Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий и анализа данных

Центр программной инженерии

ОТЧЕТ

к лабораторной работе №5 по дисциплине:

|  |
| --- |
| «Объектно-ориентированное программирование» |
| Разработка приложения с графическим пользовательским интерфейсом  пользовательским интенаследования, полиморфизма и инкапсуляции. |

наименование темы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы: | ИСТб-19-2 |  |  |  | Комогорцева Ю.В. |
|  | шифр группы |  | подпись |  | Фамилия ИО |
| Проверил: | доцент |  |  |  | Маланова Т.В. |
|  | должность |  | подпись |  | Фамилия ИО |

Иркутск. 2020 г.

Содержание

[**Содержание 2**](#_Toc67217064)

[**1 Постановка задачи 3**](#_Toc67217065)

[**2 Проектирование графического интерфейса 4**](#_Toc67217066)

[**3 Проектирование классов 6**](#_Toc67217067)

[**4 Проектирование тестов 7**](#_Toc67217068)

[**Результаты тестирования 8**](#_Toc67217069)

[**5 Листинг программы 17**](#_Toc67217070)

[**5.1 класс Main 17**](#_Toc67217071)

[**5.2 класс MyFrame 18**](#_Toc67217072)

[**5.3 класс MyTableModel 30**](#_Toc67217073)

[**5.4 класс FindTableModel 33**](#_Toc67217074)

[**5.5 класс TransportManager 36**](#_Toc67217075)

[**5.6 класс Transport 37**](#_Toc67217076)

[**5.7 класс CarTransport 38**](#_Toc67217077)

[**5.8 класс TrainTransport 39**](#_Toc67217078)

[**5.9 класс ExpressTrain 40**](#_Toc67217079)

[**Список использованных источников 41**](#_Toc67217080)

1 Постановка задачи

**Цель работы**: Применение объектно-ориентированного подхода в

разработке приложений с графическим интерфейсом.

**Задачи**:

- Получение общего представления о шаблоне программирования

Модель/Вид/Контроллер и его практического применения;

- Получение практического опыта работы с базовыми графическими

библиотеками java.

**Содержание задания**

Для выполнения задания необходимо разработать графический

пользовательский интерфейс средствами java для работы с классами

реализованными при выполнении либо 3-ей, либо 4-ой лабораторной работы.

Графический пользовательский интерфейс должен предоставлять

возможности работы со всеми возможными функциями программы (ввод и

вывод данных в соответствующей форме, изменение условий выполнения

задания, работа с файлами и т.д.). Состав используемых компонентов

определяется самостоятельно, но для них должны быть предусмотрены

соответствующие планировщики раскладки компонентов в основных

контейнерах.

Классы, реализующие графический пользовательский интерфейс,

должны содержать только обращения к методам классов для выполнения

заданий. Для этого может понадобиться модифицировать программный код

из предыдущих лабораторных работ.

**Индивидуальное задание**

*Вариант 11*-автомобиль, поезд, транспортное средство, экспресс;

2 Проектирование графического интерфейса

Главное окно программы включает в себя список транспорта компании, панель управления(вверху) , панель процесса(слева) и календарь-часы(внизу).

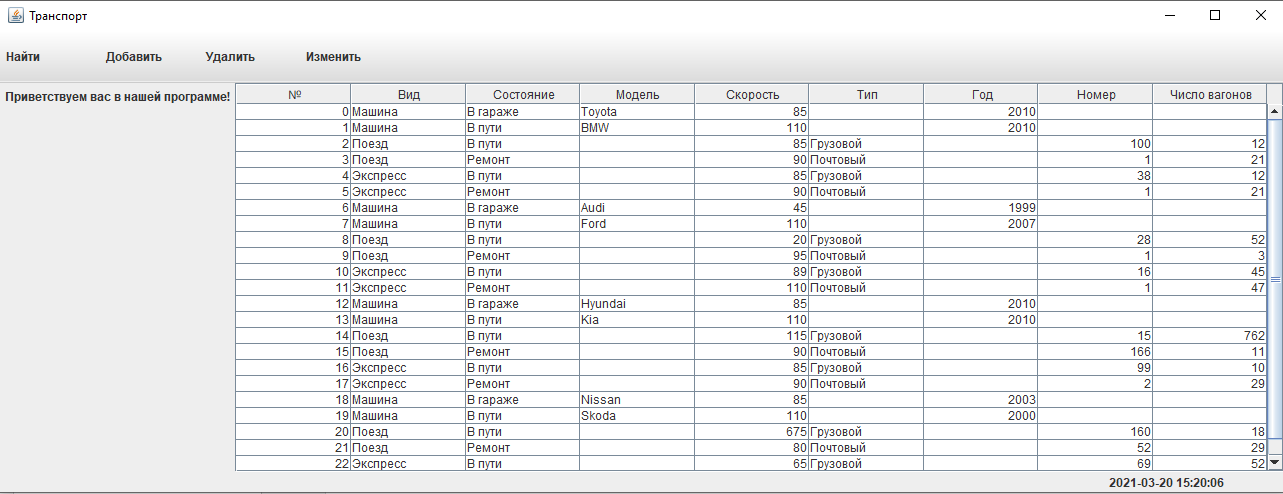


Рисунок 1 – Главное окно программы

Окно с найденной записью это окно которое содержит таблицу с одной строкой - запись которую запрашивал пользователь. Открывается при нажатие кнопки найти в главном окне после ввода номера искомой записи. Пока оно открыто, доступ к главному окну запрещен.

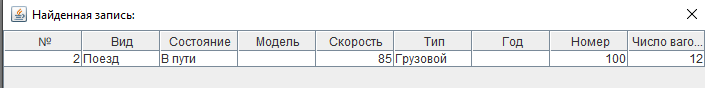


Рисунок 2 – Окно с найденной записью

Окно для обработки исключений , которое открывается если пользователь в какое то либо поле ввода ввел буквы , символы , не целые числа или пустую строку и нажал кнопку выполняющую действие. Пока оно открыто, доступ к главному окну запрещен.

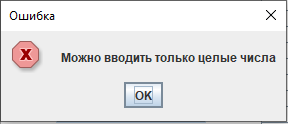


Рисунок 3 – Окно для обработки исключений

Окно для обработки исключений , которое открывается если пользователь ввел номер не существующей записи. Пока оно открыто, доступ к главному окну запрещен.

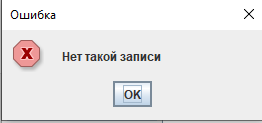


Рисунок 4 – Окно для обработки исключений

Окно для обработки исключений , которое открывается если пользователь не выбрал запись в таблице и нажал кнопку выполняющую действие. Пока оно открыто, доступ к главному окну запрещен.

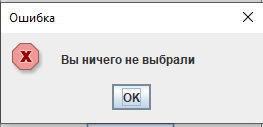


Рисунок 5 – Окно для обработки исключений

**3 Проектирование классов**

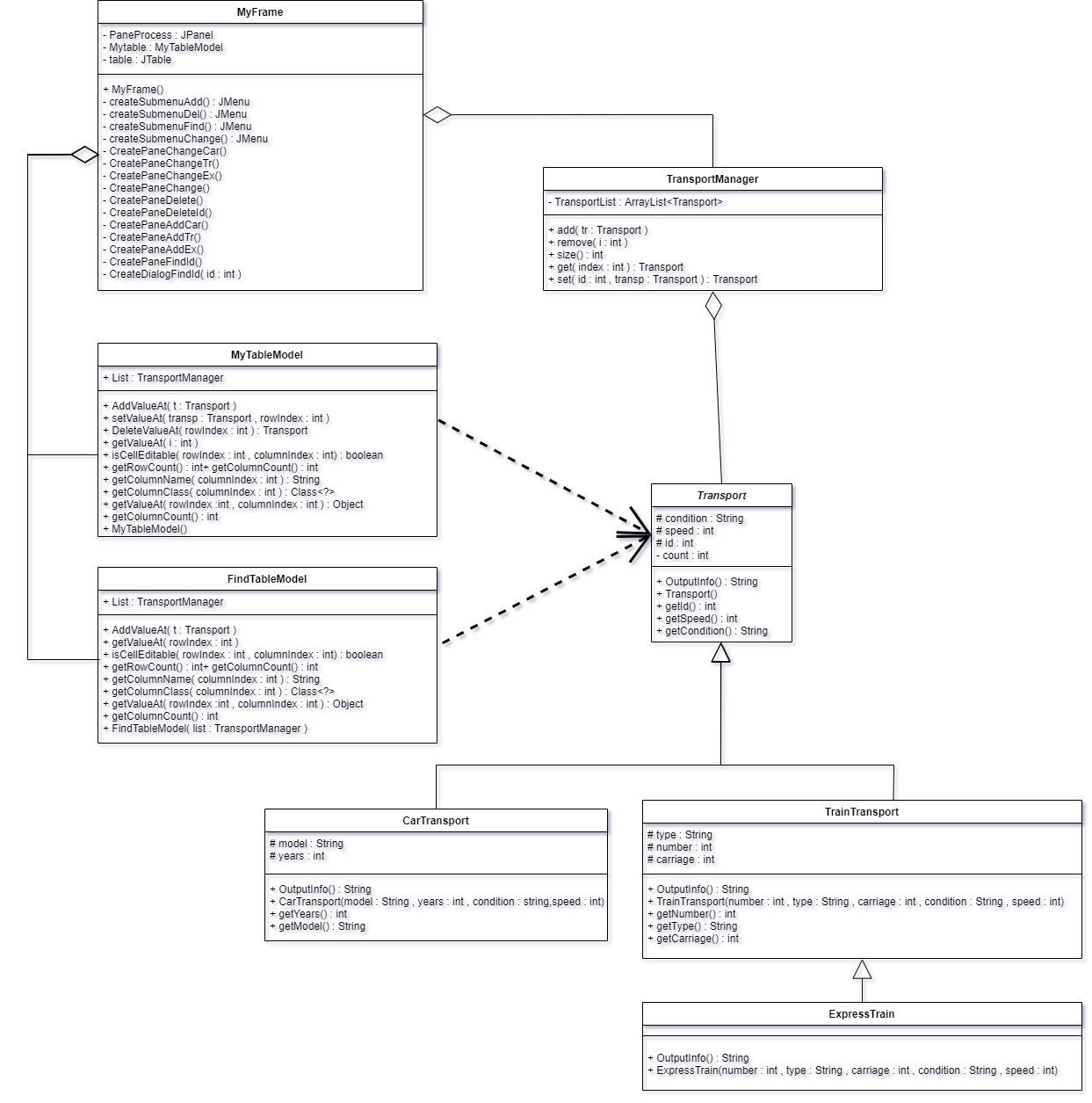


Рисунок 6 – UML-диаграмма проекта

4 Проектирование тестов

В ходе разработки было проведено тестирование программы, результаты которого представлены в табл. 1.

Таблица 1 – Тестирование программы

|  |  |  |  |
| --- | --- | --- | --- |
| № | Действие | | Результат действия |
| 1 | Первоначальный запуск программы | | Открытие главного окна |
| 2 | Нажатие на любую кнопку верхней панели | | Выпадение кнопок выбора |
| 3.1 | Добавление новой записи | Машина | Открытие панели добавления машины с необходимыми полями, добавление записи в список |
| 3.2 | Поезд | Открытие панели добавления поезд с необходимыми полями, добавление записи в список |
| 3.3 | Экспресс | Открытие панели добавления экспресс с необходимыми полями, добавление записи в список |
| 4 | Поиск записи по № | | Открытие панели поиска записи, ввод № искомой записи и открытие диалогового окна с найденной записи |
| 5.1 | Удаление записи | По № | Открытие панели удаления записи , ввод № удаляемой записи и удаление записи |
| 5.2 | В таблице | Открытие панели удаления записи в таблице , выбор удаляемой записи и удаление записи |
| 6 | Изменить запись в таблице | | Открытие панели изменения записи ,выбора записи для изменения ,открытие панели с полями изменения записи и изменения записи |
| 7 | Неверный ввод (вместо целых чисел буквы, символы или пустая строка) | | Открытие диалогового окна с сообщением " Можно вводить только целые числа" |
| 8 | Не выбрана запись | | Открытие диалогового окна с сообщением " Вы ничего не выбрали" |
| 9 | Ввод номера не существующей записи | | Открытие диалогового окна с сообщением “Нет такой записи” |

Результаты тестирования

Действие 1 – Первоначальный запуск программы – открытие главного окна

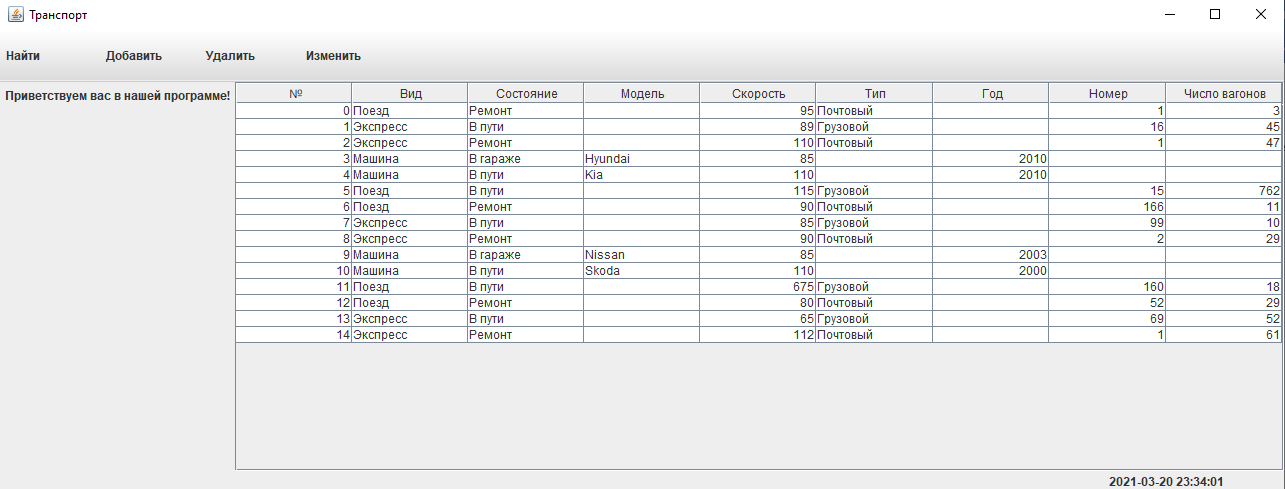


Рисунок 7 – Первоначальный запуск программы

Действие 2 – Нажатие на любую кнопку верхней панели – выпадение кнопок

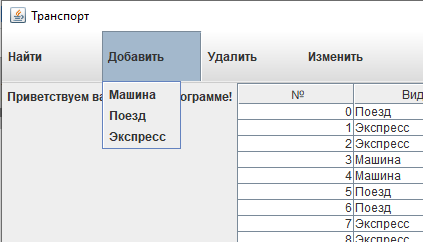


Рисунок 8 – Выпадение кнопок действия «Добавить»

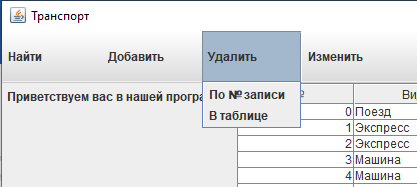


Рисунок 9 – Выпадение кнопок действия «Удаление»

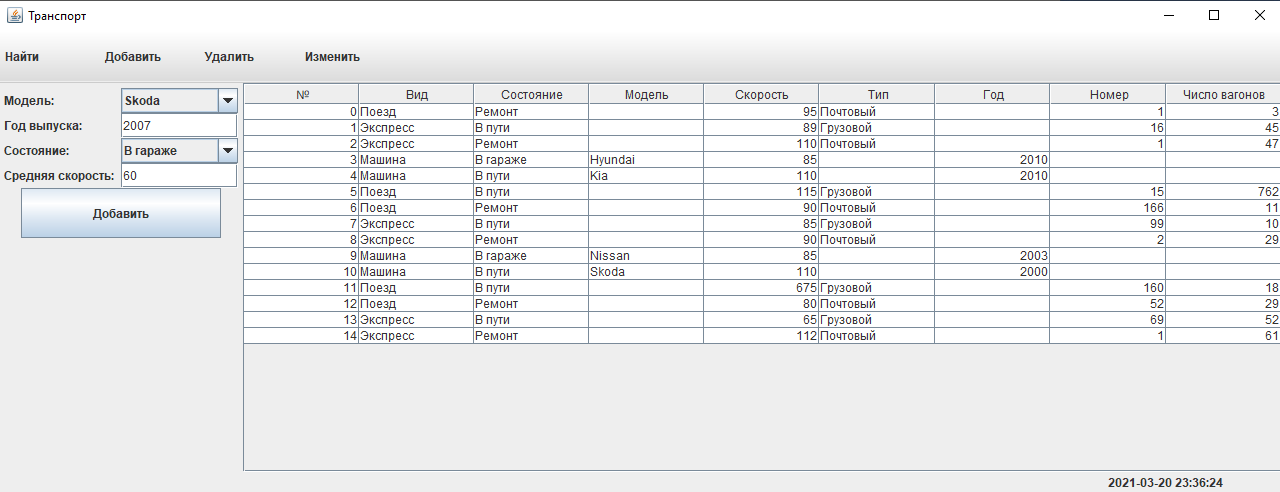
Действие 3.1 – Добавление новой записи – Машина – Открытие панели добавления машины с необходимыми полями, добавление записи в список

Рисунок 10 –Заполнение полей для добавления новой записи - машины

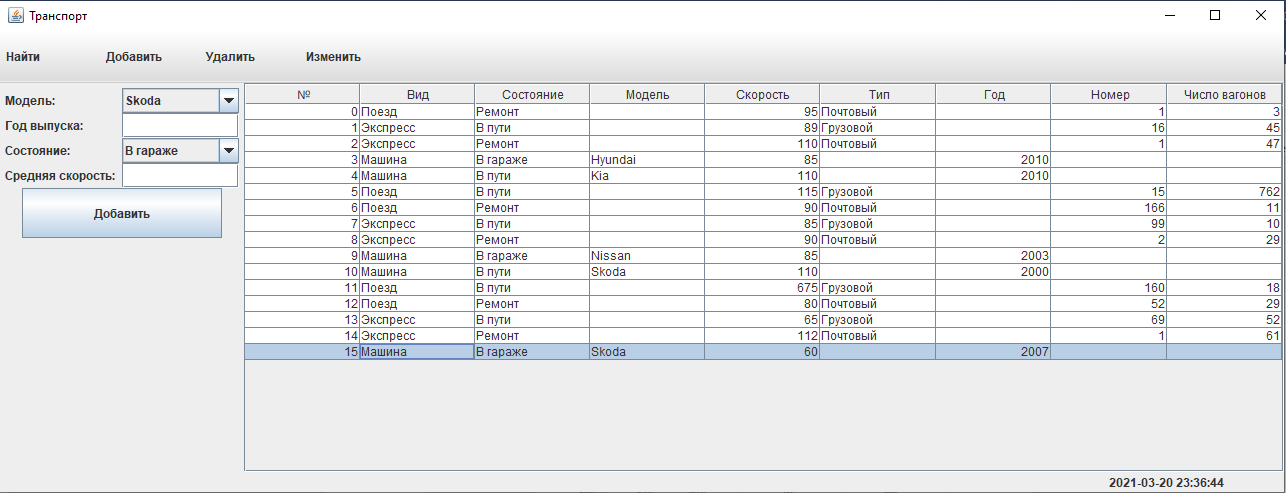


Рисунок 11 –Добавление новой записи – машины

Действие 3.2 – Добавление новой записи – Поезд – Открытие панели добавления поезд с необходимыми полями, добавление записи в список

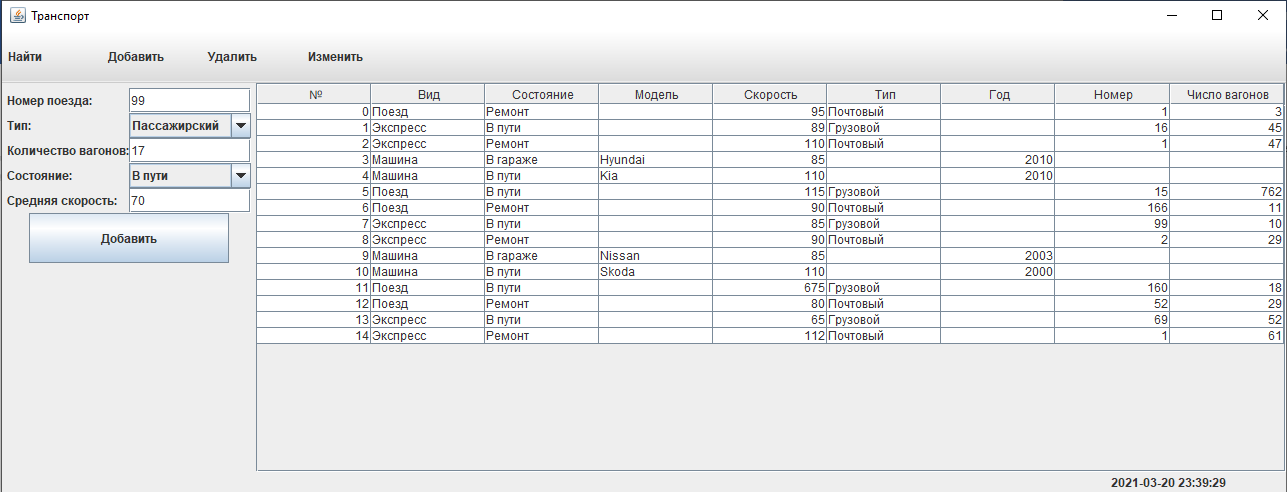


Рисунок 12 –Заполнение полей для добавления новой записи - поезда

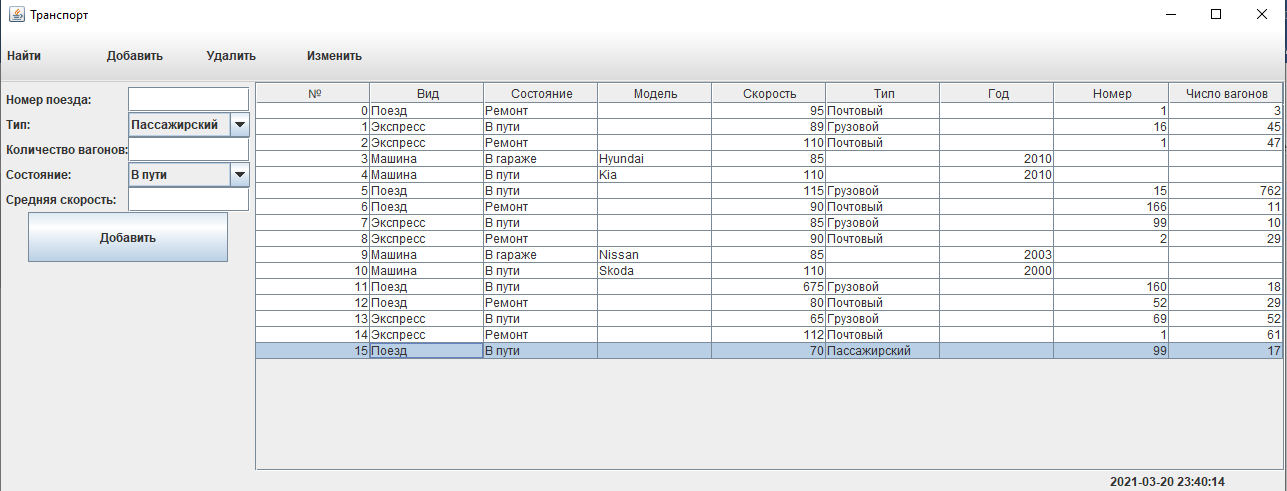


Рисунок 13 –Добавление новой записи – поезда

Действие 3.2 – Добавление новой записи – Экспресс – Открытие панели добавления экспресса с необходимыми полями, добавление записи в список

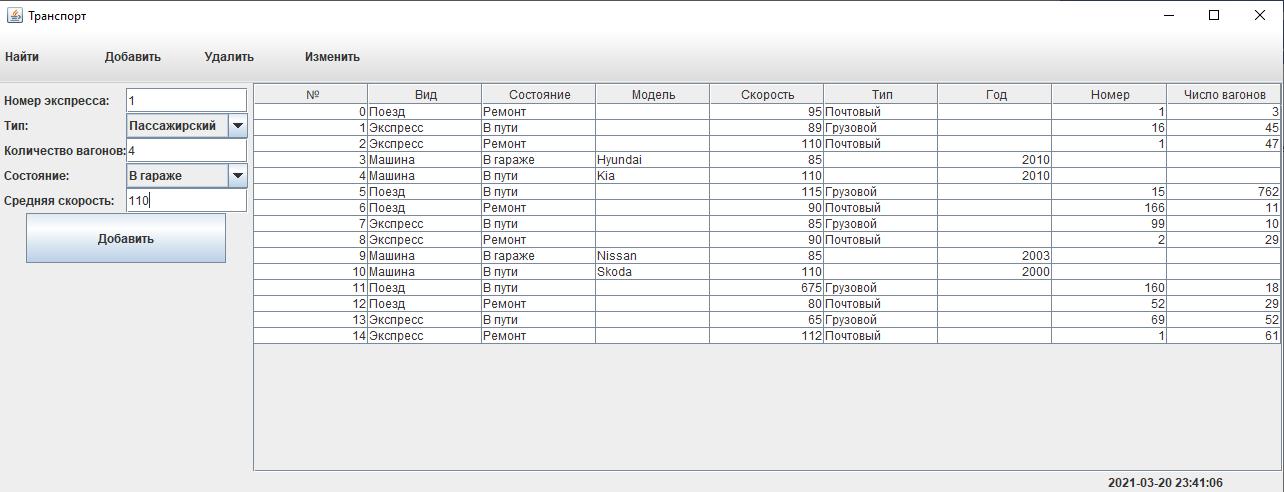


Рисунок 14 –Заполнение полей для добавления новой записи – экспресса

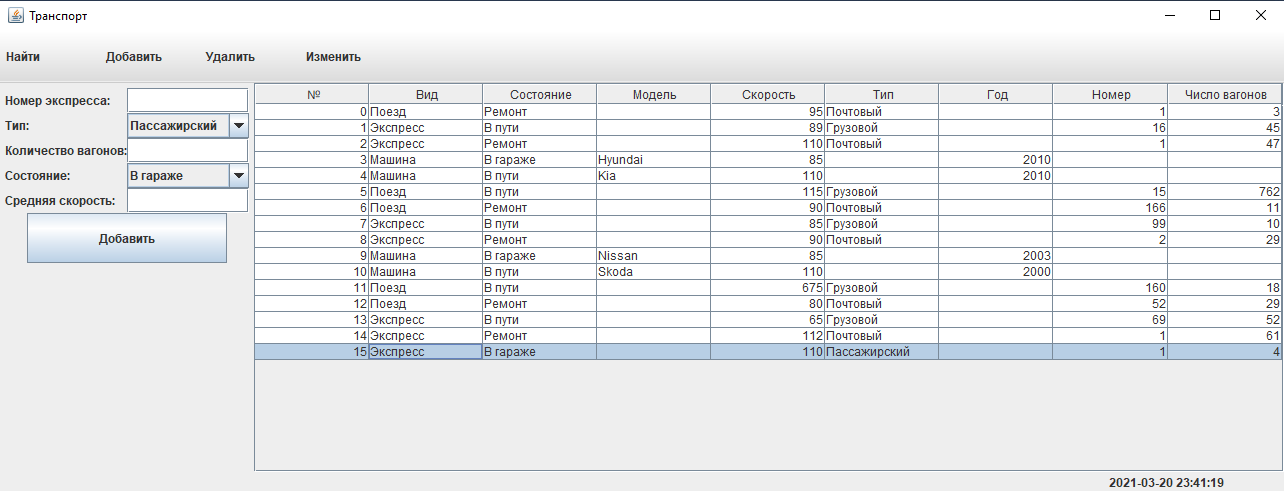


Рисунок 15 –Добавление новой записи – экспресса

Действие 4 – Поиск записи по № – Открытие панели поиска записи, ввод № искомой записи и открытие диалогового окна с найденной записи

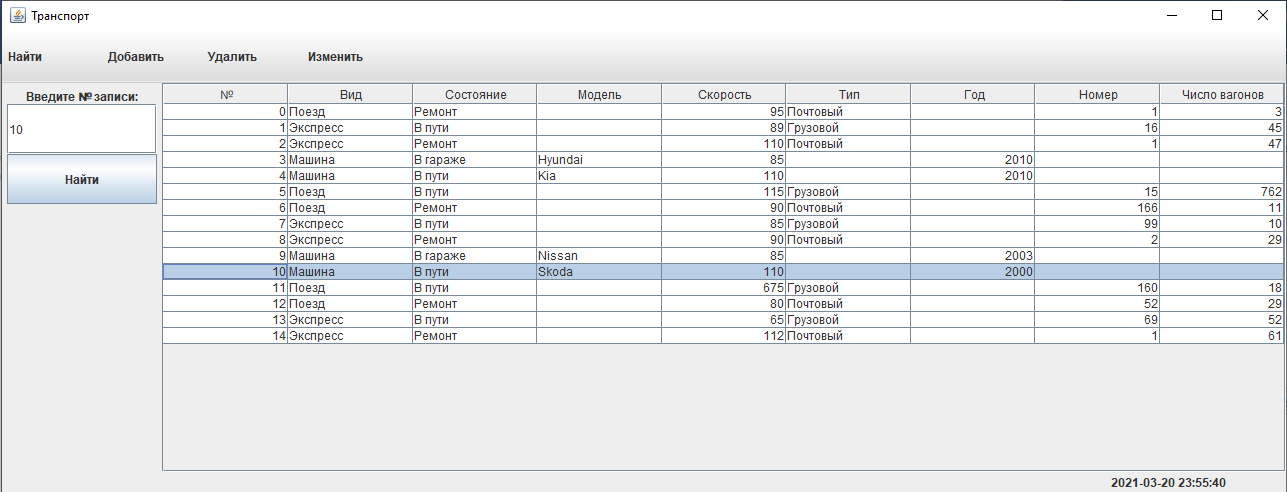


Рисунок 16 –– Открытие панели поиска записи, ввод № искомой записи

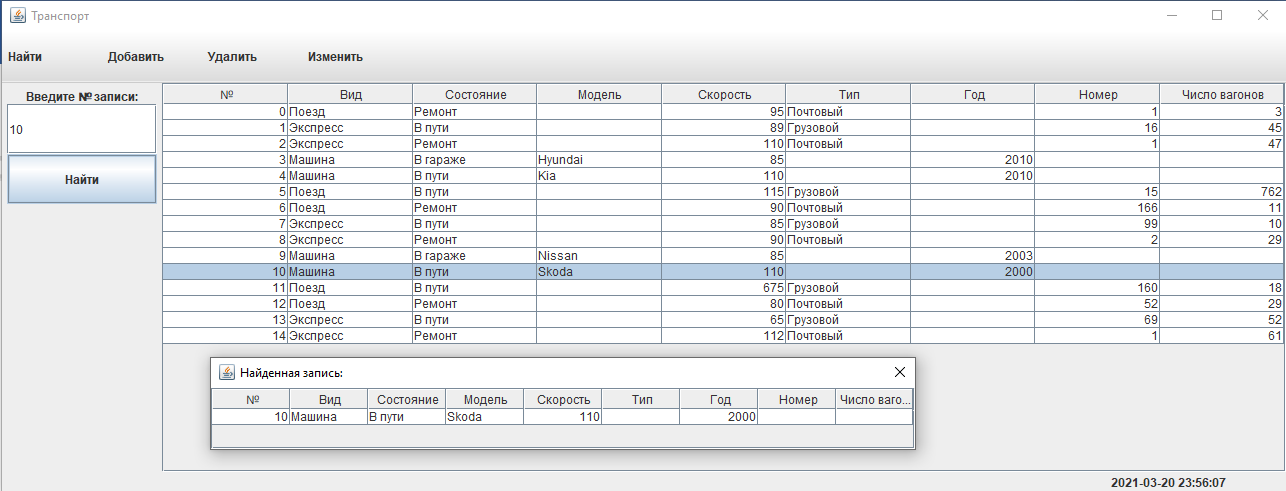


Рисунок 17 – Открытие диалогового окна с найденной записью

Действие 5.1 – Удаление записи – По № – Открытие панели удаления записи , ввод № удаляемой записи и удаление записи

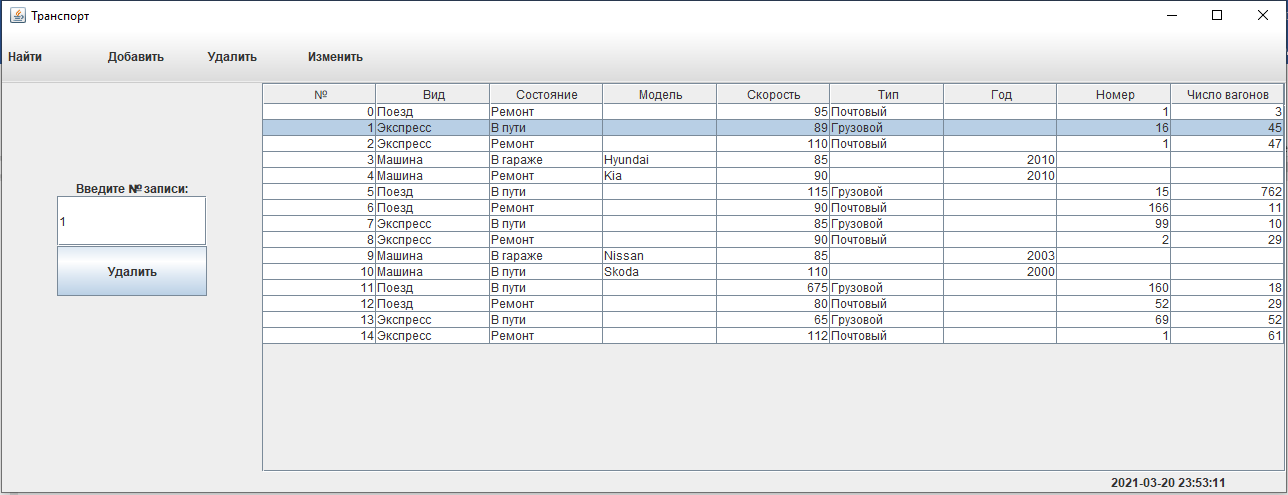


Рисунок 18 – Открытие панели удаления записи и ввод № удаляемой записи

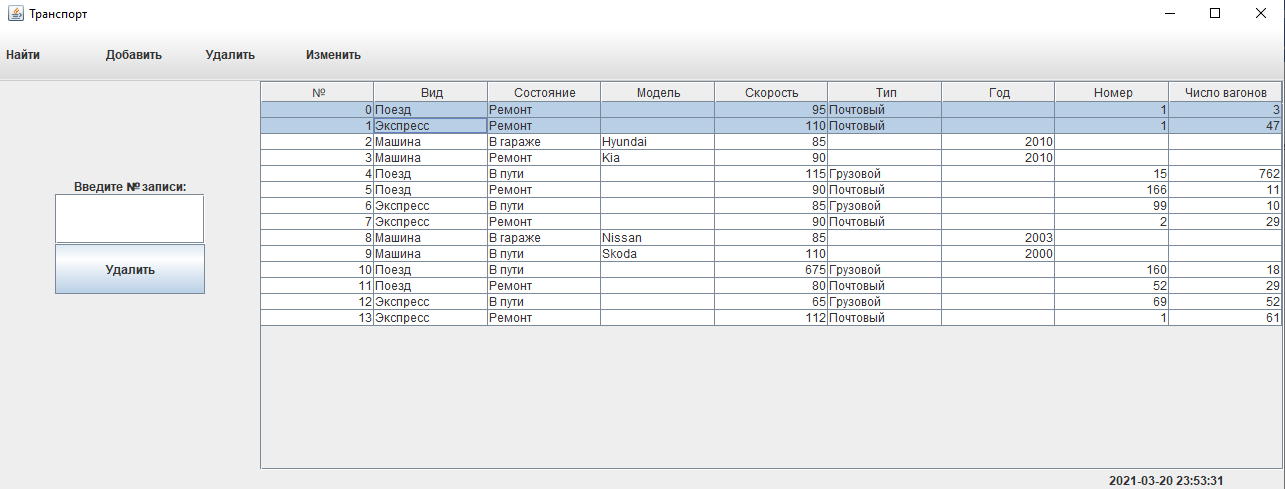


Рисунок 19 – Удаление записи

Действие 5.2 – Удаление записи – В таблице – Открытие панели удаления записи в таблице , выбор удаляемой записи и удаление записи

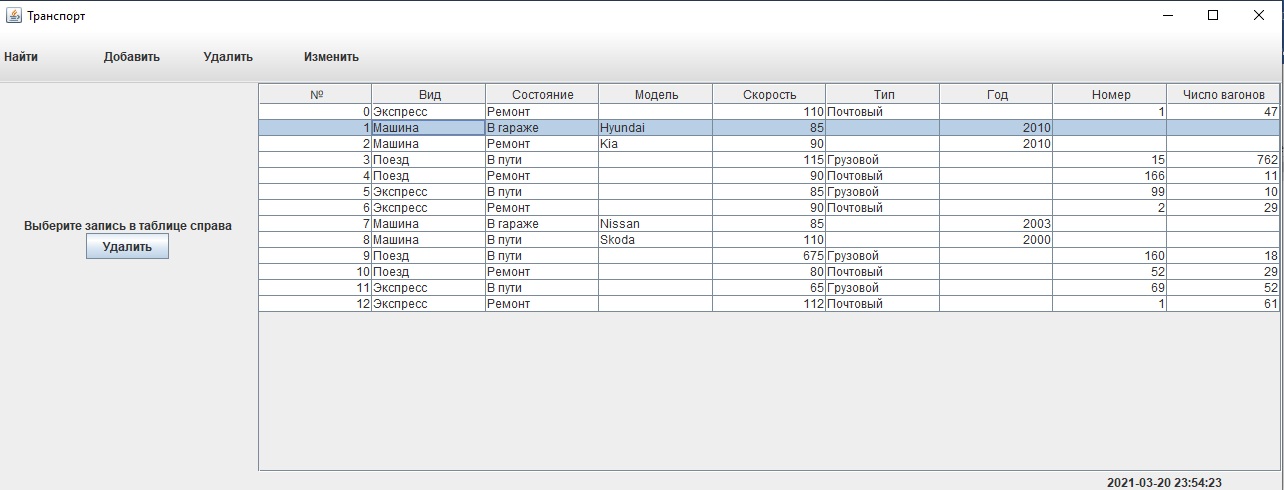


Рисунок 20 – Открытие панели удаления записи и выбор в таблице удаляемой записи

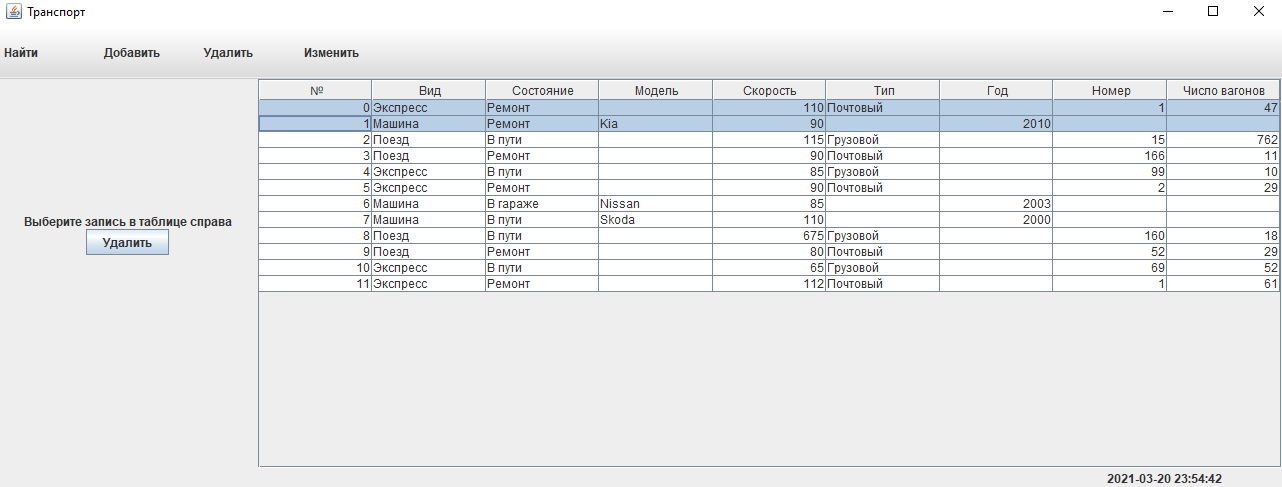
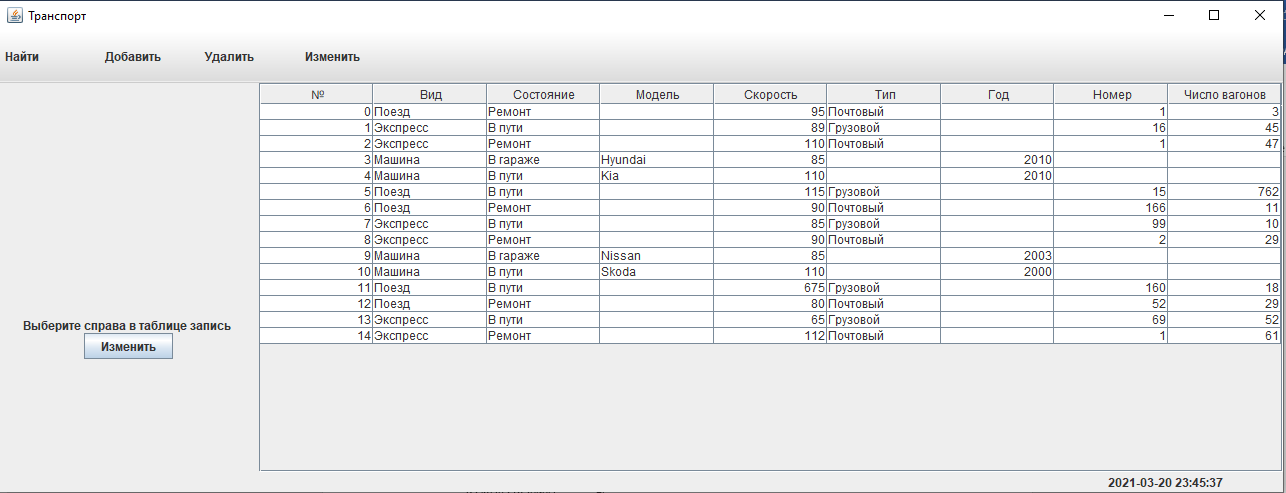


Рисунок 21 – Удаление записи

Действие 6 – Изменить запись в таблице – Открытие панели изменения записи ,выбора записи для изменения ,открытие панели с полями изменения записи и изменения записи

 Рисунок 22 – Открытие панели изменения записи , выбор записи для изменения

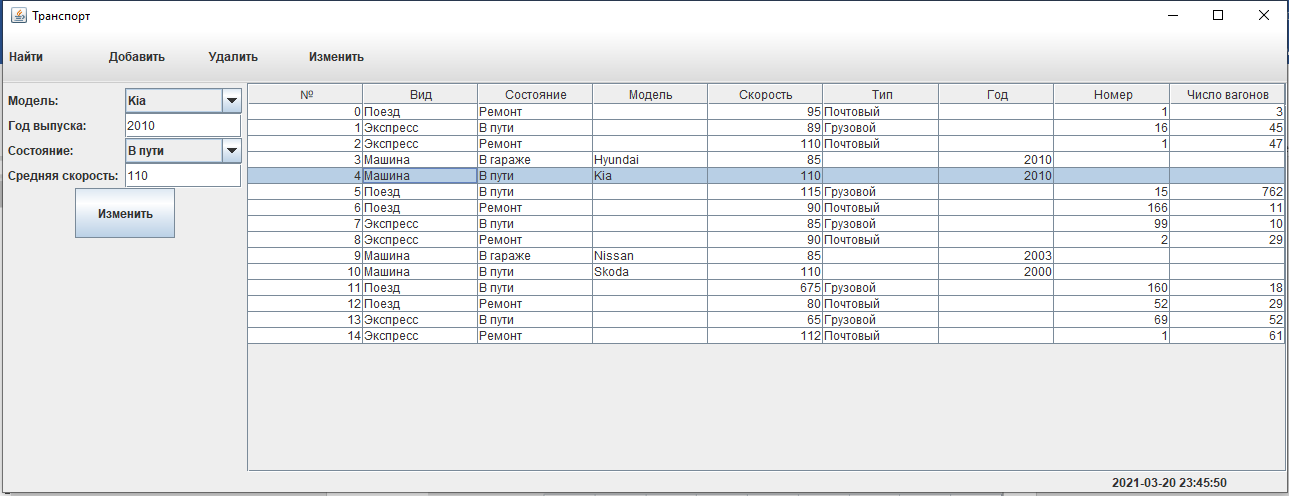


Рисунок 23 – Открытие панели с полями изменения записи

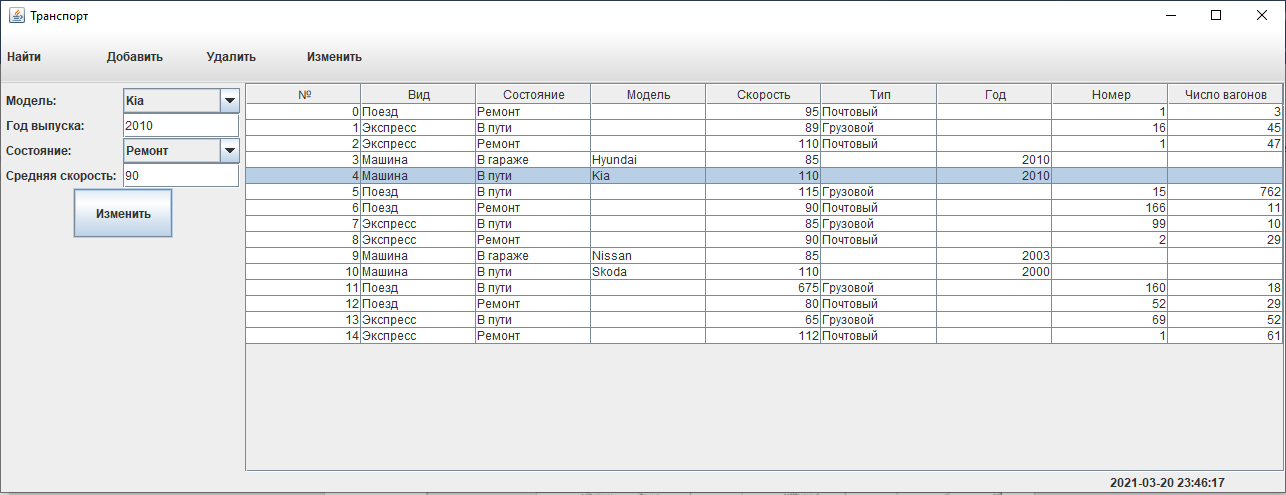


Рисунок 24 – Изменение полей «состояние» и «средняя скорость»

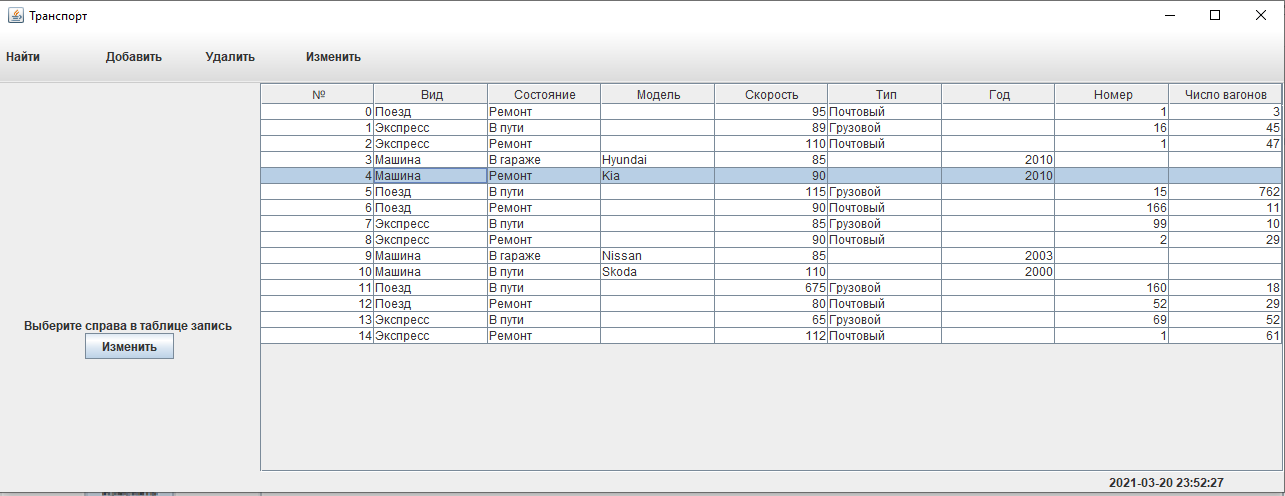


Рисунок 25 – Измененная запись

Действие 7 – Неверный ввод (вместо целых чисел буквы, символы или пустая строка) – Открытие диалогового окна с сообщением " Можно вводить только целые числа"

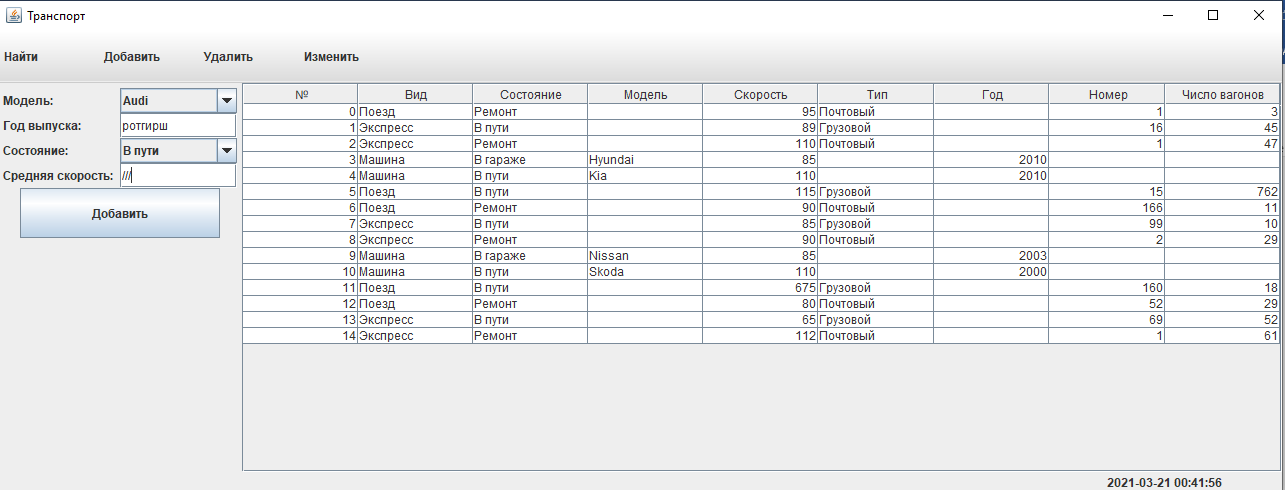


Рисунок 26 – Неверный ввод

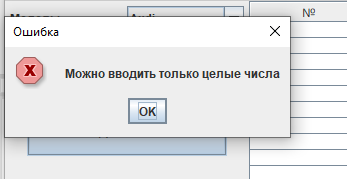
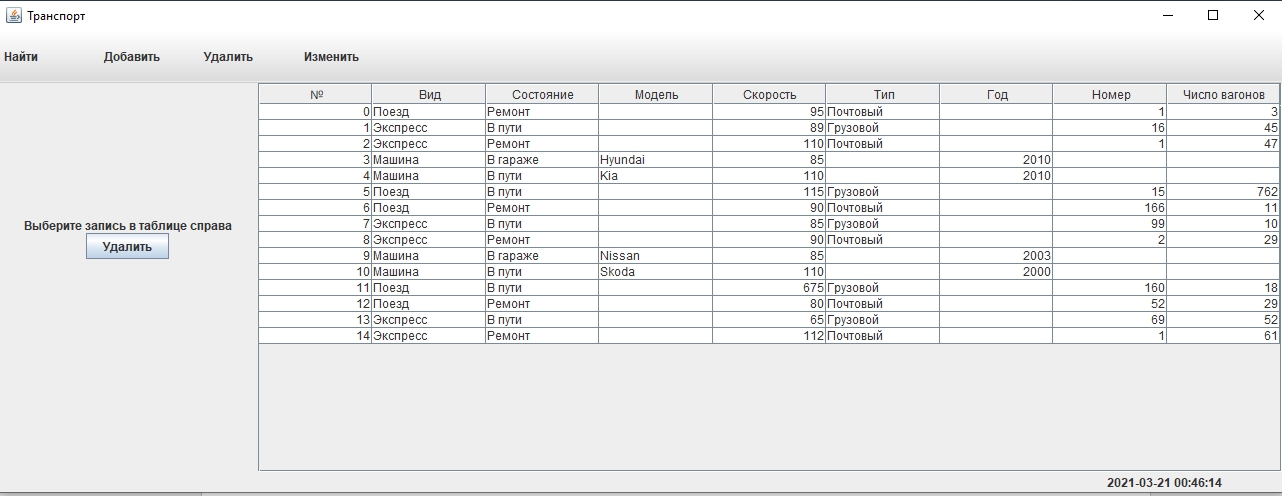


Рисунок 27 – Открытие диалогового окна с сообщением " Можно вводить только целые числа"

Действие 8 – Не выбрана запись – Открытие диалогового окна с сообщением " Вы ничего не выбрали"

 Рисунок 28 – Не выбрана запись

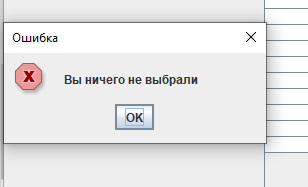


Рисунок 29 – Открытие диалогового окна с сообщением " Вы ничего не выбрали"

Действие 9 – Ввод номера не существующей записи – Открытие диалогового окна с сообщением “Нет такой записи”

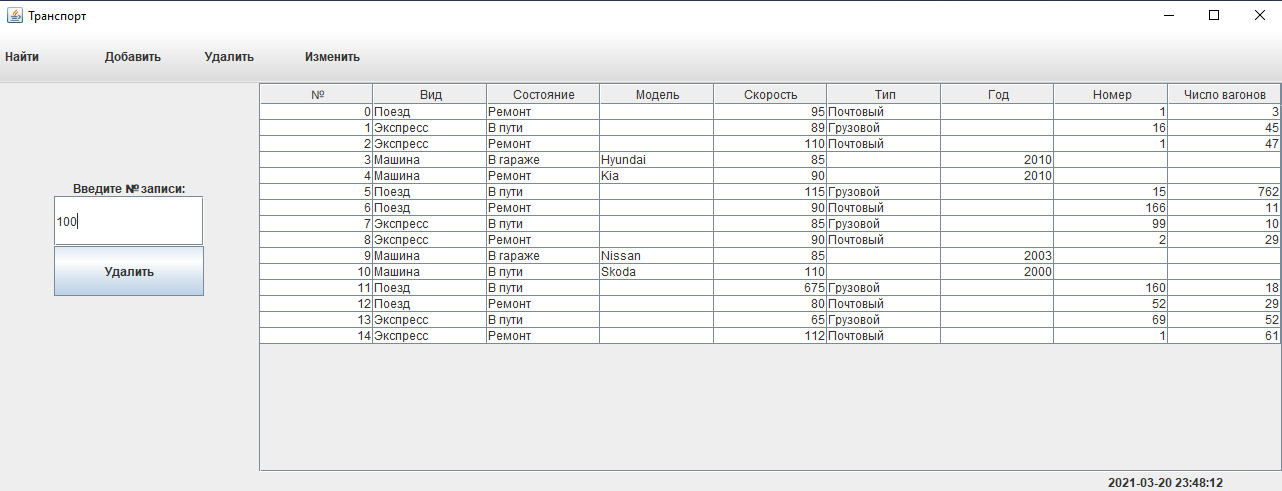


Рисунок 30 – Ввод номера не существующей записи

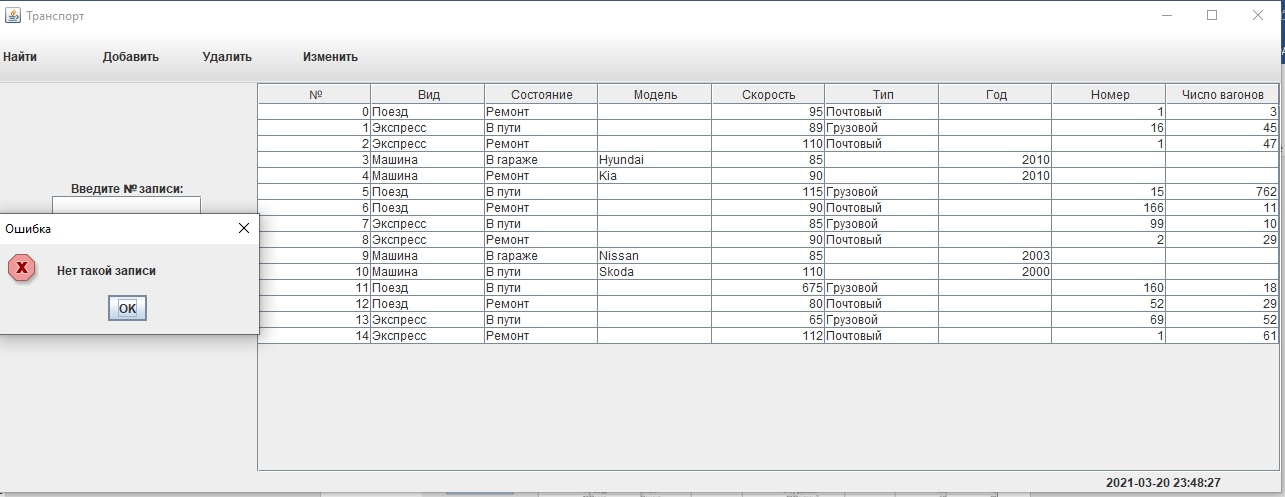


Рисунок 32 – Открытие диалогового окна с сообщением " Нет такой записи"

5 Листинг программы

## **5.1 класс Main**

**package** com.company.Interface;  
**import** javax.swing.\*;  
**public class** Main {  
 **public static void** main(String[] args) {  
 JFrame frame =**new** MyFrame();  
 frame.setDefaultCloseOperation( JFrame.***EXIT\_ON\_CLOSE*** );  
 frame.setVisible(**true**);  
 }  
}

## **5.2 класс MyFrame**

**package** com.company.Interface;  
**import** com.company.Logic.\*;  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
**import** java.text.SimpleDateFormat;  
**import** java.util.Date;  
  
**public class** MyFrame **extends** JFrame{  
  
 **private static** JPanel *PaneProcess*=**new** JPanel();  
 **private static** MyTableModel *Mytable*=**new** MyTableModel();  
 **private static** JTable *table*=**new** JTable(*Mytable*);  
  
 **public** MyFrame(){  
 **super**(**"Транспорт"**);  
 *// создаем строку главного меню* JMenuBar menuBar = **new** JMenuBar();  
 menuBar.add(createSubmenuFind());  
 menuBar.add(createSubmenuAdd());  
 menuBar.add(createSubmenuDel());  
 menuBar.add(createSubmenuChange());  
  
 JLabel j=**new** JLabel(**""**);  
 Timer timer = **new** Timer(1000,**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent arg0) {  
 Date current = **new** Date();  
 SimpleDateFormat formatter = **new** SimpleDateFormat(**"yyyy-MM-dd HH:mm:ss"**);  
 String message = formatter.format(current);  
 j.setText(message+**" "**);  
 j.repaint();  
 j.revalidate();  
 }  
 });  
 timer.start();  
 JPanel PaneTimer=**new** JPanel();  
 PaneTimer.add(j);  
 j.setHorizontalAlignment(SwingConstants.***RIGHT***);  
 j.setPreferredSize(**new** Dimension(100,20));  
 *//////заполняем frame* JScrollPane scrollPane=**new** JScrollPane(*table*);  
 scrollPane.setPreferredSize(**new** Dimension(650,300));  
 *PaneProcess*.add(**new** JLabel(**"Приветствуем вас в нашей программе!"**) );  
 *PaneProcess*.setMinimumSize(**new** Dimension(400,300));  
 setMinimumSize(**new** Dimension(1300,500));  
 **this**.setLayout(**new** BorderLayout());  
 add(menuBar,BorderLayout.***NORTH***);  
 add(scrollPane,BorderLayout.***CENTER***);  
 add(*PaneProcess*,BorderLayout.***WEST***);  
 add(j,BorderLayout.***SOUTH***);  
 scrollPane.setVisible(**true**);  
 *PaneProcess*.setVisible(**true**);  
 menuBar.setVisible(**true**);  
 PaneTimer.setVisible(**true**);  
 setDefaultCloseOperation( JFrame.***EXIT\_ON\_CLOSE*** );  
 setVisible(**true**);  
 }  
 **private** JMenu createSubmenuAdd() {  
 JMenu text = **new** JMenu(**"Добавить"**);  
 text.setPreferredSize(**new** Dimension(100,50));  
 JMenuItem AddCar = **new** JMenuItem(**"Машина"**);  
 JMenuItem AddTrain = **new** JMenuItem(**"Поезд"**);  
 JMenuItem AddExpress = **new** JMenuItem(**"Экспресс"**);  
 AddCar.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *PaneProcess*.removeAll();  
 *CreatePaneAddCar*();  
 *PaneProcess*.repaint();  
 *PaneProcess*.revalidate();  
 }  
 });  
 AddTrain.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *PaneProcess*.removeAll();  
 *CreatePaneAddTr*();  
 *PaneProcess*.repaint();  
 *PaneProcess*.revalidate();  
 }  
 });  
 AddExpress.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *PaneProcess*.removeAll();  
 *CreatePaneAddEx*();  
 *PaneProcess*.repaint();  
 *PaneProcess*.revalidate();  
 }  
 });  
 text.add(AddCar);  
 text.add(AddTrain);  
 text.add(AddExpress);  
 **return** text;  
 }  
 **private** JMenu createSubmenuDel() {  
 JMenu text = **new** JMenu(**"Удалить"**);  
 text.setPreferredSize(**new** Dimension(100,50));  
 *// и несколько вложенных меню* JMenuItem DelId = **new** JMenuItem(**"По № записи"**);  
 JMenuItem Del = **new** JMenuItem(**"В таблице"**);  
 DelId.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *PaneProcess*.removeAll();  
 *CreatePaneDeleteId*();  
 *PaneProcess*.repaint();  
 *PaneProcess*.revalidate();  
 }  
 });  
 Del.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *PaneProcess*.removeAll();  
 *CreatePaneDelete*();  
 *PaneProcess*.repaint();  
 *PaneProcess*.revalidate();  
 }  
 });  
 text.add(DelId);  
 text.add(Del);  
 **return** text;  
 }  
 **private** JMenu createSubmenuFind(){  
 JMenu text = **new** JMenu(**"Найти"**);  
 text.setPreferredSize(**new** Dimension(100,50));  
 *// и несколько вложенных меню* JMenuItem FindId = **new** JMenuItem(**"По № записи"**);  
 FindId.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *PaneProcess*.removeAll();  
 *CreatePaneFindId*();  
 *PaneProcess*.repaint();  
 *PaneProcess*.revalidate();  
 }  
 });  
 text.add(FindId);  
 **return** text;  
 }  
 **private** JMenu createSubmenuChange(){  
 JMenu text=**new** JMenu(**"Изменить"**);  
 JMenuItem t=**new** JMenuItem(**"В таблице"**);  
 text.setPreferredSize(**new** Dimension(100,50));  
 t.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *PaneProcess*.removeAll();  
 *CreatePaneChange*();  
 *PaneProcess*.repaint();  
 *PaneProcess*.revalidate();  
 }  
 });  
 text.add(t);  
 **return** text;  
 }  
  
 **private static void** CreatePaneChangeCar(){  
 JPanel ChCar=**new** JPanel();  
 Transport tr=*Mytable*.getValueAt(*table*.getSelectedRow());  
 **int** id=tr.getId();  
 ChCar.setLayout(**new** GridBagLayout());  
 GridBagConstraints constraints = **new** GridBagConstraints();  
 *// По умолчанию натуральная высота, максимальная ширина* constraints.**fill** = GridBagConstraints.***VERTICAL***;  
 constraints.**weightx** = 0.5;  
 constraints.**gridx** = 0; *// нулевая ячейка таблицы по вертикали* constraints.**gridy** = 0;  
 JPanel Pane1 = **new** JPanel(**new** GridLayout(4, 2));  
 JComboBox model = **new** JComboBox(**new** String[]{**"Audi"**, **"BMW"**, **"Ford"**, **"Honda"**, **" Hyundai"**, **"Kia"**, **"Lada(ВАЗ)"**, **"Mazda"**, **"Mercedes-Benz"**, **"Mitsubishi"**, **"Nissan"**, **"Renault"**, **"Skoda"**, **"Toyota"**});  
 model.setSelectedItem(((CarTransport) tr).getModel());  
 JTextField years = **new** JTextField();  
 years.setText(String.*valueOf*(((CarTransport) tr).getYears()));  
 JComboBox condition = **new** JComboBox(**new** String[]{**"В пути"**, **"В гараже"**, **"Ремонт"**});  
 condition.setSelectedItem(tr.getCondition());  
 JTextField speed = **new** JTextField();  
 speed.setText(String.*valueOf*(tr.getSpeed()));  
 Pane1.add(**new** JLabel(**"Модель:"**));  
 Pane1.add(model);  
 Pane1.add(**new** JLabel(**"Год выпуска:"**));  
 Pane1.add(years);  
 Pane1.add(**new** JLabel(**"Состояние:"**));  
 Pane1.add(condition);  
 Pane1.add(**new** JLabel(**"Средняя скорость:"**));  
 Pane1.add(speed);  
 ChCar.add(Pane1, constraints);  
 JButton CHCAR = **new** JButton(**"Изменить"**);  
 CHCAR.setAlignmentX(***CENTER\_ALIGNMENT***);  
 constraints.**gridy** = 1;  
 CHCAR.setPreferredSize(**new** Dimension(100, 50));  
 ChCar.add(CHCAR, constraints);  
 *PaneProcess*.add(ChCar);  
 CHCAR.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **try**{  
 *Mytable*.setValueAt(**new** CarTransport(String.*valueOf*(model.getSelectedItem()),Integer.*valueOf*(years.getText()),String.*valueOf*(condition.getSelectedItem()),Integer.*valueOf*(speed.getText())),id);  
 *PaneProcess*.removeAll();  
 *CreatePaneChange*();  
 *PaneProcess*.repaint();  
 *PaneProcess*.revalidate();  
 }**catch** (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(*PaneProcess*,  
 **" Можно вводить только целые числа"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 }  
 years.setText(**""**);  
 speed.setText(**""**);  
 }  
 });  
 }  
 **private static void** CreatePaneChangeTr(){  
 JPanel ChTr=**new** JPanel();  
 Transport tr=*Mytable*.getValueAt(*table*.getSelectedRow());  
 **int** id=tr.getId();  
 ChTr.setLayout(**new** GridBagLayout());  
 GridBagConstraints constraints = **new** GridBagConstraints();  
 *// По умолчанию натуральная высота, максимальная ширина* constraints.**fill** = GridBagConstraints.***VERTICAL***;  
 constraints.**weightx** = 0.5;  
 constraints.**gridx** = 0 ; *// нулевая ячейка таблицы по вертикали* constraints.**gridy** = 0;  
 JPanel Pane1=**new** JPanel(**new** GridLayout(5,2));  
 JTextField number=**new** JTextField();  
 number.setText(String.*valueOf*(((TrainTransport) tr).getNumber()));  
 JComboBox type=**new** JComboBox(**new** String[] {**"Грузовой"**, **"Пассажирский"**, **"Почтовый"**});  
 type.setSelectedItem(((TrainTransport) tr).getType());  
 JTextField count=**new** JTextField();  
 count.setText(String.*valueOf*(((TrainTransport) tr).getCarriage()));  
 JComboBox condition=**new** JComboBox(**new** String[] {**"В пути"**, **"В гараже"**, **"Ремонт"**});  
 condition.setSelectedItem(tr.getCondition());  
 JTextField speed=**new** JTextField();  
 speed.setText(String.*valueOf*(tr.getSpeed()));  
 Pane1.add(**new** JLabel(**"Номер поезда:"**));Pane1.add(number);  
 Pane1.add(**new** JLabel(**"Тип:"**));Pane1.add(type);  
 Pane1.add(**new** JLabel(**"Количество вагонов:"**));Pane1.add(count);  
 Pane1.add(**new** JLabel(**"Состояние:"**));Pane1.add(condition);  
 Pane1.add(**new** JLabel(**"Средняя скорость:"**));Pane1.add(speed);  
 ChTr.add(Pane1,constraints);  
 JButton CHTR=**new** JButton(**"Изменить"**);  
 CHTR.setAlignmentX(***CENTER\_ALIGNMENT***);  
 constraints.**gridy**=1;  
 CHTR.setPreferredSize(**new** Dimension(200,50));  
 ChTr.add(CHTR,constraints);  
 *PaneProcess*.add(ChTr);  
 CHTR.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **try**{  
 Transport t=**new** TrainTransport(Integer.*valueOf*(number.getText()),String.*valueOf*(type.getSelectedItem()),Integer.*valueOf*(count.getText()),String.*valueOf*(condition.getSelectedItem()),Integer.*valueOf*(speed.getText()));  
 *Mytable*.setValueAt(t,id);  
 *PaneProcess*.removeAll();  
 *CreatePaneChange*();  
 *PaneProcess*.repaint();  
 *PaneProcess*.revalidate();  
 }**catch** (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(*PaneProcess*,  
 **" Можно вводить только целые числа"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 }  
 number.setText(**""**);  
 count.setText(**""**);  
 speed.setText(**""**);  
 }  
 });  
 }  
 **private static void** CreatePaneChangeEx(){  
 JPanel ChEx=**new** JPanel();  
 Transport tr=*Mytable*.getValueAt(*table*.getSelectedRow());  
 **int** id=tr.getId();  
 ChEx.setLayout(**new** GridBagLayout());  
 GridBagConstraints constraints = **new** GridBagConstraints();  
 *// По умолчанию натуральная высота, максимальная ширина* constraints.**fill** = GridBagConstraints.***VERTICAL***;  
 constraints.**weightx** = 0.5;  
 constraints.**gridx** = 0 ; *// нулевая ячейка таблицы по вертикали* constraints.**gridy** = 0;  
 JPanel Pane1=**new** JPanel(**new** GridLayout(5,2));  
 JTextField number=**new** JTextField();  
 number.setText(String.*valueOf*(((TrainTransport) tr).getNumber()));  
 JComboBox type=**new** JComboBox(**new** String[] {**"Грузовой"**, **"Пассажирский"**, **"Почтовый"**});  
 type.setSelectedItem(((TrainTransport) tr).getType());  
 JTextField count=**new** JTextField();  
 count.setText(String.*valueOf*(((TrainTransport) tr).getCarriage()));  
 JComboBox condition=**new** JComboBox(**new** String[] {**"В пути"**, **"В гараже"**, **"Ремонт"**});  
 condition.setSelectedItem(tr.getCondition());  
 JTextField speed=**new** JTextField();  
 speed.setText(String.*valueOf*(tr.getSpeed()));  
 Pane1.add(**new** JLabel(**"Номер экспресса:"**));Pane1.add(number);  
 Pane1.add(**new** JLabel(**"Тип:"**));Pane1.add(type);  
 Pane1.add(**new** JLabel(**"Количество вагонов:"**));Pane1.add(count);  
 Pane1.add(**new** JLabel(**"Состояние:"**));Pane1.add(condition);  
 Pane1.add(**new** JLabel(**"Средняя скорость:"**));Pane1.add(speed);  
 ChEx.add(Pane1,constraints);  
 JButton ADDEX=**new** JButton(**"Изменить"**);  
 ADDEX.setAlignmentX(***CENTER\_ALIGNMENT***);  
 constraints.**gridy**=1;  
 ADDEX.setPreferredSize(**new** Dimension(200,50));  
 ChEx.add(ADDEX,constraints);  
 *PaneProcess*.add(ChEx);  
 ADDEX.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **try**{  
 *Mytable*.setValueAt(**new** ExpressTrain(Integer.*valueOf*(number.getText()),String.*valueOf*(type.getSelectedItem()),Integer.*valueOf*(count.getText()),String.*valueOf*(condition.getSelectedItem()),Integer.*valueOf*(speed.getText())),id);  
 *PaneProcess*.removeAll();  
 *CreatePaneChange*();  
 *PaneProcess*.repaint();  
 *PaneProcess*.revalidate();  
 }**catch** (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(*PaneProcess*,  
 **" Можно вводить только целые числа"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 }  
 number.setText(**""**);  
 count.setText(**""**);  
 speed.setText(**""**);  
 }  
 });  
 }  
 **private static void** CreatePaneChange(){  
 JPanel PaneCh=**new** JPanel();  
 PaneCh.setLayout(**new** GridBagLayout());  
 GridBagConstraints constraints = **new** GridBagConstraints();  
 *// По умолчанию натуральная высота, максимальная ширина* constraints.**fill** = GridBagConstraints.***VERTICAL***;  
 constraints.**weightx** = 0.5;  
 constraints.**gridx** = 0 ; *// нулевая ячейка таблицы по вертикали* constraints.**gridy**=0;  
 JLabel t=**new** JLabel(**"Выберите справа в таблице запись "**);  
 PaneCh.add(t,constraints);  
 constraints.**gridy**=1;  
 JButton t1=**new** JButton(**"Изменить"** );  
 PaneCh.setPreferredSize(**new** Dimension(250,500));  
 PaneCh.add(t1,constraints);  
 *PaneProcess*.add(PaneCh);  
 t1.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **int** row=*table*.getSelectedRow();  
 **if** (row==-1){  
 JOptionPane.*showMessageDialog*(*PaneProcess*,  
 **" Вы ничего не выбрали"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 **return**;  
 }**else** {  
 *PaneProcess*.removeAll();  
 **if**(*Mytable*.getValueAt(*table*.getSelectedRow()) **instanceof** CarTransport){  
 *CreatePaneChangeCar*();  
 }**else if**(*Mytable*.getValueAt(*table*.getSelectedRow()) **instanceof** ExpressTrain){  
 *CreatePaneChangeEx*();  
 } **else if**(*Mytable*.getValueAt(*table*.getSelectedRow()) **instanceof** TrainTransport){  
 *CreatePaneChangeTr*();  
 }  
 *PaneProcess*.repaint();  
 *PaneProcess*.revalidate();  
 }  
 }  
 });  
 }  
  
 **private static void** CreatePaneDelete(){  
 JPanel PaneDelete=**new** JPanel();  
 PaneDelete.setLayout(**new** GridBagLayout());  
 GridBagConstraints constraints = **new** GridBagConstraints();  
 constraints.**weightx** = 0.5;  
 constraints.**gridx** = 0 ; *// нулевая ячейка таблицы по вертикали* constraints.**gridy**=0;  
 JLabel t=**new** JLabel(**"Выберите запись в таблице справа"**);  
 JButton DEL=**new** JButton(**"Удалить"**);  
 DEL.setAlignmentX(***CENTER\_ALIGNMENT***);  
 PaneDelete.setPreferredSize(**new** Dimension(250,300));  
 PaneDelete.add(t,constraints);  
 constraints.**gridy**=1;  
 PaneDelete.add(DEL,constraints);  
 PaneDelete.repaint();  
 PaneDelete.revalidate();  
 DEL.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **int** row=*table*.getSelectedRow();  
 **if** (row==-1){  
 JOptionPane.*showMessageDialog*(*PaneProcess*,  
 **" Вы ничего не выбрали"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 **return**;  
 }  
 *Mytable*.DeleteValueAt(row);  
 }  
 });  
 *PaneProcess*.add(PaneDelete);  
 }  
 **private static void** CreatePaneDeleteId(){  
 JPanel PaneDeleteID=**new** JPanel();  
 PaneDeleteID.setLayout(**new** GridBagLayout());  
 GridBagConstraints constraints = **new** GridBagConstraints();  
 constraints.**weightx** = 0.5;  
 constraints.**gridx** = 0 ; *// нулевая ячейка таблицы по вертикали* JTextField id=**new** JTextField();  
 id.setPreferredSize(**new** Dimension(150,50));  
 JButton DELID=**new** JButton(**"Удалить"**);  
 DELID.setPreferredSize(**new** Dimension(150,50));  
 PaneDeleteID.setPreferredSize(**new** Dimension(250,300));  
 DELID.setAlignmentX(***CENTER\_ALIGNMENT***);  
 constraints.**gridy** = 0;  
 PaneDeleteID.add(**new** JLabel(**"Введите № записи:"**),constraints);  
 constraints.**gridy** = 1;  
 PaneDeleteID.add(id,constraints);  
 constraints.**gridy** = 2;  
 PaneDeleteID.add(DELID,constraints);  
 PaneDeleteID.repaint();  
 PaneDeleteID.revalidate();  
 *PaneProcess*.add(PaneDeleteID);  
 DELID.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **try**{  
 **if** (Integer.*parseInt*(id.getText())<*Mytable*.getRowCount() & Integer.*parseInt*(id.getText())>0){  
 *Mytable*.DeleteValueAt(Integer.*parseInt*(id.getText()));  
 }**else** {  
 JOptionPane.*showMessageDialog*(*PaneProcess*,  
 **" Нет такой записи"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 **return**;  
 }  
 }**catch** (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(PaneDeleteID,  
 **" Можно вводить только целые числа"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 }  
 id.setText(**""**);  
 }  
 });  
 }  
  
 **private static void** CreatePaneAddCar(){  
 JPanel PaneAddCar=**new** JPanel();  
 PaneAddCar.setLayout(**new** GridBagLayout());  
 GridBagConstraints constraints = **new** GridBagConstraints();  
 constraints.**weightx** = 0.5;  
 constraints.**gridx** = 0 ;  
 constraints.**gridy** = 0;  
 JPanel Pane1=**new** JPanel(**new** GridLayout(4,2));  
 JComboBox model=**new** JComboBox(**new** String[] {**"Audi"**, **"BMW"**, **"Ford"**, **"Honda"**, **" Hyundai"**, **"Kia"**, **"Lada(ВАЗ)"**, **"Mazda"**, **"Mercedes-Benz"**, **"Mitsubishi"**, **"Nissan"**, **"Renault"**, **"Skoda"**, **"Toyota"**});  
 JTextField years=**new** JTextField();  
 JComboBox condition=**new** JComboBox(**new** String[] {**"В пути"**, **"В гараже"**, **"Ремонт"**});  
 JTextField speed=**new** JTextField();  
 Pane1.add(**new** JLabel(**"Модель:"**));Pane1.add(model);  
 Pane1.add(**new** JLabel(**"Год выпуска:"**));Pane1.add(years);  
 Pane1.add(**new** JLabel(**"Состояние:"**));Pane1.add(condition);  
 Pane1.add(**new** JLabel(**"Средняя скорость:"**));Pane1.add(speed);  
 PaneAddCar.add(Pane1,constraints);  
 JButton ADDCAR=**new** JButton(**"Добавить"**);  
 ADDCAR.setAlignmentX(***CENTER\_ALIGNMENT***);  
 constraints.**gridy**=1;  
 ADDCAR.setPreferredSize(**new** Dimension(200,50));  
 PaneAddCar.add(ADDCAR,constraints);  
 *PaneProcess*.add(PaneAddCar);  
 ADDCAR.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **try**{  
 Integer.*parseInt*(years.getText());  
 Integer.*parseInt*(speed.getText());  
 Transport t=**new** CarTransport(String.*valueOf*(model.getSelectedItem()),Integer.*parseInt*(years.getText()),String.*valueOf*(condition.getSelectedItem()),Integer.*parseInt*(speed.getText()));  
 *Mytable*.AddValueAt(t);  
 }**catch** (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(PaneAddCar,  
 **" Можно вводить только целые числа"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 }  
 years.setText(**""**);  
 speed.setText(**""**);  
 }  
 });  
 }  
 **private static void** CreatePaneAddTr(){  
 JPanel PaneAddTrain=**new** JPanel();  
 PaneAddTrain.setLayout(**new** GridBagLayout());  
 GridBagConstraints constraints = **new** GridBagConstraints();  
 constraints.**fill** = GridBagConstraints.***VERTICAL***;  
 constraints.**weightx** = 0.5;  
 constraints.**gridx** = 0 ;  
 constraints.**gridy** = 0;  
 JPanel Pane1=**new** JPanel(**new** GridLayout(5,2));  
 JTextField number=**new** JTextField();  
 JComboBox type=**new** JComboBox(**new** String[] {**"Грузовой"**, **"Пассажирский"**, **"Почтовый"**});  
 JTextField count=**new** JTextField();  
 JComboBox condition=**new** JComboBox(**new** String[] {**"В пути"**, **"В гараже"**, **"Ремонт"**});  
 JTextField speed=**new** JTextField();  
 Pane1.add(**new** JLabel(**"Номер поезда:"**));Pane1.add(number);  
 Pane1.add(**new** JLabel(**"Тип:"**));Pane1.add(type);  
 Pane1.add(**new** JLabel(**"Количество вагонов:"**));Pane1.add(count);  
 Pane1.add(**new** JLabel(**"Состояние:"**));Pane1.add(condition);  
 Pane1.add(**new** JLabel(**"Средняя скорость:"**));Pane1.add(speed);  
 PaneAddTrain.add(Pane1,constraints);  
 JButton ADDTR=**new** JButton(**"Добавить"**);  
 ADDTR.setAlignmentX(***CENTER\_ALIGNMENT***);  
 constraints.**gridy**=1;  
 ADDTR.setPreferredSize(**new** Dimension(200,50));  
 PaneAddTrain.add(ADDTR,constraints);  
 *PaneProcess*.add(PaneAddTrain);  
 ADDTR.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **try**{  
 Integer.*parseInt*(number.getText());  
 Integer.*parseInt*(count.getText());  
 Integer.*parseInt*(speed.getText());  
 *Mytable*.AddValueAt(**new** TrainTransport(Integer.*valueOf*(number.getText()),String.*valueOf*(type.getSelectedItem()),Integer.*valueOf*(count.getText()),String.*valueOf*(condition.getSelectedItem()),Integer.*valueOf*(speed.getText())));  
 }**catch** (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(PaneAddTrain,  
 **" Можно вводить только целые числа"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 }  
 number.setText(**""**);  
 count.setText(**""**);  
 speed.setText(**""**);  
 }  
 });  
 }  
 **private static void** CreatePaneAddEx(){  
 JPanel PaneAddExp=**new** JPanel();  
 PaneAddExp.setLayout(**new** GridBagLayout());  
 GridBagConstraints constraints = **new** GridBagConstraints();  
 constraints.**weightx** = 0.5;  
 constraints.**gridx** = 0 ;  
 constraints.**gridy** = 0;  
 JPanel Pane1=**new** JPanel(**new** GridLayout(5,2));  
 JTextField number=**new** JTextField();  
 JComboBox type=**new** JComboBox(**new** String[] {**"Грузовой"**, **"Пассажирский"**, **"Почтовый"**});  
 JTextField count=**new** JTextField();  
 JComboBox condition=**new** JComboBox(**new** String[] {**"В пути"**, **"В гараже"**, **"Ремонт"**});  
 JTextField speed=**new** JTextField();  
 Pane1.add(**new** JLabel(**"Номер экспресса:"**));Pane1.add(number);  
 Pane1.add(**new** JLabel(**"Тип:"**));Pane1.add(type);  
 Pane1.add(**new** JLabel(**"Количество вагонов:"**));Pane1.add(count);  
 Pane1.add(**new** JLabel(**"Состояние:"**));Pane1.add(condition);  
 Pane1.add(**new** JLabel(**"Средняя скорость:"**));Pane1.add(speed);  
 PaneAddExp.add(Pane1,constraints);  
 JButton ADDEX=**new** JButton(**"Добавить"**);  
 ADDEX.setAlignmentX(***CENTER\_ALIGNMENT***);  
 constraints.**gridy**=1;  
 ADDEX.setPreferredSize(**new** Dimension(200,50));  
 PaneAddExp.add(ADDEX,constraints);  
 *PaneProcess*.add(PaneAddExp);  
 ADDEX.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **try**{  
 Integer.*parseInt*(number.getText());  
 Integer.*parseInt*(count.getText());  
 Integer.*parseInt*(speed.getText());  
 *Mytable*.AddValueAt(**new** ExpressTrain(Integer.*valueOf*(number.getText()),String.*valueOf*(type.getSelectedItem()),Integer.*valueOf*(count.getText()),String.*valueOf*(condition.getSelectedItem()),Integer.*valueOf*(speed.getText())));  
 }**catch** (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(PaneAddExp,  
 **" Можно вводить только целые числа"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 }  
 number.setText(**""**);  
 count.setText(**""**);  
 speed.setText(**""**);  
 }  
 });  
 }  
  
 **private static void** CreatePaneFindId(){  
 JPanel PaneFindID=**new** JPanel();  
 PaneFindID.setLayout(**new** GridBagLayout());  
 GridBagConstraints constraints = **new** GridBagConstraints();  
 *// По умолчанию натуральная высота, максимальная ширина* constraints.**fill** = GridBagConstraints.***VERTICAL***;  
 constraints.**weightx** = 0.5;  
 constraints.**gridx** = 0 ; *// нулевая ячейка таблицы по вертикали* JTextField id=**new** JTextField();  
 id.setPreferredSize(**new** Dimension(150,50));  
 JButton FINDID=**new** JButton(**"Найти"**);  
 FINDID.setPreferredSize(**new** Dimension(150,50));  
 constraints.**gridy** = 0;  
 PaneFindID.add(**new** JLabel(**"Введите № записи:"**),constraints);  
 constraints.**gridy** = 1;  
 PaneFindID.add(id,constraints);  
 constraints.**gridy** = 2;  
 PaneFindID.add(FINDID,constraints);  
 PaneFindID.repaint();  
 PaneFindID.revalidate();  
 *PaneProcess*.add(PaneFindID);  
 FINDID.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **try**{  
 **if** (Integer.*parseInt*(id.getText())<*Mytable*.getRowCount() & Integer.*parseInt*(id.getText())>=0){  
 *CreateDialogFindId*(Integer.*parseInt*(id.getText()));  
 id.setText(**""**);  
 }**else** {  
 JOptionPane.*showMessageDialog*(PaneFindID,  
 **" Нет такой записи"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 **return**;  
 }  
 }**catch** (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(PaneFindID,  
 **" Можно вводить только целые числа"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 }  
  
 }  
 });  
 }  
 **private static void** CreateDialogFindId(**int** id){  
 FindTableModel table1=**new** FindTableModel(**new** TransportManager());  
 **if**(table1.getRowCount()!=0){  
 table1.DeleteValueAt(0);  
 }  
 table1.AddValueAt(*Mytable*.getValueAt(id));  
 JDialog dialogFindId = **new** JDialog();  
 dialogFindId.setLayout(**new** BorderLayout());  
 dialogFindId.setModal(**true**);  
 dialogFindId.setMinimumSize(**new** Dimension(720,100));  
 dialogFindId.setMaximumSize(**new** Dimension(800,120));  
 dialogFindId.setLocationRelativeTo(*PaneProcess*);  
 dialogFindId.setTitle(**"Найденная запись:"**);  
 JTable tableFind=**new** JTable(table1);  
 dialogFindId.add((**new** JScrollPane(tableFind)),BorderLayout.***CENTER***);  
 dialogFindId.setDefaultCloseOperation(***DISPOSE\_ON\_CLOSE***);  
 dialogFindId.setVisible(**true**);  
 dialogFindId.repaint();  
 dialogFindId.revalidate();  
 }  
}

## **5.3 класс MyTableModel**

**package** com.company.Interface;  
**import** com.company.Logic.CarTransport;  
**import** com.company.Logic.ExpressTrain;  
**import** com.company.Logic.TrainTransport;  
**import** com.company.Logic.TransportManager;  
**import** com.company.Logic.Transport;  
  
**import** javax.swing.table.AbstractTableModel;  
  
**public class** MyTableModel **extends** AbstractTableModel {  
 **public static** TransportManager *List* =**new** TransportManager();  
  
 **public** MyTableModel() {  
 *List*.add(**new** TrainTransport(1, **"Почтовый"**, 3, **"Ремонт"**, 95));  
 *List*.add(**new** ExpressTrain(16, **"Грузовой"**, 45, **"В пути"**, 89));  
 *List*.add(**new** ExpressTrain(1, **"Почтовый"**, 47, **"Ремонт"**, 110));  
 *List*.add(**new** CarTransport(**"Hyundai"**, 2010, **"В гараже"**, 85));  
 *List*.add(**new** CarTransport(**"Kia"**, 2010, **"В пути"**, 110));  
 *List*.add(**new** TrainTransport(15, **"Грузовой"**, 762, **"В пути"**, 115));  
 *List*.add(**new** TrainTransport(166, **"Почтовый"**, 11, **"Ремонт"**, 90));  
 *List*.add(**new** ExpressTrain(99, **"Грузовой"**, 10, **"В пути"**, 85));  
 *List*.add(**new** ExpressTrain(2, **"Почтовый"**, 29, **"Ремонт"**, 90));  
 *List*.add(**new** CarTransport(**"Nissan"**, 2003, **"В гараже"**, 85));  
 *List*.add(**new** CarTransport(**"Skoda"**, 2000, **"В пути"**, 110));  
 *List*.add(**new** TrainTransport(160, **"Грузовой"**, 18, **"В пути"**, 675));  
 *List*.add(**new** TrainTransport(52, **"Почтовый"**, 29, **"Ремонт"**, 80));  
 *List*.add(**new** ExpressTrain(69 ,**"Грузовой"**, 52, **"В пути"**, 65));  
 *List*.add(**new** ExpressTrain(1, **"Почтовый"**, 61, **"Ремонт"**, 112));  
 }  
  
 @Override  
 **public int** getRowCount() {  
 **return** *List*.size();  
 }  
  
 @Override  
 **public int** getColumnCount() {  
 **return** 9;  
 }  
  
 @Override  
 **public** String getColumnName(**int** columnIndex) {  
 **switch** (columnIndex) {  
 **case** 0:  
 **return "№"**;  
 **case** 1:  
 **return "Вид"**;  
 **case** 2:  
 **return "Состояние"**;  
 **case** 3:  
 **return "Модель"**;  
 **case** 4:  
 **return "Скорость"**;  
 **case** 5:  
 **return "Тип"**;  
 **case** 6:  
 **return "Год"**;  
 **case** 7:  
 **return "Номер"**;  
 **case** 8:  
 **return "Число вагонов"**;  
 }  
 **return ""**;  
 }  
  
 @Override  
 **public** Class<?> getColumnClass(**int** columnIndex) {  
 **switch** (columnIndex) {  
 **case** 0:  
 **return** Integer.**class**;  
 **case** 1:  
 **return** String.**class**;  
 **case** 2:  
 **return** String.**class**;  
 **case** 3:  
 **return** String.**class**;  
 **case** 4:  
 **return** Integer.**class**;  
 **case** 5:  
 **return** String.**class**;  
 **case** 6:  
 **return** Integer.**class**;  
 **case** 7:  
 **return** Integer.**class**;  
 **case** 8:  
 **return** Integer.**class**;  
 }  
 **return** String.**class**;  
 }  
  
 @Override  
 **public boolean** isCellEditable(**int** rowIndex, **int** columnIndex) {  
 **return false**;  
 }  
  
 @Override  
 **public** Object getValueAt(**int** rowIndex, **int** columnIndex) {  
 Transport transp= *List*.get(rowIndex);  
 **if** (transp **instanceof** CarTransport){  
 **switch** (columnIndex) {  
 **case** 0:  
 **return** rowIndex;  
 **case** 1:  
 **return "Машина"**;  
 **case** 2:  
 **return** transp.getCondition();  
 **case** 3:  
 **return** ((CarTransport) transp).getModel();  
 **case** 4:  
 **return** transp.getSpeed();  
 **case** 6:  
 **return** ((CarTransport) transp).getYears();  
 }  
 }  
  
 **if** (transp **instanceof** ExpressTrain){  
 **switch** (columnIndex) {  
 **case** 0:  
 **return** rowIndex;  
 **case** 1:  
 **return "Экспресс"**;  
 **case** 2:  
 **return** transp.getCondition();  
 **case** 5:  
 **return** ((TrainTransport) transp).getType();  
 **case** 4:  
 **return** transp.getSpeed();  
 **case** 7:  
 **return** ((TrainTransport) transp).getNumber();  
 **case** 8:  
 **return** ((TrainTransport) transp).getCarriage();  
 }  
 }  
 **if** (transp **instanceof** TrainTransport){  
 **switch** (columnIndex) {  
 **case** 0:  
 **return** rowIndex;  
 **case** 1:  
 **return "Поезд"**;  
 **case** 2:  
 **return** transp.getCondition();  
 **case** 5:  
 **return** ((TrainTransport) transp).getType();  
 **case** 4:  
 **return** transp.getSpeed();  
 **case** 7:  
 **return** ((TrainTransport) transp).getNumber();  
 **case** 8:  
 **return** ((TrainTransport) transp).getCarriage();  
 }  
 }  
 **return null**;  
 }  
  
 **public void** setValueAt(Transport t, **int** rowIndex) {  
 *List*.set(rowIndex,t);  
 **this**.fireTableDataChanged();  
 }  
 **public** Transport getValueAt(**int** rowIndex) {  
 **return** *List*.get(rowIndex);  
 }  
 **public void** AddValueAt(Transport transp) {  
 *List*.add(transp);  
 **this**.fireTableDataChanged();  
 }  
 **public void** DeleteValueAt(**int** i) {  
 *List*.remove(i);  
 **this**.fireTableDataChanged();  
 }  
}

## **5.4 класс FindTableModel**

**package** com.company.Interface;  
  
**import** com.company.Logic.\*;  
  
**import** javax.swing.table.AbstractTableModel;  
  
**public class** FindTableModel **extends** AbstractTableModel {  
  
 **public static** TransportManager *List*;  
  
 **public** FindTableModel(TransportManager list){  
 *List*=list;  
 }  
 @Override  
 **public int** getRowCount() {  
 **return** *List*.size();  
 }  
  
 @Override  
 **public int** getColumnCount() {  
 **return** 9;  
 }  
  
 @Override  
 **public** Object getValueAt(**int** rowIndex, **int** columnIndex) {  
 Transport transp= *List*.get(rowIndex);  
 **if** (transp **instanceof** CarTransport){  
 **switch** (columnIndex) {  
 **case** 0:  
 **return** transp.getId();  
 **case** 1:  
 **return "Машина"**;  
 **case** 2:  
 **return** transp.getCondition();  
 **case** 3:  
 **return** ((CarTransport) transp).getModel();  
 **case** 4:  
 **return** transp.getSpeed();  
 **case** 6:  
 **return** ((CarTransport) transp).getYears();  
 }  
 }  
  
 **if** (transp **instanceof** ExpressTrain){  
 **switch** (columnIndex) {  
 **case** 0:  
 **return** transp.getId();  
 **case** 1:  
 **return "Экспресс"**;  
 **case** 2:  
 **return** transp.getCondition();  
 **case** 5:  
 **return** ((TrainTransport) transp).getType();  
 **case** 4:  
 **return** transp.getSpeed();  
 **case** 7:  
 **return** ((TrainTransport) transp).getNumber();  
 **case** 8:  
 **return** ((TrainTransport) transp).getCarriage();  
 }  
 }  
 **if** (transp **instanceof** TrainTransport){  
 **switch** (columnIndex) {  
 **case** 0:  
 **return** transp.getId();  
 **case** 1:  
 **return "Поезд"**;  
 **case** 2:  
 **return** transp.getCondition();  
 **case** 5:  
 **return** ((TrainTransport) transp).getType();  
 **case** 4:  
 **return** transp.getSpeed();  
 **case** 7:  
 **return** ((TrainTransport) transp).getNumber();  
 **case** 8:  
 **return** ((TrainTransport) transp).getCarriage();  
 }  
 }  
 **return null**;  
 }  
  
 @Override  
 **public** String getColumnName(**int** columnIndex) {  
 **switch** (columnIndex) {  
 **case** 0:  
 **return "№"**;  
 **case** 1:  
 **return "Вид"**;  
 **case** 2:  
 **return "Состояние"**;  
 **case** 3:  
 **return "Модель"**;  
 **case** 4:  
 **return "Скорость"**;  
 **case** 5:  
 **return "Тип"**;  
 **case** 6:  
 **return "Год"**;  
 **case** 7:  
 **return "Номер"**;  
 **case** 8:  
 **return "Число вагонов"**;  
 }  
 **return ""**;  
 }  
  
 @Override  
 **public** Class<?> getColumnClass(**int** columnIndex) {  
 **switch** (columnIndex) {  
 **case** 0:  
 **return** Integer.**class**;  
 **case** 1:  
 **return** String.**class**;  
 **case** 2:  
 **return** String.**class**;  
 **case** 3:  
 **return** String.**class**;  
 **case** 4:  
 **return** Integer.**class**;  
 **case** 5:  
 **return** String.**class**;  
 **case** 6:  
 **return** Integer.**class**;  
 **case** 7:  
 **return** Integer.**class**;  
 **case** 8:  
 **return** Integer.**class**;  
 }  
 **return** String.**class**;  
 }  
  
 @Override  
 **public boolean** isCellEditable(**int** rowIndex, **int** columnIndex) {  
 **return false**;  
 }  
  
 **public** Transport getValueAt(**int** rowIndex) {  
 **return** *List*.get(rowIndex);  
 }  
  
 **public void** AddValueAt(Transport transp) {  
 *List*.add(transp);  
 **this**.fireTableDataChanged();  
 }  
  
 **public void** DeleteValueAt(**int** i) {  
 *List*.remove(i);  
 **this**.fireTableDataChanged();  
 }  
}

## **5.5 класс TransportManager**

**package** com.company.Logic;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** TransportManager {  
 **private** List<Transport> **TransportList**=**new** ArrayList();  
 **public void** add(Transport tr) {  
 **this**.**TransportList**.add(tr);  
 }  
 **public void** remove(**int** i){  
 **this**.**TransportList**.remove(i);  
 }  
 **public int** size() {  
 **return this**.**TransportList**.size();  
 }  
 **public** Transport get(**int** index) {  
 **return TransportList**.get(index);  
 }  
 **public** Transport set(**int** id,Transport transp) {  
 **return TransportList**.set(id,transp);  
 }  
}

## **5.6 класс Transport**

**package** com.company.Logic;  
  
**public abstract class** Transport {  
 *//поля и метод абстрактного класса,их наследую дети и внуки* **protected** String **condition**;  
 **protected int speed**;  
 **public abstract** String[] OutputInfo();  
 **protected int id**;  
 *//конструктор по умолчанию* **public** Transport(){  
 **this**.**id**=*count*;  
 *count*++;  
 }  
 **private static int** *count*=0;  
 *//геттеры и сеттеры* **public int** getId() {  
 **return id**;  
 }  
 **public int** getSpeed() {  
 **return speed**;  
 }  
 **public** String getCondition() {  
 **return condition**;  
 }  
}

## **5.7 класс CarTransport**

**package** com.company.Logic;  
  
**public class** CarTransport **extends** Transport {  
 *//поля класса CarTransport* **protected int years**;  
 **protected** String **model**;  
 *//геттеры* **public** String getModel() {  
 **return model**;  
 }  
 **public int** getYears() {  
 **return years**;  
 }  
 *//конструктор с пармаметром* **public** CarTransport(String model, **int** years, String condition, **int** speed) {  
 **this**.**model**=model;  
 **this**.**years**=years;  
 **this**.**condition**=condition;  
 **this**.**speed**=speed;  
 }  
 *//переопределенный метод для вывода информации* @Override  
 **public** String[] OutputInfo() {  
 String [] str=**new** String[4];  
 str[0]=**this**.getModel();  
 str[1]=String.*valueOf*(**this**.getYears());  
 str[2]=**this**.getCondition();  
 str[3]=String.*valueOf*(**this**.getSpeed());  
 **return** str;  
 }  
}

## **5.8 класс TrainTransport**

**package** com.company.Logic;  
  
**public class** TrainTransport **extends** Transport {  
 *//поля класса TrainTransport* **protected** String **type**;  
 **protected int carriage**;  
 **protected int number**;  
 *//конструктор с параметром* **public** TrainTransport(**int** number,String type,**int** carriage,String condition,**int** speed){  
 **this**.**number**=number;  
 **this**.**type**=type;  
 **this**.**carriage**=carriage;  
 **this**.**condition**=condition;  
 **this**.**speed**=speed;  
 }  
 *//геттеры* **public** String getType() {  
 **return type**;  
 }  
 **public int** getNumber() {  
 **return number**;  
 }  
 **public int** getCarriage() {  
 **return carriage**;  
 }  
  
 *//переопределенный метод для вывода информации* @Override  
 **public** String[] OutputInfo() {  
 String [] str=**new** String[5];  
 str[0]=String.*valueOf*(**this**.getNumber());  
 str[1]=**this**.getType();  
 str[2]=String.*valueOf*(**this**.getCarriage());  
 str[3]=**this**.getCondition();  
 str[4]=String.*valueOf*(**this**.getSpeed());  
 **return** str;  
 }  
}

## **5.9 класс ExpressTrain**

**package** com.company.Logic;  
  
*//класс внук по отношению к классу Transport  
//класс-ребенок по отношению к классу TrainTransport***public class** ExpressTrain **extends** TrainTransport {  
 *//все поля наслудуются от TrainTransport  
 //конструктор с параметром* **public** ExpressTrain(**int** number, String type, **int** carriage, String condition, **int** speed) {  
 **super**(number, type, carriage, condition, speed);  
 }  
 *//переопределенный метод для вывода информации* @Override  
 **public** String[] OutputInfo() {  
 String [] str=**new** String[5];  
 str[0]=String.*valueOf*(**this**.getNumber());  
 str[1]=**this**.getType();  
 str[2]=String.*valueOf*(**this**.getCarriage());  
 str[3]=**this**.getCondition();  
 str[4]=String.*valueOf*(**this**.getSpeed());  
 **return** str;  
 }  
  
}

Список использованных источников

1. Объектно-ориентированное программирование. Метод. указания по

выполнению лабораторных работ / сост.: В.Л. Аршинский. – Иркутск : Изд-во

ИРНИТУ, 2017. – 24 c.