**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники**

**Пояснительная записка к курсовой работе**

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

Вариант 16

|  |  |  |
| --- | --- | --- |
| Студент гр. 9308 |  | Ручкин Д.А. |
| Преподаватель |  | Гречухин М.Н. |

Санкт-Петербург

2021

Содержание

[Техническое задание 3](#_Toc3134)

[Требования к программе 4](#_Toc31895)

[Описание процесса проектирования ПК 5](#_Toc30032)

[Диаграмма классов 6](#_Toc22534)

[Описание классов 7](#_Toc2735)

[Класс Preparat 7](#_Toc24066)

[Класс SoldPreparat 9](#_Toc28430)

[Класс Illness 11](#_Toc29710)

[Класс Pharmacy 12](#_Toc28951)

[Класс Application 14](#_Toc30863)

[Класс Export 18](#_Toc15960)

[Класс PharmFieldsException 19](#_Toc4253)

[Класс OpenException 20](#_Toc18017)

[Описание таблиц БД 21](#_Toc6412)

[Исходный код 24](#_Toc6163)

[Интерфейс 75](#_Toc32023)

[Вывод 82](#_Toc4659)

# Техническое задание

Разработать ПК для администратора аптеки. В ПК должны храниться сведения о болезнях и лекарствах. Администратор аптеки может добавлять, изменять и удалять эти сведения. Ему может потребоваться следующая информация:

* какие лекарства применяются для лечения указанной болезни;
* имеется ли лекарство в аптеке и в каком количестве;
* какие лекарства и в каком количестве проданы за указанный период времени;
* на какую сумму проданы лекарства за месяц.

# Требования к программе

ПК должен обеспечивать выполнение следующих функций:

* Просмотр информации из БД;
* Добавление новых записей в БД;
* Удаление записей БД;
* Изменение уже существующих записей в БД.

ПК должен предоставлять пользователю возможность работать со следующими сведениями:

* Сведений о аптеках;
* Сведения о лекарствах;
* Сведения о болезнях;
* Сведения о проданных лекарствах.

# Описание процесса проектирования ПК

При выполнении курсовой работы было пройдено несколько этапов разработки приложения.

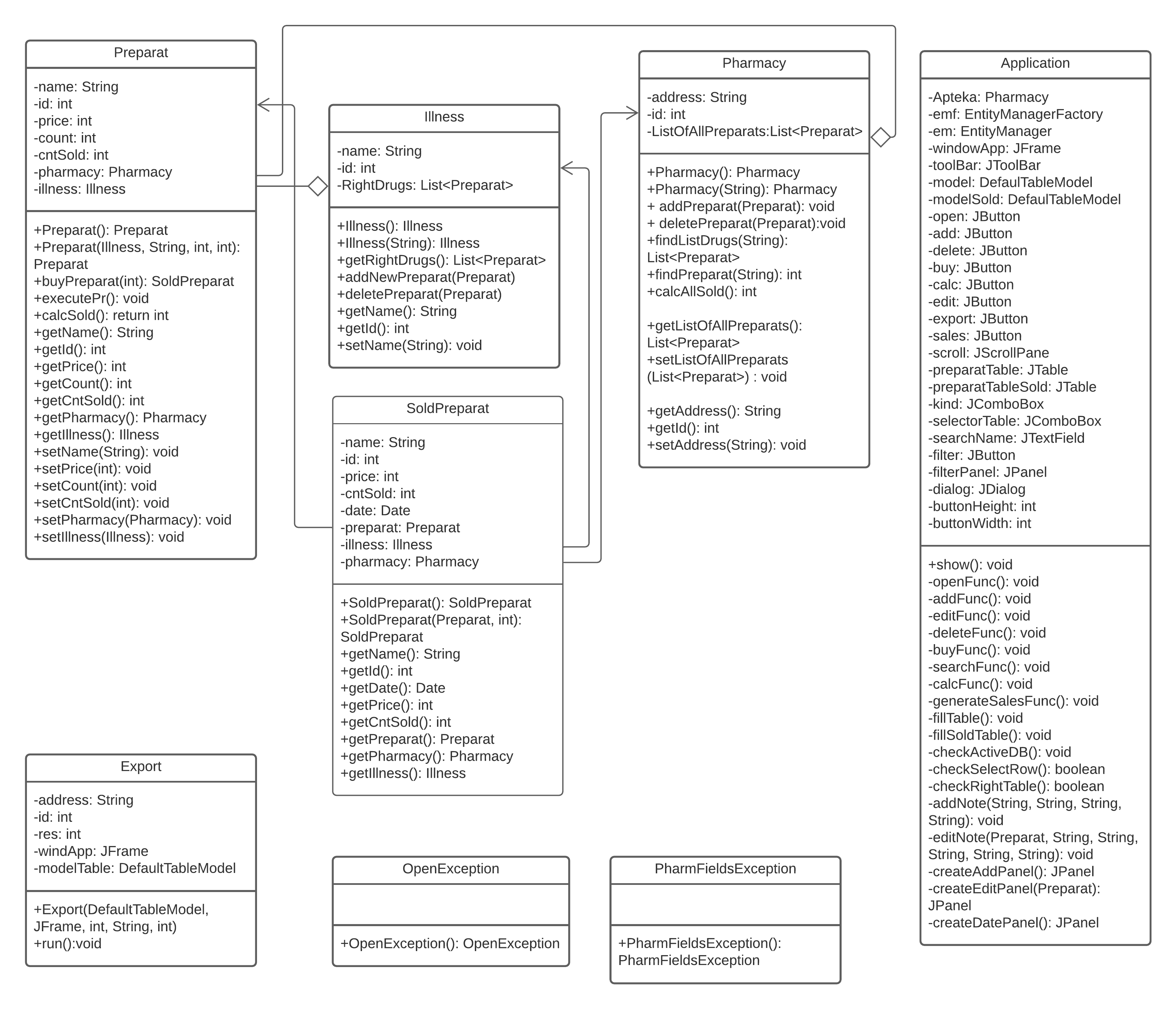
Сначала была разработана архитектура проекта, основные классы, таблицы БД и связи между ними.

Затем был разработан интерфейс приложения, добавлено меню с кнопками действия, таблица данных, поиск.

После были добавлены: обработка исключений, генерация PDF отчетов, JUnit тестирование, логирование, использование многопоточности для некоторых действий.

Также было решено добавить возможность работы с большими количеством аптек (фармацевтической сетью), для этого в начале работы необходимо выбрать аптеку, с которой планируется совершать действия.

# Диаграмма классов



*Рисунок 1. Диаграмма классов*

# Описание классов

# Класс Preparat

Класс используется для хранения информации о лекарстве, используемом в ПК.

Свойства.

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Семантика |
| id | int | id объекта в БД |
| name | String | Название лекарства |
| price | int | Цена лекарства |
| count | int | Количество лекарства в аптеке |
| cntSold | int | Количество проданных единиц лекарства |
| pharmacy | Pharmacy | Аптека, в которой находится лекарство |
| illness | Illness | Болезнь, которую лечит это лекарство |

Методы.

|  |  |  |
| --- | --- | --- |
| Название | Параметры | Семантика |
| Preparat |  | Конструктор |
| Preparat | Illness i - болезнь, String nm - название, int pr - цена, int c - количество препарата | Конструктор |
| buyPreparat | int k - сколько товара нужно купить | Покупка данного лекарства |
| executePr |  | Разрывает связь препарата с болезнью и аптекой |
| calcSold |  | Считает на какую сумму было продано это лекарство |
| getName |  | Возвращает название препарата |
| getId |  | Возвращает id препарата |
| getPrice |  | Возвращает цену препарата |
| getCount |  | Возвращает количество препарата |
| getCntSold |  | Возвращает количество проданного препарата |
| getPharmacy |  | Возвращает аптеку, в которой находится препарат |
| getIllness |  | Возвращает болезнь, лечимую этим препаратом |
| setName | String name - название | Устанавливает название препарата |
| setPrice | int p - цена | Устанавливает цену препарата |
| setCount | int a - количество | Устанавливает количество препарата |
| setCntSold | int a - количество проданных | Устанавливает количество проданного препарата |
| setPharmacy | Pharmacy p - аптека | Устанавливает аптеку, в которой находится этот препарат |
| setIllness | Illness i - болезнь | Устанавливает болезнь, лечимую препаратом |

# Класс SoldPreparat

Класс используется для хранения информации о уже проданном лекарстве.

Свойства.

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Семантика |
| id | int | id объекта в БД |
| name | String | Название проданного лекарства |
| price | int | Цена проданного лекарства |
| cntSold | int | Количество проданных единиц лекарства |
| pharmacy | Pharmacy | Аптека, в которой было продано лекарство |
| illness | Illness | Болезнь, которую лечит проданное лекарство |
| date | Date | Дата и время, когда было продано лекарство |
| preparat | Preparat | Лекарство, которое было продано |

Методы.

|  |  |  |
| --- | --- | --- |
| Название | Параметры | Семантика |
| SoldPreparat |  | Конструктор |
| SoldPreparat | Preparat ParentPreparat - лекарство, int cnt - сколько продано | Конструктор |
| getName |  | Возвращает название проданного препарата |
| getId |  | Возвращает id проданного препарата |
| getPrice |  | Возвращает цену препарата |
| getDate |  | Возвращает дату и время продажи |
| getCntSold |  | Возвращает количество проданного препарата |
| getPharmacy |  | Возвращает аптеку, в которой продан препарат |
| getIllness |  | Возвращает болезнь, лечимую этим препаратом |
| getPreparat |  | Возвращает лекарство, которое было продано |

# Класс Illness

Класс используется для хранения информации о болезни, используемой в ПК.

Свойства.

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Семантика |
| id | int | id объекта в БД |
| name | String | Название болезни |
| RightDrugs | List<Preparat> | Список лекарств, которые подходят для лечения этой болезни |

Методы.

|  |  |  |
| --- | --- | --- |
| Название | Параметры | Семантика |
| Illness |  | Конструктор |
| Illness | String s - название болезни | Конструктор |
| addNewPreparat | Preparat p - лекарство | Добавить препарат в список лекарств, которые лечат эту болезнь |
| deletePreparat | Preparat p - лекарство | Удалить препарат из списка лекарств, которые лечат эту болезнь |
| getRightDrugs |  | Возвращает список лекарств, которые лечат эту болезнь |
| getName |  | Возвращает название болезни |
| getId |  | Возвращает id болезни |
| setName | String name - название болезни | Устанавливает название болезни |

# Класс Pharmacy

Класс используется для хранения информации о аптеке, используемой в ПК.

Свойства.

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Семантика |
| id | int | id объекта в БД |
| address | String | Адрес аптеки |
| ListOfAllPreparats | List<Preparat> | Список лекарств, которые находятся в этой аптеке |

Методы.

|  |  |  |
| --- | --- | --- |
| Название | Параметры | Семантика |
| Pharmacy |  | Конструктор |
| Pharmacy | String s - адрес аптеки | Конструктор |
| addPreparat | Preparat p - лекарство | Добавить препарат в список лекарств, которые находятся в этой аптеке |
| deletePreparat | Preparat p - лекарство | Удалить препарат из списка лекарств, которые находятся в этой аптеке |
| findPreparat | String s - название лекарства | Находит лекарство в аптеке по названию |
| findListDrugs | String nameIll - название болезни | Находит список лекарств в аптеке, подходящих для лечения определенной болезни |
| calcAllSold |  | Считает на какую сумму было продано всех лекарств в аптеке |
| getListOfAllPreparats |  | Возвращает список лекарств, которые находятся в этой аптеке |
| setListOfAllPreparats | List<Preparat> preparats - список препаратов | Устанавливает список лекарств, которые находятся в этой аптеке |
| getAddress |  | Возвращает адрес аптеки |
| getId |  | Возвращает id аптеки |
| setAddress | String address- адрес аптеки | Устанавливает адрес аптеки |

# Класс Application

Класс используется для вывода окон интерфейса ПК и взаимодействия с базой данных, посредством работы с элементами интерфейса (кнопки, выпадающие списки и тд).

Свойства.

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Семантика |
| Apteka | pharmacy | Аптека, с которой происходит работа |
| emf | EntityManagerFactory | Фабрика |
| em | EntityManager | Менеджер сущностей |
| windowApp | JFrame | Главное окно интерфейса |
| toolBar | JToolBar | Панель кнопок меню |
| model | DefaultTableModel | Модель таблицы препаратов в аптеке |
| modelSold | DefaultTableModel | Модель таблицы проданных препаратов в аптеке |
| preparatTable | JTable | Таблица препаратов в аптеке |
| preparatTableSold | JTable | Таблица проданных препаратов в аптеке |
| open | JButton | Кнопка открытия БД аптеки |
| add | JButton | Кнопка добавления нового препарата |
| delete | JButton | Кнопка удаления препарата |
| buy | JButton | Кнопка покупки препарата |
| calc | JButton | Кнопка подсчета прибыли |
| edit | JButton | Кнопка изменения информации о препарате |
| export | JButton | Кнопка экспорта таблицы препаратов в PDF файл |
| sales | JButton | Кнопка вывода информации о продажах |
| filter | JButton | Кнопка поиска препаратов |
| scroll | JScrollPane | Панель прокрутки с активной таблицей |
| kind | JComboBox | Выпадающий список выбора объекта поиска |
| selectorTable | JComboBox | Выпадающий список выбора активной таблицы |
| searchName | JTextField | Поле ввода для поиска |
| dialog | JDialog | Диалоговое окно |
| filterPanel | JPanel | Панель поиска |
| buttonHeight | int | Высота изображения на иконке кнопки |
| buttonWidth | int | Ширина изображения на иконке кнопки |

Методы.

|  |  |  |
| --- | --- | --- |
| Название | Параметры | Семантика |
| show |  | Запускает приложение |
| openFunc |  | Открытие БД аптеки |
| addFunc |  | Добавление нового препарата в БД аптеки |
| deleteFunc |  | Удаление препарата из БД аптеки |
| editFunc |  | Изменение информации о препарате |
| buyFunc |  | Покупка препарата |
| searchFunc |  | Поиск подходящего препарата по названию или болезни |
| calcFunc |  | Подсчет прибыли за указанный период |
| generateSalesFunc |  | Показывает продажи аптеки за указанный период |
| fillTable |  | Заполняет таблицу препаратов в аптеке и делает её активной |
| fillSoldTable |  | Заполняет таблицу проданных препаратов в аптеке и делает её активной |
| addNote | String name - название препарата, String ill - название болезни, String price\_ - цена , String cnt\_ - количество препарата | Добавляет новую запись в БД в виде нового препарата |
| editNote | Preparat pr - препарат, String name - новое название препарата, String ill - новое название болезни, String price\_ - новая цена , String cnt\_ - новое количество препарата, String cntSold\_ - новое количество проданного препарата | Изменяет поля существующего препарата и сохраняет изменения в БД |
| checkActiveDB |  | Проверяет открыта ли БД аптеки |
| checkRightTable |  | Проверяет открыта ли таблица препаратов |
| checkSelectRow |  | Проверяет выбрана ли одна строчка |
| createAddPanel |  | Создание панели с информацией о добавлении нового препарата |
| createEditPanel | Preparat pr - препарат, который необходимо изменить | Создание панели с информацией о изменении полей существующего препарата |
| createDatePanel |  | Создание панели с выбором промежутка времени |

# Класс Export

Класс используется для формирования PDF отчета о препаратах в аптеке в отдельном потоке.

Свойства.

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Семантика |
| id | int | id аптеки |
| res | int | На какую сумму всего продано лекарств в аптеке |
| address | String | Адрес аптеки |
| windApp | JFrame | Окно, поверх которого выводится сообщение о успешной/неуспешной генерации отчета |
| modelTable | DefaultTableModel | Модель таблицы препаратов |

Методы.

|  |  |  |
| --- | --- | --- |
| Название | Параметры | Семантика |
| Export | DefaultTableModel tab - модель таблицы препаратов, JFrame jf - главное окно, Pharmacy apteka - текущая аптека | Конструктор |
| run |  | Запускает поток формирования PDF отчета |

# Класс PharmFieldsException

Класс собственных исключений, обрабатывающий исключительную ситуацию, когда пользователь некорректно вводит текстовые поля. Класс наследуется от Exception.

Методы.

|  |  |  |
| --- | --- | --- |
| Название | Параметры | Семантика |
| PharmFieldsException |  | Конструктор, который передает в конструктор базового класса исключений сообщение об ошибке |

# Класс OpenException

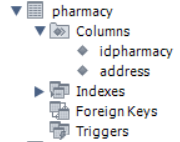
Класс собственных исключений, обрабатывающий исключительную ситуацию, когда пользователь пытается сделать какое-нибудь действие, для которого требуется открытая БД аптеки, предварительно не открыв эту БД. Класс наследуется от Exception.

Методы.

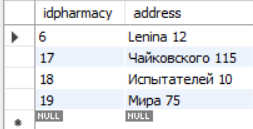
|  |  |  |
| --- | --- | --- |
| Название | Параметры | Семантика |
| OpenFieldsException |  | Конструктор, который передает в конструктор базового класса исключений сообщение об ошибке |

# Описание таблиц БД

1. Таблица аптек



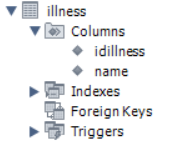
*Рисунок 2. Структура таблицы аптек*



*Рисунок 3. Таблица аптек*

idpharmacy является первичным ключом.

1. Таблица болезней



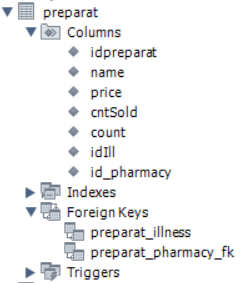
*Рисунок 4. Структура таблицы болезней*



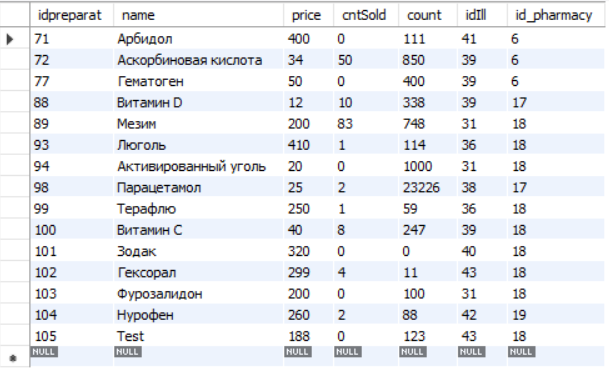
*Рисунок 5. Таблица болезней*

idillness является первичным ключом.

1. Таблица препаратов



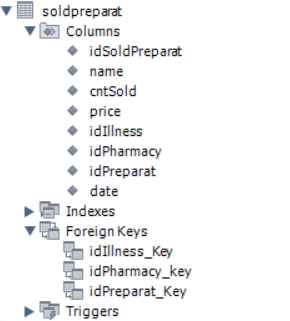
*Рисунок 6. Структура таблицы препаратов*



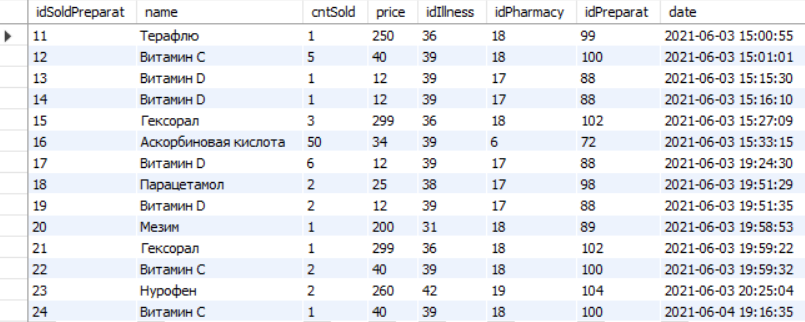
*Рисунок 7. Таблица препаратов*

idpreparat является первичным ключом. preparat\_illness является внешним ключом для связи с болезнью. preparat\_pharmacy\_fk является внешним ключом для связи с аптекой.

1. Таблица проданных препаратов



*Рисунок 8. Структура таблицы проданных препаратов*



*Рисунок 9. Таблица проданных препаратов*

idSoldPreparat является первичным ключом. idIllness\_key является внешним ключом для связи с болезнью. idPharmacy\_key является внешним ключом для связи с аптекой. idPreparat\_key является внешним ключом для связи проданного объекта с самим препаратом.

# Исходный код

**Pharmacy:**

package App;  
import org.apache.log4j.Logger;  
  
import javax.persistence.\*;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
  
@Entity  
@Table(name = "pharmacy")  
public class Pharmacy{  
  
 @Id  
 @Column(name = "idpharmacy")  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private int id;  
  
 @Column(name="address")  
 private String address;  
  
 @OneToMany(mappedBy = "pharmacy", cascade = CascadeType.ALL)  
 private List<Preparat> ListOfAllPreparats = new ArrayList<>();  
  
 static Scanner in = new Scanner(System.in);  
 final static Logger logger = Logger.getLogger(Pharmacy.class);  
 public Pharmacy() {  
 logger.info("a pharmacy without an address was created");  
 }  
 public Pharmacy(String s) { //с указанием адреса аптеки  
 this.address = s;  
 logger.debug("the address " + this.getAddress() + " has been successfully assigned to the object");  
 logger.info("pharmacy was created");  
 }

public void addPreparat(Preparat p) {  
 ListOfAllPreparats.add(p);  
 p.setPharmacy(this);  
 }  
  
 public void deletePreparat(Preparat p) {  
 ListOfAllPreparats.remove(p);  
 }  
  
 public List<Preparat> findListDrugs(String nameIll) {//находит лекарства по болезни  
 List<Preparat> preparatList = new ArrayList<>();  
 for (int i=0; i<ListOfAllPreparats.size(); ++i){  
 if (ListOfAllPreparats.get(i).getIllness().getName().equals(nameIll))  
 preparatList.add(ListOfAllPreparats.get(i));  
 }  
 return preparatList;  
  
 }  
  
 public int findPreparat(String s){//находит лекарство по названию препарата  
 for (int i=0; i<ListOfAllPreparats.size(); ++i)  
 if ((ListOfAllPreparats.get(i).getName().equals(s)) && (ListOfAllPreparats.get(i).getCount() != 0))  
 return i;//нашлось  
 return -1;//ничего не нашлось  
 }  
  
 public int calcAllSold(){  
 int res = 0;  
 for (int i =0 ; i<ListOfAllPreparats.size(); ++i)  
 res += ListOfAllPreparats.get(i).calcSold();  
 return res;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public List<Preparat> getListOfAllPreparats()  
 {  
 if (ListOfAllPreparats.size() == 0)  
 return null;  
 else  
 return ListOfAllPreparats;  
 }  
  
 public void setListOfAllPreparats(List<Preparat> preparats)  
 { this.ListOfAllPreparats = preparats; }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
}

**Illness:**

package App;  
  
import javax.persistence.\*;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
  
@Entity  
@Table(name = "illness")  
public class Illness{  
  
 @Id  
 @Column(name = "idillness")  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private int id;  
 @Column(name="name")  
 private String name;  
  
 @OneToMany(mappedBy = "illness", cascade = CascadeType.ALL)  
 private List<Preparat> RightDrugs = new ArrayList<Preparat>();  
  
 public List<Preparat> getRightDrugs(){  
 if (RightDrugs.size() == 0)  
 return null;  
 else  
 return RightDrugs;  
 }  
  
 static Scanner in = new Scanner(System.in);  
  
 public Illness() { }  
  
 public Illness(String s){  
 this.setName(s);  
 }  
  
 public void addNewPreparat(Preparat p) { RightDrugs.add(p); }  
 public void deletePreparat(Preparat p){ RightDrugs.remove(p);}  
  
 public int getId() {  
 return id;  
 }  
 public String getName() { return name; }  
  
 public void setName(String name) { this.name = name; }  
}

**Preparat:**

package App;  
  
import org.apache.log4j.Logger;  
import javax.persistence.\*;  
import java.util.Scanner;  
  
@Entity  
@Table(name = "preparat")  
public class Preparat{  
  
 static Scanner in = new Scanner(System.in);  
  
 @Id  
 @Column(name = "idpreparat")  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private int id;  
  
 @Column(name="name")  
 private String name;  
  
 @Column(name = "price")  
 private int price;  
  
 @Column(name = "cntSold")  
 private int cntSold=0;  
  
 @Column(name = "count")  
 private int count=0;  
  
 @ManyToOne(fetch = FetchType.EAGER)  
 @JoinColumn(name = "id\_pharmacy")  
 private Pharmacy pharmacy;  
  
 @ManyToOne(fetch = FetchType.EAGER)  
 @JoinColumn(name = "idIll")  
 private Illness illness;  
  
 final static Logger logger = Logger.getLogger(Preparat.class);  
  
 public Preparat() { logger.info("a drug was created without specifying fields");};  
  
 public Preparat(Illness i, String s, int pr, int c){  
 this.setName(s);  
 logger.debug("the name " + this.getName() + " has been successfully assigned to the object");  
 this.price = pr;  
 logger.debug("the price " + this.getPrice() + " has been successfully assigned to the object");  
 this.count = c;  
 logger.debug("the count " + this.getCount() + " has been successfully assigned to the object");  
 i.addNewPreparat(this);  
 this.setIllness(i);  
 logger.debug("the illness " + this.getIllness().getName() + " has been successfully assigned to the object");  
 logger.info("the drug was created");  
 }  
  
 public SoldPreparat buyPreparat(int k){ // возвращает купленный товар  
 this.count -= k;  
 this.cntSold += k;  
 SoldPreparat sldPr = new SoldPreparat(this, k);  
 return sldPr;  
 }  
 public int calcSold(){ return cntSold\*price; }  
  
 public void executePr(){//метод разрывает связь препарата с болезнью и аптекой  
 if (pharmacy != null)  
 pharmacy = null;  
 if (illness != null)  
 illness = null;  
 }  
 public int getId() {  
 return id;  
 }  
 public String getName(){  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
 public void setPrice(int p) { this.price = p;}  
 public void setCntSold(int a) { this.cntSold = a;}  
 public void setCount(int a) {this.count = a;}  
  
 public int getPrice(){ return price; }  
 public int getCount(){ return count; }  
 public int getCntSold(){ return cntSold; }  
 public Pharmacy getPharmacy() { return pharmacy; }  
 public Illness getIllness() { return illness; }  
  
 public void setPharmacy(Pharmacy p) {  
 this.pharmacy = p;  
 }  
 public void setIllness(Illness i) { this.illness = i; }  
}

**SoldPreparat:**

package App;  
import javax.persistence.\*;  
import java.util.Date;  
@Entity  
@Table(name = "soldpreparat")  
public class SoldPreparat {  
 @Id  
 @Column(name = "idSoldPreparat")  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private int id;  
 @Column(name="name")  
 private String name;  
 @Column(name = "price")  
 private int price;  
 @Column(name = "cntSold")  
 private int cntSold=0;  
 @Column(name = "date")  
 private Date date;  
 @OneToOne(fetch = FetchType.LAZY)  
 @JoinColumn(name = "idPreparat")  
 private Preparat preparat;  
 @OneToOne(fetch = FetchType.LAZY)  
 @JoinColumn(name = "idPharmacy")  
 private Pharmacy pharmacy;  
 @OneToOne(fetch = FetchType.LAZY)  
 @JoinColumn(name = "idIllness")  
 private Illness illness;

public SoldPreparat(){}  
 public SoldPreparat(Preparat ParentPreparat, int cnt){  
 this.name = ParentPreparat.getName();  
 this.preparat = ParentPreparat;  
 this.cntSold = cnt;  
 this.price = ParentPreparat.getPrice();  
 this.illness = preparat.getIllness();  
 this.pharmacy = preparat.getPharmacy();  
 this.date = new Date();  
 }  
 public int getId() { return id; }  
 public String getName(){ return name; }  
 public int getPrice(){ return price; }  
 public int getCntSold(){ return cntSold; }  
 public Date getDate() {return date;}  
 public Pharmacy getPharmacy() { return pharmacy; }  
 public Preparat getPreparat() {return preparat;}  
 public Illness getIllness() { return illness; }  
}

**Application:**

package App;  
import java.awt.\*;  
import java.awt.event.\*;  
import javax.persistence.\*;  
import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Calendar;  
import java.util.List;  
import org.apache.log4j.Logger;  
  
public class Application {  
 private EntityManagerFactory emf = Persistence.createEntityManagerFactory("Database persistence");  
 private EntityManager em = emf.createEntityManager();  
 final static Logger logger = Logger.getLogger(Application.class);  
 private Pharmacy Apteka;  
  
 private JFrame windowApp;  
  
 private JToolBar toolBar;  
 private DefaultTableModel model;  
 private DefaultTableModel modelSold;  
 private JButton open;  
 private JButton add;  
 private JButton delete;  
 private JButton buy;  
 private JButton calc;  
 private JButton edit;  
 private JButton export;  
 private JButton sales;  
 private JScrollPane scroll = new JScrollPane();  
 private JTable preparatTable;  
 private JTable preparatTableSold;  
 private JComboBox kind;  
 private JComboBox selectorTable;  
 private JTextField searchName;  
 private JButton filter;  
 private JDialog dialog;  
 private JPanel filterPanel;  
  
 private int buttonHeight = 20;  
 private int buttonWidth = 20;  
  
 public void show() {//открытие приложения  
 // / Создание окна  
 windowApp = new JFrame("Pharmacy management");  
 windowApp.setSize(700, 480);  
 windowApp.setLocation(100, 100);  
 windowApp.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
  
 // Создание кнопок, прикрепление иконок и масштабирование их картинок  
 open = new JButton(new ImageIcon(new ImageIcon("pic/openDB.png").getImage().getScaledInstance(buttonWidth, buttonHeight, java.awt.Image.SCALE\_SMOOTH)));  
 add = new JButton(new ImageIcon(new ImageIcon("pic/add.png").getImage().getScaledInstance(buttonWidth, buttonHeight, java.awt.Image.SCALE\_SMOOTH)));  
 delete = new JButton(new ImageIcon(new ImageIcon("pic/delete.png").getImage().getScaledInstance(buttonWidth, buttonHeight, java.awt.Image.SCALE\_SMOOTH)));  
 buy = new JButton(new ImageIcon(new ImageIcon("pic/buy.png").getImage().getScaledInstance(buttonWidth, buttonHeight, java.awt.Image.SCALE\_SMOOTH)));  
 calc = new JButton(new ImageIcon(new ImageIcon("pic/calc.png").getImage().getScaledInstance(buttonWidth, buttonHeight, java.awt.Image.SCALE\_SMOOTH)));  
 edit = new JButton(new ImageIcon(new ImageIcon("pic/edit.png").getImage().getScaledInstance(buttonWidth, buttonHeight, java.awt.Image.SCALE\_SMOOTH)));  
 export = new JButton(new ImageIcon(new ImageIcon("pic/export.png").getImage().getScaledInstance(buttonWidth, buttonHeight, java.awt.Image.SCALE\_SMOOTH)));  
 sales = new JButton(new ImageIcon(new ImageIcon("pic/sales.png").getImage().getScaledInstance(buttonWidth, buttonHeight, java.awt.Image.SCALE\_SMOOTH)));  
 open.setActionCommand("open");  
 add.setActionCommand("add");  
 delete.setActionCommand("delete");  
 buy.setActionCommand("buy");  
 edit.setActionCommand("edit");  
 calc.setActionCommand("calc");  
 export.setActionCommand("export");  
 sales.setActionCommand("sales");  
// Настройка подсказок для кнопок  
 open.setToolTipText("Открыть БД аптеки");  
 add.setToolTipText("Добавить запись");  
 delete.setToolTipText("Удалить запись");  
 buy.setToolTipText("Покупка товара");  
 calc.setToolTipText("Посчитать прибыль");  
 edit.setToolTipText("Изменить запись");  
 export.setToolTipText("Экспортировать таблицу");  
 sales.setToolTipText("Список продаж");  
// Добавление кнопок на панель инструментов  
 toolBar = new JToolBar("Панель инструментов");  
 toolBar.add(open);  
 toolBar.add(add);  
 toolBar.add(delete);  
 toolBar.add(edit);  
 toolBar.add(buy);  
 toolBar.add(calc);  
 toolBar.add(export);  
 toolBar.add(sales);  
// Размещение панели инструментов  
 windowApp.setLayout(new BorderLayout());  
 windowApp.add(toolBar, BorderLayout.NORTH);  
// Подготовка компонентов нижней панели  
 kind = new JComboBox(new String[]{"Препарат", "Болезнь"});  
 searchName = new JTextField("", 20);  
 filter = new JButton("Поиск");  
 selectorTable = new JComboBox(new String[]{"Товары в аптеке", "Проданные товары"});  
// Добавление компонентов на панель  
 filterPanel = new JPanel();  
 filterPanel.add(kind);  
 filterPanel.add(searchName);  
 filterPanel.add(filter);  
// Размещение панели поиска внизу окна  
 windowApp.add(filterPanel, BorderLayout.SOUTH);  
  
 // Визуализация экранной формы  
 windowApp.setVisible(true);  
 logger.info("The App is running");  
  
 //Слушатель для всех кнопок  
 ActionListener actionPressButton = new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 switch (e.getActionCommand()){  
 case("open"):  
 logger.info("loading database");  
 openFunc();  
 logger.info("loading database id " + Apteka.getId() + " completed");  
 break;  
 case("add"):  
 try {  
 checkActiveDB();//при неудаче создается исключение  
 }  
 catch (OpenException ex){  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), ex.getMessage());  
 break;  
 }  
 if (checkRightTable())  
 addFunc();  
 break;  
 case("delete"):  
 try {  
 checkActiveDB();  
 }  
 catch (OpenException ex){  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), ex.getMessage());  
 break;  
 }  
 if (checkRightTable() && checkSelectRow())  
 deleteFunc();  
 break;  
 case("edit"):  
 try {  
 checkActiveDB();  
 }  
 catch (OpenException ex){  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), ex.getMessage());  
 break;  
 }  
 if (checkRightTable() && checkSelectRow())  
 editFunc();  
 break;  
 case("buy"):  
 try {  
 checkActiveDB();  
 }  
 catch (OpenException ex){  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), ex.getMessage());  
 break;  
 }  
 if (checkRightTable() && checkSelectRow())  
 buyFunc();  
 break;  
 case("calc"):  
 try {  
 checkActiveDB();  
 }  
 catch (OpenException ex){  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), ex.getMessage());  
 break;  
 }  
 calcFunc();  
 break;  
 case("export"):  
 try {  
 checkActiveDB();  
 }  
 catch (OpenException ex){  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), ex.getMessage());  
 break;  
 }  
  
 Export export = new Export(model, windowApp, Apteka);  
 export.start();  
 break;  
 case("sales"):  
 try {  
 checkActiveDB();  
 }  
 catch (OpenException ex){  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), ex.getMessage());  
 break;  
 }  
 generateSalesFunc();  
 break;  
 case("Поиск"):  
 try {  
 checkActiveDB();  
 }  
 catch (OpenException ex){  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), ex.getMessage());  
 break;  
 }  
 if (!checkRightTable())  
 break;  
 if (searchName.getText().equals(""))  
 {  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), "<html>Введите название");  
 break;  
 }  
 searchFunc();  
 break;  
 }  
 }  
 };  
  
 //слушатель для селектора таблиц  
 selectorTable.addItemListener(new ItemListener() {  
 @Override  
 public void itemStateChanged(ItemEvent e) {  
 if (e.getStateChange() == ItemEvent.SELECTED){  
 try {  
 checkActiveDB();  
 }  
 catch (OpenException ex){  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), ex.getMessage());  
 return;  
 }  
  
 if (selectorTable.getSelectedItem() == "Товары в аптеке")  
 {  
 scroll.setViewportView(preparatTable);  
 windowApp.add(scroll, BorderLayout.CENTER);  
 }  
 else{  
 if (preparatTableSold == null)  
 fillSoldTable();  
 else  
 {  
 scroll.setViewportView(preparatTableSold);  
 windowApp.add(scroll, BorderLayout.CENTER);  
 }  
 }  
 }  
 }  
 });  
 open.addActionListener(actionPressButton);  
 add.addActionListener(actionPressButton);  
 delete.addActionListener(actionPressButton);  
 buy.addActionListener(actionPressButton);  
 calc.addActionListener(actionPressButton);  
 edit.addActionListener(actionPressButton);  
 export.addActionListener(actionPressButton);  
 sales.addActionListener(actionPressButton);  
 filter.addActionListener(actionPressButton);  
 }  
  
 private JPanel createAddPanel(){  
 JPanel panel = new JPanel();  
 panel.setLayout(new GridLayout(0, 2, 15, 27));  
 panel.add(new JLabel("Название препарата: "));  
 panel.add(new JTextField("введите название",45));  
 panel.add(new JLabel("Болезнь: "));  
 panel.add(new JTextField("введите болезнь",45));  
 panel.add(new JLabel("Цена: "));  
 panel.add(new JTextField( "0",45));  
 panel.add(new JLabel("Количество: "));  
 panel.add(new JTextField("0",45));  
 return panel;  
 }

private JPanel createEditPanel(Preparat pr){  
 JPanel panel = new JPanel();  
 panel.setLayout(new GridLayout(0, 2, 15, 27));  
 panel.add(new JLabel("Название препарата: "));  
 panel.add(new JTextField(pr.getName(),45));  
 panel.add(new JLabel("Болезнь: "));  
 panel.add(new JTextField(pr.getIllness().getName(),45));  
 panel.add(new JLabel("Цена: "));  
 panel.add(new JTextField( Integer.toString(pr.getPrice()),45));  
 panel.add(new JLabel("Количество: "));  
 panel.add(new JTextField(Integer.toString(pr.getCount()),45));  
 panel.add(new JLabel("Проданное количество: "));  
 panel.add(new JTextField(Integer.toString(pr.getCntSold()),45));  
 return panel;  
 }  
  
 private void addNote(String name, String ill, String price\_, String cnt\_) throws PharmFieldsException {  
 int price, cnt;  
 try{  
 price = Integer.parseInt(price\_);  
 cnt = Integer.parseInt(cnt\_);  
 }  
 catch (NumberFormatException | NullPointerException nfe){  
 throw new PharmFieldsException();//вызов ошибки добавления препарата  
 }  
 if ((name.equals("введите название")) ||(ill.equals("введите болезнь"))||(price <= 0)||(cnt <= 0))  
 throw new PharmFieldsException();//вызов ошибки добавления препарата  
 if (!em.getTransaction().isActive())  
 em.getTransaction().begin();  
  
 //проверка на существование болезни  
 String querystr = "SELECT i FROM Illness i WHERE i.name like '" + ill + "'";//создание запроса  
 List<Illness> query = em.createQuery(querystr).getResultList();  
 Illness ill\_search;  
 if (query.size() != 0)//если нашли болезнь  
 ill\_search = query.get(0);  
 else//создаём болезнь  
 {  
 ill\_search = new Illness(ill);  
 em.persist(ill\_search);  
 }  
  
 Preparat pr = new Preparat(ill\_search, name, price, cnt);  
  
 Apteka.addPreparat(pr);  
 em.persist(pr);  
 em.persist(Apteka);  
 em.getTransaction().commit();  
 model.addRow(new Object[]{ pr.getId(), name, ill, cnt, 0, price});  
 return;  
 }  
  
 private void editNote(Preparat pr, String name, String ill, String price\_, String cnt\_, String cntSold\_) throws PharmFieldsException {  
 int price, cnt, cntSold;  
 try{  
 price = Integer.parseInt(price\_);  
 cnt = Integer.parseInt(cnt\_);  
 cntSold = Integer.parseInt(cntSold\_);  
 if ((price <= 0)||(cnt < 0)||(cntSold < 0))  
 throw new NumberFormatException();  
 }  
 catch (NumberFormatException | NullPointerException nfe){  
 throw new PharmFieldsException();//вызов ошибки изменения препарата  
 }  
 if (!em.getTransaction().isActive())  
 em.getTransaction().begin();  
  
 //проверка на существование болезни  
 String querystr = "SELECT i FROM Illness i WHERE i.name like '" + ill + "'";//создание запроса  
 List<Illness> query = em.createQuery(querystr).getResultList();  
 Illness ill\_search;  
 if (query.size() != 0)//если нашли болезнь  
 ill\_search = query.get(0);  
 else//создаём болезнь  
 {  
 ill\_search = new Illness(ill);  
 em.persist(ill\_search);  
 }  
  
 pr.setName(name);  
 pr.setIllness(ill\_search);  
 pr.setCntSold(cntSold);  
 pr.setCount(cnt);  
 pr.setPrice(price);  
 em.persist(pr);  
 em.getTransaction().commit();  
  
 preparatTable.setValueAt(pr.getCount(), preparatTable.getSelectedRow(), 3);//изменяем значения в ячейках  
 preparatTable.setValueAt(pr.getCntSold(), preparatTable.getSelectedRow(), 4);  
 preparatTable.setValueAt(pr.getName(), preparatTable.getSelectedRow(), 1);  
 preparatTable.setValueAt(pr.getIllness().getName(), preparatTable.getSelectedRow(), 2);  
 preparatTable.setValueAt(pr.getPrice(), preparatTable.getSelectedRow(), 5);  
 model.fireTableCellUpdated(preparatTable.getSelectedRow(), 3);//обновляем ячейки  
 model.fireTableCellUpdated(preparatTable.getSelectedRow(), 4);  
 model.fireTableCellUpdated(preparatTable.getSelectedRow(), 1);  
 model.fireTableCellUpdated(preparatTable.getSelectedRow(), 2);  
 model.fireTableCellUpdated(preparatTable.getSelectedRow(), 5);  
 return;  
 }  
  
 private void checkActiveDB() throws OpenException {  
 if (Apteka == null)//если не открыта аптека  
 throw new OpenException();  
 }  
  
 private boolean checkRightTable(){  
 if (selectorTable.getSelectedItem().toString() != "Товары в аптеке")  
 {  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), "<html>Выберите таблицу товаров");  
 return false;  
 }  
 return true;  
 }  
  
 private boolean checkSelectRow(){//проверяет выделена ли одна строка  
 if (preparatTable.getSelectedRowCount() != 1)  
 {  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), "<html>Выберите одну строку в таблице");  
 return false;  
 }  
 return true;  
 }  
  
 private void fillTable(){  
 TypedQuery<Preparat> query = em.createQuery(  
 "SELECT p FROM Preparat p WHERE p.pharmacy.id = :id", Preparat.class);  
 List<Preparat> preparats = query.setParameter("id", Apteka.getId()).getResultList();  
 String [][] data = new String[preparats.size()][6];  
 for (int i = 0; i < preparats.size(); i++) {  
 data[i][0] = Integer.toString(preparats.get(i).getId());  
 data[i][1] = preparats.get(i).getName();  
 data[i][2] = preparats.get(i).getIllness().getName();  
 data[i][3] = Integer.toString(preparats.get(i).getCount());  
 data[i][4] = Integer.toString(preparats.get(i).getCntSold());  
 data[i][5] = Integer.toString(preparats.get(i).getPrice());  
 }  
 String [] columns = {"Id", "Название", "Болезнь", "Количество", "Продано", "Цена"};  
 model= new DefaultTableModel(data, columns){  
 @Override  
 public boolean isCellEditable(int row, int column) {  
 return false;  
 }  
 };  
 preparatTable.setModel(model);  
 scroll.setViewportView(preparatTable);  
 windowApp.add(scroll, BorderLayout.CENTER);  
 }  
  
 private void fillSoldTable(){  
 preparatTableSold = new JTable();  
 SimpleDateFormat formatForDateNow = new SimpleDateFormat("HH:mm dd.MM.yyyy");  
 TypedQuery<SoldPreparat> query = em.createQuery(  
 "SELECT s FROM SoldPreparat s WHERE s.pharmacy.id = :id", SoldPreparat.class);  
 List<SoldPreparat> sldPreparats = query.setParameter("id", Apteka.getId()).getResultList();

String [][] data = new String[sldPreparats.size()][7];  
 for (int i = 0; i < sldPreparats.size(); i++) {  
 data[i][0] = Integer.toString(sldPreparats.get(i).getId());  
 data[i][1] = Integer.toString(sldPreparats.get(i).getPreparat().getId());  
 data[i][2] = sldPreparats.get(i).getName();  
 data[i][3] = sldPreparats.get(i).getIllness().getName();  
 data[i][4] = Integer.toString(sldPreparats.get(i).getCntSold());  
 data[i][5] = Integer.toString(sldPreparats.get(i).getPrice());  
 data[i][6] = formatForDateNow.format(sldPreparats.get(i).getDate());  
 }  
 String [] columns = {"Id","Id препарата", "Название", "Болезнь", "Продано", "Цена", "Время продажи"};  
 modelSold= new DefaultTableModel(data, columns){  
 @Override  
 public boolean isCellEditable(int row, int column) {  
 return false;  
 }  
 };  
 preparatTableSold.setModel(modelSold);  
 scroll.setViewportView(preparatTableSold);  
 windowApp.add(scroll, BorderLayout.CENTER);  
 }  
  
 private void addFunc(){  
 JPanel addWin = createAddPanel();  
 dialog = new JDialog(windowApp, "Добавление записи", true);  
 //нажатие "добавить"  
 JButton addBtn = new JButton("Добавить");  
 ActionListener actionPressAdd = new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 Component[] components = addWin.getComponents();  
 try{  
 addNote(((JTextField)components[1]).getText(), ((JTextField)components[3]).getText(), ((JTextField)components[5]).getText(), ((JTextField)components[7]).getText());  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), "<html>Препарат добавлен");  
 dialog.dispose();  
 }  
 catch (PharmFieldsException aex){  
 logger.error("an attempt to enter incorrect fields when adding a preparat");  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true),aex.getMessage());  
 }  
 }  
 };  
 addBtn.addActionListener(actionPressAdd);  
  
 dialog.setResizable(false);//окно нельзя изменять в размере  
 dialog.setPreferredSize(new Dimension(300, 250));  
 dialog.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);  
 dialog.setLocation(250, 250);  
 dialog.add(addWin, BorderLayout.NORTH);  
 dialog.add(addBtn, BorderLayout.SOUTH);  
 dialog.pack();  
 dialog.setVisible(true);  
 }  
  
 private void deleteFunc(){  
 if (!em.getTransaction().isActive())  
 em.getTransaction().begin();  
 Preparat prd = em.find(Preparat.class, Integer.parseInt(preparatTable.getValueAt(preparatTable.getSelectedRow(), 0).toString()));  
 Illness il = em.find(Illness.class, prd.getIllness().getId());  
 Apteka.deletePreparat(prd);  
 il.deletePreparat(prd);  
 prd.executePr();  
 em.remove(prd);  
 em.getTransaction().commit();  
 model.removeRow(preparatTable.getSelectedRow());  
 model.fireTableDataChanged();  
 }  
  
 private void editFunc(){  
 if (!em.getTransaction().isActive())  
 em.getTransaction().begin();  
 Preparat pre = em.find(Preparat.class, Integer.parseInt(preparatTable.getValueAt(preparatTable.getSelectedRow(), 0).toString()));  
 JPanel editWin = createEditPanel(pre);  
 dialog = new JDialog(windowApp, "Изменить информацию о товаре", true);  
 JButton editBtn = new JButton("Изменить информацию");  
 ActionListener actionPressEdit = new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 Component[] components = editWin.getComponents();  
 try{  
 editNote( pre, ((JTextField)components[1]).getText(), ((JTextField)components[3]).getText(), ((JTextField)components[5]).getText(), ((JTextField)components[7]).getText(), ((JTextField)components[9]).getText());  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), "<html>Информация изменена");  
 dialog.dispose();  
 }  
 catch (PharmFieldsException aex){  
 logger.error("an attempt to enter incorrect fields when editing a preparat");  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true),aex.getMessage());  
 }  
 }  
 };  
 editBtn.addActionListener(actionPressEdit);  
 dialog.setResizable(false);//окно нельзя изменять в размере  
 dialog.setPreferredSize(new Dimension(300, 300));  
 dialog.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);  
 dialog.setLocation(250, 250);  
 dialog.add(editBtn, BorderLayout.SOUTH);  
 dialog.add(editWin, BorderLayout.NORTH);  
 dialog.pack();  
 dialog.setVisible(true);  
 em.persist(pre);  
 if (!em.getTransaction().isActive())  
 em.getTransaction().begin();  
 em.getTransaction().commit();  
 preparatTable.setValueAt(pre.getCount(), preparatTable.getSelectedRow(), 3);//изменяем значения в ячейках  
 preparatTable.setValueAt(pre.getCntSold(), preparatTable.getSelectedRow(), 4);  
 model.fireTableCellUpdated(preparatTable.getSelectedRow(), 3);//обновляем ячейки  
 model.fireTableCellUpdated(preparatTable.getSelectedRow(), 4);  
 }  
  
 private void buyFunc(){  
 if (!em.getTransaction().isActive())  
 em.getTransaction().begin();  
 Preparat pr = em.find(Preparat.class, Integer.parseInt(preparatTable.getValueAt(preparatTable.getSelectedRow(), 0).toString()));  
 dialog = new JDialog(windowApp, "Покупка", true);  
 JButton buyBtn = new JButton("Приобрести товар");  
 JTextField buyTxt = new JTextField("1");  
 ActionListener actionPressBuy = new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 try {  
 int c = Integer.parseInt(buyTxt.getText());  
 if (c<0)  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), "<html>Ошибка! Число должно быть положительным");  
 else  
 {  
 if (c > pr.getCount())  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), String.format("<html>В наличии только %d единиц", pr.getCount()));  
 else  
 {  
 SoldPreparat sldPr = pr.buyPreparat(c);  
 em.persist(pr);  
 em.persist(sldPr);  
 em.getTransaction().commit();  
 preparatTable.setValueAt(pr.getCount(), preparatTable.getSelectedRow(), 3);//изменяем значения в ячейках  
 preparatTable.setValueAt(pr.getCntSold(), preparatTable.getSelectedRow(), 4);  
 model.fireTableCellUpdated(preparatTable.getSelectedRow(), 3);//обновляем ячейки  
 model.fireTableCellUpdated(preparatTable.getSelectedRow(), 4);  
 if (preparatTableSold != null)  
 modelSold.addRow(new Object[]{ sldPr.getId(), pr.getId(), sldPr.getName(), sldPr.getIllness().getName(), sldPr.getCntSold(), sldPr.getPrice(), new SimpleDateFormat("HH:mm dd.MM.yyyy").format(sldPr.getDate())});  
 dialog.dispose();  
 }  
 }  
 } catch (NumberFormatException | NullPointerException nfe) {  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), "<html>Ошибка! Нужно ввести число");  
 }  
 }  
 };  
 buyBtn.addActionListener(actionPressBuy);  
 dialog.setResizable(false);//окно нельзя изменять в размере  
 dialog.setPreferredSize(new Dimension(250, 115));  
 dialog.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);  
 dialog.setLocation(250, 250);  
 dialog.add(new JLabel("Введите количество товара: "), BorderLayout.NORTH);  
 dialog.add(buyTxt, BorderLayout.CENTER);  
 dialog.add(buyBtn, BorderLayout.SOUTH);  
 dialog.pack();  
 dialog.setVisible(true);  
 }  
  
 private void searchFunc(){  
 String a =(String) kind.getSelectedItem();  
 if(a.equals("Препарат"))  
 {  
 int b = Apteka.findPreparat(searchName.getText());  
 if (b != -1)  
 preparatTable.setRowSelectionInterval(b, b);//выделяем строку с таким препаратом  
 else  
 {  
 preparatTable.clearSelection();  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), "<html>Такого препарата нет");  
 }  
 }  
 if(a.equals("Болезнь"))  
 {  
 List<Preparat> rightPrep = Apteka.findListDrugs(searchName.getText());  
 if (rightPrep.size() == 0)  
 {  
 preparatTable.clearSelection();  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), "<html>Таких препаратов нет");  
 }  
 else  
 {  
 String str = "Подходящие препараты: ";  
 for (int i=0; i < rightPrep.size(); ++i)  
 str += "\n" + rightPrep.get(i).getId() + " " + rightPrep.get(i).getName();  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), str);  
 }  
  
 }  
 }  
  
 private void openFunc(){  
 if (!em.getTransaction().isActive())  
 em.getTransaction().begin();  
 preparatTableSold = null;  
 preparatTable = new JTable();  
 dialog = new JDialog(windowApp, "Открытие БД", true);  
 JButton openBtn = new JButton("Открыть БД");  
 JButton addPharmBtn = new JButton("Добавить новую БД аптеки");  
 JTextField adresPharm = new JTextField("Введите адрес новой аптеки", 20);  
 List<Pharmacy> listPharmacy = em.createQuery("SELECT p FROM Pharmacy p").getResultList();  
 String [][] data = new String[listPharmacy.size()][2];  
 for (int i = 0; i < listPharmacy.size(); i++) {  
 data[i][0] = Integer.toString(listPharmacy.get(i).getId());  
 data[i][1] = listPharmacy.get(i).getAddress();  
 }  
 String [] columns = {"Id", "Адрес"};  
 DefaultTableModel modelPh = new DefaultTableModel(data, columns){  
 @Override  
 public boolean isCellEditable(int row, int column) {  
 return false;  
 }  
 };  
 JTable pharmTable = new JTable(modelPh);  
 JScrollPane pane = new JScrollPane(pharmTable);  
 JPanel addPharmPanel = new JPanel();  
 addPharmPanel.setLayout(new GridLayout(0, 1, 1, 1));  
 addPharmPanel.add(adresPharm);  
 addPharmPanel.add(addPharmBtn);  
 addPharmPanel.add(openBtn);  
  
 ActionListener actionPressOpen = new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (pharmTable.getSelectedRow() == -1 || pharmTable.getSelectedRowCount() != 1)//если не выбрана строчка  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), "<html>Выберите одну строку в таблице");  
 else  
 {  
 Apteka = em.find(Pharmacy.class, Integer.parseInt(pharmTable.getValueAt(pharmTable.getSelectedRow(), 0).toString()));  
 dialog.dispose();  
 fillTable();//открываем таблицу  
 filterPanel.add(selectorTable);

filterPanel.add(new JLabel("Аптека: " + Apteka.getId() + " " + Apteka.getAddress()));  
 selectorTable.setSelectedIndex(0);  
 windowApp.add(filterPanel, BorderLayout.SOUTH);  
 windowApp.setVisible(true);  
 }  
 }  
 };  
 ActionListener actionPressAddPharm = new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (adresPharm.getText().equals("Введите адрес новой аптеки"))//если не выбрана строчка  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), "<html>Введите адрес аптеки");  
 else  
 {  
 Apteka = new Pharmacy(adresPharm.getText());  
 em.persist(Apteka);  
 dialog.dispose();  
 em.getTransaction().commit();  
 fillTable();//открываем таблицу  
 filterPanel.add(selectorTable);  
 windowApp.add(filterPanel, BorderLayout.SOUTH);  
 windowApp.setVisible(true);  
 }  
 }  
 };  
 openBtn.addActionListener(actionPressOpen);  
 addPharmBtn.addActionListener(actionPressAddPharm);  
 dialog.add(pane, BorderLayout.CENTER);  
 dialog.add(addPharmPanel, BorderLayout.SOUTH);  
 dialog.setResizable(false);//окно нельзя изменять в размере  
 dialog.setPreferredSize(new Dimension(300, 250));  
 dialog.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);  
 dialog.setLocation(250, 250);  
 dialog.pack();  
 dialog.setVisible(true);  
 }  
 private void calcFunc(){  
 int res=0;  
 Calendar a = Calendar.getInstance();  
 Calendar b = Calendar.getInstance();  
  
 dialog = new JDialog(windowApp, "Посчитать прибыль", true);  
 JButton defDateBtn = new JButton("Посчитать за последний месяц");  
 JButton dateBtn = new JButton("Посчитать за указанный период");  
 JPanel panel = createDatePanel();  
  
 dateBtn.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 int df, mf, yf, ds, ms, ys;  
 Component[] components = panel.getComponents();  
 try{  
 try  
 {  
 df = Integer.parseInt(((JTextField)components[1]).getText());  
 mf = Integer.parseInt(((JTextField)components[2]).getText());  
 yf = Integer.parseInt(((JTextField)components[3]).getText());  
 ds = Integer.parseInt(((JTextField)components[5]).getText());  
 ms = Integer.parseInt(((JTextField)components[6]).getText());  
 ys = Integer.parseInt(((JTextField)components[7]).getText());  
 }  
 catch (NumberFormatException | NullPointerException nfe){  
 throw new PharmFieldsException();}  
  
 a.set(yf, mf-1, df, 0, 0);  
 b.set(ys, ms-1, ds, 23, 59);  
 if ((df <= 0)||(mf <= 0)||(yf<=0)||(ds <= 0)||(ms <= 0)||(ys<=0)||(a.getTimeInMillis() > b.getTimeInMillis()))  
 throw new PharmFieldsException();  
 dialog.dispose();  
 }  
 catch ( PharmFieldsException aex){  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true),"Введите все числовые поля корректно");  
 }  
 }  
 });  
 defDateBtn.addActionListener((new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 a.set(b.get(Calendar.YEAR), b.get(Calendar.MONTH)-1, b.get(Calendar.DAY\_OF\_MONTH));  
 dialog.dispose();  
 }  
 }));  
 JPanel btnPanel = new JPanel();  
 btnPanel.setLayout(new GridLayout(0, 1, 1, 1));  
 btnPanel.add(dateBtn, BorderLayout.NORTH);  
 btnPanel.add(defDateBtn, BorderLayout.SOUTH);  
 dialog.add(panel, BorderLayout.NORTH);  
 dialog.add(btnPanel, BorderLayout.SOUTH);  
 dialog.setResizable(false);//окно нельзя изменять в размере  
 dialog.setPreferredSize(new Dimension(420, 180));  
 dialog.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);  
 dialog.setLocation(250, 250);  
 dialog.pack();  
 dialog.setVisible(true);  
 if (preparatTableSold == null)  
 {  
 fillSoldTable();  
 scroll.setViewportView(preparatTable);  
 windowApp.add(scroll, BorderLayout.CENTER);  
 }

TypedQuery<SoldPreparat> query = em.createQuery(  
 "SELECT s FROM SoldPreparat s WHERE s.date >=:fd AND s.date <= :sd AND s.pharmacy.id = :id", SoldPreparat.class);  
 query.setParameter("fd", a.getTime());  
 query.setParameter("sd", b.getTime());  
 query.setParameter("id", Apteka.getId());  
 List<SoldPreparat> sldPreparats = query.getResultList();  
 for (int i=0; i<sldPreparats.size(); ++i)  
 res += sldPreparats.get(i).getCntSold()\*sldPreparats.get(i).getPrice();  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true), String.format("<html>Прибыль: %d рублей", res));  
 }  
  
 private JPanel createDatePanel(){  
 JPanel panel = new JPanel();  
 panel.setLayout(new GridLayout(0, 4, 1, 10));  
 panel.add(new JLabel("С какого числа: "));  
 panel.add(new JTextField("День",45));  
 panel.add(new JTextField("Месяц",45));  
 panel.add(new JTextField("Год",45));  
 panel.add(new JLabel("По какое число: "));  
 panel.add(new JTextField("День",45));  
 panel.add(new JTextField("Месяц",45));  
 panel.add(new JTextField("Год",45));  
 return panel;  
 }  
  
 private void generateSalesFunc(){  
 Calendar a = Calendar.getInstance();  
 Calendar b = Calendar.getInstance();  
 dialog = new JDialog(windowApp, "Показать продажи", true);  
 JButton dateBtn = new JButton("Показать за указанный период");  
 JPanel panel = createDatePanel();  
  
 dateBtn.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 int df, mf, yf, ds, ms, ys;  
 Component[] components = panel.getComponents();  
 try{  
 try  
 {  
 df = Integer.parseInt(((JTextField)components[1]).getText());  
 mf = Integer.parseInt(((JTextField)components[2]).getText());  
 yf = Integer.parseInt(((JTextField)components[3]).getText());  
 ds = Integer.parseInt(((JTextField)components[5]).getText());  
 ms = Integer.parseInt(((JTextField)components[6]).getText());  
 ys = Integer.parseInt(((JTextField)components[7]).getText());  
 }  
 catch (NumberFormatException | NullPointerException nfe){  
 throw new PharmFieldsException();}  
 a.set(yf, mf-1, df, 0, 0);  
 b.set(ys, ms-1, ds, 23, 59);  
 if ((df <= 0)||(mf <= 0)||(yf<=0)||(ds <= 0)||(ms <= 0)||(ys<=0)||(a.getTimeInMillis() > b.getTimeInMillis()))  
 throw new PharmFieldsException();  
 dialog.dispose();  
 }  
 catch ( PharmFieldsException aex){  
 JOptionPane.showMessageDialog(new JDialog(windowApp, "", true),"Введите все числовые поля корректно");  
 }  
 }  
 });  
  
 dialog.add(panel, BorderLayout.NORTH);  
 dialog.add(dateBtn, BorderLayout.SOUTH);  
 dialog.setResizable(false);//окно нельзя изменять в размере  
 dialog.setPreferredSize(new Dimension(420, 180));  
 dialog.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);  
 dialog.setLocation(250, 250);  
 dialog.pack();  
 dialog.setVisible(true);  
  
 if (preparatTableSold == null)  
 {  
 fillSoldTable();  
 scroll.setViewportView(preparatTable);  
 windowApp.add(scroll, BorderLayout.CENTER);  
 }  
 TypedQuery<SoldPreparat> query = em.createQuery(  
 "SELECT s FROM SoldPreparat s WHERE s.date >=:fd AND s.date <= :sd AND s.pharmacy.id = :id", SoldPreparat.class);  
 query.setParameter("fd", a.getTime());  
 query.setParameter("sd", b.getTime());  
 query.setParameter("id", Apteka.getId());  
 List<SoldPreparat> sldPreparats = query.getResultList();  
 dialog = new JDialog(windowApp, "Проданные товары за период с " + new SimpleDateFormat("dd.MM.yyyy").format(a.getTime()) + " по " + new SimpleDateFormat("dd.MM.yyyy").format(b.getTime()), false);  
  
 String [][] data = new String[sldPreparats.size()][4];  
 List<SoldPreparat> uniqueSldPreparats = new ArrayList<SoldPreparat>();  
 boolean check = false;  
 int k=0;  
 for (int i=0; i< sldPreparats.size(); ++i) {  
 for (int j = 0; j < uniqueSldPreparats.size(); ++j)  
 if (uniqueSldPreparats.get(j).getPreparat().getId() == sldPreparats.get(i).getPreparat().getId())  
 {  
 check = true;  
 k=j;  
 j=uniqueSldPreparats.size();  
 }  
 if (check == false) {  
 data[uniqueSldPreparats.size()][0] = Integer.toString(sldPreparats.get(i).getPreparat().getId());  
 data[uniqueSldPreparats.size()][1] = sldPreparats.get(i).getName();  
 data[uniqueSldPreparats.size()][2] = sldPreparats.get(i).getIllness().getName();  
 data[uniqueSldPreparats.size()][3] = Integer.toString(sldPreparats.get(i).getCntSold());  
 uniqueSldPreparats.add(sldPreparats.get(i));  
 }  
 else{  
 data[k][3] = Integer.toString(Integer.parseInt(data[k][3]) + sldPreparats.get(i).getCntSold());  
 check = false;  
 }  
 }  
 String [][] data1 = new String[uniqueSldPreparats.size()][4];  
 for (int i = 0; i < uniqueSldPreparats.size(); ++i) {  
 data1[i][0] = data[i][0];  
 data1[i][1] = data[i][1];  
 data1[i][2] = data[i][2];  
 data1[i][3] = data[i][3];  
 }  
 String [] columns = {"Id", "Название", "Болезнь", "Количество"};  
 DefaultTableModel modelSales = new DefaultTableModel(data1, columns){  
 @Override  
 public boolean isCellEditable(int row, int column) {  
 return false;  
 }  
 };  
 JTable pharmSales = new JTable(modelSales);  
 JScrollPane pane = new JScrollPane(pharmSales);  
 dialog.add(pane, BorderLayout.CENTER);  
 dialog.setResizable(false);//окно нельзя изменять в размере  
 dialog.setPreferredSize(new Dimension(420, 360));  
 dialog.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);  
 dialog.setLocation(250, 250);  
 dialog.pack();  
 dialog.setVisible(true);  
 }  
}

**Export:**

package App;  
import java.awt.\*;  
import java.io.FileOutputStream;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import com.itextpdf.text.\*;  
import com.itextpdf.text.Document;  
import com.itextpdf.text.Font;  
import com.itextpdf.text.pdf.\*;  
import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
  
public class Export extends Thread {  
 private DefaultTableModel modelTable;  
 private JFrame windApp;  
 int id, res;  
 String addres;  
  
 public Export(DefaultTableModel tab, JFrame jf, Pharmacy apteka) {  
 this.modelTable = tab;  
 this.windApp = jf;  
 this.id = apteka.getId();  
 this.addres = apteka.getAddress();  
 this.res = apteka.calcAllSold();  
 }  
  
 public void run() {  
 synchronized (modelTable)// тк общий ресурс для нескольких потоков  
 {  
 // Блок который подлежит синхронизации  
 try {  
 SimpleDateFormat formatForDateNow1 = new SimpleDateFormat("HH-mm-ss dd.MM.yyyy");//изменяем формат вывода данных, тк нельзя использовать ":"  
 SimpleDateFormat formatForDateNow2 = new SimpleDateFormat("HH:mm dd.MM.yyyy");  
 String[] columns = {"Id", "Название", "Болезнь", "Количество", "Продано", "Цена"};  
 String path = ".\\Reports\\PDFreport from " + formatForDateNow1.format(new Date()) + ".pdf";  
  
 Document document = new Document();  
 PdfWriter.getInstance(document, new FileOutputStream(path));  
 document.open();  
  
 Font fontHead = FontFactory.getFont(".\\fonts\\DejaVuSans.ttf", "cp1251", BaseFont.EMBEDDED, 10, Font.BOLD);  
 Font font = FontFactory.getFont(".\\fonts\\DejaVuSans.ttf", "cp1251", BaseFont.EMBEDDED, 10);  
  
 Paragraph title = new Paragraph("Таблица товаров. Аптека id:" + id + " Адрес: " + addres + "\nДата создания отчета: " + formatForDateNow2.format(new Date()) + "\n\n", font);  
 document.add(title);  
  
 PdfPTable tbl = new PdfPTable(modelTable.getColumnCount());  
 for (int i = 0; i < columns.length; ++i) {  
 Phrase tx = new Phrase(columns[i], fontHead);  
 PdfPCell cell = new PdfPCell(tx);  
 cell.setBackgroundColor(new BaseColor(Color.lightGray.getRGB()));  
 tbl.addCell(cell);  
 }  
  
 for (int i = 0; i < modelTable.getRowCount(); i++)  
 for (int j = 0; j < modelTable.getColumnCount(); j++)  
 tbl.addCell(new Phrase(modelTable.getValueAt(i, j).toString(), font));  
 document.add(tbl);  
  
 Paragraph totalSales = new Paragraph("Всего было продано товаров на сумму: " + res + " рублей.", font);  
 document.add(totalSales);  
  
 document.close();  
 JOptionPane.showMessageDialog(windApp, "Готово. Создан PDF-отчет (папка Reports)");  
 } catch (Exception e) {  
 JOptionPane.showMessageDialog(windApp, e.toString());  
 }  
 }  
 }  
}

**PharmFieldsException:**

package App;  
  
public class PharmFieldsException extends Exception{  
 public PharmFieldsException()  
 {  
 super("Ошибка добавления/изменений, введите все поля корректно");  
 }  
}

**OpenException:**

package App;  
  
public class OpenException extends Exception{  
 public OpenException()  
 {  
 super("Сначала необходимо открыть БД аптеки");  
 }  
}

**Классы JUnit тестов**

**PharmacyTest:**

package App;  
import org.junit.Test;  
import java.util.ArrayList;  
import java.util.List;  
import static org.junit.Assert.\*;  
  
public class PharmacyTest {  
  
 @Test  
 public void addPreparat() {  
 Pharmacy pharmacy = new Pharmacy();  
 Preparat preparat = new Preparat();  
 Preparat preparat2 = new Preparat();  
 pharmacy.addPreparat(preparat);  
 assertSame(preparat, pharmacy.getListOfAllPreparats().get(pharmacy.getListOfAllPreparats().size()-1));  
 assertNotSame(preparat2, pharmacy.getListOfAllPreparats().get(pharmacy.getListOfAllPreparats().size()-1));//проверяем что не добавился другой  
 }  
  
 @Test  
 public void deletePreparat() {  
 Pharmacy pharmacy = new Pharmacy();  
 Preparat preparat = new Preparat();  
 pharmacy.addPreparat(preparat);  
 assertSame(preparat, pharmacy.getListOfAllPreparats().get(pharmacy.getListOfAllPreparats().size()-1));//проверка добавилось ли  
 pharmacy.deletePreparat(preparat);  
 assertNull(pharmacy.getListOfAllPreparats());//проверка удаления  
 }  
  
 @Test  
 public void findPreparat() {  
 Pharmacy pharmacy = new Pharmacy();  
 Preparat preparat1 = new Preparat();  
 preparat1.setName("Лекарство№1");  
 preparat1.setCount(1);  
 pharmacy.addPreparat(preparat1);  
  
 assertSame(preparat1, pharmacy.getListOfAllPreparats().get(pharmacy.findPreparat("Лекарство№1")));//нашёлся препарат  
 assertEquals(-1, pharmacy.findPreparat("Лекарство№2"));//не нашелся другой препарат  
 }  
  
 @Test  
 public void calcAllSold() {  
 Pharmacy pharmacy = new Pharmacy();  
 Preparat preparat1 = new Preparat();  
 int cntS1 = 15, cntS2 = 100, price1\_ = 500, price2\_ = 20;  
 preparat1.setCntSold(cntS1);  
 preparat1.setPrice(price1\_);  
 pharmacy.addPreparat(preparat1);  
  
 Preparat preparat2 = new Preparat();  
 preparat2.setCntSold(cntS2);  
 preparat2.setPrice(price2\_);  
 pharmacy.addPreparat(preparat2);  
  
 assertEquals(price1\_\*cntS1 + price2\_\*cntS2, pharmacy.calcAllSold());//9500  
 }  
  
 @Test  
 public void getListOfAllPreparats() {  
 Pharmacy pharmacy = new Pharmacy();  
 Pharmacy pharmacyEmpty = new Pharmacy();  
 Preparat preparat1 = new Preparat();  
 Preparat preparat2 = new Preparat();  
  
 List<Preparat> list = new ArrayList<Preparat>();  
 list.add(preparat1);  
 list.add(preparat2);  
  
 pharmacy.setListOfAllPreparats(list);  
 assertSame(list, pharmacy.getListOfAllPreparats());  
 assertNull(pharmacyEmpty.getListOfAllPreparats());//возвращает null, если нет лекарств в аптеке  
 }  
  
  
  
 @Test  
 public void getAddress() {  
 Pharmacy pharmacy = new Pharmacy();  
 String s = "Popova 5";  
 pharmacy.setAddress(s);  
 assertEquals(s, pharmacy.getAddress());  
 }  
}

**IllnessTest:**

package App;  
import org.junit.Test;  
import java.util.ArrayList;  
import java.util.List;  
import static org.junit.Assert.\*;

public class IllnessTest {

@Test  
 public void getRightDrugs() {  
 Illness illness = new Illness();  
 Preparat preparat1 = new Preparat();  
 Preparat preparat2 = new Preparat();  
 Preparat preparat3 = new Preparat();  
 illness.addNewPreparat(preparat1);  
 illness.addNewPreparat(preparat2);  
 illness.addNewPreparat(preparat3);  
 List<Preparat> list = new ArrayList<Preparat>();  
 list.add(preparat1);  
 list.add(preparat2);  
 list.add(preparat3);  
 for (int i=0; i<list.size() && i<illness.getRightDrugs().size(); ++i)  
 assertSame(list.get(i), illness.getRightDrugs().get(i));  
 }

@Test  
 public void addNewPreparat() {  
 Illness illness = new Illness();  
 Preparat preparat = new Preparat();  
 illness.addNewPreparat(preparat);  
 assertSame(preparat, illness.getRightDrugs().get(illness.getRightDrugs().size()-1));  
 }

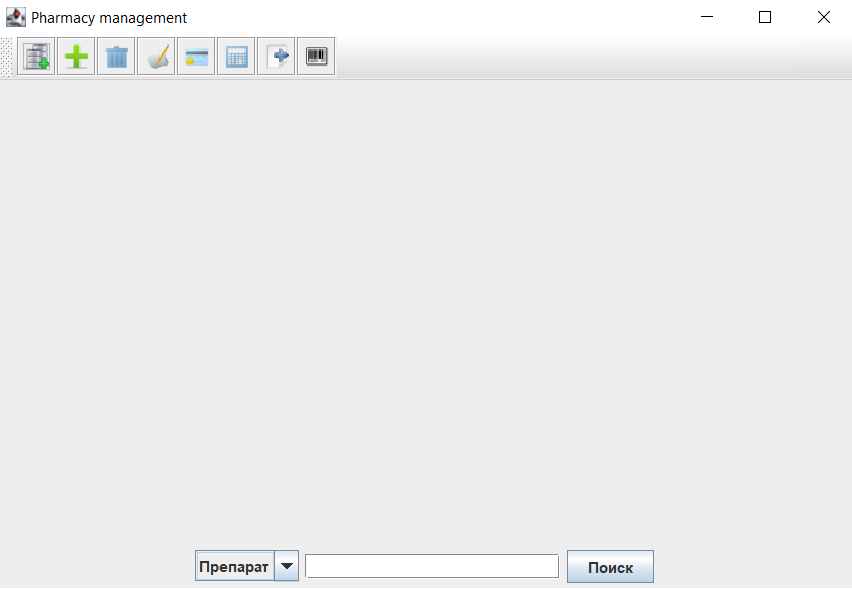
@Test  
 public void deletePreparat() {  
 Illness illness = new Illness();  
 Preparat preparat = new Preparat();  
 illness.addNewPreparat(preparat);  
 assertSame(preparat, illness.getRightDrugs().get(illness.getRightDrugs().size()-1));//проверка добавилось ли  
 illness.deletePreparat(preparat);  
 assertNull(illness.getRightDrugs());//проверка удаления  
 }

@Test  
 public void getName() {  
 Illness illness = new Illness();  
 String s = "Ill";  
 illness.setName(s);  
 assertEquals(s, illness.getName());  
 }  
}

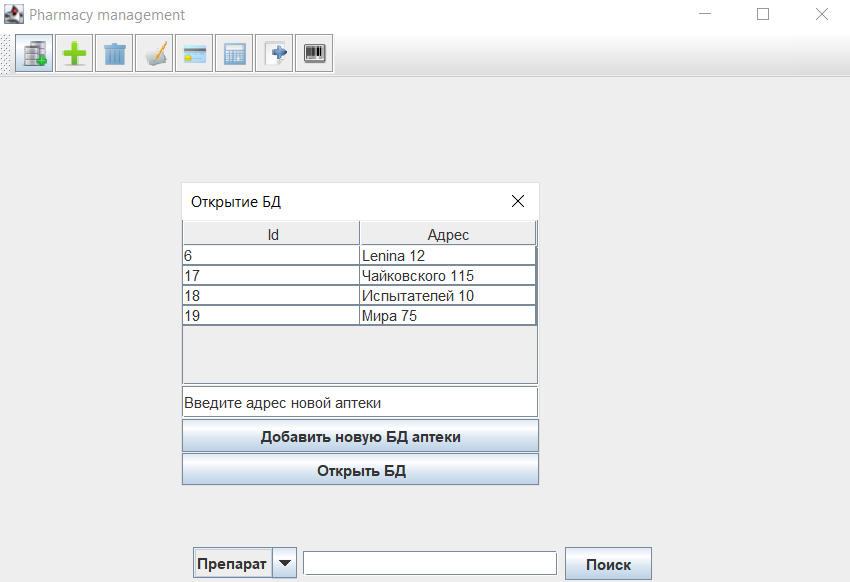
**PreparatTest:**

package App;  
import org.junit.Test;  
import java.io.ByteArrayOutputStream;  
import java.io.PrintStream;  
import static org.junit.Assert.\*;  
public class PreparatTest {  
  
 @Test  
 public void buyPreparat() {  
 Preparat preparat = new Preparat();  
 int cnt = 100, buyCnt = 5;  
 preparat.setCount(cnt);  
 preparat.buyPreparat(buyCnt);  
 assertEquals(cnt - buyCnt, preparat.getCount());  
 assertEquals(buyCnt, preparat.getCntSold());  
 }  
  
 @Test  
 public void calcSold() {  
 Preparat preparat = new Preparat();  
 int cnt = 100, price\_ = 200, buyCnt = 5;  
 preparat.setCount(cnt);  
 preparat.setPrice(price\_);  
 preparat.buyPreparat(buyCnt);  
 assertEquals(buyCnt\*price\_, preparat.calcSold());  
 }  
  
 @Test  
 public void executePr() {  
 Preparat preparat = new Preparat();  
 Illness illness = new Illness();  
 Pharmacy pharmacy = new Pharmacy();  
 preparat.setPharmacy(pharmacy);  
 preparat.setIllness(illness);  
 preparat.executePr();  
 //assertNotSame(illness, preparat.getIllness());  
 //assertNotSame(pharmacy, preparat.getPharmacy());  
 assertSame(null, preparat.getIllness());  
 assertSame(null, preparat.getPharmacy());  
 }  
  
 @Test  
 public void getName() {  
 Preparat preparat = new Preparat();  
 String s = "Drug";  
 preparat.setName(s);  
 assertEquals(s, preparat.getName());  
 }  
  
 @Test  
 public void getIdIll() {  
 Preparat preparat = new Preparat();  
 Illness illness = new Illness();  
 preparat.setIllness(illness);  
 assertEquals(illness.getId(), preparat.getIllness().getId());  
 }  
  
 @Test  
 public void getNameIll() {  
 Preparat preparat = new Preparat();  
 String s = "Суставная боль";  
 Illness illness = new Illness(s);  
 preparat.setIllness(illness);  
 assertEquals(s, preparat.getIllness().getName());  
 }  
  
 @Test  
 public void getIdPhar() {  
 Preparat preparat = new Preparat();  
 Pharmacy pharmacy = new Pharmacy();  
 preparat.setPharmacy(pharmacy);  
 assertEquals(pharmacy.getId(), preparat.getPharmacy().getId());  
 }  
  
 @Test  
 public void getPrice() {  
 Preparat preparat = new Preparat();  
 int price\_ = 500;  
 preparat.setPrice(price\_);  
 assertEquals(price\_, preparat.getPrice());  
 }  
  
 @Test  
 public void getCount() {  
 Preparat preparat = new Preparat();  
 int cnt = 10;  
 preparat.setCount(cnt);  
 assertEquals(cnt, preparat.getCount());  
 }  
  
 @Test  
 public void getCntSold() {  
 Preparat preparat = new Preparat();  
 int cntSold = 1;  
 preparat.setCntSold(cntSold);  
 assertEquals(cntSold, preparat.getCntSold());  
 }  
  
 @Test  
 public void getPharmacy() {  
 Preparat preparat = new Preparat();  
 Pharmacy pharmacy = new Pharmacy();  
 preparat.setPharmacy(pharmacy);  
 assertSame(pharmacy, preparat.getPharmacy());  
 }  
  
 @Test  
 public void getIllness() {  
 Preparat preparat = new Preparat();  
 Illness illness = new Illness();  
 preparat.setIllness(illness);  
 assertSame(illness, preparat.getIllness());  
 }  
}

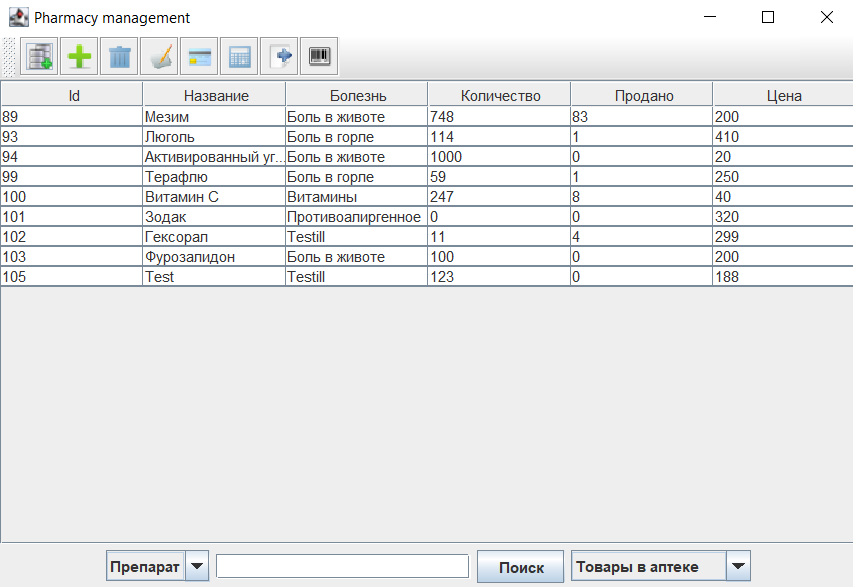
# Интерфейс



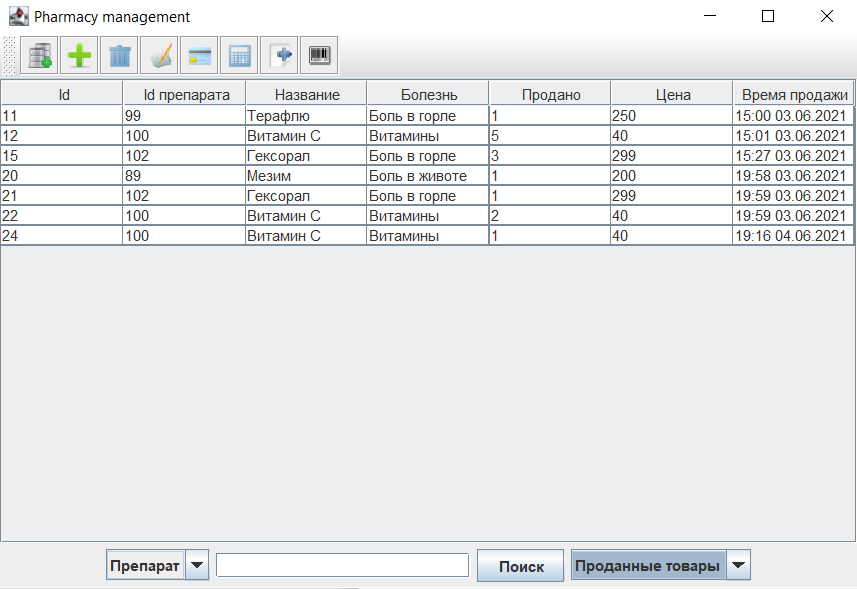
*Рисунок 10. Окно приложения сразу после запуска*



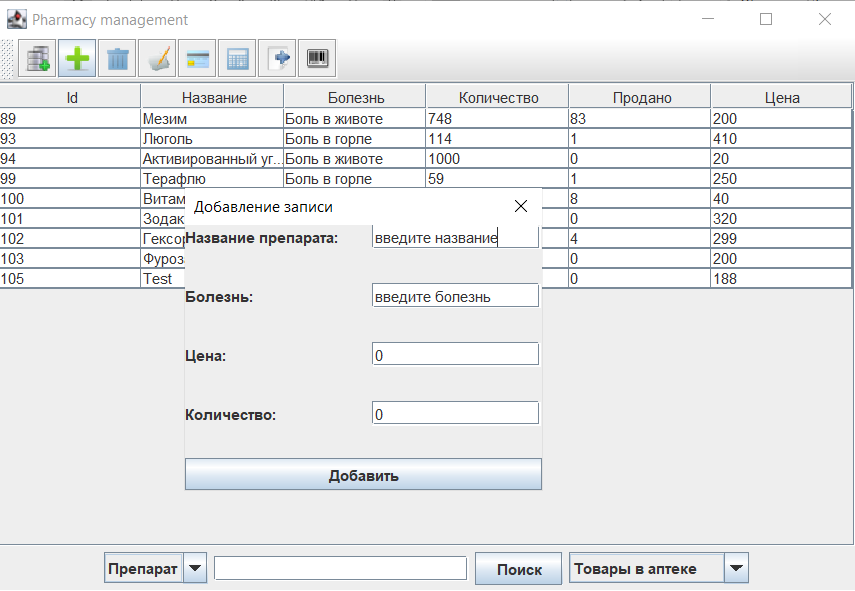
*Рисунок 11. Окно выбора аптеки*



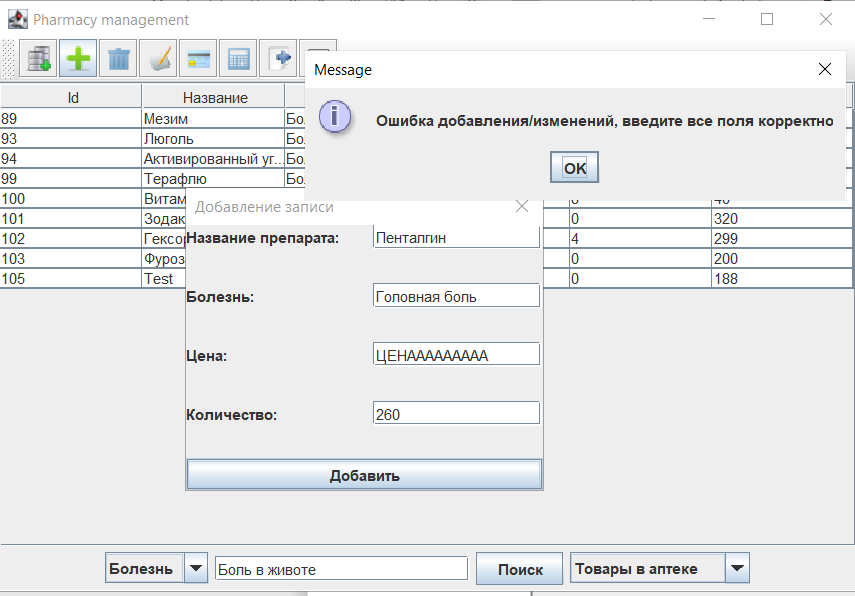
*Рисунок 12. Окно приложения с активной таблицей препаратов в аптеке*



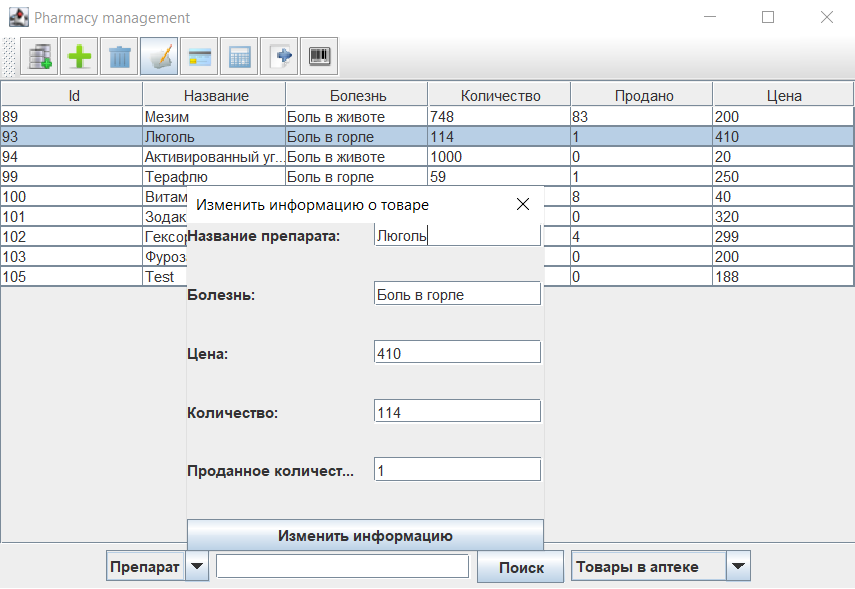
*Рисунок 13. Окно приложения с активной таблицей проданных товаров*



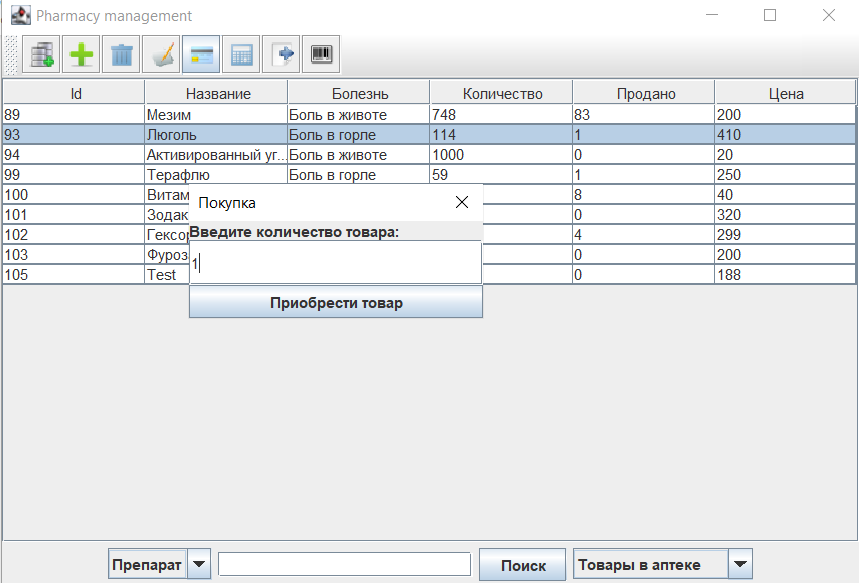
*Рисунок 14. Окно добавления нового препарата*



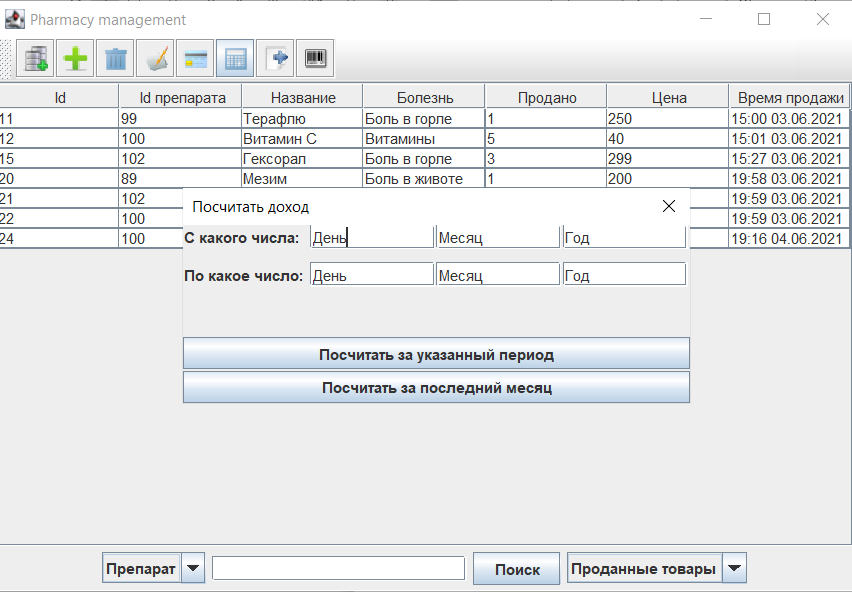
*Рисунок 15. Сообщение об ошибке в информационном поле*



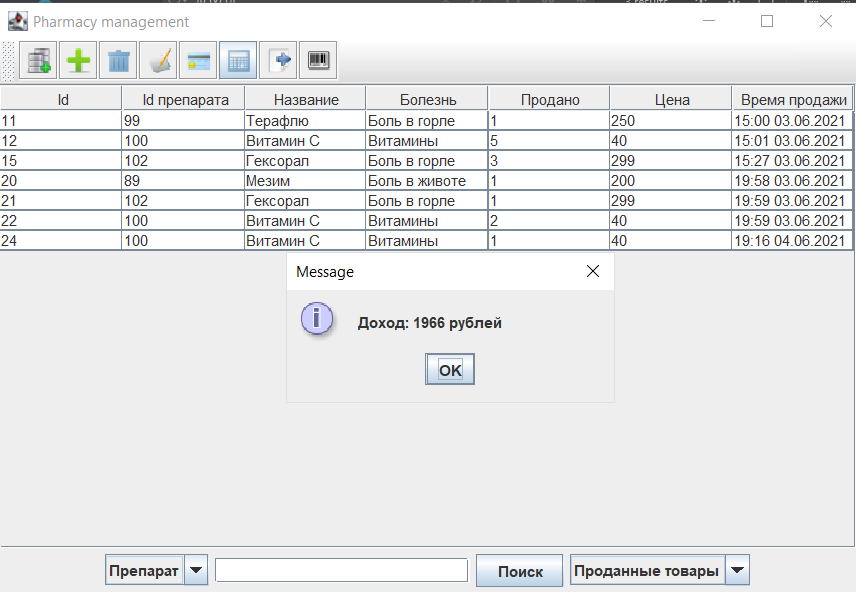
*Рисунок 16. Окно изменения информации о препарате*



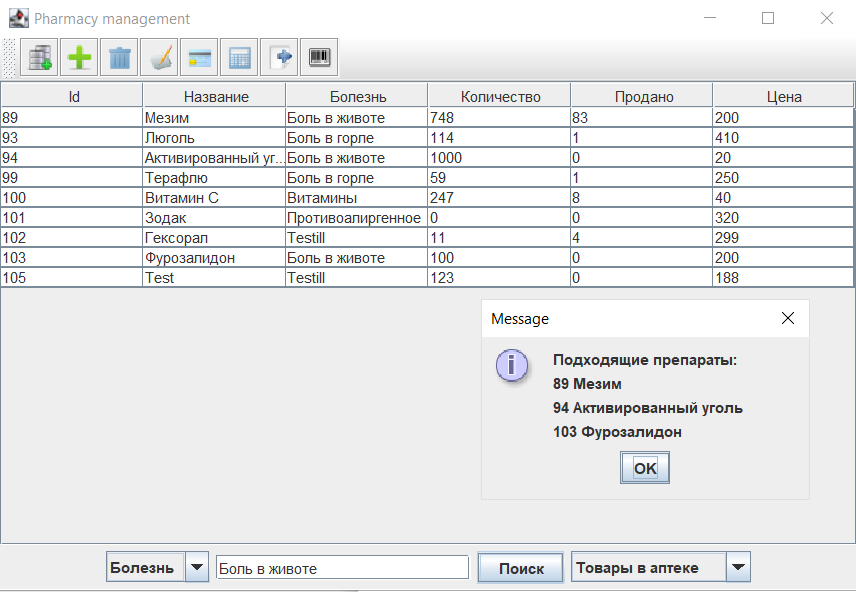
*Рисунок 17. Окно покупки товара*



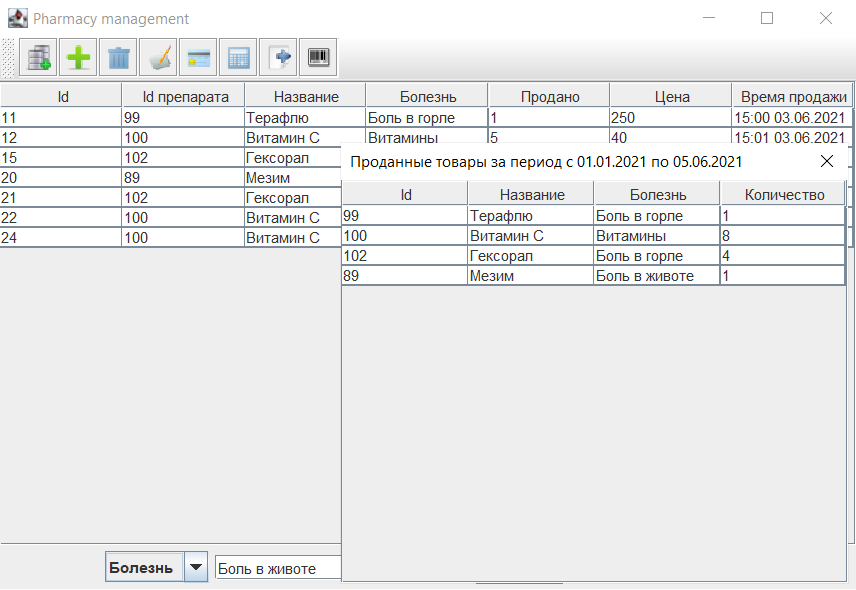
*Рисунок 18. Окно подсчета дохода за период времени*



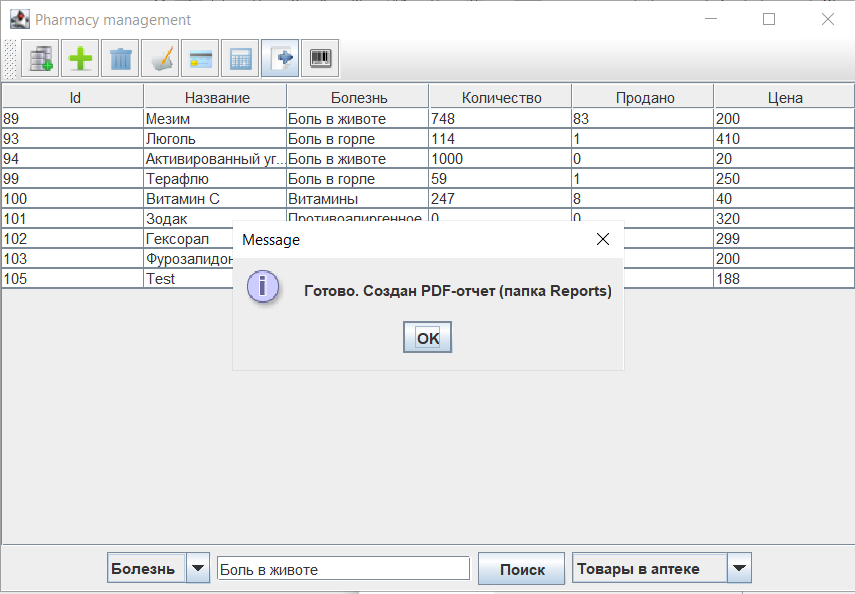
*Рисунок 19. Сообщение о посчитанном доходе*



*Рисунок 20. Сообщение о подходящих лекарствах для лечения определенной болезни*



*Рисунок 21. Окно с информацией о проданных препаратах за определенный период времени*



*Рисунок 22. Сообщение о успешном создании PDF отчета*

# Вывод

В результате выполнения этой работы был разработан ПК для администратора аптеки по управлению данными с помощью графического интерфейса. Благодаря этому были закреплены теоретические знания и получены практические навыки в проектировании и разработке ПО на языке Java с использованием большого количества технологий.