**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники**

отчет

**по лабораторной работе №9**

**по дисциплине «Объектно-ориентированное программирование»**

Тема: Многопоточность

|  |  |  |
| --- | --- | --- |
| Студент гр. 9308 |  | Яловега Н.В. |
| Преподаватель |  | Гречухин М.Н. |

Санкт-Петербург

2021

# Цель работы

Знакомство с правилами и классами построения параллельных приложений в языке Java.

# Задача

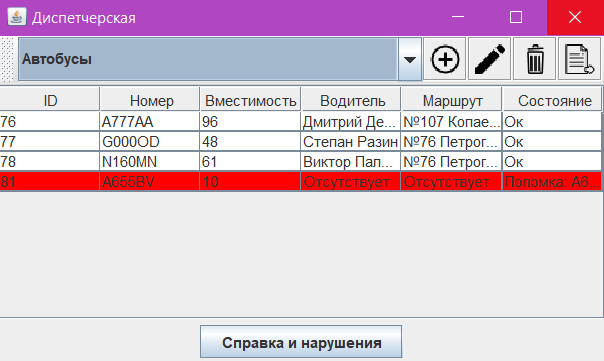
Интерфейс приложения предоставляет функцию составления отчёта о имеющихся маршрутах и всевозможных нарушениях.

Диспетчер может открыть любое количество отчётов с разницей лишь по времени их создания. Поскольку каждый из множества отчётов можно принудительно Обновить (чтобы получить актуальный в настоящий момент времени отчёт), может возникнуть проблема с Экспортом отчёта, т.к. информация, хранимая в общем текстовом поле, может меняться.

Поэтому для каждого из окон отчётов было решено создавать новый поток ReportThread, синхронизируемый по этому текстовому полю во избежание проблем Экспорта.

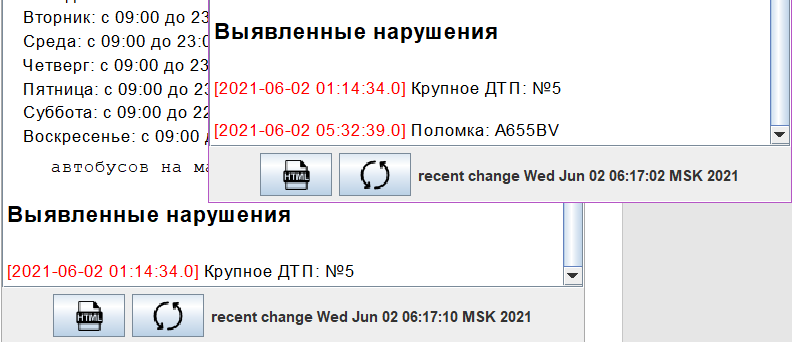
# Демонстрация работы

Новый поток ReportThread создаётся при нажатии на кнопку «Справка и нарушения»:



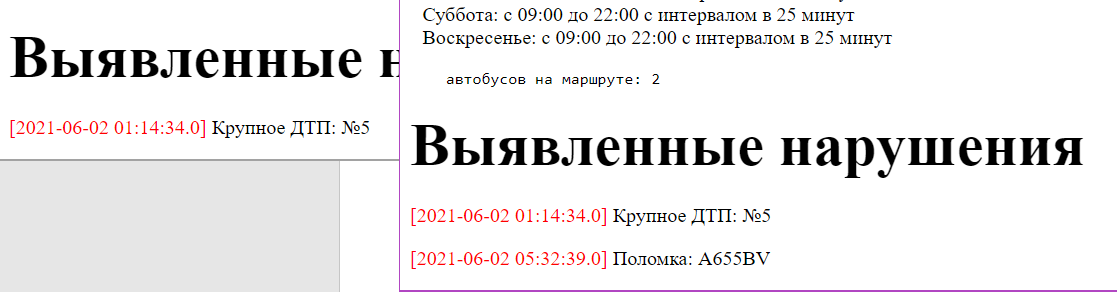
*Рисунок 1. Интерфейс приложения*

Откроем несколько отчётов в разные периоды времени с немного различающимися данными:



*Рисунок 2. Работа нескольких потоков*

Теперь информацию из каждого из них можно экспортировать (информация будет соответствовать времени составления отчёта):



*Рисунок 3. Результат работы*

# Фрагменты кода, отвечающего за многопоточность

**Поток ReportThread:**

package app;  
  
import entitys.Route;  
import entitys.Violation;  
  
import javax.persistence.EntityManager;  
import javax.swing.\*;  
import java.awt.\*;  
import java.util.Date;  
import java.util.List;  
  
public class ReportThread extends Thread{  
 final private EntityManager em;  
 private JTextPane textEditor = null;  
 final private JButton export = new JButton(new ImageIcon("./src/main/resources/img/html.png"));  
 final private JButton refresh = new JButton(new ImageIcon("./src/main/resources/img/refresh.png"));  
 final private JLabel date = new JLabel();  
  
 public ReportThread(String name, EntityManager \_em) {  
 super(name);  
 em = \_em;  
 }  
 public void run() {  
 createUI();  
 synchronized (textEditor) {  
 // Загрузка документа в приложении  
 loadTextToHTML();  
 }  
 }  
 private void createUI() {  
 JFrame frame = new JFrame("Справка и выявленные нарушения");  
 frame.setSize(400, 300);  
 frame.setLocation(100, 100);  
 frame.setDefaultCloseOperation(JFrame.*DISPOSE\_ON\_CLOSE*);  
 frame.setLayout(new BorderLayout());  
  
 // Создание редактора  
 textEditor = new JTextPane();  
 textEditor.setEditable(false);  
 textEditor.setContentType("text/html");  
 // возможные действия с отчётом и информация  
 export.addActionListener(e -> {  
 synchronized (textEditor) {  
 // Запись отчёта в HTML файл  
 new HTMLExporter(textEditor.getText());  
 }  
 });  
 refresh.addActionListener(e -> {  
 synchronized (textEditor) {  
 // Загрузка документа в приложении  
 loadTextToHTML();  
 }  
 setDate();  
 });  
 export.setToolTipText("Экпорт в HTML");  
 refresh.setToolTipText("Обновить данные");  
 setDate();  
 JPanel actionPanel = new JPanel();  
 actionPanel.add(export);  
 actionPanel.add(refresh);  
 actionPanel.add(date);  
 // добавление в фрейм  
 frame.add(new JScrollPane(textEditor), BorderLayout.*CENTER*);  
 frame.add(actionPanel, BorderLayout.*SOUTH*);  
  
 //pack();  
 frame.setSize(480, 560);  
 frame.setVisible(true);  
 }  
 private void setDate() {  
 date.setText("recent change " + new Date().toString());  
 }  
 private void loadTextToHTML() {  
 StringBuffer doc = new StringBuffer();  
 // начало HTML файла  
 doc.append("<html><body style=\"font-size:12px\" style=\"font-family: FreeSans\">");  
 doc.append("<head>" + "<meta http-equiv=\"content-type\" content=\"application/xhtml+xml; charset=UTF-8\"/>" + "</head>");  
 doc.append("<p style=\"margin: 5px\">");  
 // заголовок отчёта  
 doc.append("<h1><strong><span style=\"font-size:16px\">" + "Отчёт от " + new Date() + "</span></strong></h1><hr>");  
 // добавляем информацию о маршрутах  
 doc.append("<h3><strong><span style=\"font-size:14px\">" + "Информация о маршрутах" + "</span></strong></h3>");  
 java.util.List<Route> routeList = em.createQuery("SELECT r FROM Route r").getResultList();  
 for (int i = 0; i < routeList.size(); ++i) {  
 doc.append("<p style=\"margin-left: 10px\">");  
 doc.append(routeList.get(i).toHTMLFormat());  
 doc.append("</p>");  
 }  
 // информация о нарушениях  
 doc.append("<h3><strong><span style=\"font-size:14px\">" + "Выявленные нарушения" + "</span></strong></h3>");  
 List<Violation> violationList = em.createQuery("SELECT v FROM Violation v WHERE fixdatetime=null").getResultList();  
 for (int i = 0; i < violationList.size(); ++i) {  
 doc.append(violationList.get(i).toHTMLFormat());  
 }  
 doc.append("</p>");  
 // конец HTML файла  
 doc.append("</html></body>");  
 textEditor.setText(doc.toString());  
 textEditor.setCaretPosition(0);  
 }  
}

**Вызов нового потока:**

report = new JButton("Справка и нарушения");  
report.addActionListener(e -> new ReportThread("report",*em*).start());

# Выводы

Во время выполнения данной лабораторной работы мы ознакомились с правилами построения параллельных приложений в языке Java, реализовали свой класс потока ReportThread, ответственного за составление отчётов по ТЗ курсовой работы.