Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Санкт-Петербургский государственный электротехнический университет

“ЛЭТИ” им.В.И.Ульянова (Ленина) »

Кафедра МОЭВМ

**ОТЧЕТ**

**по лабораторно-практической работе № 4**

**«Обработка исключений»**

**по дисциплине «Объектно - ориентированное программирование на языке Java»**

Выполнил Баймухамедов Р. Р.

Факультет КТИ

Группа № 3311

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2024 г

**Цель работы**

Знакомство с механизмом обработки исключений в языке Java.

**Добавление нового функционала**

На данном этапе добавлен функционал кнопкам “Добавить данные” и “Удалить данные”.

**Перечень ситуаций, которые контролируются с помощью исключений**

— Попытка нажатия кнопки поиска, при не введённом в поле ввода значении (пример 1)

— Попытка ввода значения неправильного формата времени, даты или количества проданных билетов при добавлении данных. (пример 2, 3 и 4 соответственно)

**Ссылка на репозиторий**

[https://github.com/brick1ng5654/course-2/tree/main/OOP/lab\_0](https://github.com/brick1ng5654/course-2/tree/main/OOP/lab_02)4

В этом репозитории находятся исходные файлы проекта:

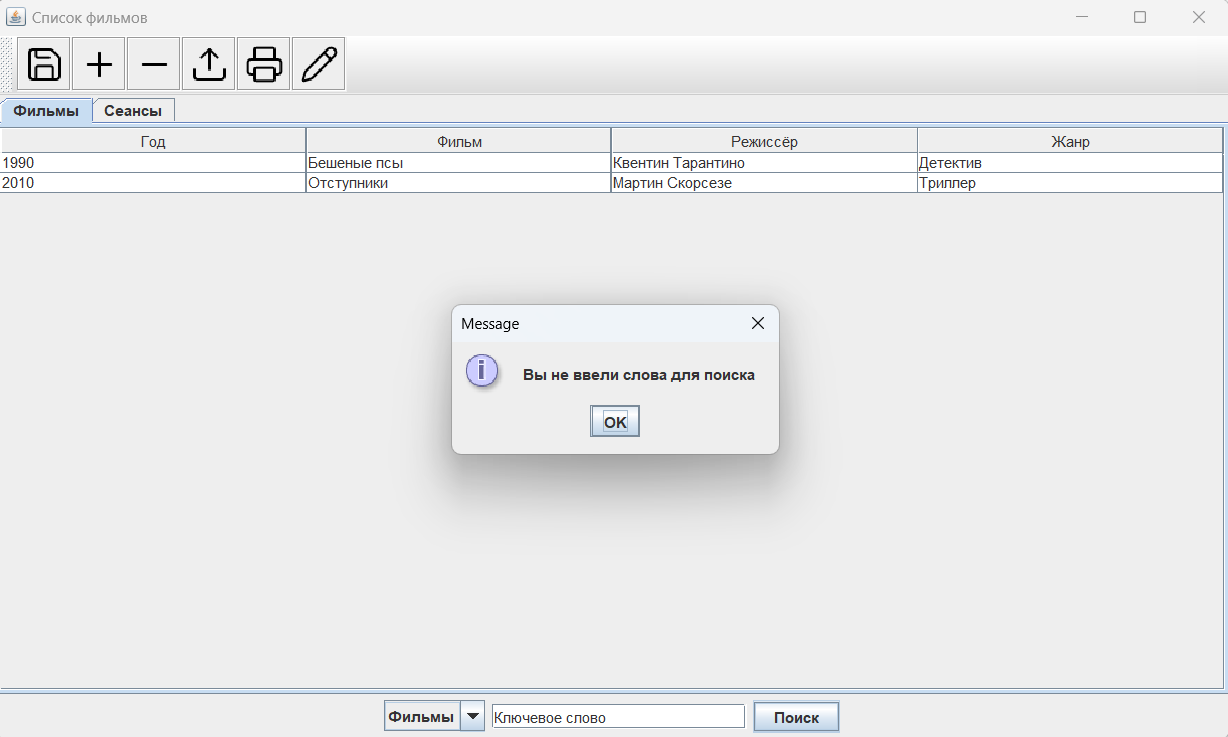
В /src/edu/java/lab04 находится файл CinemaList.java

В /doc находится документация, сгенерированная JavaDoc

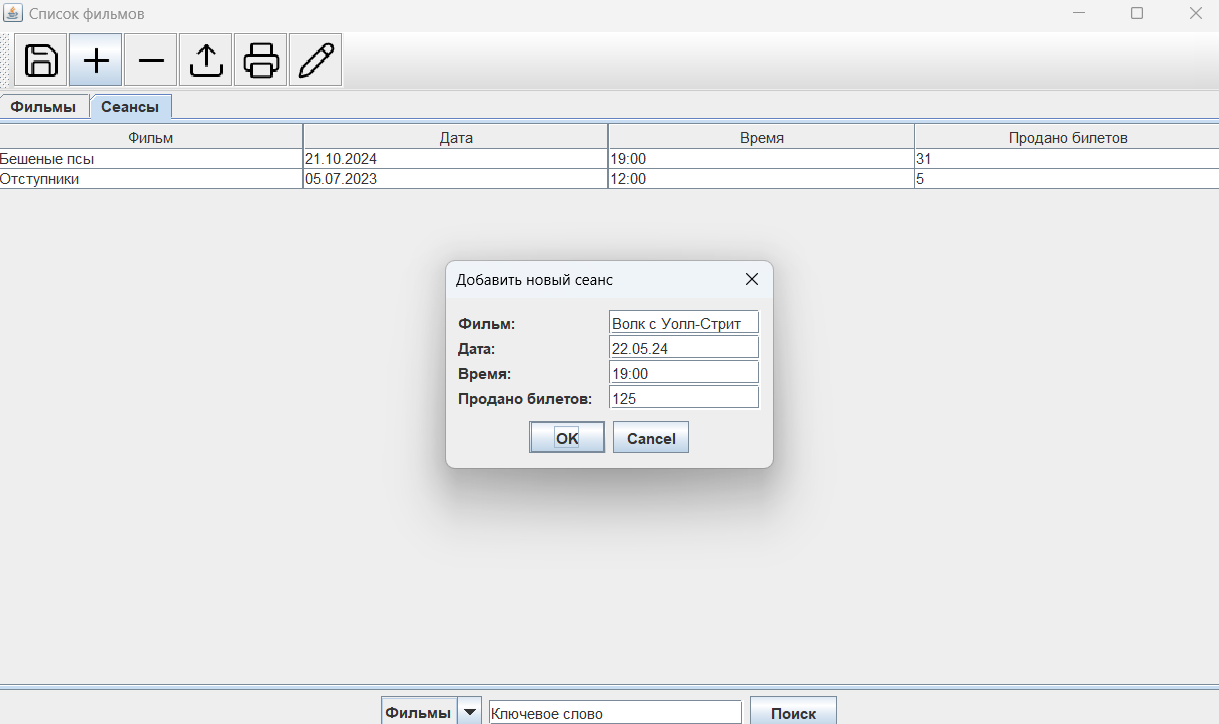
Ссылка на видеоотчёт

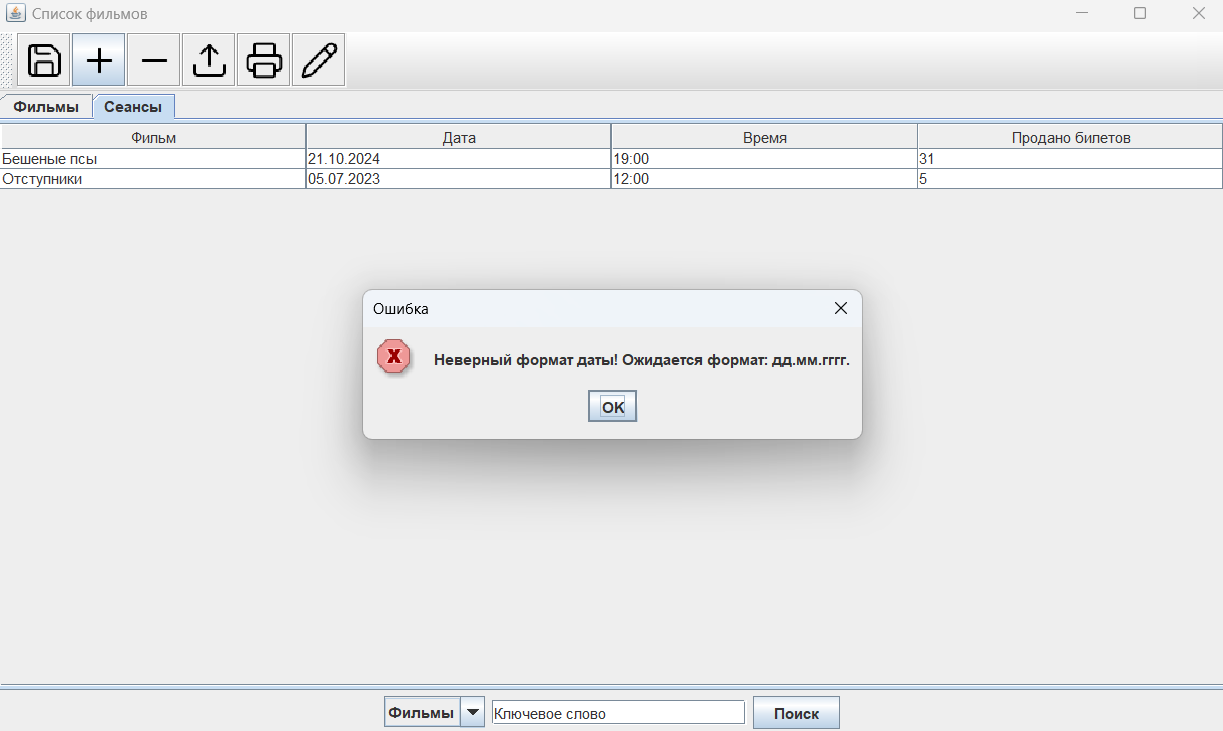
ht

**Скриншоты, иллюстрирующие работу обработчиков ситуаций**

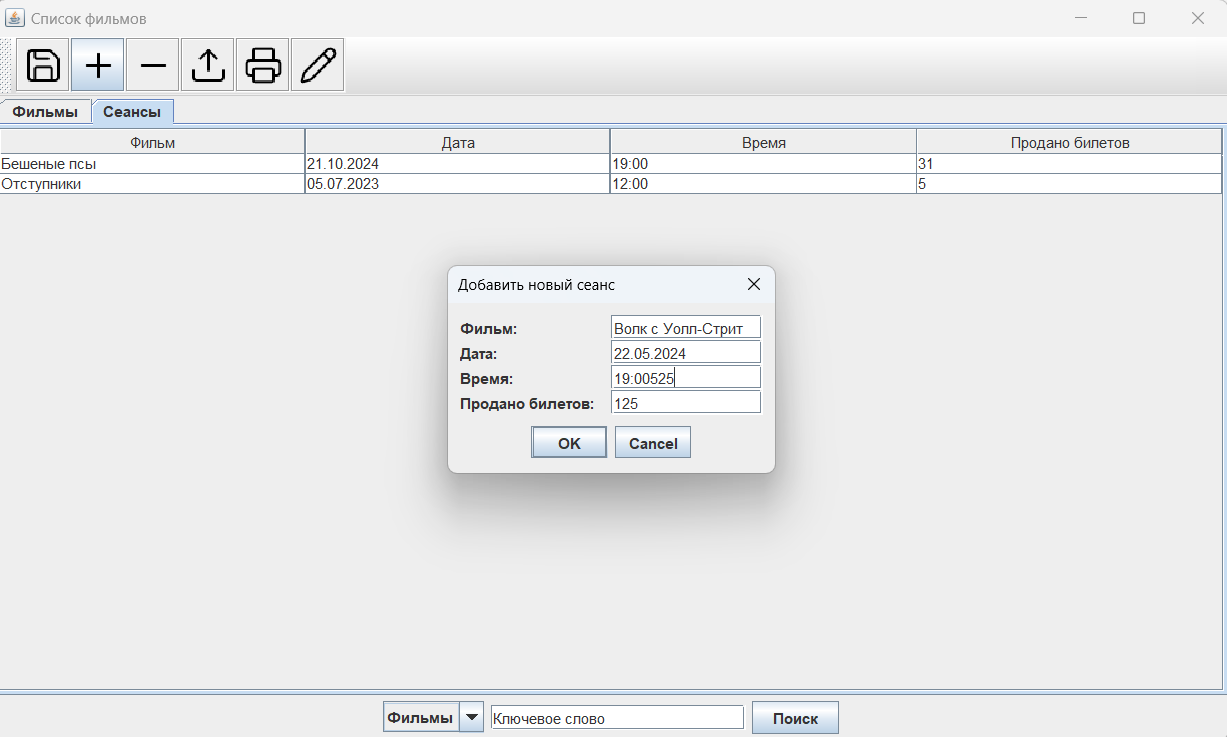


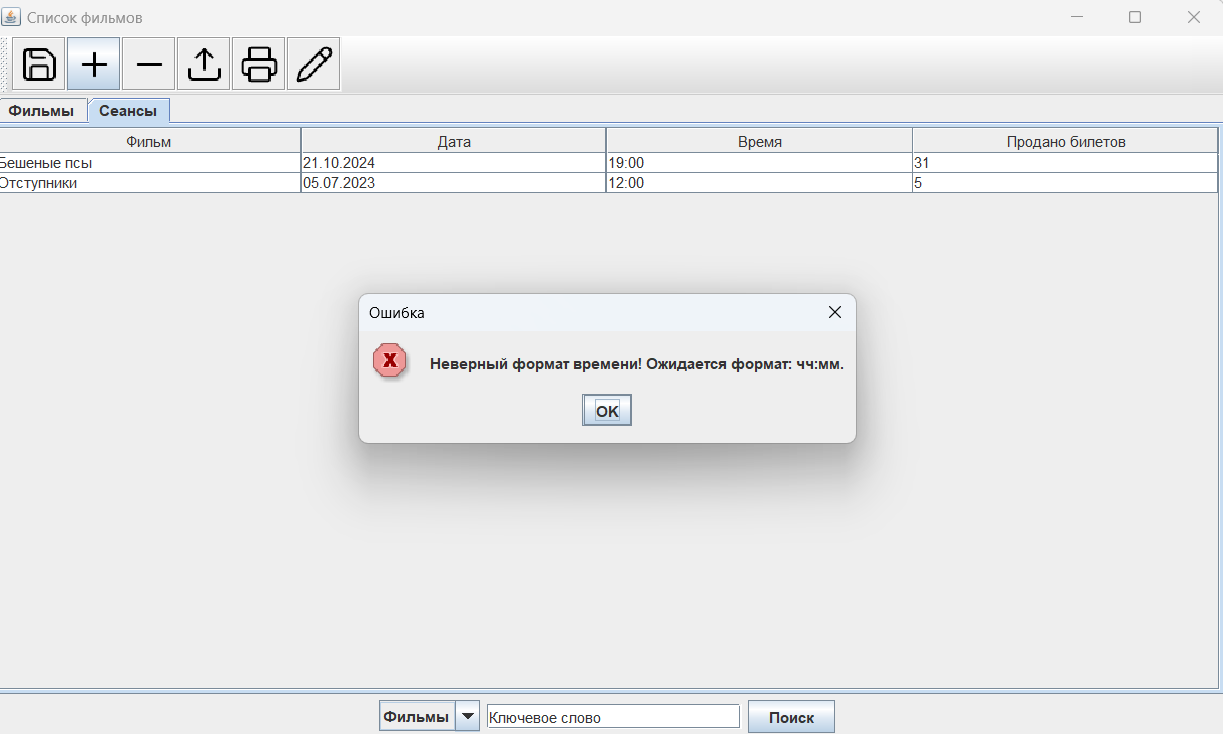
Пример



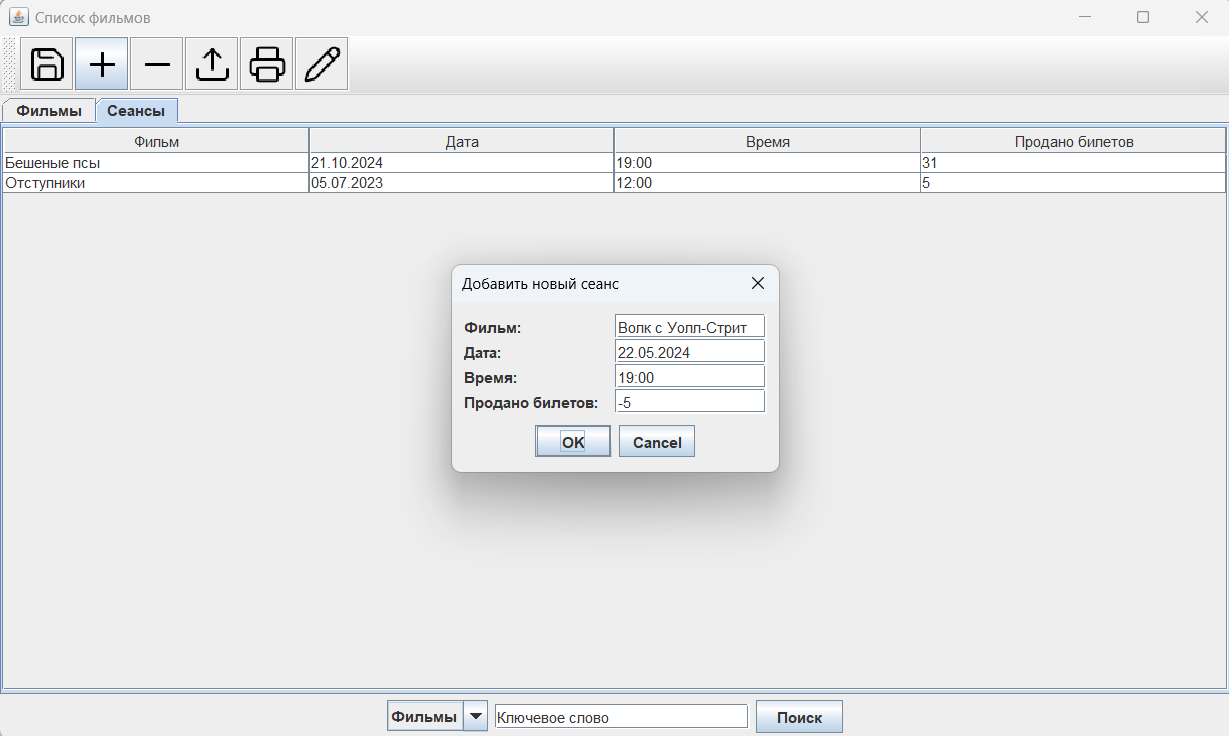


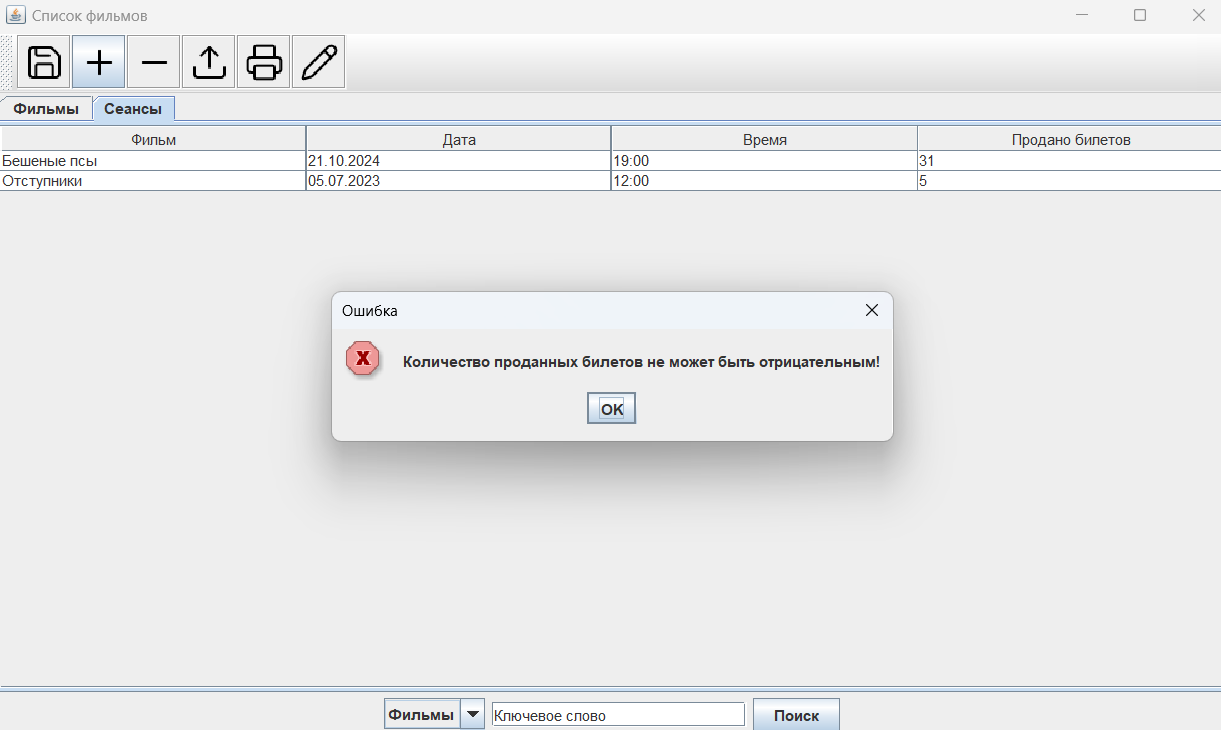
Пример





Пример





Пример

**Исходные тексты классов собственных исключений, методов, где генерируются и обрабатываются исключительные ситуации**

filter.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent arg0) { // Обработка исключения - пустая строка  
 try { checkName(wordToFind);  
 }  
 catch(NullPointerException ex) {  
 JOptionPane.*showMessageDialog*(cinemaList, ex.toString());  
 }  
 catch(InvalidFindException myEx) {  
 JOptionPane.*showMessageDialog*(null, myEx.getMessage());  
 }}});

delete.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 int selectedTab = tabbedPane.getSelectedIndex(); // Определяем, какая вкладка выбрана  
  
 if (selectedTab == 0) { // Если выбрана вкладка "Фильмы"  
 int selectedRow = tableMovies.getSelectedRow();  
 if (selectedRow != -1) { // Проверяем, выбрана ли строка  
 modelMovies.removeRow(selectedRow); // Удаляем выбранную строку из таблицы фильмов  
 } else {  
 JOptionPane.*showMessageDialog*(cinemaList, "Пожалуйста, выберите фильм для удаления.");  
 }  
 } else if (selectedTab == 1) { // Если выбрана вкладка "Сеансы"  
 int selectedRow = tableSessions.getSelectedRow();  
 if (selectedRow != -1) { // Проверяем, выбрана ли строка  
 modelSessions.removeRow(selectedRow); // Удаляем выбранную строку из таблицы сеансов  
 } else {  
 JOptionPane.*showMessageDialog*(cinemaList, "Пожалуйста, выберите сеанс для удаления.");  
 }  
 }  
 }  
});

private class InvalidFindException extends Exception { // Исключение для неверного ввода поиска  
 public InvalidFindException(String message) {  
 super (message);  
 }}  
private void checkName (JTextField bName) throws InvalidFindException,NullPointerException { // Метод проверки поля поиска  
 String sName = bName.getText();  
 if (sName.contains("Ключевое слово")) throw new InvalidFindException("Вы не ввели слова для поиска");  
 if (sName.isEmpty()) throw new NullPointerException();  
}  
  
class InvalidDateException extends Exception { // Исключение для неверного формата даты  
 public InvalidDateException(String message) {  
 super(message);  
 }  
}  
private void checkDate(String date) throws InvalidDateException { // Метод проверки формата даты  
 if (!date.matches("\\d{2}\\.\\d{2}\\.\\d{4}")) {  
 throw new InvalidDateException("Неверный формат даты! Ожидается формат: дд.мм.гггг.");  
 }  
}  
  
class InvalidTimeException extends Exception { // Исключение для неверного формата времени  
 public InvalidTimeException(String message) {  
 super(message);  
 }  
}  
private void checkTime(String time) throws InvalidTimeException { // Метод проверки формата времени  
 if (!time.matches("\\d{2}:\\d{2}")) {  
 throw new InvalidTimeException("Неверный формат времени! Ожидается формат: чч:мм.");  
 }  
}  
  
class InvalidTicketCountException extends Exception { // Исключение для неверного формата количества билетов  
 public InvalidTicketCountException(String message) {  
 super(message);  
 }  
}  
private void checkTicketCount(String ticketsSold) throws InvalidTicketCountException, NumberFormatException { // Метод проверки количества проданных билетов  
 try {  
 int count = Integer.*parseInt*(ticketsSold);  
 if (count < 0) {  
 throw new InvalidTicketCountException("Количество проданных билетов не может быть отрицательным!");  
 }  
 } catch (NumberFormatException ex) {  
 throw new InvalidTicketCountException("Количество проданных билетов должно быть числом!");  
 }  
}