Если введён несуществующий ID, то выводится окно с ошибкой (рис. 21).

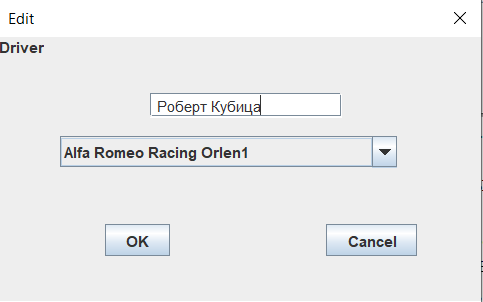


Рис. 26 Окно для редактирования данных

При нажатии кнопки «Delite» появляется окно для ввода ID поля (рис.19) для удаления позиции. Если введён несуществующий ID, то выводится окно с ошибкой (рис. 21).

При нажатии кнопки «Print» выводится окно для выбора параметров печати данных таблицы (рис. 22).

При нажатии кнопки «Report» выводится окно для выбора формата (PDF/ HTML) формирования отчёта (рис. 23).

При выборе пункта «Grand prix» автоматически загружаются данные из БД и появляется основная экранная форма.

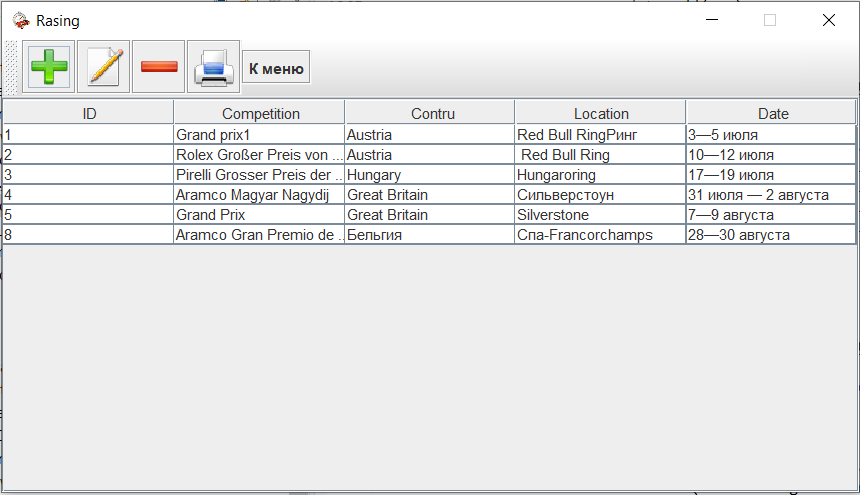


Рис. 27 Экранная форма «Список команд»

При нажатии кнопки «Add» появляется экранная форма для ввода данных.

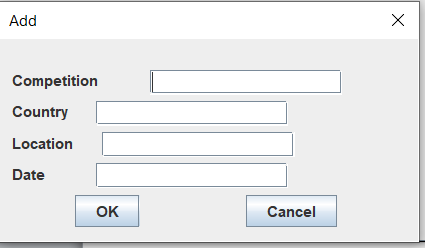


Рис. 28 Добавление позиции

Если заполнены не все поля, то выводится соответствующие сообщение (рис.18).

При нажатии кнопки «Edit» появляется окно для ввода ID поля для редактирования (рис. 19). Затем выводится окно для редактирования данных (рис. 26). Если введён несуществующий ID, то выводится окно с ошибкой (рис. 21).

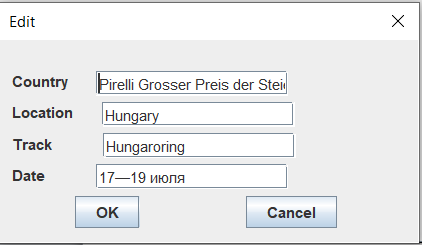


Рис. 29 Окно для редактирования данных

При нажатии кнопки «Delete» появляется окно для ввода ID поля (рис.19) для удаления позиции. Если введён несуществующий ID, то выводится окно с ошибкой (рис. 21).

При нажатии кнопки «Print» выводится окно для выбора параметров печати данных таблицы (рис. 22).

# Заключение

В результате проделанной работы разработан ПК, для менеджера соревнований автогонщиков.

В процессе проектирования созданы описание вариантов использования ПК, прототип интерфейса пользователя, объектная модель ПК, диаграмма классов, описание поведения ПК, диаграмма действия ПК.

Курсовой проект удовлетворяет поставленным требованиям.

# Исходный код

## Data

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.math.BigDecimal;

import java.util.Vector;

import javax.swing.JOptionPane;

import static javax.swing.WindowConstants.DISPOSE\_ON\_CLOSE;

import java.sql.\*;

/\*\*

\* Class for work to data

\*/

public class Data {

private String url = "jdbc:mysql://localhost:3306/rasing?serverTimezone=Europe/Moscow";

private String username = "root";

private String password = "1234567890";

/\*\*

\* Auxiliary method

\* @param flag

\* @return string

\*/

Vector <String> getData(int flag) {

Vector <String> s = new Vector<>();

try{

Class.forName("com.mysql.cj.jdbc.Driver").getDeclaredConstructor().newInstance();

try (Connection conn = DriverManager.getConnection(url, username, password)){

Statement statement = conn.createStatement();

ResultSet resultSet = statement.executeQuery("SELECT \* FROM rasing.info order by info.Team");

while(resultSet.next()){

if(flag == 1) {

s.add(String.valueOf(resultSet.getInt(1)));

}

else if(flag == 2) {

s.add(resultSet.getNString(2));

}

}

System.out.println("Connection sucdefull!");

}

}

catch(Exception ex){

System.out.println("Connection failed...");

System.out.println(ex);

}

return s;

}

Vector <String> getData1(int flag) {

Vector <String> s = new Vector<>();

try{

Class.forName("com.mysql.cj.jdbc.Driver").getDeclaredConstructor().newInstance();

try (Connection conn = DriverManager.getConnection(url, username, password)){

Statement statement = conn.createStatement();

ResultSet resultSet = statement.executeQuery("SELECT \* FROM rasing.grand\_prix order by grand\_prix.Competition");

while(resultSet.next()){

if(flag == 1) {

s.add(String.valueOf(resultSet.getInt(1)));

}

else if(flag == 2) {

s.add(resultSet.getNString(2));

}

}

System.out.println("Connection sucdefull!");

}

}

catch(Exception ex){

System.out.println("Connection failed...");

System.out.println(ex);

}

return s;

}

/\*\*

\* method for search number row in table

\* @param model

\* @param number

\* @return number row

\*/

private int findRow(DefaultTableModel model,String number) {

int num = -1;

for(int i = 0;i< model.getRowCount();i++) {

if(model.getValueAt(i,0).equals(number)) {

num = i;

System.out.println(num);

break;

}

}

return num;

}

/\*\*

\* method for search in table db

\* @param s

\* @return ID

\*/

private int maxID(String s) {

int id = 0;

try{

Class.forName("com.mysql.cj.jdbc.Driver").getDeclaredConstructor().newInstance();

try (Connection conn = DriverManager.getConnection(url, username, password)){

Statement statement = conn.createStatement();

ResultSet resultSet = statement.executeQuery("SELECT MAX(ID) FROM " + s);

while(resultSet.next()){

id = resultSet.getInt(1);

}

System.out.println("Connection sucdefull!");

}

}

catch(Exception ex){

System.out.println("Connection failed...");

System.out.println(ex);

}

System.out.println(id);

return id;

}

/\*\*

\* method for add position

\* @param model

\* @param frame

\*/

public void AddI(final DefaultTableModel model, final JFrame frame) {

final JDialog window = new JDialog(frame,"Add");

window.setModal(true);

window.setSize(350,325);

window.setLocation(300,300);

window.setResizable(false);

Container contentPane = window.getContentPane();

SpringLayout layout = new SpringLayout();

contentPane.setLayout(layout);

Component Team = new JLabel("Team");

final JTextField tTeam = new JTextField(15);

contentPane.add(Team);

contentPane.add(tTeam);

layout.putConstraint(SpringLayout.WEST , Team, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Team, 25,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tTeam, 25,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tTeam, 23,

SpringLayout.EAST , Team );

Component Designer = new JLabel("Designer");

final JTextField tDesigner = new JTextField(15);

layout.putConstraint(SpringLayout.WEST , Designer, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Designer, 50,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tDesigner, 50,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tDesigner, 23,

SpringLayout.EAST , Team);

contentPane.add(Designer);

contentPane.add(tDesigner);

Component Chassis = new JLabel("Chassis");

final JTextField tChassis= new JTextField(15);

layout.putConstraint(SpringLayout.WEST , Chassis, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Chassis, 75,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tChassis, 75,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tChassis, 23,

SpringLayout.EAST , Chassis);

contentPane.add(Chassis);

contentPane.add(tChassis);

Component Power\_plant = new JLabel("Power\_plant");

final JTextField tPower\_plant = new JTextField(15);

layout.putConstraint(SpringLayout.WEST , Power\_plant, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Power\_plant, 100,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tPower\_plant, 100,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tPower\_plant, 11,

SpringLayout.EAST , Power\_plant);

contentPane.add(Power\_plant);

contentPane.add(tPower\_plant);

Component Driver\_1 = new JLabel("Driver\_1");

final JTextField tDriver\_1 = new JTextField(15);

layout.putConstraint(SpringLayout.WEST , Driver\_1, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Driver\_1, 125,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tDriver\_1, 125,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tDriver\_1, 23,

SpringLayout.EAST , Driver\_1);

contentPane.add(Driver\_1);

contentPane.add(tDriver\_1);

Component Driver\_2 = new JLabel("Driver\_2");

final JTextField tDriver\_2 = new JTextField(15);

layout.putConstraint(SpringLayout.WEST , Driver\_2, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Driver\_2, 150,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tDriver\_2, 150,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tDriver\_2, 23,

SpringLayout.EAST , Driver\_2 );

contentPane.add(Driver\_2);

contentPane.add(tDriver\_2);

Component Status = new JLabel("Status");

final String tStatus[] = {"Winner","Competitor"};

final JComboBox comboStatus = new JComboBox<>(tStatus);

layout.putConstraint(SpringLayout.WEST , Status, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Status, 175,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, comboStatus, 175,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , comboStatus, 25,

SpringLayout.EAST,Status );

contentPane.add(Status);

contentPane.add(comboStatus);

JButton ok = new JButton("OK");

JButton cancel = new JButton("Cancel");

layout.putConstraint(SpringLayout.WEST , ok, 60,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, ok, 200,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, cancel, 200,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , cancel, 85,

SpringLayout.EAST , ok );

contentPane.add(ok);

contentPane.add(cancel);

ok.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Vector <String> s = new Vector<>();

s.add(tTeam.getText());

s.add(tDesigner.getText());

s.add(tChassis.getText());

s.add(tPower\_plant.getText());

s.add(tDriver\_1.getText());

s.add(tDriver\_2.getText());

s.add(tStatus[comboStatus.getSelectedIndex()]);

int count = 0;

for(String s1:s) {

if((s1.length() > 0) && (!s1.equals(" "))) {

count++;

}

}

if(count == 7) {

try{

Class.forName("com.mysql.cj.jdbc.Driver").getDeclaredConstructor().newInstance();

try (Connection conn = DriverManager.getConnection(url, username, password)){

try {

PreparedStatement st = conn.prepareStatement("INSERT INTO `rasing`.`info` (Team, Designer, Chassis, Power\_plant, Driver\_1, Driver\_2, Status) VALUES (?, ?, ?, ?, ?, ?, ?)");

st.setString(1,s.get(0));

st.setString(2, s.get(1));

st.setString(3, s.get(2));

st.setString(4, s.get(3));

st.setString(5, s.get(4));

st.setString(6, s.get(5));

st.setString(7, s.get(6));

st.execute();

} catch (SQLException exception) {

exception.printStackTrace();

}

System.out.println("Connection sucdefull!");

}

}

catch(Exception ex){

System.out.println("Connection failed...");

System.out.println(ex);

}

s.add(0,String.valueOf(maxID("info")));

model.addRow(s);

window.dispose();

}

else {

JOptionPane.showMessageDialog (frame,"Заполнены не все поля!");

}

}

});

cancel.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int n = JOptionPane.showConfirmDialog(window,"Вы уверены, что хотите закрыть окно?","Message",JOptionPane.YES\_NO\_OPTION);

if(n == JOptionPane.YES\_OPTION) {

window.dispose();

}

}

});

window.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

window.setVisible(true);

}

/\*\*

\* method for edit position

\* @param model

\* @param frame

\*/

public void editI(DefaultTableModel model, JFrame frame) {

String number = JOptionPane.showInputDialog("Введите ID для редактирование");

int num = findRow(model,number);

if(num >= 0) {

final JDialog window = new JDialog(frame,"Edit");

window.setModal(true);

window.setSize(350,325);

window.setLocation(300,300);

window.setResizable(false);

Container contentPane = window.getContentPane();

SpringLayout layout = new SpringLayout();

contentPane.setLayout(layout);

Component Team = new JLabel("Team");

final JTextField tTeam = new JTextField(String.valueOf(model.getValueAt(num,1)),15);

contentPane.add(Team);

contentPane.add(tTeam);

layout.putConstraint(SpringLayout.WEST , Team, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Team, 25,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tTeam, 25,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tTeam, 23,

SpringLayout.EAST , Team );

Component Designer = new JLabel("Designer");

final JTextField tDesigner = new JTextField(String.valueOf(model.getValueAt(num,2)),15);

layout.putConstraint(SpringLayout.WEST , Designer, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Designer, 50,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tDesigner, 50,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tDesigner, 23,

SpringLayout.EAST , Team);

contentPane.add(Designer);

contentPane.add(tDesigner);

Component Chassis = new JLabel("Chassis");

final JTextField tChassis = new JTextField(String.valueOf(model.getValueAt(num,3)),15);

layout.putConstraint(SpringLayout.WEST , Chassis, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Chassis, 75,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tChassis, 75,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tChassis, 23,

SpringLayout.EAST , Chassis);

contentPane.add(Chassis);

contentPane.add(tChassis);

Component Power\_plant = new JLabel("Power\_plant");

final JTextField tPower\_plant = new JTextField(String.valueOf(model.getValueAt(num,4)),15);

layout.putConstraint(SpringLayout.WEST , Power\_plant, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Power\_plant, 100,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tPower\_plant, 100,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tPower\_plant, 11,

SpringLayout.EAST , Power\_plant);

contentPane.add(Power\_plant);

contentPane.add(tPower\_plant);

Component Driver\_1 = new JLabel("Driver\_1");

final JTextField tDriver\_1 = new JTextField(String.valueOf(model.getValueAt(num,5)),15);

layout.putConstraint(SpringLayout.WEST , Driver\_1, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Driver\_1, 125,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tDriver\_1, 125,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tDriver\_1, 23,

SpringLayout.EAST , Driver\_1);

contentPane.add(Driver\_1);

contentPane.add(tDriver\_1);

Component Driver\_2 = new JLabel("Driver\_2");

final JTextField tDriver\_2 = new JTextField(String.valueOf(model.getValueAt(num,6)),15);

layout.putConstraint(SpringLayout.WEST , Driver\_2, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Driver\_2, 150,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tDriver\_2, 150,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tDriver\_2, 23,

SpringLayout.EAST , Driver\_2 );

contentPane.add(Driver\_2);

contentPane.add(tDriver\_2);

Component Status = new JLabel("Status");

final String gunType[] = {"Winner","Competitor"};

final JComboBox comboGunType = new JComboBox<>(gunType);

layout.putConstraint(SpringLayout.WEST , Status, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Status, 175,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, comboGunType, 175,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , comboGunType, 25,

SpringLayout.EAST,Status );

contentPane.add(Status);

contentPane.add(comboGunType);

JButton ok = new JButton("OK");

JButton cancel = new JButton("Cancel");

layout.putConstraint(SpringLayout.WEST , ok, 60,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, ok, 200,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, cancel, 200,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , cancel, 85,

SpringLayout.EAST , ok );

contentPane.add(ok);

contentPane.add(cancel);

ok.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Vector <String> s = new Vector<>();

s.add(tTeam.getText());

s.add(tDesigner.getText());

s.add(tChassis.getText());

s.add(tPower\_plant.getText());

s.add(tDriver\_1.getText());

s.add(tDriver\_2.getText());

s.add(gunType[comboGunType.getSelectedIndex()]);

int count = 0;

for(String s1:s) {

if((s1.length() > 0) && (!s1.equals(" "))) {

count++;

}

}

if(count == 7) {

s.add(0, number);

try{

Class.forName("com.mysql.cj.jdbc.Driver").getDeclaredConstructor().newInstance();

try (Connection conn = DriverManager.getConnection(url, username, password)){

Statement statement = conn.createStatement();

statement.executeUpdate("UPDATE info SET Team = '" + s.get(1) + "' WHERE ID = " + number);

statement.executeUpdate("UPDATE info SET Designer = '" + s.get(2) + "' WHERE ID = " + number);

statement.executeUpdate("UPDATE info SET Chassis = '" + s.get(3) + "' WHERE ID = " + number);

statement.executeUpdate("UPDATE info SET Power\_plant = '" + s.get(4) + "' WHERE ID = " + number);

statement.executeUpdate("UPDATE info SET Driver\_1 = " + s.get(5) + " WHERE ID = " + number);

statement.executeUpdate("UPDATE info SET Driver\_2 = " + s.get(6) + " WHERE ID = " + number);

statement.executeUpdate("UPDATE info SET Status = " + s.get(7) + " WHERE ID = " + number);

System.out.println("Connection sucdefull!");

}

}

catch(Exception ex){

System.out.println("Connection failed...");

System.out.println(ex);

}

for(int i = 1;i < s.size();i++) {

model.setValueAt(s.get(i),num,i);

}

window.dispose();

}

else {

JOptionPane.showMessageDialog (frame,"Заполнены не все поля!");

}

}

});

cancel.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int n = JOptionPane.showConfirmDialog(window,"Вы уверены, что хотите закрыть окно?","Message",JOptionPane.YES\_NO\_OPTION);

if(n == JOptionPane.YES\_OPTION) {

window.dispose();

}

}

});

window.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

window.setVisible(true);

}

else {

JOptionPane.showMessageDialog (frame,"Ошибка! Несуществующий ID!","Error", JOptionPane.ERROR\_MESSAGE);

}

}

/\*\*

\* method for add new personal

\* @param model

\* @param frame

\*/

public void addD(DefaultTableModel model, JFrame frame) {

final JDialog window = new JDialog(frame,"Add");

window.setModal(true);

window.setSize(400,200);

window.setLocation(300,300);

window.setResizable(false);

Container contentPane = window.getContentPane();

SpringLayout layout = new SpringLayout();

contentPane.setLayout(layout);

Component Driver = new JLabel("Driver");

final JTextField tDriver = new JTextField(12);

layout.putConstraint(SpringLayout.WEST , Driver, 100,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Driver, 100,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tDriver, 25,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tDriver, 100,

SpringLayout.EAST , Driver);

contentPane.add(Driver);

contentPane.add(tDriver);

Component Team = new JLabel("Team");

Vector <String> aTeam = getData(2);

JComboBox Teaml = new JComboBox(aTeam);

layout.putConstraint(SpringLayout.WEST , Team, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Team, 50,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, Teaml, 50,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , Teaml, 20,

SpringLayout.EAST , Driver );

contentPane.add(Driver);

contentPane.add(Teaml);

JButton ok = new JButton("OK");

JButton cancel = new JButton("Cancel");

layout.putConstraint(SpringLayout.WEST , ok, 25,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, ok, 100,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, cancel, 100,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , cancel, 45,

SpringLayout.EAST , ok );

contentPane.add(ok);

contentPane.add(cancel);

ok.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Vector <String> s = new Vector<>();

Vector <String> DriverID = getData(1);

int num = Teaml.getSelectedIndex();

s.add( DriverID.get(num));

s.add(tDriver.getText());

int count = 0;

for(String s1:s) {

if((s1.length() > 0) && (!s1.equals(" "))) {

count++;

}

}

if(count == 2) {

System.out.println(s);

try{

Class.forName("com.mysql.cj.jdbc.Driver").getDeclaredConstructor().newInstance();

try (Connection conn = DriverManager.getConnection(url, username, password)){

try {

PreparedStatement st = conn.prepareStatement("INSERT INTO `rasing`.`reserv\_drivers` (`Driver`, `Team`) VALUES (?, ?)");

st.setInt(1,Integer.parseInt(s.get(0)));

st.setString(2,s.get(1));

st.execute();

} catch (SQLException exception) {

exception.printStackTrace();

}

System.out.println("Connection succefull!");

}

}

catch(Exception ex){

System.out.println("Connection failed...");

System.out.println(ex);

}

s.add(0,String.valueOf(maxID("reserv\_drivers")));

s.remove(1);

s.add(aTeam.get(Teaml.getSelectedIndex()));

model.addRow(s);

window.dispose();

}

else {

JOptionPane.showMessageDialog (frame,"Заполнены не все поля!");

}

}

});

cancel.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int n = JOptionPane.showConfirmDialog(window,"Вы уверены, что хотите закрыть окно?","Message",JOptionPane.YES\_NO\_OPTION);

if(n == JOptionPane.YES\_OPTION) {

window.dispose();

}

}

});

window.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

window.setVisible(true);

}

/\*\*

\* method for edit personal

\* @param model

\* @param frame

\*/

public void editD(DefaultTableModel model, JFrame frame) {

String number = JOptionPane.showInputDialog("Введите ID для редактирование");

int num = findRow(model,number);

if(num >= 0) {

final JDialog window = new JDialog(frame,"Edit");

window.setModal(true);

window.setSize(400,250);

window.setLocation(300,300);

window.setResizable(false);

Container contentPane = window.getContentPane();

SpringLayout layout = new SpringLayout();

contentPane.setLayout(layout);

Component Driver = new JLabel("Driver");

final JTextField tDriver = new JTextField(String.valueOf(model.getValueAt(num,1)),15);

contentPane.add(Driver);

contentPane.add(tDriver);

layout.putConstraint(SpringLayout.WEST , Driver, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Driver, 45,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tDriver, 45,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tDriver, 85,

SpringLayout.EAST , Driver);

Component Team = new JLabel("Team");

Vector <String> aTeam = getData(2);

JComboBox Teaml = new JComboBox(aTeam);

layout.putConstraint(SpringLayout.WEST , Team, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Team, 80,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, Teaml, 80,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , Teaml, 13,

SpringLayout.EAST , Driver );

contentPane.add(Driver);

contentPane.add(Teaml);

JButton ok = new JButton("OK");

JButton cancel = new JButton("Cancel");

layout.putConstraint(SpringLayout.WEST , ok, 85,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, ok, 150,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, cancel, 150,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , cancel, 125,

SpringLayout.EAST , ok );

contentPane.add(ok);

contentPane.add(cancel);

ok.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Vector <String> s = new Vector<>();

Vector <String> DriverID = getData(1);

int num = Teaml.getSelectedIndex();

s.add( DriverID.get(num));

s.add(tDriver.getText());

int count = 0;

for(String s1:s) {

if((s1.length() > 0) && (!s1.equals(" "))) {

count++;

}

}

if(count == 2) {

s.add(0, number);

try{

Class.forName("com.mysql.cj.jdbc.Driver").getDeclaredConstructor().newInstance();

try (Connection conn = DriverManager.getConnection(url, username, password)){

Statement statement = conn.createStatement();

statement.executeUpdate("UPDATE reserv\_drivers SET Driver = '" + s.get(1) + "' WHERE ID = " + number);

statement.executeUpdate("UPDATE reserv\_drivers SET Team = '" + s.get(2) + "' WHERE ID = " + number);

System.out.println("Connection sucdefull!");

}

}

catch(Exception ex){

System.out.println("Connection failed...");

System.out.println(ex);

}

s.setElementAt(aTeam.get(Teaml.getSelectedIndex()),2);

for(int i = 1;i < 3;i++) {

System.out.println(s.get(i));

model.setValueAt(s.get(i),num,i);

}

window.dispose();

}

else {

JOptionPane.showMessageDialog (frame,"Заполнены не все поля!");

}

}

});

cancel.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int n = JOptionPane.showConfirmDialog(window,"Вы уверены, что хотите закрыть окно?","Message",JOptionPane.YES\_NO\_OPTION);

if(n == JOptionPane.YES\_OPTION) {

window.dispose();

}

}

});

window.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

window.setVisible(true);

}

else {

JOptionPane.showMessageDialog (frame,"Ошибка! Несуществующий ID!","Error", JOptionPane.ERROR\_MESSAGE);

}

}

/\*\*

\* method for delete

\* @param model

\* @param frame

\* @param s

\*/

public void DeletePosision(DefaultTableModel model, JFrame frame, String s) {

String number = JOptionPane.showInputDialog("Введите ID для удаления");

int num = findRow(model,number);

if(num > 0) {

try {

try{

Class.forName("com.mysql.cj.jdbc.Driver").getDeclaredConstructor().newInstance();

try (Connection conn = DriverManager.getConnection(url, username, password)){

System.out.println("Connection sucсessfully!");

Statement statement = conn.createStatement();

String sqlCommand = "DELETE FROM "+ s + " WHERE ID = " + number;

System.out.println(sqlCommand);

int rows = statement.executeUpdate(sqlCommand);

model.removeRow(num);

}

}

catch(Exception ex){

System.out.println("Connection failed...");

System.out.println(ex);

}

}

catch (IndexOutOfBoundsException ev) {

JOptionPane.showMessageDialog (frame,"Ошибка! Таблица пустая!","Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

/\*\*

\* method for add store

\* @param model

\* @param frame

\*/

public void addC(DefaultTableModel model, JFrame frame) {

final JDialog window = new JDialog(frame,"Add");

window.setModal(true);

window.setSize(350,200);

window.setLocation(300,300);

window.setResizable(false);

Container contentPane = window.getContentPane();

SpringLayout layout = new SpringLayout();

contentPane.setLayout(layout);

Component Competition = new JLabel("Competition");

final JTextField tCompetition = new JTextField(15);

layout.putConstraint(SpringLayout.WEST ,Competition, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Competition, 25,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tCompetition, 25,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tCompetition, 41,

SpringLayout.EAST , Competition );

contentPane.add(Competition);

contentPane.add(tCompetition);

Component Country = new JLabel("Country");

final JTextField tCountry = new JTextField(15);

contentPane.add(Country);

contentPane.add(tCountry);

layout.putConstraint(SpringLayout.WEST , Country, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Country, 50,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tCountry, 50,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tCountry, 22,

SpringLayout.EAST , Country );

Component Location = new JLabel("Location");

final JTextField tLocation = new JTextField(15);

layout.putConstraint(SpringLayout.WEST , Location, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Location, 75,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tLocation, 75,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tLocation, 23,

SpringLayout.EAST , Location );

contentPane.add(Location);

contentPane.add(tLocation);

Component Date = new JLabel("Date");

final JTextField tDate = new JTextField(15);

layout.putConstraint(SpringLayout.WEST , Date, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Date, 100,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tDate, 100,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tDate, 41,

SpringLayout.EAST ,Date);

contentPane.add(Date);

contentPane.add(tDate);

JButton ok = new JButton("OK");

JButton cancel = new JButton("Cancel");

layout.putConstraint(SpringLayout.WEST , ok, 60,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, ok, 125,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, cancel, 125,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , cancel, 85,

SpringLayout.EAST , ok );

contentPane.add(ok);

contentPane.add(cancel);

ok.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Vector <String> s = new Vector<>();

s.add(tCompetition.getText());

s.add(tCountry.getText());

s.add(tLocation.getText());

s.add(tDate.getText());

int count = 0;

for(String s1:s) {

if((s1.length() > 0) && (!s1.equals(" "))) {

count++;

}

}

if(count == 4) {

try{

Class.forName("com.mysql.cj.jdbc.Driver").getDeclaredConstructor().newInstance();

try (Connection conn = DriverManager.getConnection(url, username, password)){

try {

PreparedStatement st = conn.prepareStatement("INSERT INTO grand\_prix (Competition, Country, Location, Date) VALUES (?, ?, ?, ?)");

st.setString(1,s.get(0));

st.setString(2, s.get(1));

st.setString(3, s.get(2));

st.setString(4, s.get(3));

st.execute();

} catch (SQLException exception) {

exception.printStackTrace();

}

System.out.println("Connection sucdefull!");

}

}

catch(Exception ex){

System.out.println("Connection failed...");

System.out.println(ex);

}

s.add(0,String.valueOf(maxID("grand\_prix")));

model.addRow(s);

window.dispose();

}

else {

JOptionPane.showMessageDialog (frame,"Заполнены не все поля!");

}

}

});

cancel.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int n = JOptionPane.showConfirmDialog(window,"Вы уверены, что хотите закрыть окно?","Message",JOptionPane.YES\_NO\_OPTION);

if(n == JOptionPane.YES\_OPTION) {

window.dispose();

}

}

});

window.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

window.setVisible(true);

}

/\*\*

\* method for edit store

\* @param model

\* @param frame

\*/

public void editC(DefaultTableModel model, JFrame frame) {

String number = JOptionPane.showInputDialog("Введите ID для редактирование");

int num = findRow(model,number);

if(num >= 0) {

final JDialog window = new JDialog(frame,"Edit");

window.setModal(true);

window.setSize(350,200);

window.setLocation(300,300);

window.setResizable(false);

Container contentPane = window.getContentPane();

SpringLayout layout = new SpringLayout();

contentPane.setLayout(layout);

Component Country = new JLabel("Competition");

final JTextField tCountry = new JTextField(String.valueOf(model.getValueAt(num,1)),15);

contentPane.add(Country);

contentPane.add(tCountry);

layout.putConstraint(SpringLayout.WEST , Country, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Country, 25,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tCountry, 25,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tCountry, 22,

SpringLayout.EAST , Country );

Component Location = new JLabel("Country");

final JTextField tLocation = new JTextField(String.valueOf(model.getValueAt(num,2)),15);

layout.putConstraint(SpringLayout.WEST , Location, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Location, 50,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tLocation, 50,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tLocation, 23,

SpringLayout.EAST , Location );

contentPane.add(Location);

contentPane.add(tLocation);

Component Track = new JLabel("Location");

final JTextField tTrack = new JTextField(String.valueOf(model.getValueAt(num,3)),15);

layout.putConstraint(SpringLayout.WEST , Track, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Track, 75,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tTrack, 75,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tTrack, 41,

SpringLayout.EAST , Track );

contentPane.add(Track);

contentPane.add(tTrack);

Component Date = new JLabel("Date");

final JTextField tDate = new JTextField(String.valueOf(model.getValueAt(num,4)),15);

layout.putConstraint(SpringLayout.WEST , Date, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, Date, 100,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, tDate, 100,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , tDate, 41,

SpringLayout.EAST , Date );

contentPane.add(Date);

contentPane.add(tDate);

JButton ok = new JButton("OK");

JButton cancel = new JButton("Cancel");

layout.putConstraint(SpringLayout.WEST , ok, 60,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, ok, 125,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, cancel, 125,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , cancel, 85,

SpringLayout.EAST , ok );

contentPane.add(ok);

contentPane.add(cancel);

ok.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Vector <String> s = new Vector<>();

s.add(tCountry.getText());

s.add(tLocation.getText());

s.add(tTrack.getText());

s.add(tDate.getText());

int count = 0;

for(String s1:s) {

if((s1.length() > 0) && (!s1.equals(" "))) {

count++;

}

}

if(count == 4) {

s.add(0, number);

try{

Class.forName("com.mysql.cj.jdbc.Driver").getDeclaredConstructor().newInstance();

try (Connection conn = DriverManager.getConnection(url, username, password)){

Statement statement = conn.createStatement();

statement.executeUpdate("UPDATE grand\_prix SET Country = '" + s.get(0) + "' WHERE ID = " + number);

statement.executeUpdate("UPDATE grand\_prix SET Location = '" + s.get(1) + "' WHERE ID = " + number);

statement.executeUpdate("UPDATE grand\_prix SET Track = '" + s.get(2) + "' WHERE ID = " + number);

statement.executeUpdate("UPDATE grand\_prix SET Date= '" + s.get(3) + "' WHERE ID = " + number);

System.out.println("Connection sucdefull!");

}

}

catch(Exception ex){

System.out.println("Connection failed...");

System.out.println(ex);

}

model.removeRow(num);

model.insertRow(num,s);

window.dispose();

}

else {

JOptionPane.showMessageDialog (frame,"Заполнены не все поля!");

}

}

});

cancel.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int n = JOptionPane.showConfirmDialog(window,"Вы уверены, что хотите закрыть окно?","Message",JOptionPane.YES\_NO\_OPTION);

if(n == JOptionPane.YES\_OPTION) {

window.dispose();

}

}

});

window.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

window.setVisible(true);

}

else {

JOptionPane.showMessageDialog (frame,"Ошибка! Несуществующий ID!","Error", JOptionPane.ERROR\_MESSAGE);

}

}

/\*\*

\* method for load data from data base

\* @param data

\* @param flag

\* @param id

\*/

public void loadData(Vector <Vector<String>> data, int flag, int id) {

try{

Class.forName("com.mysql.cj.jdbc.Driver").getDeclaredConstructor().newInstance();

try (Connection conn = DriverManager.getConnection(url, username, password)){

Statement statement = conn.createStatement();

ResultSet resultSet;

switch (flag) {

case 1:

resultSet = statement.executeQuery("SELECT \* FROM info");

while(resultSet.next()){

Vector <String> row = new Vector<>();

row.add(String.valueOf(resultSet.getInt(1)));

row.add(resultSet.getNString(2));

row.add(resultSet.getNString(3));

row.add(resultSet.getNString(4));

row.add(resultSet.getNString(5));

row.add(resultSet.getNString(6));

row.add(resultSet.getNString(7));

row.add(resultSet.getNString(8));

data.add(row);

}

break;

case 2:

resultSet = statement.executeQuery("SELECT \* FROM reserv\_drivers");//("select reserv\_drivers.ID, reserv\_drivers.Driver, grand\_prix.Country from reserv\_drivers join grand\_prix on reserv\_drivers.ID = grand\_prix.ID order by ID");

while(resultSet.next()){

Vector <String> row = new Vector<>();

row.add(String.valueOf(resultSet.getInt(1)));

row.add(resultSet.getNString(2));

row.add(resultSet.getNString(3));

data.add(row);

}

break;

case 3:

resultSet = statement.executeQuery("SELECT \* FROM grand\_prix");

while(resultSet.next()){

Vector <String> row = new Vector<>();

row.add(String.valueOf(resultSet.getInt(1)));

row.add(resultSet.getNString(2));

row.add(resultSet.getNString(3));

row.add(resultSet.getNString(4));

row.add(resultSet.getNString(5));

data.add(row);

}

break;

case 4:

data.clear();

resultSet = statement.executeQuery("select info.ID,info.Team,coalesce(statistic.Score,0)\n" +

"from info \n" +

"left join statistic on info.ID = statistic.ID \n" + //and statistic.ID = " + String.valueOf(id) + "

"order by info.Team");

while(resultSet.next()){

Vector <String> row = new Vector<>();

row.add(String.valueOf(resultSet.getInt(1)));

row.add(resultSet.getString(2));

row.add(resultSet.getString(3));

data.add(row);

}

break;

}

}

}

catch(Exception ex){

System.out.println("Connection failed...");

System.out.println(ex);

}

}

}

## Window

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableModel;

import javax.swing.table.TableRowSorter;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.print.PrinterException;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.security.NoSuchAlgorithmException;

import java.text.MessageFormat;

import java.util.Vector;

/\*\*

\* class for create interface this program

\*/

public class Window extends JFrame {

/\*\*

\* method for create second window

\*/

private void createSecondWindow() {

JFrame frame = new JFrame("Rasing");

frame.setSize(500,400);

frame.setLocation(100,100);

frame.setResizable(false);

//Просим программу закрыться при закрытии окна

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

//Установка иконки приложения

ImageIcon icon = new ImageIcon("./pictures\_OOP/car1.png");

frame.setIconImage(icon.getImage());

// Создание панели инструментов

JToolBar toolBar = new JToolBar("Панель инструментов");

JButton save = new JButton(new ImageIcon("./pictures\_OOP/save.png"));

save.setToolTipText("Сохранить");

JButton add = new JButton(new ImageIcon("./pictures\_OOP/add.png"));

add.setToolTipText("Добавить");

JButton delete = new JButton(new ImageIcon("./pictures\_OOP/delete.png"));

delete.setToolTipText("Удалить");

JButton print = new JButton(new ImageIcon("./pictures\_OOP/print.png"));

print.setToolTipText("Печать");

JButton back = new JButton("К меню");

JButton report = new JButton(new ImageIcon("./pictures\_OOP/report.png"));

report.setToolTipText("Отчёт");

JButton edit = new JButton(new ImageIcon("./pictures\_OOP/edit.png"));

report.setToolTipText("Редактирование");

toolBar.add(save);

toolBar.add(add);

toolBar.add(edit);

toolBar.add(delete);

toolBar.add(print);

toolBar.add(report);

toolBar.add(back);

//Размещение панели инструментов в окне

frame.setLayout(new BorderLayout());

frame.add(toolBar,BorderLayout.NORTH);

//Создание таблицы

final String column[] = {"ID", "Driver", "Team"};

Vector<String> columns = new Vector<>();

for(int i = 0;i < column.length;i++) {

columns.add(column[i]);

}

Vector <Vector<String>> data = new Vector<>();

new Data().loadData(data,2,0);

DefaultTableModel model = new DefaultTableModel(data,columns);

JTable rasing = new JTable(model);

final TableRowSorter <TableModel> sorter = new TableRowSorter<>(model);

rasing.setRowSorter(sorter);

JScrollPane scroll = new JScrollPane(rasing);

// Размещение таблицы в окне

frame.add(scroll, BorderLayout.CENTER);

rasing.setEnabled(false);

//Создание панели поиска

JTextField weaponName = new JTextField("Team");

JButton search = new JButton("Search");

JPanel filterPanel = new JPanel();

filterPanel.add(weaponName);

filterPanel.add(search);

// Размещение панели поиска внизу окна

frame.add(filterPanel, BorderLayout.SOUTH);

//Блок обработки событий

edit.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

new Data().editD(model,frame);

}

});

back.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

frame.setVisible(false);

createMenuWindow();

}

});

report.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String s[] = {"PDF", "HTML"};

Object result = JOptionPane.showInputDialog(

frame,

"В каком формате хотите сохранить отчёт:",

"Отчёт",

JOptionPane.QUESTION\_MESSAGE,

icon, s, s[0]);

if(result.equals(s[0])) {

Report rp = new Report();

try {

rp.createPDF(model,column,2);

} catch (FileNotFoundException fileNotFoundException) {

fileNotFoundException.printStackTrace();

} catch (IOException ioException) {

ioException.printStackTrace();

}

}

else if(result.equals(s[1])) {

Report rp = new Report();

rp.createPersonalHTML(model);

}

}

});

// Обработка нажатия кнопки поиск

search.addActionListener (new ActionListener()

{

public void actionPerformed (ActionEvent event)

{

//JOptionPane.showMessageDialog (frame, "Проверка нажатия на кнопку");

String text = weaponName.getText();

if (text.length() == 0) {

sorter.setRowFilter(null);

} else {

sorter.setRowFilter(RowFilter.regexFilter(text));

}

}

});

// Обработка нажатия кнопки добавления новой позиции

add.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

new Data().addD(model,frame);

}

});

// Обработка нажатия кнопки удаление

delete.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new Data().DeletePosision(model,frame,"reserv\_drivers");

}

});

//Обработка нажатия кнопки сохранения

save.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

new Xml\_file(2).xmlSaveDriver(model,frame);

} catch (Exception exception) {

exception.printStackTrace();

}

}

});

print.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

MessageFormat headerFormat = new MessageFormat("Page {0}");

MessageFormat footerFormat = new MessageFormat("- {0} -");

rasing.print(JTable.PrintMode.FIT\_WIDTH, headerFormat, footerFormat);

} catch (PrinterException pe) {

System.err.println("Error printing: " + pe.getMessage());

}

}

});

//Показать окно

frame.setVisible(true);

}

/\*\*

\* method for create first window

\*/

private void createfirstWindow() {

JFrame frame = new JFrame("Rasing");

frame.setSize(1200,600);

frame.setLocation(100,100);

frame.setResizable(false);

//Просим программу закрыться при закрытии окна

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

//Установка иконки приложения

ImageIcon icon = new ImageIcon("./pictures\_OOP/car1.png");

frame.setIconImage(icon.getImage());

// Создание панели инструментов

JToolBar toolBar = new JToolBar("Instrument panel");

JButton save = new JButton(new ImageIcon("./pictures\_OOP/save.png"));

save.setToolTipText("Сохранить");

JButton add = new JButton(new ImageIcon("./pictures\_OOP/add.png"));

add.setToolTipText("Добавить");

JButton delete = new JButton(new ImageIcon("./pictures\_OOP/delete.png"));

delete.setToolTipText("Удалить");

JButton print = new JButton(new ImageIcon("./pictures\_OOP/print.png"));

print.setToolTipText("Печать");

JButton report = new JButton(new ImageIcon("./pictures\_OOP/report.png"));

report.setToolTipText("Отчёт");

////////////////////////////////////////////////////////////////////////////

JButton back = new JButton("К меню");

/////////////////////////////////////////////////////////////////////////////

JButton edit = new JButton(new ImageIcon("./pictures\_OOP/edit.png"));

edit.setToolTipText("Редактирование");

toolBar.add(save);

toolBar.add(add);

toolBar.add(edit);

toolBar.add(delete);

toolBar.add(print);

toolBar.add(report);

toolBar.add(back);

//Размещение панели инструментов в окне

frame.setLayout(new BorderLayout());

frame.add(toolBar,BorderLayout.NORTH);

//Создание таблицы

String column[] = {"ID","Team", "Designer", "Chassis", "Power\_plant", "Driver\_1","Driver\_2", "Status"};

Vector<String> columns = new Vector<>();

for(int i = 0;i < column.length;i++) {

columns.add(column[i]);

}

Vector <Vector<String>> data = new Vector<>();

new Data().loadData(data,1,0);

DefaultTableModel model = new DefaultTableModel(data,columns);

model.setDataVector(data,columns);

JTable rasing = new JTable(model);

final TableRowSorter <TableModel> sorter = new TableRowSorter<>(model);

rasing.setRowSorter(sorter);

JScrollPane scroll = new JScrollPane(rasing);

// Размещение таблицы в окне

frame.add(scroll, BorderLayout.CENTER);

rasing.setEnabled(false);

//Создание панели поиска

JTextField Name = new JTextField("Team");

JButton search = new JButton("Search");

JPanel filterPanel = new JPanel();

filterPanel.add(Name);

filterPanel.add(search);

// Размещение панели поиска внизу окна

frame.add(filterPanel, BorderLayout.SOUTH);

//Блок обработки событий

edit.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

new Data().editI(model,frame);

}

});

back.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

frame.setVisible(false);

createMenuWindow();

}

});

// Обработка нажатия кнопки поиск

search.addActionListener (new ActionListener()

{

public void actionPerformed (ActionEvent event)

{

String text = Name.getText();

if (text.length() == 0) {

sorter.setRowFilter(null);

} else {

sorter.setRowFilter(RowFilter.regexFilter(text));

}

}

});

report.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String s[] = {"PDF", "HTML"};

Object result = JOptionPane.showInputDialog(

frame,

"В каком формате хотите сохранить отчёт:",

"Отчёт",

JOptionPane.QUESTION\_MESSAGE,

icon, s, s[0]);

if(result.equals(s[0])) {

Report rp = new Report();

try {

rp.createPDF(model,column,1);

} catch (FileNotFoundException fileNotFoundException) {

fileNotFoundException.printStackTrace();

} catch (IOException ioException) {

ioException.printStackTrace();

}

}

else if(result.equals(s[1])) {

Report rp = new Report();

rp.createPriceHTML(model);

}

}

});

// Обработка нажатия кнопки добавления новой позиции

add.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

new Data().AddI(model,frame);

}

});

// Обработка нажатия кнопки удаление

delete.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new Data().DeletePosision(model,frame,"info");

}

});

//Обработка нажатия кнопки сохранения

save.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

new Xml\_file(1).xmlSaveInfo(model,frame);

} catch (Exception exception) {

exception.printStackTrace();

}

}

});

print.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

MessageFormat headerFormat = new MessageFormat("Page {0}");

MessageFormat footerFormat = new MessageFormat("- {0} -");

rasing.print(JTable.PrintMode.FIT\_WIDTH, headerFormat, footerFormat);

} catch (PrinterException pe) {

System.err.println("Error printing: " + pe.getMessage());

}

}

});

//Показать окно

frame.setVisible(true);

}

/\*\*

\* method for create menu window

\*/

private void createMenuWindow() {

JFrame frame = new JFrame("Menu");

frame.setSize(225,170);

frame.setLocation(600,300);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setResizable(false);

ImageIcon icon = new ImageIcon("./pictures\_OOP/car1.png");

frame.setIconImage(icon.getImage());

Container container = new Container();

container.setLayout(new BoxLayout(container, BoxLayout.Y\_AXIS));

JButton price = new JButton(" Information ");

price.setAlignmentX(Component.CENTER\_ALIGNMENT);

price.setSize(20,20);

container.add(price);

price.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

frame.setVisible(false);

createfirstWindow();

}

});

JButton list = new JButton("Reserv drivers");

list.setAlignmentX(Component.CENTER\_ALIGNMENT);

list.setSize(20,20);

container.add(list);

list.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

frame.setVisible(false);

createSecondWindow();

}

});

JButton report = new JButton("Grand prix");

report.setSize(20,20);

report.setAlignmentX(Component.CENTER\_ALIGNMENT);

container.add(report);

report.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

frame.setVisible(false);

createLastWindow();

}

});

JButton quant = new JButton("Results");

quant.setSize(20,20);

quant.setAlignmentX(Component.CENTER\_ALIGNMENT);

container.add(quant);

quant.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

frame.setVisible(false);

createThirdWindow();

}

});

frame.setContentPane(container);

frame.setVisible(true);

}

/\*\*

\* method for create third window

\*/

private void createLastWindow() {

JFrame frame = new JFrame("Rasing");

frame.setSize(700,400);

frame.setLocation(100,100);

frame.setResizable(false);

//Установка иконки приложения

ImageIcon icon = new ImageIcon("./pictures\_OOP/car1.png");

frame.setIconImage(icon.getImage());

// Создание панели инструментов

JToolBar toolBar = new JToolBar("Панель инструментов");

JButton add = new JButton(new ImageIcon("./pictures\_OOP/add.png"));

add.setToolTipText("Добавить");

JButton delete = new JButton(new ImageIcon("./pictures\_OOP/delete.png"));

delete.setToolTipText("Удалить");

JButton print = new JButton(new ImageIcon("./pictures\_OOP/print.png"));

print.setToolTipText("Печать");

///////////////////////////////////////////////////////

JButton back = new JButton("К меню");

/////////////////////////////////////////////////////////////

JButton edit = new JButton(new ImageIcon("./pictures\_OOP/edit.png"));

edit.setToolTipText("Редактирование");

toolBar.add(add);

toolBar.add(edit);

toolBar.add(delete);

toolBar.add(print);

toolBar.add(back);

//Размещение панели инструментов в окне

frame.setLayout(new BorderLayout());

frame.add(toolBar,BorderLayout.NORTH);

String column[] = { "ID","Competition","Contru","Location","Date"};

Vector<String> columns = new Vector<>();

for(int i = 0;i < column.length;i++) {

columns.add(column[i]);

}

Vector <Vector<String>> data = new Vector<>();

//DefaultTableModel model= new DefaultTableModel(data, columns);

new Data().loadData(data,3,0);

DefaultTableModel model = new DefaultTableModel(data,columns);

JTable grand\_prix = new JTable(model);

final TableRowSorter <TableModel> sorter = new TableRowSorter<>(model);

grand\_prix.setRowSorter(sorter);

JScrollPane scroll = new JScrollPane(grand\_prix);

// Размещение таблицы в окне

frame.add(scroll, BorderLayout.CENTER);

grand\_prix.setEnabled(false);

add.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

new Data().addC(model,frame);

}

});

edit.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

new Data().editC(model,frame);

}

});

// Обработка нажатия кнопки удаление

delete.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new Data().DeletePosision(model,frame,"grand\_prix");

}

});

back.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

frame.setVisible(false);

createMenuWindow();

}

});

print.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

MessageFormat headerFormat = new MessageFormat("Page {0}");

MessageFormat footerFormat = new MessageFormat("- {0} -");

grand\_prix.print(JTable.PrintMode.FIT\_WIDTH, headerFormat, footerFormat);

} catch (PrinterException pe) {

System.err.println("Error printing: " + pe.getMessage());

}

}

});

frame.setVisible(true);

}

/\*\*

\* method for create last window

\*/

private void createThirdWindow() {

JFrame frame = new JFrame("Rasing");

frame.setSize(700,400);

frame.setLocation(100,100);

frame.setResizable(false);

//Установка иконки приложения

ImageIcon icon = new ImageIcon("./pictures\_OOP/car1.png");

frame.setIconImage(icon.getImage());

// Создание панели инструментов

JToolBar toolBar = new JToolBar("Панель инструментов");

Vector <String> Competition = new Data().getData1(2);

Vector <String> ID = new Data().getData1(1);

JComboBox Comp = new JComboBox(Competition);

JButton select = new JButton("Выбрать");

JButton back = new JButton("К меню");

JButton print = new JButton("Печать");

toolBar.add(Comp);

toolBar.add(select);

toolBar.add(print);

toolBar.add(back);

frame.setLayout(new BorderLayout());

frame.add(toolBar,BorderLayout.NORTH);

String column[] = { "ID","Team","Score"};

Vector<String> columns = new Vector<>();

for(int i = 0;i < column.length;i++) {

columns.add(column[i]);

}

Vector <Vector<String>> data = new Vector<>();

DefaultTableModel model = new DefaultTableModel(data,columns);

JTable statistic = new JTable(model);

final TableRowSorter <TableModel> sorter = new TableRowSorter<>(model);

statistic.setRowSorter(sorter);

JScrollPane scroll = new JScrollPane(statistic);

// Размещение таблицы в окне

frame.add(scroll, BorderLayout.CENTER);

statistic.setEnabled(false);

select.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

new Data().loadData(data,4,Integer.valueOf(ID .get(Comp.getSelectedIndex())));

model.setDataVector(data,columns);

}

});

back.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

frame.setVisible(false);

createMenuWindow();

}

});

print.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

MessageFormat headerFormat = new MessageFormat("Page {0}");

MessageFormat footerFormat = new MessageFormat("- {0} -");

statistic.print(JTable.PrintMode.FIT\_WIDTH, headerFormat, footerFormat);

} catch (PrinterException pe) {

System.err.println("Error printing: " + pe.getMessage());

}

}

});

frame.setVisible(true);

}

/\*\*

\* method for create login window

\*/

public void createLoginWindow() {

JFrame frame = new JFrame("Login");

frame.setSize(300,175);

frame.setLocation(600,300);

frame.setResizable(false);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

ImageIcon icon = new ImageIcon("./pictures\_OOP/car1.png");

frame.setIconImage(icon.getImage());

Container contentPane = frame.getContentPane();

SpringLayout layout = new SpringLayout();

contentPane.setLayout(layout);

Component labelLogin = new JLabel("Login");

JTextField loginField = new JTextField(15);

contentPane.add(labelLogin);

contentPane.add(loginField);

layout.putConstraint(SpringLayout.WEST , labelLogin, 25,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, labelLogin, 25,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, loginField, 25,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , loginField, 22,

SpringLayout.EAST , labelLogin );

Component pass = new JLabel("Password");

JPasswordField passwordField = new JPasswordField(15);

contentPane.add(pass);

contentPane.add(passwordField);

layout.putConstraint(SpringLayout.WEST , pass, 10,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, pass, 55,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, passwordField, 55,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , passwordField, 10,

SpringLayout.EAST , pass );

JButton ok = new JButton("OK");

JButton cancel = new JButton("Cancel");

contentPane.add(ok);

contentPane.add(cancel);

layout.putConstraint(SpringLayout.WEST , ok, 60,

SpringLayout.WEST , contentPane);

layout.putConstraint(SpringLayout.NORTH, ok, 85,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.NORTH, cancel, 85,

SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST , cancel, 80,

SpringLayout.EAST , ok );

ok.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

if((new Password().checkPassword(passwordField.getText())) && (loginField.getText().equals("12345"))) {

frame.setVisible(false);

createMenuWindow();

}

else{

JOptionPane.showMessageDialog(frame,"Неверный пароль или логин");

}

} catch (NoSuchAlgorithmException noSuchAlgorithmException) {

noSuchAlgorithmException.printStackTrace();

}

}

});

cancel.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

frame.dispose();

}

});

frame.setVisible(true);

}

}

## Main

/\*\*

\* Main class program

\*/

**public** **class** Main {

**public** **static** **void** main(String[] args) {

**new** Window().createLoginWindow();

}

}

## Password

**import** javax.crypto.SecretKeyFactory;

**import** java.security.NoSuchAlgorithmException;

**import** java.security.spec.InvalidKeySpecException;

**import** java.security.spec.KeySpec;

**import** javax.crypto.spec.PBEKeySpec;

**import** java.util.Base64;

/\*\*

\* class for check password

\*/

**public** **class** Password {

**public** **boolean** checkPassword(String s) **throws** NoSuchAlgorithmException {

**boolean** flag = **false**;

**byte**[] salt = **new** **byte**[16];

KeySpec spec = **new** PBEKeySpec(s.toCharArray(), salt, 65536, 128);

SecretKeyFactory f = SecretKeyFactory.*getInstance*("PBKDF2WithHmacSHA1");

**byte**[] hash = **new** **byte**[0];

**try** {

hash = f.generateSecret(spec).getEncoded();

} **catch** (InvalidKeySpecException e) {

e.printStackTrace();

}

Base64.Encoder enc = Base64.*getEncoder*();

**if**(enc.encodeToString(hash).equals("g4MCaU7GcQ4oc9qsvsjD7Q==")) {

flag = **true**;

}

**return** flag;

}

}

## Xml\_file

import org.w3c.dom.\*;

import org.xml.sax.SAXException;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.io.FileOutputStream;

import java.lang.Exception;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.transform.\*;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import java.awt.\*;

import java.io.IOException;

/\*\*

\* Class for save data from XML file

\*/

public class Xml\_file {

private static String fileNameSave;

private static String tags[];

Xml\_file(int flag) {

if(flag == 1) {

tags = new String[]{"ID", "Team", "Designer", "Chassis", "Pover plant", "Driver 1", "Driver 2", "Status"};

}

else if(flag == 2) {

tags = new String[]{"ID", "Driver", "Team"};

}

}

/\*\*

\* method getting document for save

\*/

private static Document getDocumentSave() throws Exception {

try {

DocumentBuilderFactory f = DocumentBuilderFactory.newInstance();

DocumentBuilder builder = f.newDocumentBuilder();

return builder.newDocument();

} catch (Exception exception) {

throw new Exception("XML parsing error!");

}

}

/\*\*

\* method for save xml document

\* @param model - model of table

\* @param frame - object type of window

\* @throws Exception standart exception

\*/

public void xmlSaveInfo(DefaultTableModel model,JFrame frame) throws Exception {

FileDialog saveXml = new FileDialog(frame,"Save File",FileDialog.SAVE);

saveXml.setFile("\*.xml");

saveXml.setVisible(true);

fileNameSave = saveXml.getDirectory() + saveXml.getFile();

if(fileNameSave == null) return;

Document doc = getDocumentSave();

Node list = doc.createElement("rasingList");

doc.appendChild(list);

for(int i = 0; i < model.getRowCount(); i++){

Element weapon = doc.createElement("rasing");

list.appendChild(weapon);

for(int j = 0;j < tags.length;j++) {

weapon.setAttribute(tags[j],(String) model.getValueAt(i,j));

}

}

try{

Transformer trans= TransformerFactory.newInstance().newTransformer();

trans.setOutputProperty(OutputKeys.METHOD, "xml");

trans.setOutputProperty(OutputKeys.INDENT, "yes");

trans.transform(new DOMSource(doc), new StreamResult(new FileOutputStream(fileNameSave)));

}catch(TransformerConfigurationException e){

e.printStackTrace();

}catch(TransformerException e){

e.printStackTrace();

}catch(IOException e){

e.printStackTrace();

}

}

public void xmlSaveDriver(DefaultTableModel model,JFrame frame) throws Exception {

FileDialog saveXml = new FileDialog(frame,"Save File",FileDialog.SAVE);

saveXml.setFile("\*.xml");

saveXml.setVisible(true);

fileNameSave = saveXml.getDirectory() + saveXml.getFile();

if(fileNameSave == null) return;

Document doc = getDocumentSave();

Node weaponlist = doc.createElement("driverList");

doc.appendChild(weaponlist);

for(int i = 0; i < model.getRowCount(); i++){

Element weapon = doc.createElement("driver");

weaponlist.appendChild(weapon);

for(int j = 0;j < tags.length;j++) {

weapon.setAttribute(tags[j],(String) model.getValueAt(i,j));

}

}

try{

Transformer trans= TransformerFactory.newInstance().newTransformer();

trans.setOutputProperty(OutputKeys.METHOD, "xml");

trans.setOutputProperty(OutputKeys.INDENT, "yes");

trans.transform(new DOMSource(doc), new StreamResult(new FileOutputStream(fileNameSave)));

}catch(TransformerConfigurationException e){

e.printStackTrace();

}catch(TransformerException e){

e.printStackTrace();

}catch(IOException e){

e.printStackTrace();

}

}

}

## Report

import com.itextpdf.io.font.PdfEncodings;

import com.itextpdf.kernel.font.PdfFont;

import com.itextpdf.kernel.font.PdfFontFactory;

import com.itextpdf.kernel.pdf.PdfDocument;

import com.itextpdf.kernel.pdf.PdfWriter;

import com.itextpdf.layout.Document;

import com.itextpdf.layout.element.Cell;

import com.itextpdf.layout.element.Paragraph;

import com.itextpdf.layout.element.Table;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.io.FileNotFoundException;

import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

import static com.itextpdf.kernel.pdf.PdfName.BaseFont;

import static com.itextpdf.kernel.pdf.PdfName.FontFamily;

/\*\*

\* class for create report

\*/

public class Report {

private String FONT = "./arial.ttf";

/\*\*

\* method for create report in PDF

\* @param model object type of DefaultTableModel

\* @throws FileNotFoundException exception throws if file not found

\*/

void createPDF(DefaultTableModel model, String [] column, int flag) throws IOException {

String text = new String();

if (flag == 1) {

text = "ReportInfo";

}

else if(flag == 2) {

text = "ReportDrivers";

}

PdfFont font = PdfFontFactory.createFont(FONT, PdfEncodings.IDENTITY\_H);

PdfDocument pdfDoc = new PdfDocument(new PdfWriter("./" + text + ".pdf"));

pdfDoc.addNewPage();

Document doc = new Document(pdfDoc);

Paragraph paragraph = new Paragraph("Report");

doc.add(paragraph);

Table table = new Table(column.length);

table.setFont(font);

for(String s: column) {

// table.addCell(new Cell().add(s));

}

for(int i = 0;i < model.getRowCount();i++) {

for(int j = 0;j < model.getColumnCount();j++) {

// table.addCell(new Cell().add(model.getValueAt(i,j).toString()));

}

}

doc.add(table);

doc.close();

}

/\*\*

\* method for create report in HTML

\* @param model object type of DefaultTableModel

\*/

void createPriceHTML(DefaultTableModel model) {

PrintWriter pw = null;

try {

pw = new PrintWriter(new FileWriter("./ReportInfo.html"));

} catch (IOException e) {

e.printStackTrace();

}

pw.println("<TABLE BORDER><TR><TH>ID<TH>Team<TH>Designer<TH>Chassis<TH>Power plant<TH>Driver 1<TH>Driver 2<TH>Status</TR>");

for(int i = 0; i < model.getRowCount(); i++) {

pw.println("<TR><TD>" + (String) model.getValueAt(i,0)

+ "<TD>" + (String) model.getValueAt(i,1)

+ "<TD>" + (String) model.getValueAt(i,2)

+ "<TD>" + (String) model.getValueAt(i,3)

+ "<TD>" + (String) model.getValueAt(i,4)

+ "<TD>" + (String) model.getValueAt(i,5)

+ "<TD>" + (String) model.getValueAt(i,6)

+ "<TD>" + (String) model.getValueAt(i,7));

}

pw.println("</TABLE>");

pw.close();

}

/\*\*

\* method for create report in HTML

\* @param model object type of DefaultTableModel

\*/

void createPersonalHTML(DefaultTableModel model) {

PrintWriter pw = null;

try {

pw = new PrintWriter(new FileWriter("./ReportDrivers.html"));

} catch (IOException e) {

e.printStackTrace();

}

pw.println("<TABLE BORDER><TR><TH>ID<TH>Driver<TH>Team</TR>");

for(int i = 0; i < model.getRowCount(); i++) {

pw.println("<TR><TD>" + (String) model.getValueAt(i,0)

+ "<TD>" + (String) model.getValueAt(i,1)

+ "<TD>" + (String) model.getValueAt(i,2));

}

pw.println("</TABLE>");

pw.close();

}

}