

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ ПО
ДИСЦИПЛИНЕ «ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ
ПРОГРАММИРОВАНИЕ»

Тема: Создание программного комплекса средствами объектно-ориентированного программирования

Студент гр. 1305

Фомин К.С.

Преподаватель

Павловский М.Г.

Санкт-Петербург

2023

Цель работы.

1. Техническое задание

1.1 Введение

Программный комплекс (ПК) администрирования магазина графических процессоров предназначен для использования в составе системы программно-информационного обеспечения учета и администрирования.

1.2 Основание для разработки

Основанием для разработки ПК «Учет прейскуранта товаров и списка работников магазина графических процессоров» является курсовой проект по дисциплине «Объектно-ориентированное программирование».

1.3 Назначение разработки

ПК «Учет прейскуранта товаров и списка работников магазина графических процессоров» должен входить в состав автоматизированной системы учета и администрирования информации, и предназначен для автоматизации деятельности лица (ОЛ), ответственного за учет прейскуранта товаров и списка работников магазина.

ПК «Учет прейскуранта товаров и списка работников магазина графических процессоров» предназначен для автоматизации следующих процессов:

- учет и администрирование информации о наличии, количестве и цене товаров в магазине;
- учет и администрирование информации о работниках магазина;
- получение справочной информации о товарах в магазине;

1.4 Требования к программе

1.4.1 Требования к функциональным характеристикам

1.4.1.1 Перечень функций

ПК «Учет прейскуранта товаров и списка работников магазина графических процессоров» должен обеспечивать выполнение следующих функций:

- просмотр, добавление, удаление и изменение данных в источниках информации;
- выдача справочной информации, хранимой в источниках информации, по запросам ОЛ.

1.4.1.2 Требования к составу выполняемых функций

1.4.1.2.1 Функция «просмотр, добавление, удаление и изменение данных в источниках информации».

Функция должна обеспечивать ведение и хранение следующих данных:

- прейскуранте товаров;
- данные о списке работников.

1.4.1.2 Требования к организации и форме представления выходных данных

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

1.4.1.3 Требования к организации и форме представления входных данных

Входная информация для задачи «Учет прейскуранта товаров и списка работников магазина графических процессоров» содержится в приходно-расходной документации. Ввод исходных данных должен осуществляется ОЛ в режиме диалога. Вводимые данные являются значениями характеристик (атрибутов) информационных объектов.

1.4.2 Требования к надежности

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

1.4.3 Условия эксплуатации

Выполнение ПК «Учет прејскуранта товаров и списка работников магазина графических процессоров» своих функций должно быть обеспечено для однопользовательского режима работы с монопольным доступом к базе данных.

1.4.4 Требование к составу и параметрам технических средств

Задача должна решаться на ПЭВМ типа IBM PC или совместимой с ней с процессором Pentium III 500 и выше, ОЗУ не менее 128Мб, HDD не менее 4 Гб, монитор SVGA (цветной)15", видеокарта 64 Мб, клавиатура 102 кл., манипулятор типа "мышь".

1.4.5 Требование к информационной и программной совместимости

Выходная и входная информация ПК «Учет прејскуранта товаров и списка работников магазина графических процессоров» должна быть удобна для визуального восприятия. ПК должен быть выполнен на языке программирования высокого уровня Java и должен быть совместим с операционной системой Windows.

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

- закрытые и открытые члены классов;
- наследование;
- конструкторы с параметрами;
- абстрактные базовые классы;
- виртуальные функции;
- обработка исключительных ситуаций;
- динамическое создание объектов.

1.5 Требования к программной документации

Программная документация (ПД) должна удовлетворять требованиям стандартов ЕСПД.

Документация должна быть представлена в следующем составе:

1. описание процесса проектирования ПК;
2. руководство оператора;
3. исходные тексты ПК.

1.6 Стадии и этапы разработки

1. Разработка технического задания;
2. Описание вариантов использования ПК;
3. Создание прототипа интерфейса пользователя;
4. Разработка объектной модели ПК;
5. Построение диаграмм программных классов;
6. Описание поведения ПК;
7. Построение диаграмм действий;

1.7 Порядок контроля и приемки

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

2 Проектирование ПК

2.1 Описание вариантов использования ПК

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

Связь коммуникации — это связь между прецедентом и актором. На языке UML связь коммуникации изображают в виде стрелки. Направление стрелки показывает, кто инициирует коммуникацию. При задании коммуникации необходимо указать данные, которые вводит или получает пользователь. Кроме

данных на концах стрелки можно указать кратности отношения, которые характеризуют количество взаимодействующих между собой акторов и прецедентов. На диаграммах прецедентов наиболее распространенными являются две формы записи кратности 1 и 1 .. *. Первая форма записи означает, что один актор (прецедент) участвует во взаимодействии, а вторая форма записи, что один или несколько акторов (прецедентов) участвуют во взаимодействии.

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

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

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

Диаграмма прецедентов представлена на рис. 2.1.

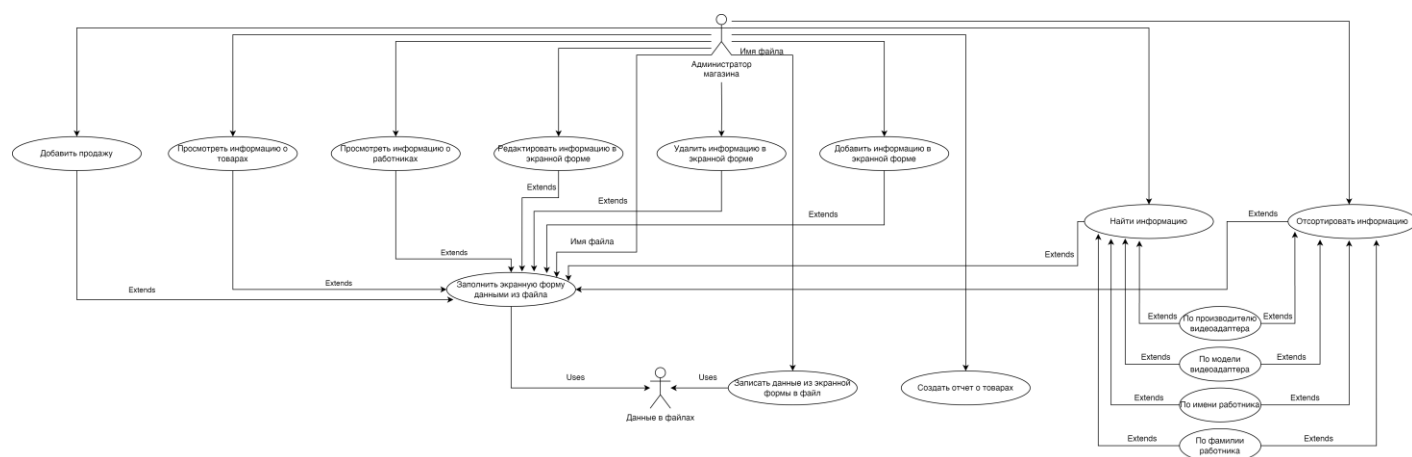


Рис.2.1 Диаграмма прецедентов

[Ссылка на диаграмму в полном размере](#)

2.3 Создание прототипа интерфейса пользователя

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

Рис. 2.2 – главная экранная форма, рис. 2.3.1. – экранная форма добавления продажи, рис. 2.3.2. – экранная форма подтверждения продажи. В табл. 2.1 представлено описание экранных форм.

Таблица 2.1

Экранная форма	Элементы управления	Действия пользователя	Отклик системы
Главная экранная форма	Кнопка «Сохранить»	Нажатие на кнопку «Сохранить»	Открытие диалогового окна выбора места и названия файла для записи в него информации из таблиц.
	Кнопка «Открыть файл»	Нажатие на кнопку «Открыть файл»	Открытие диалогового окна выбора файла для открытия, считывания и заполнения таблиц полученной информацией (рис. 2.3.)
	Кнопка «Создать отчет»	Нажатие на кнопку «Создать отчет»	Создание отчета о преЙскуранте товаров в формате pdf.
	Кнопка «Добавить строку»	Нажатие на кнопку «Добавить строку»	Добавление строки в таблицу, активную в экранной форме (выбор осуществляется на панели выбора

			«Прейскурант, список сотрудников», «Список продаж»)
	Кнопка «Удалить строку»	Нажатие на кнопку «Удалить строку»	Удаляет выбранную строку в таблице, активной в экранной форме
	Кнопка «Режим редактирования»	Нажатие на кнопку «Режим редактирования» (при условии, что режим редактирования выключен)	Включение режима редактирования таблицы. Все поля таблицы становятся доступными к внесению изменений
		Нажатие на кнопку «Режим редактирования» (при условии, что режим редактирования включен)	Выключение режима редактирования
	Выпадающий список «Поиск по»	Нажатие на выпадающий список «Поиск по» и выбор одного из элементов	Установка столбца, в котором будет произведен поиск по ключевому слову из поля ввода «Ключевое слово» (рис. 2.4.)

	Поле ввода «Ключевое слово»	Нажатие на поле ввода «Ключевое слово» и ввод слова	Установка слова, по которому по нажатии на кнопку «Поиск» или кнопку «Enter» на клавиатуре будет произведен поиск в выбранном в списке «Поиск по» столбце.
	Кнопка «Поиск»	Нажатие кнопки «Поиск»	Выполнение поиска по ключевому слову, указанному в поле «Ключевое слово», в столбце, выбранном в списке «Поиск по»
		Нажатие на кнопку «Enter» на клавиатуре	Выполнение поиска по ключевому слову, указанному в поле «Ключевое слово», в столбце, выбранном в списке «Поиск по»
		Нажатие на кнопку «Поиск» или «Enter» на клавиатуре при условии отсутствия ключевого слова в поле	Сброс результатов поиска, приведение таблицы к изначальному виду.

		«Ключевое слово»	
	Панель выбора списка с информацией «Прейскурант, список сотрудников»	Нажатие на одну из кнопок панели с названиями таблиц с информацией	Отображение в экранной форме выбранной таблицы (прейскуранта либо списка работников)
	Таблица с информацией	Нажатие на заголовок колонки	Сортировка полей таблицы по значениями выбранной колонки
		Двойное нажатие на поле таблицы при условии включенного режима редактирования	Ожидание редактирования информации в выбранном поле, сохранение информации в нем (рис 2.6.)
Экранная форма добавления продажи	Поле «ID продавца» Поле «ID проданного видеоадаптера» Кнопка «ОК» Кнопка «Отмена»	Ввод корректных данных и нажатие на кнопку «ОК»	Обработка данных и отображения экранной формы подтверждения покупки.
		Ввод некорректных данных	Отображение сообщения об ошибке с содержательной информацией.

		Нажатие кнопки «Отмена»	Отмена операции
Экранная форма подтверждения продажи	Текстовые поля «Имя продавца», «Название проданного видеоадаптера». Кнопки «ОК», «Отмена»	Нажатие на кнопку «ОК»	Добавление продажи, обновление соответствующей информации в прейскуранте товаров.
		Нажатие на кнопку «Отмена»	Отмена добавления продажи.

Manufacturer	Model	Price, USD	Quantity	ID
Asrock	GeForce RTX 4090	5000	17	0
Gigabyte	GeForce RTX 4090	5000	12	1
Asus ROG	GeForce RTX 4080	4800	11	2
Palit	GeForce RTX 4080	4800	10	3
Palit	Radeon RX 7900 XTX	4600	17	4
XFX	Radeon RX 7900 XTX	4600	9	5
MSI	Radeon RX 7900 XTX	4600	1	6
Palit	GeForce RTX 3090 Ti	4400	6	7
Asrock	GeForce RTX 3090 Ti	4400	8	8
Asrock	GeForce RTX 4070 Ti	4200	15	9
XFX	GeForce RTX 4070 Ti	4200	4	10
MSI	GeForce RTX 4070 Ti	4200	3	11
XFX	GeForce RTX 3080 Ti	3800	11	12
ASUS	GeForce RTX 3080 Ti	3800	8	13
Gigabyte	GeForce RTX 3080 Ti	3800	4	14
EVGA	GeForce RTX 3080	3400	18	15
Asrock	GeForce RTX 3080	3400	8	16
MSI	Radeon RX 7900 XT	3200	13	17
EVGA	Radeon RX 7900 XT	3200	7	18
Palit	Radeon RX 6900 XT	2800	14	19
MSI	Radeon RX 6900 XT	2800	15	20
MSI	GeForce Titan RTX	2600	19	21
EVGA	GeForce Titan RTX	2600	4	22
XFX	GeForce RTX 3070 Ti	2400	15	23
Asrock	GeForce RTX 3070	2200	3	24
EVGA	GeForce RTX 3070	2200	3	25
ASUS	GeForce RTX 3070	2200	18	26
XFX	GeForce RTX 2080 Ti	2000	18	27
Asrock	GeForce RTX 4060 Ti	1800	2	28
Palit	GeForce RTX 4060 Ti	1800	18	29

Рис. 2.2.1. Главная экранная форма

Price list

Search by name Enter information Search

Price list Employee list

Name	Surname	Title	Phone number
Michael	De Santa	CEO	+12025550180
Trevor	Phillips	Co-CEO	+12025550145
Isaac	Bowman	Computer technician	+12021006334
Dominick	Little	Computer technician	+12021015724
Kieran	Lowe	Computer technician	+12021026962
Lewis	Graham	Computer technician	+12021028145
Martin	Allen	Computer technician	+12021009961
Bo	Johnston	Web-programmer	+12021011942
Everest	Payne	Web-programmer	+12021032391
Lionel	Stewart	Web-programmer	+12021000153
Colin	Olson	Web-programmer	+12021017421
Legacy	Gibson	Web-programmer	+12021019895
Santino	Robinson	Salesman	+12021014771
Julio	Stevens	Salesman	+12021019912
Keith	Edwards	Salesman	+12021017035
Jrue	Brown	Salesman	+12021023811
Hector	Austin	Salesman	+12021017673
Ryland	Barnes	Salesman	+12021007711
Finley	Garza	Salesman	+12021025547
Eliam	Ortiz	Salesman	+12021032757
Isaiah	Walters	Salesman	+12021008723
Kaiser	Jacobs	Salesman	+12021000778
Ari	Dean	Salesman	+12021022190
Ira	Barnett	Salesman	+12021030106
Charles	Elliott	Salesman	+12021019264
Chris	Mason	Salesman	+12021023805
Yousef	Hughes	Salesman	+12021024370
Mario	Delgado	Salesman	+12021031101
Edgar	Parker	Salesman	+12021019629
Gustavo	Pena	Salesman	+12021019954

Рис. 2.2.2 Главная экранная форма (выбран список работников)

Price list

Search by Seller's ID Enter information Search

Price list Employee list Sales list

Seller's name	Seller's ID	Sold item	Sold item ID	Date
Martin Allen	6	XFX GeForce RTX 3080 Ti	12	01.05.2002
Bo Johnston	7	Asus ROG GeForce Titan V	32	02.05.2002
Jrue Brown	15	XFX GeForce RTX 3070 Ti	23	05.05.2002

Рис. 2.2.3 Главная экранная форма (выбран список продаж)

Please, enter sale data

? Seller's id: Sold item id: Sale date:

OK Cancel

Рис. 2.3.1. Экранная форма добавления продажи

Please, confirm the data.

? Seller's name:
Bo Johnston

Sold item name:
Palit Radeon RX 7900 XTX

OK Cancel

Рис. 2.3.2. Экранная форма подтверждения продажи

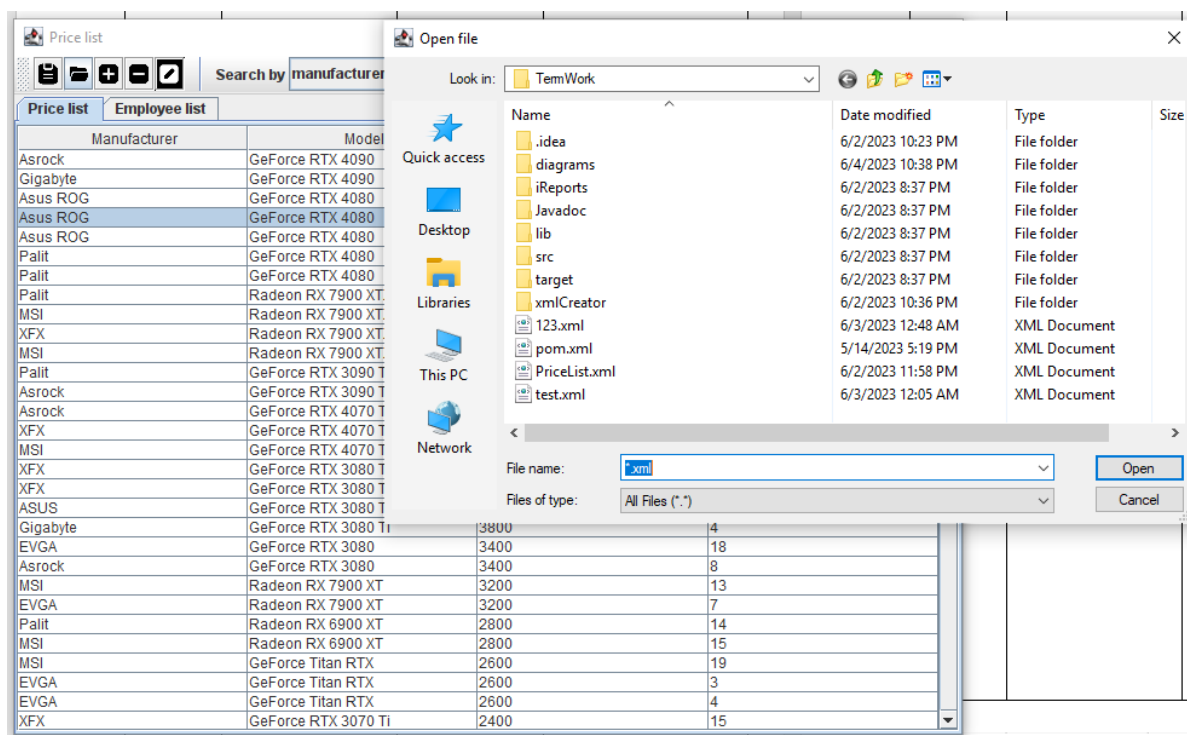


Рис. 2.4. Нажатие на кнопку «Открыть файл»

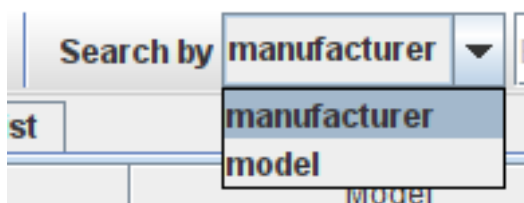


Рис. 2.5. Выпадающий список «Поиск по»

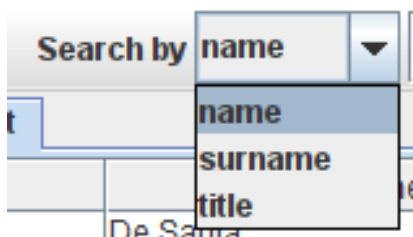


Рис. 2.6. Выпадающий список «Поиск по» (выбран список работников)

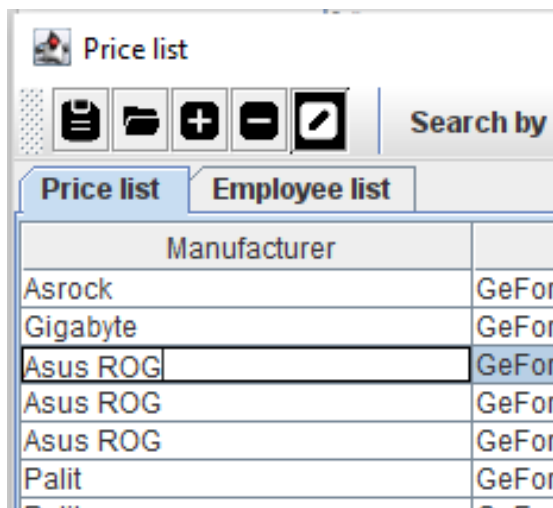


Рис. 2.7. Включение режима редактирования и двойное нажатие по полю таблицы дает возможность изменять информацию

2.3. Разработка объектной модели ПК

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

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

Описание операций помогает определить поведение объектов сущности. На этом этапе, прежде всего, определяется внутреннее поведение каждого объекта сущности, без учета взаимодействия с другими объектами предметной

области. На диаграмме обычно указывается только имя операции, а ее подробное описание приводится в отдельной таблице. В таблице должно содержаться краткое описание назначения операции, ее имя и список входных и выходных параметров.

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

0 ..* - нуль или больше,

1 .. * - один или больше,

1 – ровно один.

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

Диаграмма сущностей представлена на рис. 2.7. Детальное описание операций представлено в табл. 2.2.

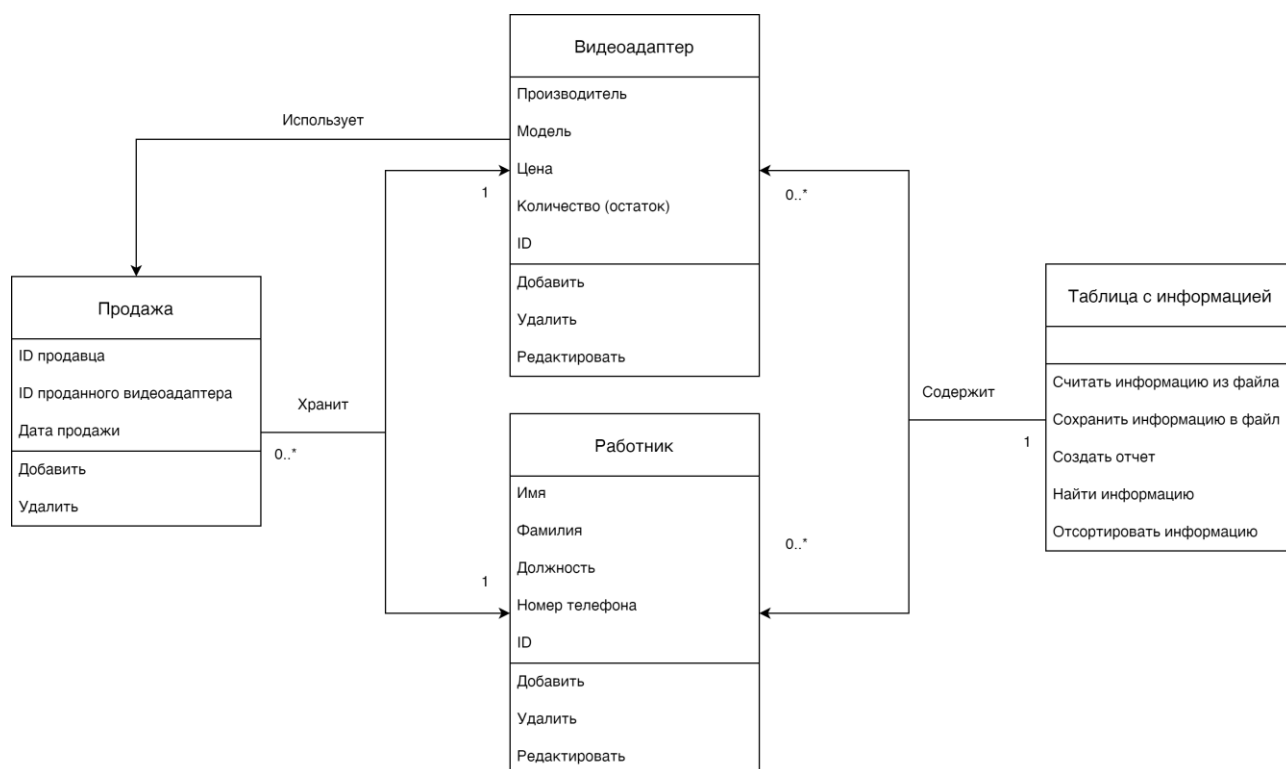


Рис. 2.7. Диаграмма сущностей

Таблица 2.2

Сущность	Имя операции	Параметры операции			Тип возвращаемого значения	Назначение операции
		Вид	Название	Тип		
Видеоадаптер	Добавить	Вх.	Производитель	Строка	Пусто	Создает пустую строку в таблице
		Вх.	Модель	Строка	Пусто	
		Вх.	Цена	Строка	Пусто	
		Вх.	Количество	Строка	Пусто	
	Удалить	Вх.	Номер строки	Число	Пусто	Удаляет выбранную строку из таблицы
	Редактировать	Вх.	Номер строки	Число	Пусто	Редактирует информацию в

		Вх.	Номер столбца	Число	Пусто	выбранной строке и столбце
		Вх.	Новая информация	Строка	Пусто	
Работник	Добавить	Вх.	Производитель	Строка	Пусто	Создает пустую строку в таблице
		Вх.	Модель	Строка	Пусто	
		Вх.	Цена	Строка	Пусто	
		Вх.	Количество	Строка	Пусто	
	Удалить	Вх.	Номер строки	Число	Пусто	Удаляет выбранную строку из таблицы
	Редактировать	Вх.	Номер строки	Число	Пусто	Редактирует информацию в выбранной строке и столбце
Продажа	Добавить	Вх.	ID продавца	Строка	Пусто	Добавляет продажу в список продаж.
		Вх.	ID проданного видеоадаптера	Строка	Пусто	
		Вх.	Дата продажи	Строка	Пусто	
	Удалить	Вх.	Номер	Число	Пусто	Удаляет выбранную продажу.
Таблица с информацией	Сохранить информацию в файл	Вх.	Цена	Строка	Пусто	Создает файл и сохраняет в него

						информацию из экранной формы
	Создать отчет	Вх.	Количество	Строка	Пусто	Создает отчет о прейскуранте товаров в формате pdf
	Удалить Редактиро -вать	Вх.	Номер строки	Число	Пусто	Удаляет выбранную строку из таблицы Редактирует информацию в выбранной строке и столбце
	Удалить Редактиро -вать	Вх.	Номер строки	Число	Пусто	Удаляет выбранную строку из таблицы
	Отсортиро вать информац ию	Вх.	Критерий сортировки	Строка	Пусто	Редактирует информацию в выбранной строке и столбце Сортирует элементы в экранной форме в соответствии с заданными параметрами

2.4 Построение диаграммы программных классов

Диаграмма классов (class diagram) иллюстрирует спецификации будущих программных классов и интерфейсов. Она строится на основе объектной модели. В описание класса указываются три раздела: имя класса, состав компонентов класса и методы класса. Графически класс изображается в виде прямоугольника.

Имя программного класса может совпадать с именем сущности или быть другим. Но поскольку для записи идентификаторов переменных в языках программирования используют латинские буквы, то и имена программных классов и имена их атрибутов, как правило, записываются латинскими буквами. Атрибуты и операции класса перечисляются в горизонтальных отделениях этого прямоугольника. Атрибутам и методам классов должны быть присвоены права доступа. Права доступа помечаются специальными знаками:

- + - означает открытый (public) доступ;
- - означает скрытый (private) доступ;
- # - означает наследуемый (protected) доступ.

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

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

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

Агрегирование - это отношение между классами типа целое/часть. Агрегируемый класс в той или иной форме является частью агрегата. На практике это может быть реализовано по-разному. Например, объект класса-агрегата может хранить объект агрегируемого класса, или хранить ссылку на него. Агрегирование изображается на диаграмме полым ромбом на конце линии со стороны агрегирующего класса (агрегата). Если агрегируемый объект может

быть создан только тогда, когда создан агрегат, а с уничтожением агрегата уничтожаются и все агрегируемые объекты, то такое агрегирование называется сильным и отображается в виде закрашенного ромба.

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

Диаграмма классов представлена на рис. 2.8.

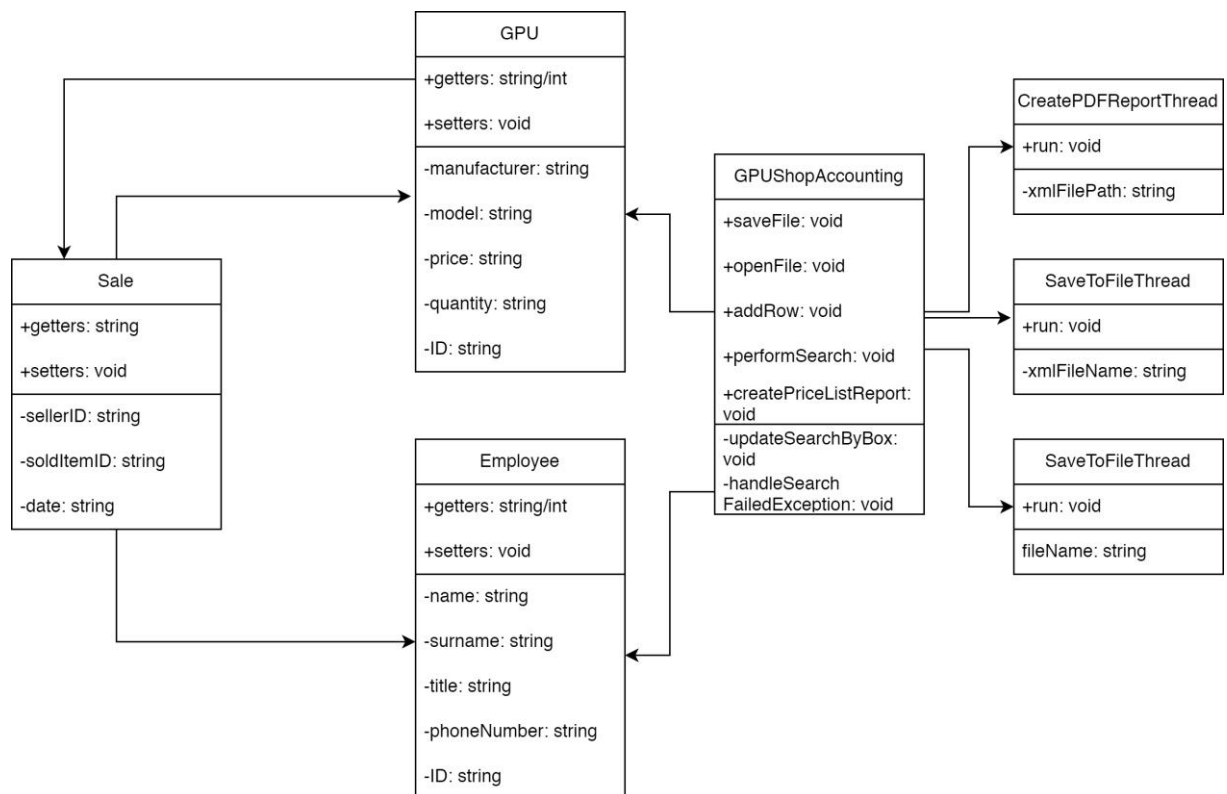


Рис.2.8. Диаграмма классов

Пояснение: данные о всех видеоадаптерах хранятся в списке объектов класса GPU, аналогично данные о работниках хранятся в списке объектов класса Employee. При добавлении объекта класса Sale, пользователь вводит только ID продавца и ID проданного видеоадаптера, таким образом, класс Sale

обращается к спискам объектов *GPU* и *Employee* в целях получения дополнительной информации (данные работника, данные видеоадаптера). С другой стороны, после создания объекта класса *Sale*, объекты класса *GPU* обращаются к созданному объекту класса *Sale*, чтобы обновить данные о себе (добавилась продажа видеоадаптера – количество проданной единицы уменьшилось).

В XML файле под отдельными тегами хранятся продавцы, видеоадаптеры и продажи. Тег `<sale>` имеет атрибуты `sellerID` и `soldItemID`.

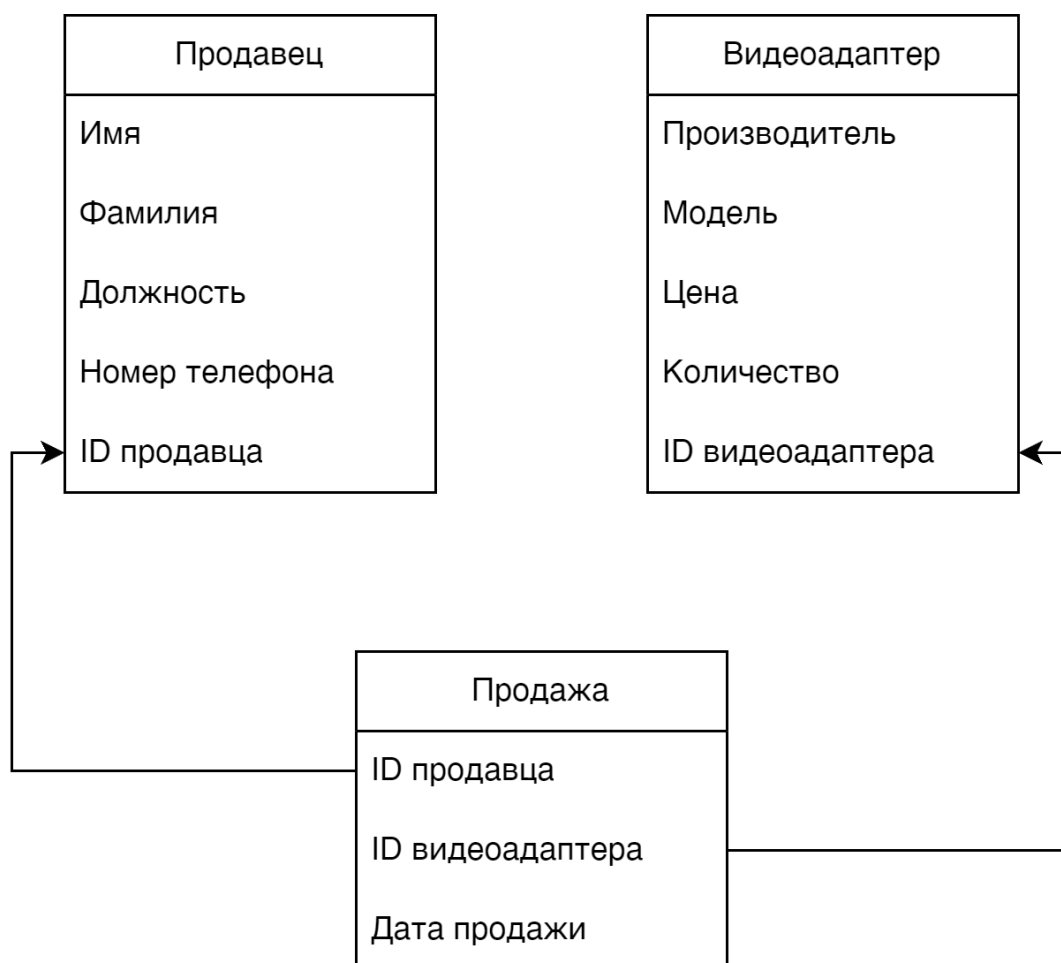


Рис.2.9.1 Класс «продажа» хранит в себе ID продавца и ID видеоадаптера

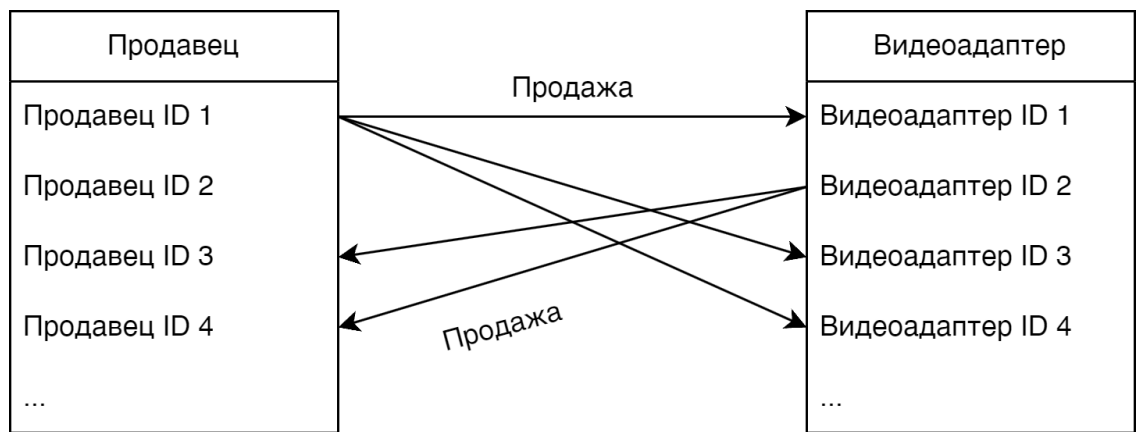


Рис.2.9.2 Путем обращения к продажам, можно найти связи между продавцом и проданными им единицами товара и наоборот.

2.5 Описание поведения ПК

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

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

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

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

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

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

Диаграмма последовательностей для операции создания отчета о
прейскуранте товаров представлена на рис. 2.9.

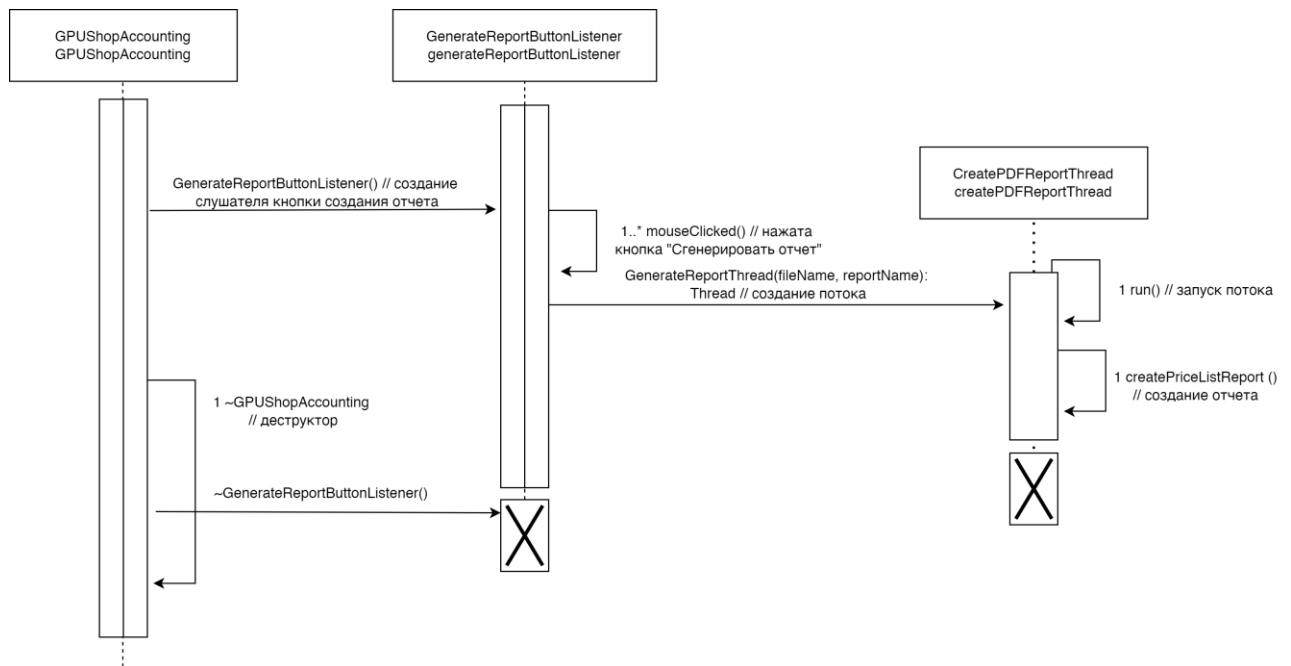


Рис.2.9 Диаграмма последовательностей для операции создания отчета о
прейскуранте товаров

2.6 Построение диаграммы действий

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

потоков управления. При этом разделение имеет один входящий переход и несколько выходящих, а слияние, наоборот, имеет несколько входящих переходов и один выходящий.

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

Диаграмма действий для операции создания отчета представлена на рис. 2.10.

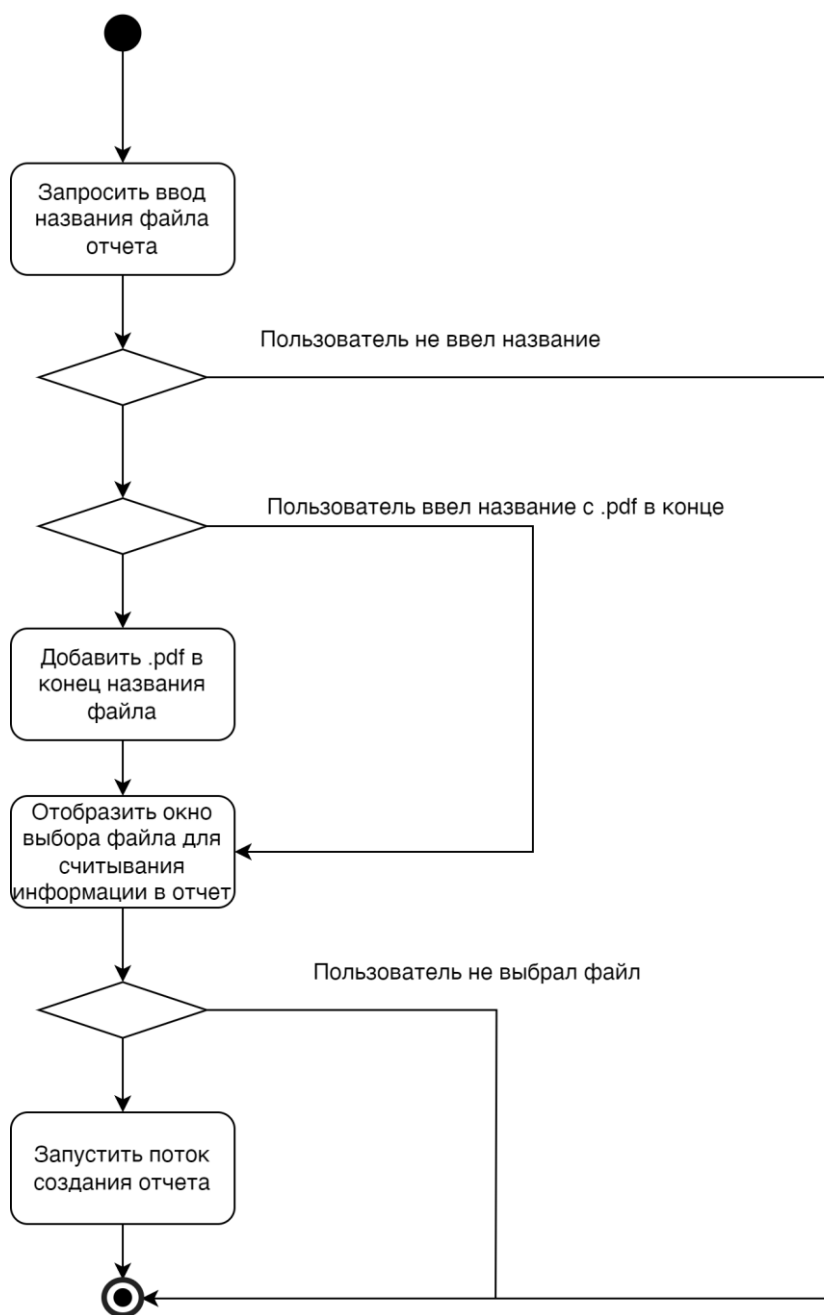


Рис. 2.10 Диаграмма действий для операции создания отчета

3. Руководство оператора

3.1 Назначение программы

ПК «Учет прејскуранта товаров и списка работников магазина графических процессоров» должен входить в состав автоматизированной системы учета и администрирования информации, и предназначен для автоматизации деятельности лица (ОЛ), ответственного за учет прејскуранта товаров и списка работников магазина.

ПК «Учет прейскуранта товаров и списка работников магазина графических процессоров» предназначен для автоматизации следующих процессов:

- учет и администрирование информации о наличии, количестве и цене товаров в магазине;
- учет и администрирование информации о работниках магазина;
- получение справочной информации о товарах в магазине;

3.2 Условия выполнения программы

ПК предназначен для функционирования под операционной средой Windows (XP, 2х).

Персональная электронно-вычислительная машина (ПЭВМ) должна обладать следующими характеристиками:

1. тип процессора Pentium III 500 и выше;
2. объем ОЗУ – не менее 128Мб;
3. объем жесткого диска – не менее 4Гб;
4. видеокарта – 64Мб;
5. стандартная клавиатура;
6. манипулятор типа "мышь".

3.3 Описание задачи

ПК «Учет прейскуранта товаров и списка работников магазина графических процессоров» должен обеспечивать выполнение следующих функций:

- просмотр, добавление, удаление и изменение данных в источниках информации;
- выдача справочной информации, хранимой в источниках информации, по запросам ОЛ.

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

- закрытые и открытые члены классов;
- наследование;

- конструкторы с параметрами;
- абстрактные базовые классы;
- виртуальные функции;
- обработка исключительных ситуаций;
- динамическое создание объектов.

С целью выполнения поставленной задачи в процессе проектирования разработана общая модель ПК с выявлением основных объектов и связей между ними. На основании полученной модели разработаны программные классы. Также в процессе проектирования принято решение о создании двух взаимосвязанных информационных объектов, первый из которых предназначен для хранения информации о книгах, второй – для хранения информации о выданных книгах. Данные об информационных объектах хранятся в базе данных.

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

3.4 Входные и выходные данные

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

Входная информация для задачи «Учет Регистрация, учет, администрирование, редактирования и выдача сведений о книгах» содержится в приходно-расходной документации. Ввод исходных данных должен осуществляется ОЛ в режиме диалога. Вводимые данные являются значениями характеристик (атрибутов) информационных объектов. Вводимая информация может выбираться или набираться из списка предлагаемых значений.

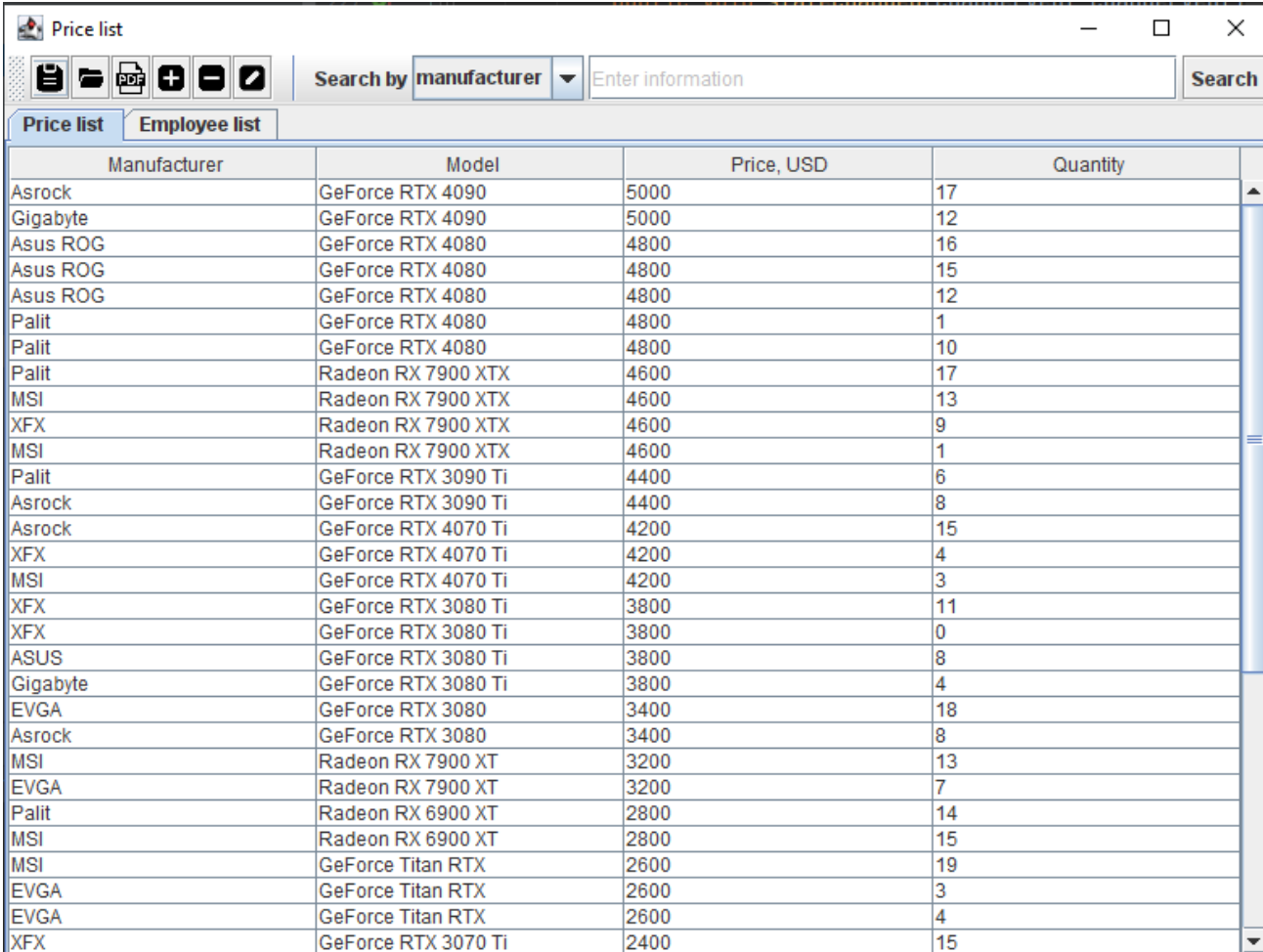
3.5 Выполнение программы

3.5.1 Подготовка к запуску не требуется.

3.5.2. Запуск программы

При запуске программы на экране появится экранная представленное на рис. 3.1.1. (в случае, если в корневой папке уже существует файл PriceList.xml:

он указан файлом хранения данных по умолчанию) или та же экранная форма, но не заполненная данными (если файла PriceList.xml не существует) - рис.3.1.2.



Manufacturer	Model	Price, USD	Quantity
Asrock	GeForce RTX 4090	5000	17
Gigabyte	GeForce RTX 4090	5000	12
Asus ROG	GeForce RTX 4080	4800	16
Asus ROG	GeForce RTX 4080	4800	15
Asus ROG	GeForce RTX 4080	4800	12
Palit	GeForce RTX 4080	4800	1
Palit	GeForce RTX 4080	4800	10
Palit	Radeon RX 7900 XTX	4600	17
MSI	Radeon RX 7900 XTX	4600	13
XFX	Radeon RX 7900 XTX	4600	9
MSI	Radeon RX 7900 XTX	4600	1
Palit	GeForce RTX 3090 Ti	4400	6
Asrock	GeForce RTX 3090 Ti	4400	8
Asrock	GeForce RTX 4070 Ti	4200	15
XFX	GeForce RTX 4070 Ti	4200	4
MSI	GeForce RTX 4070 Ti	4200	3
XFX	GeForce RTX 3080 Ti	3800	11
XFX	GeForce RTX 3080 Ti	3800	0
ASUS	GeForce RTX 3080 Ti	3800	8
Gigabyte	GeForce RTX 3080 Ti	3800	4
EVGA	GeForce RTX 3080	3400	18
Asrock	GeForce RTX 3080	3400	8
MSI	Radeon RX 7900 XT	3200	13
EVGA	Radeon RX 7900 XT	3200	7
Palit	Radeon RX 6900 XT	2800	14
MSI	Radeon RX 6900 XT	2800	15
MSI	GeForce Titan RTX	2600	19
EVGA	GeForce Titan RTX	2600	3
EVGA	GeForce Titan RTX	2600	4
XFX	GeForce RTX 3070 Ti	2400	15

Рис. 3.1.1. Главная экранная форма, заполненная данными

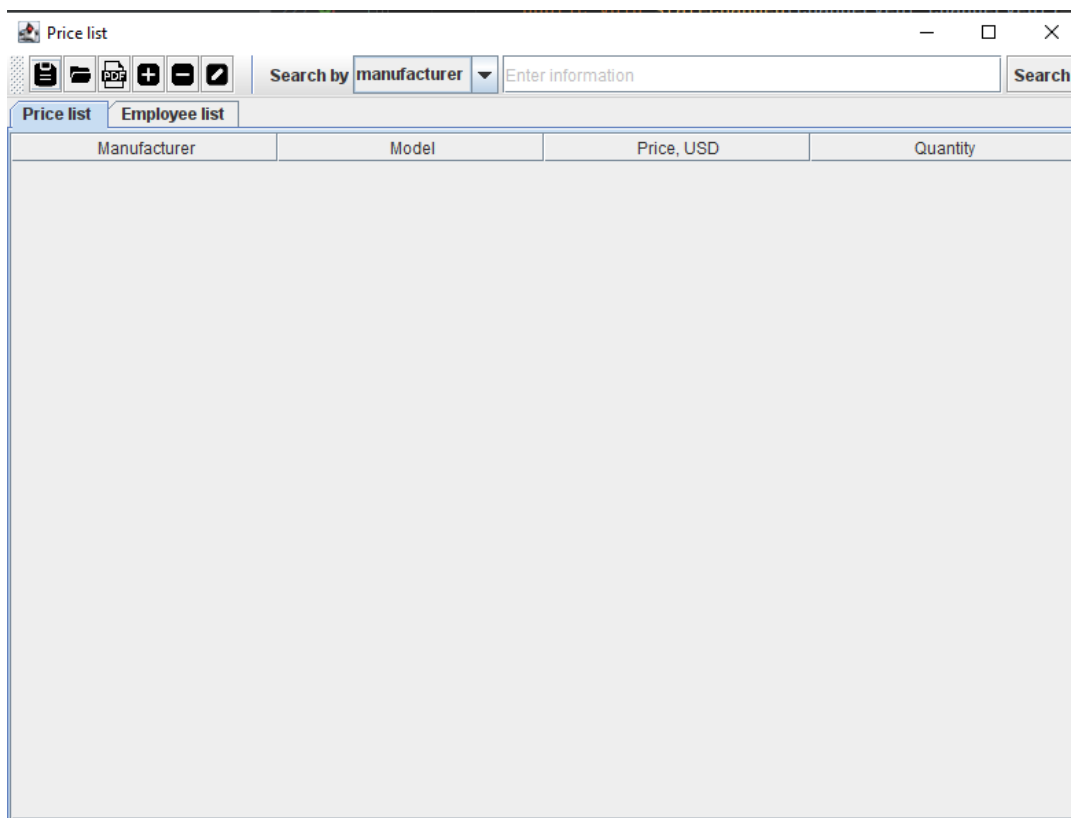


Рис. 3.1.2. Главная экранная форма, не заполненная данными

3.5.3. Выполнение основных функций

3.5.3.1 Открытие файла с данными

В случае отсутствия данных в таблице, но наличия файла с ними (например, если файл с ними находится не в корневой папке) можно открыть файл вручную: для этого необходимо нажать на кнопку «Открыть файл» (вторая на панели управления) и выбрать необходимый файл – рис 3.2.

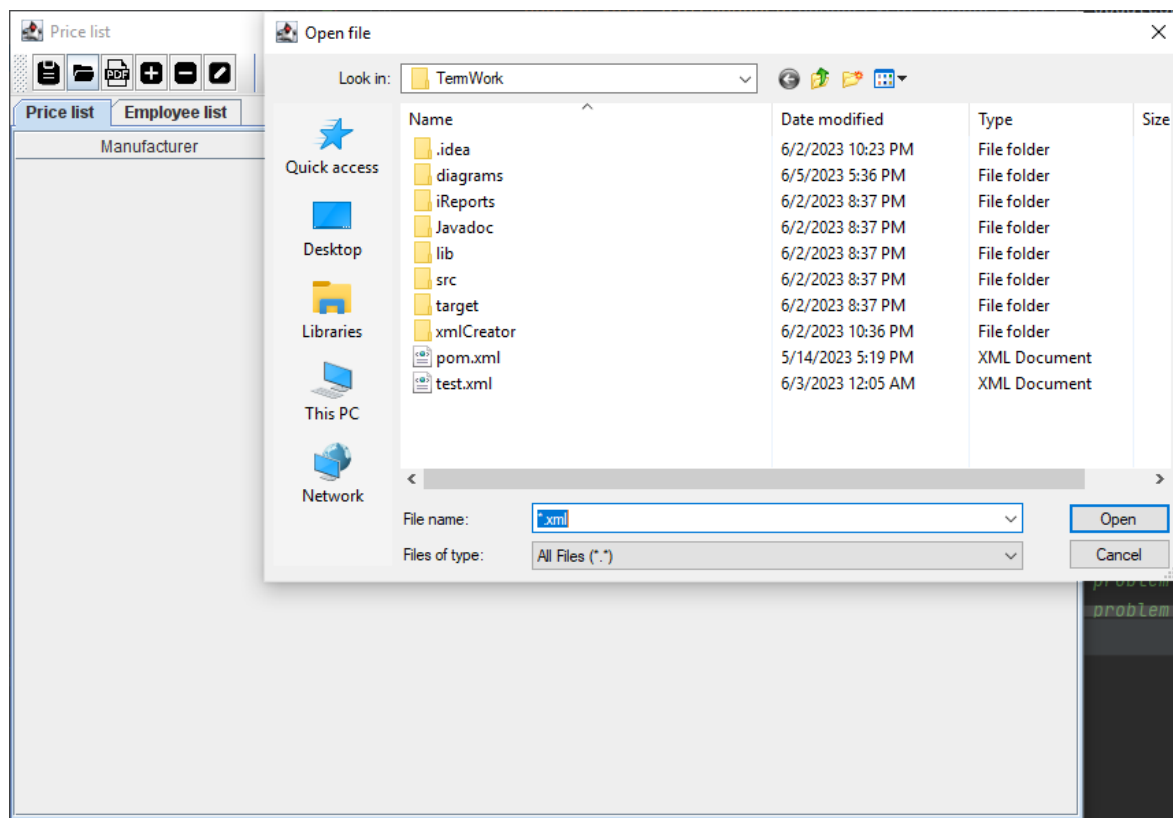


Рис. 3.2 Диалоговое окно открытия файла

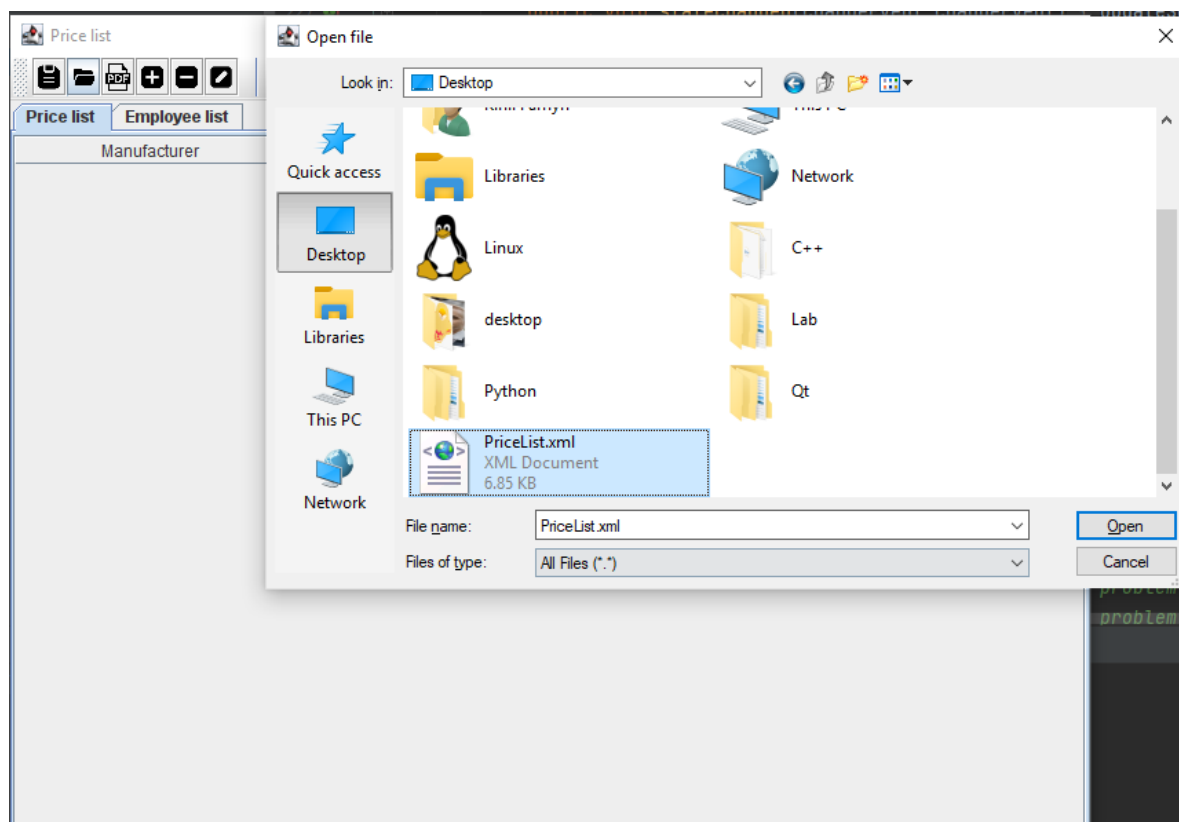


Рис. 3.3 Выбор файла с данными

The screenshot shows a window titled "Price list" with a search bar and two tabs: "Price list" (active) and "Employee list". The table contains the following data:

Manufacturer	Model	Price, USD	Quantity
Asrock	GeForce RTX 4090	5000	17
Gigabyte	GeForce RTX 4090	5000	12
Asus ROG	GeForce RTX 4080	4800	16
Asus ROG	GeForce RTX 4080	4800	15
Asus ROG	GeForce RTX 4080	4800	12
Palit	GeForce RTX 4080	4800	1
Palit	GeForce RTX 4080	4800	10
Palit	Radeon RX 7900 XTX	4600	17
MSI	Radeon RX 7900 XTX	4600	13
XFX	Radeon RX 7900 XTX	4600	9
MSI	Radeon RX 7900 XTX	4600	1
Palit	GeForce RTX 3090 Ti	4400	6
Asrock	GeForce RTX 3090 Ti	4400	8
Asrock	GeForce RTX 4070 Ti	4200	15
XFX	GeForce RTX 4070 Ti	4200	4
MSI	GeForce RTX 4070 Ti	4200	3
XFX	GeForce RTX 3080 Ti	3800	11
XFX	GeForce RTX 3080 Ti	3800	0
ASUS	GeForce RTX 3080 Ti	3800	8
Gigabyte	GeForce RTX 3080 Ti	3800	4
EVGA	GeForce RTX 3080	3400	18
Asrock	GeForce RTX 3080	3400	8
MSI	Radeon RX 7900 XT	3200	13
EVGA	Radeon RX 7900 XT	3200	7
Palit	Radeon RX 6900 XT	2800	14
MSI	Radeon RX 6900 XT	2800	15
MSI	GeForce Titan RTX	2600	19
EVGA	GeForce Titan RTX	2600	3
EVGA	GeForce Titan RTX	2600	4
XFX	GeForce RTX 3070 Ti	2400	15

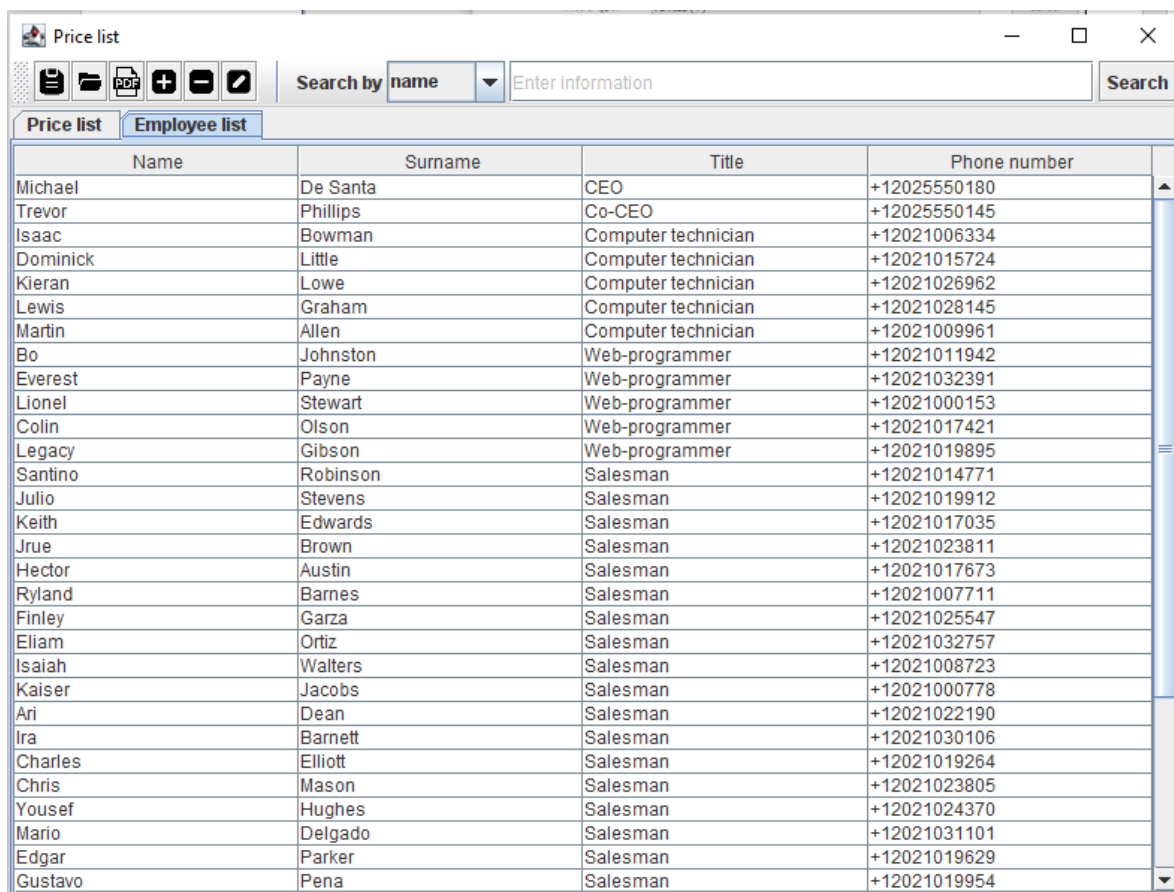
Рис. 3.4 Результат – таблица, заполненная данными

3.5.3.2 Работа с информацией

Когда данные загружены в таблицу (или при желании создать новый файл с данными) можно начать работу с ними.

3.5.3.2.1 Выбор данных для работы

Над таблицей с данными находятся вкладки выбора информации: прейскурант товаров или список работников. Путем нажатия на одну из вкладок можно переключаться между вкладками со списками – рис.3.5.



Name	Surname	Title	Phone number
Michael	De Santa	CEO	+12025550180
Trevor	Phillips	Co-CEO	+12025550145
Isaac	Bowman	Computer technician	+12021006334
Dominick	Little	Computer technician	+12021015724
Kieran	Lowe	Computer technician	+12021026962
Lewis	Graham	Computer technician	+12021028145
Martin	Allen	Computer technician	+12021009961
Bo	Johnston	Web-programmer	+12021011942
Everest	Payne	Web-programmer	+12021032391
Lionel	Stewart	Web-programmer	+12021000153
Colin	Olson	Web-programmer	+12021017421
Legacy	Gibson	Web-programmer	+12021019895
Santino	Robinson	Salesman	+12021014771
Julio	Stevens	Salesman	+12021019912
Keith	Edwards	Salesman	+12021017035
Jrue	Brown	Salesman	+12021023811
Hector	Austin	Salesman	+12021017673
Ryland	Barnes	Salesman	+12021007711
Finley	Garza	Salesman	+12021025547
Eliam	Ortiz	Salesman	+12021032757
Isaiah	Walters	Salesman	+12021008723
Kaiser	Jacobs	Salesman	+12021000778
Ari	Dean	Salesman	+12021022190
Ira	Barnett	Salesman	+12021030106
Charles	Elliott	Salesman	+12021019264
Chris	Mason	Salesman	+12021023805
Yousef	Hughes	Salesman	+12021024370
Mario	Delgado	Salesman	+12021031101
Edgar	Parker	Salesman	+12021019629
Gustavo	Pena	Salesman	+12021019954

Рис.3.5. – выбрана вкладка «Список работников»

3.5.3.2.2 Отображение данных

3.5.4.2.2.1 Сортировка данных

Данные в таблицах можно сортировать в лексико-графическом порядке. Для этого необходимо нажать на название одного из столбцов: по информации из данного столбца произойдет сортировка – рис. 3.6.1.

Manufacturer	Model	Price, USD ▲	Quantity
Gigabyte	Radeon RX 6800 XT	1400	2
ZOTAC	Radeon RX 6800 XT	1400	19
Asrock	GeForce Titan V	1600	8
ASUS	GeForce Titan V	1600	2
Asus ROG	GeForce Titan V	1600	15
MSI	GeForce Titan V	1600	2
MSI	GeForce Titan V	1600	13
ASUS	GeForce Titan V	1600	4
Asrock	GeForce RTX 4060 Ti	1800	2
Palit	GeForce RTX 4060 Ti	1800	18
EVGA	GeForce RTX 4060 Ti	1800	17
XFX	GeForce RTX 2080 Ti	2000	18
XFX	GeForce RTX 2080 Ti	2000	18
Asrock	GeForce RTX 3070	2200	3
EVGA	GeForce RTX 3070	2200	3
ASUS	GeForce RTX 3070	2200	18
XFX	GeForce RTX 3070 Ti	2400	15
MSI	GeForce Titan RTX	2600	19
EVGA	GeForce Titan RTX	2600	3
EVGA	GeForce Titan RTX	2600	4
Palit	Radeon RX 6900 XT	2800	14
MSI	Radeon RX 6900 XT	2800	15
MSI	Radeon RX 7900 XT	3200	13
EVGA	Radeon RX 7900 XT	3200	7
EVGA	GeForce RTX 3080	3400	18
Asrock	GeForce RTX 3080	3400	8
XFX	GeForce RTX 3080 Ti	3800	11
XFX	GeForce RTX 3080 Ti	3800	0
ASUS	GeForce RTX 3080 Ti	3800	8
Gigabyte	GeForce RTX 3080 Ti	3800	4

Рис. 3.6.1. – данные отсортированы по возрастанию цены

3.5.4.2.2 Поиск данных

Используя панель поиска (в правом верхнем углу программы) можно производить поиск по таблице. Поиск осуществляется по следующему алгоритму:

1. Выбрать критерий поиска в выпадающем списке «Поиск по» (рис. 3.6.2.1., рис. 3.6.2.2.)
2. Ввести ключевое слово в поле «Ключевое слово» (справа от списка «Поиск по»), по которому будет произведен поиск по выбранному в п.1 критерию (рис. 3.6.2.3.).
3. Нажать кнопку «Поиск» (справа от поля «Ключевое слово») или клавишу «Enter» на клавиатуре.

Результаты поиска отобразятся в исходной экранной форме (рис. 3.6.2.4.)

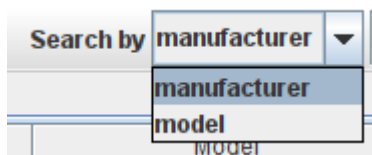


рис. 3.6.2.1. Выпадающий список «Поиск по» для таблицы с прейскурантом товаров

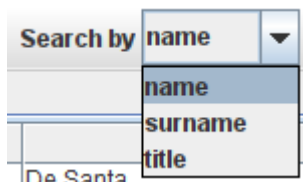


рис. 3.6.2.2. Выпадающий список «Поиск по» для таблицы со списком работников



рис. 3.6.2.3. Ввод ключевого слова в поле «Ключевое слово»

Price list

Search by manufacturer msi Search

Price list Employee list


Manufacturer	Model	Price, USD ▲	Quantity
MSI	GeForce Titan V	1600	2
MSI	GeForce Titan V	1600	13
MSI	GeForce Titan RTX	2600	19
MSI	Radeon RX 6900 XT	2800	15
MSI	Radeon RX 7900 XT	3200	13
MSI	GeForce RTX 4070 Ti	4200	3
MSI	Radeon RX 7900 XTX	4600	13
MSI	Radeon RX 7900 XTX	4600	1

рис. 3.6.2.4. Результаты поиска по ключевому слову «msi» в колонке «Производитель»

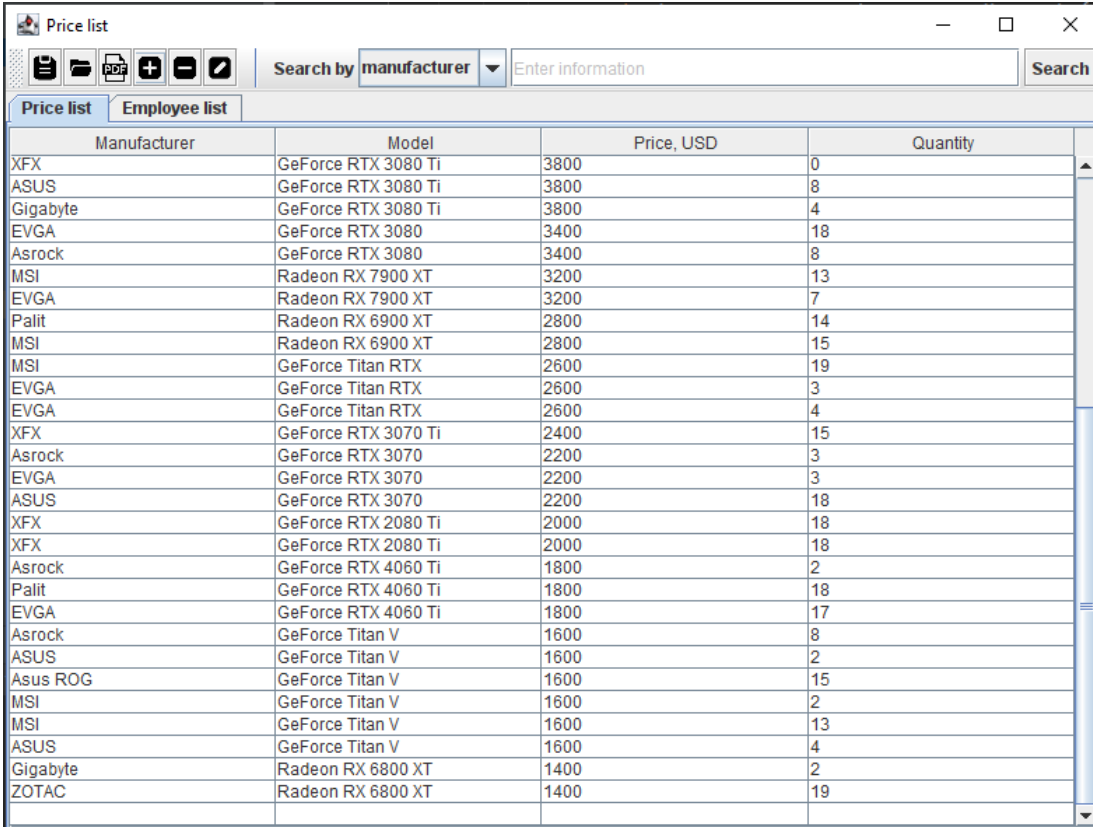
Примечание: ключевое слово не зависит от регистра.

3.5.3.2.2 Изменение данных

3.5.3.2.2.1 Добавление строк

Для списков «Прейскурант товаров» и «Список работников» добавление строк происходит в активный список (выбранный на панели выбора вкладок) в конец таблицы (если ни одна строка не выбрана – рис.3.6.3.1.) или после выбранной строки – рис.3.6.3.2.  Для добавления строки необходимо нажать кнопку «Добавить строку» на панели инструментов.

Для списка «Список продаж» добавление продажи происходит также нажатием на кнопку «Добавить строку». После нажатия на кнопку, пользователя встретит диалоговое окно, в котором необходимо ввести ID продавца, ID проданного видеоадаптера и дату продажи (рис. 3.6.3.3.1.). В случае корректного введения всех данных, отобразится окно подтверждения, в котором пользователю будет предоставлена подробная информация о введенных ID (рис. 3.6.3.3.2.). В случае некорректного введения данных, пользователь будет уведомлен об ошибке содержательным сообщением, из которого можно понять, в каком ID была допущена ошибка (рис.3.6.3.3.3.). После подтверждения данных, продажа добавится в список продаж, а в таблице «Прейскурант товаров» будет обновлена информация в соответствии с новой продажей (рис. 3.6.3.3.4 – 3.6.3.3.6)



Manufacturer	Model	Price, USD	Quantity
XFx	GeForce RTX 3080 Ti	3800	0
ASUS	GeForce RTX 3080 Ti	3800	8
Gigabyte	GeForce RTX 3080 Ti	3800	4
EVGA	GeForce RTX 3080	3400	18
Asrock	GeForce RTX 3080	3400	8
MSI	Radeon RX 7900 XT	3200	13
EVGA	Radeon RX 7900 XT	3200	7
Palit	Radeon RX 6900 XT	2800	14
MSI	Radeon RX 6900 XT	2800	15
MSI	GeForce Titan RTX	2600	19
EVGA	GeForce Titan RTX	2600	3
EVGA	GeForce Titan RTX	2600	4
XFx	GeForce RTX 3070 Ti	2400	15
Asrock	GeForce RTX 3070	2200	3
EVGA	GeForce RTX 3070	2200	3
ASUS	GeForce RTX 3070	2200	18
XFx	GeForce RTX 2080 Ti	2000	18
XFx	GeForce RTX 2080 Ti	2000	18
Asrock	GeForce RTX 4060 Ti	1800	2
Palit	GeForce RTX 4060 Ti	1800	18
EVGA	GeForce RTX 4060 Ti	1800	17
Asrock	GeForce Titan V	1600	8
ASUS	GeForce Titan V	1600	2
Asus ROG	GeForce Titan V	1600	15
MSI	GeForce Titan V	1600	2
MSI	GeForce Titan V	1600	13
ASUS	GeForce Titan V	1600	4
Gigabyte	Radeon RX 6800 XT	1400	2
ZOTAC	Radeon RX 6800 XT	1400	19

рис. 3.6.3.1. Результат добавления строки, не выбрав ни одной строки в таблице

Price list

Search by **manufacturer** Enter information Search

Price list Employee list

Manufacturer	Model	Price, USD	Quantity
Asrock	GeForce RTX 4090	5000	17
Gigabyte	GeForce RTX 4090	5000	12
Asus ROG	GeForce RTX 4080	4800	16
Asus ROG	GeForce RTX 4080	4800	15
Asus ROG	GeForce RTX 4080	4800	12
Palit	GeForce RTX 4080	4800	1
Palit	GeForce RTX 4080	4800	10
Palit	Radeon RX 7900 XTX	4600	17
MSI	Radeon RX 7900 XTX	4600	13
XFX	Radeon RX 7900 XTX	4600	9
MSI	Radeon RX 7900 XTX	4600	1
Palit	GeForce RTX 3090 Ti	4400	6
Asrock	GeForce RTX 3090 Ti	4400	8
Asrock	GeForce RTX 4070 Ti	4200	15
XFX	GeForce RTX 4070 Ti	4200	4
MSI	GeForce RTX 4070 Ti	4200	3
XFX	GeForce RTX 3080 Ti	3800	11
XFX	GeForce RTX 3080 Ti	3800	0
ASUS	GeForce RTX 3080 Ti	3800	8
Gigabyte	GeForce RTX 3080 Ti	3800	4
EVGA	GeForce RTX 3080	3400	18
Asrock	GeForce RTX 3080	3400	8
MSI	Radeon RX 7900 XT	3200	13
EVGA	Radeon RX 7900 XT	3200	7
Palit	Radeon RX 6900 XT	2800	14
MSI	Radeon RX 6900 XT	2800	15
MSI	GeForce Titan RTX	2600	19
EVGA	GeForce Titan RTX	2600	3
EVGA	GeForce Titan RTX	2600	4

Рис. 3.6.3.2. Результат добавления строки, выбрав одну из строк таблицы

Price list

Search by **Seller's ID** Enter information Search

Price list Employee list Sales list

Seller's name	Seller's ID	Sold item	Sold item ID	Date
Bo Johnston	7	Asus ROG GeForce Titan V	32	02.05.2002
Jrue Brown	15	XFX GeForce RTX 3070 Ti	23	05.05.2002
Martin Allen	6	XFX Radeon RX 7900 XTX	5	07.05.2002

Please, enter sale data

?
Seller's id:
Sold item id:
Sale date:

OK Cancel

Рис. 3.6.3.3.1. Экранная форма добавления продажи

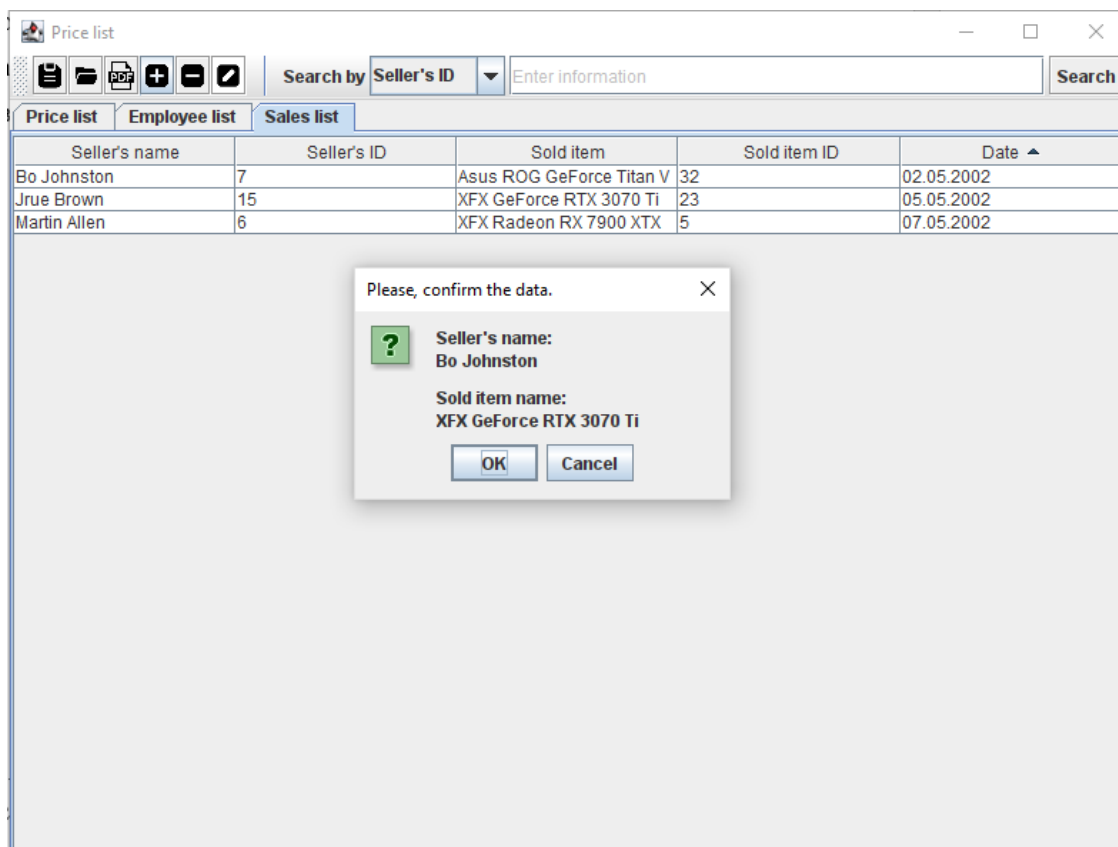


Рис. 3.6.3.3.2. Экранная форма подтверждения продажи

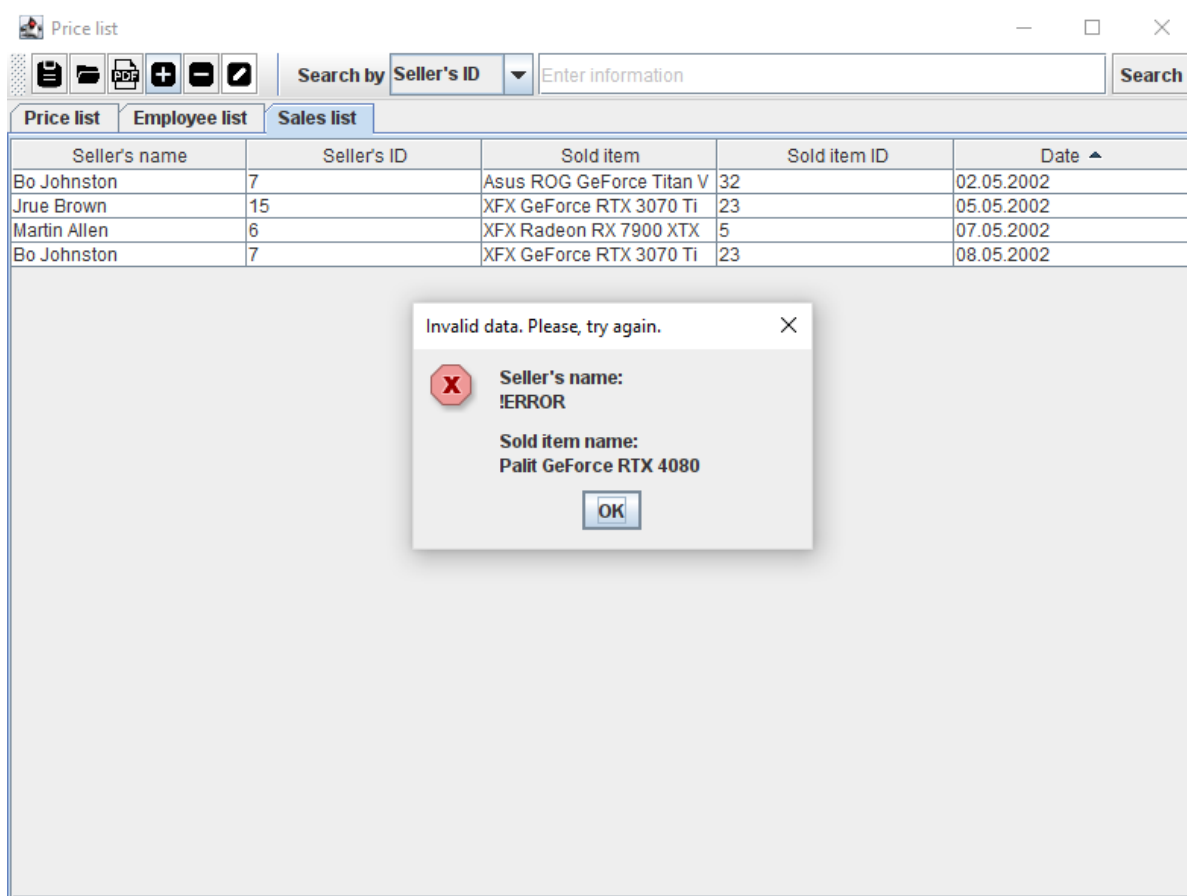
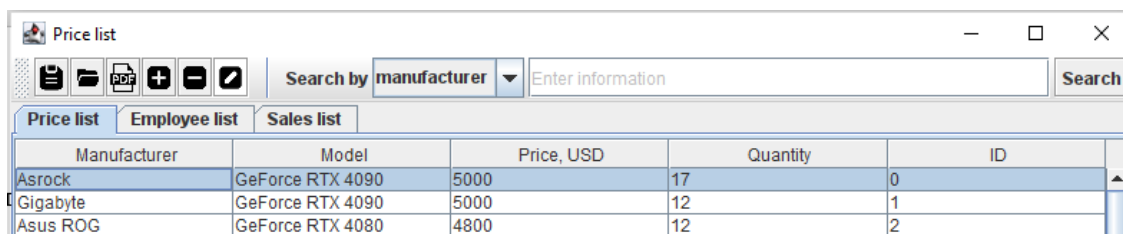


Рис. 3.6.3.3.3. Уведомление об ошибке при вводе ID



Price list window with tabs: Price list, Employee list, Sales list. Search by manufacturer. Search bar: Enter information. Search button.

Manufacturer	Model	Price, USD	Quantity	ID
Asrock	GeForce RTX 4090	5000	17	0
Gigabyte	GeForce RTX 4090	5000	12	1
Asus ROG	GeForce RTX 4080	4800	12	2

Рис. 3.6.3.3.4. Прейскурант товаров до добавления продажи

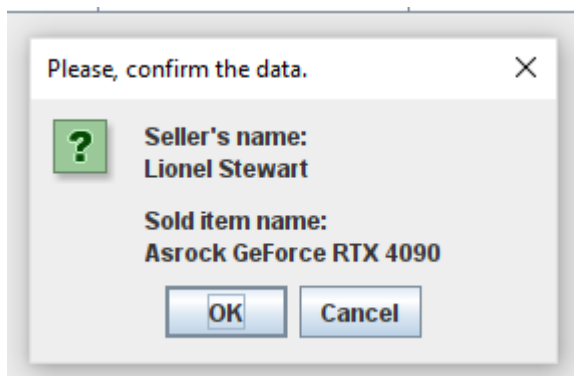
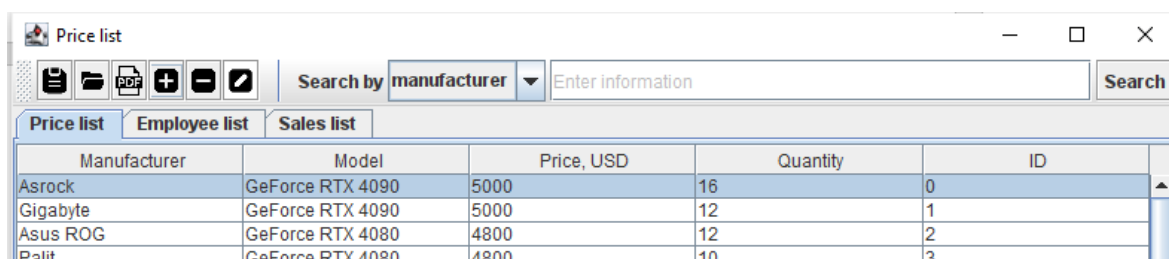


Рис. 3.6.3.3.5. Добавление продажи видеоадаптера с ID 0



Price list window with tabs: Price list, Employee list, Sales list. Search by manufacturer. Search bar: Enter information. Search button.

Manufacturer	Model	Price, USD	Quantity	ID
Asrock	GeForce RTX 4090	5000	16	0
Gigabyte	GeForce RTX 4090	5000	12	1
Asus ROG	GeForce RTX 4080	4800	12	2
Polix	GeForce RTX 4080	4800	10	3

Рис. 3.6.3.3.6. Обновленный вид прейскуранта после добавления продажи.

3.5.3.2.2.2 Удаление строк

Удаление строк происходит нажатием на кнопку «Удалить строку» 

В случае, если не одна строка не выбрана, программа уведомит пользователя об ошибке – рис.3.6.4.

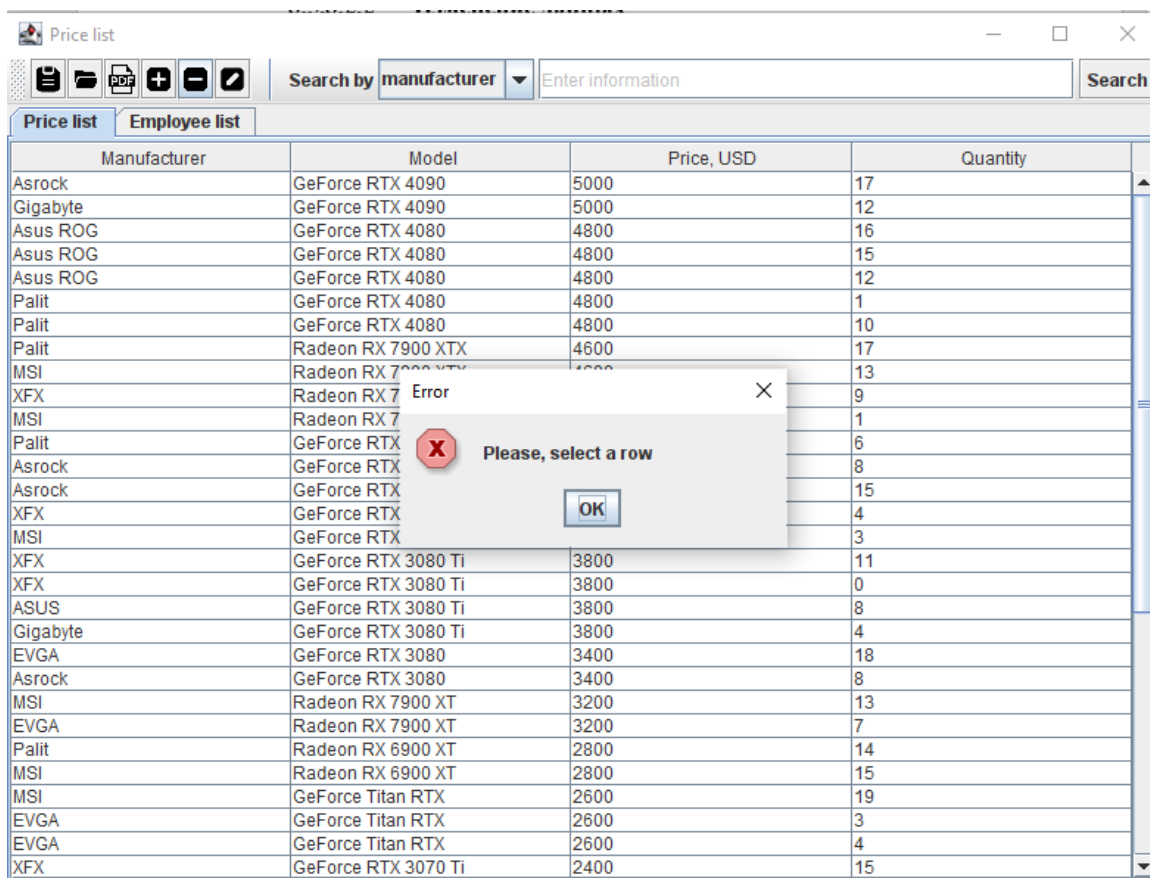




Рис.3.6.4 Попытка удаления строки, не выбрав ни одной

3.5.3.2.2.3. Редактирование информации

Редактирование информации происходит путем нажатия кнопки «Режим редактирования».  При включенном режиме редактирования, кнопка «Режим редактирования» инвертирует цвета (чтобы пользователь знал, когда режим редактирования включен)  и открывает доступ к редактированию таблицы. Для редактирования данных в поле таблицы, необходимо дважды кликнуть по полю, в которое необходимо внести изменения – рис. 3.6.5.


Price list			
<div> <div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> <div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> </div>			
Search by		manufacturer	Enter information
Search			
Price list		Employee list	
Manufacturer	Model	Price, USD	Quantity
Asrock	GeForce RTX 4090	5000	17
Gigabyte	GeForce RTX 4090	5000	12
Asus ROG	GeForce RTX 4080	4800	16
Asus ROG	GeForce RTX 4080	4800	15
Asus ROG	GeForce RTX 4080	4800	12
Palit	GeForce RTX 4080	4700	1
Palit	GeForce RTX 4080	4800	10
Palit	Radeon RX 7900 XTX	4600	17
MSI	Radeon RX 7900 XTX	4600	13
XFX	Radeon RX 7900 XTX	4600	9
MSI	Radeon RX 7900 XTX	4600	1
Palit	GeForce RTX 3090 Ti	4400	6
Asrock	GeForce RTX 3090 Ti	4400	8
Asrock	GeForce RTX 4070 Ti	4200	15
XFX	GeForce RTX 4070 Ti	4200	4
MSI	GeForce RTX 4070 Ti	4200	3
XFX	GeForce RTX 3080 Ti	3800	11
XFX	GeForce RTX 3080 Ti	3800	0
ASUS	GeForce RTX 3080 Ti	3800	8
Gigabyte	GeForce RTX 3080 Ti	3800	4
EVGA	GeForce RTX 3080	3400	18
Asrock	GeForce RTX 3080	3400	8
MSI	Radeon RX 7900 XT	3200	13
EVGA	Radeon RX 7900 XT	3200	7
Palit	Radeon RX 6900 XT	2800	14
MSI	Radeon RX 6900 XT	2800	15
MSI	GeForce Titan RTX	2600	19
EVGA	GeForce Titan RTX	2600	3
EVGA	GeForce Titan RTX	2600	4
XFX	GeForce RTX 3070 Ti	2400	15

Рис.3.6.5. – изменение данных в таблице в режиме редактирования

3.5.3.3. Сохранение данных

3.5.3.3.1. Формирование отчета о товарах

Для формирования отчета о товарах необходимо выполнить следующие действия:

1. Нажать на кнопку «Создать отчет» 
2. В появившемся диалоговом окне ввести название отчета (если пользователь не добавит в название файла расширение .pdf, программа сделает это автоматически) – рис.3.6.6.1.
3. В появившемся диалоговом окне выбрать файл, из которого будут взяты данные для формирования отчета – рис.3.6.6.2.

Отчет будет сохранен в корневой папке – рис.3.6.6.3.

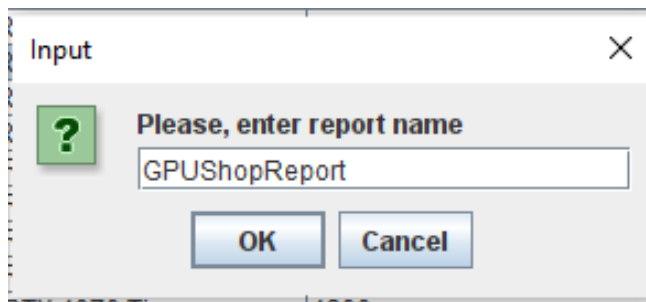


Рис.3.6.6.1. – диалоговое окно выбора имени отчета

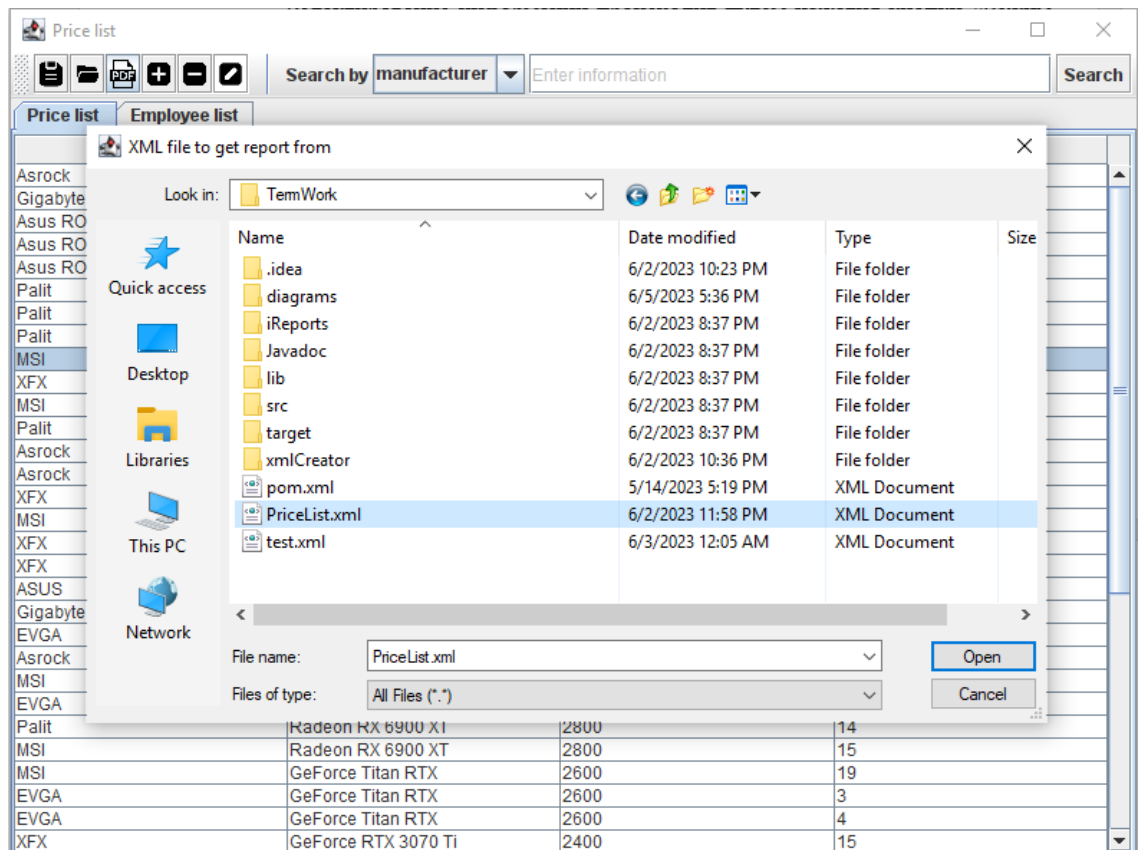



Рис.3.6.6.2. – диалоговое окно выбора файла для обработки информации

EITech market report
2415 Via Campo, Montebello, CA 90640, United States
CEO: Michael Se Santa, michaelseesanta@eitech.com

Manufacturer	Model	Price	Quantity
Asrock	GeForce RTX 4090	5000	17
Gigabyte	GeForce RTX 4090	5000	12
Aus ROG	GeForce RTX 4080	4800	16
Aus ROG	GeForce RTX 4080	4800	15
Aus ROG	GeForce RTX 4080	4800	12
Palit	GeForce RTX 4080	4800	1
Palit	GeForce RTX 4080	4800	10
Palit	Radeon RX 7900 XTX	4600	17
MSI	Radeon RX 7900 XTX	4600	13
XFX	Radeon RX 7900 XTX	4600	9
MSI	Radeon RX 7900 XTX	4600	1
Palit	GeForce RTX 3090 Ti	4400	6
Asrock	GeForce RTX 3090 Ti	4400	8
Asrock	GeForce RTX 4070 Ti	4200	15

Рис.3.6.6.3. – Отчет о товарах в магазине

3.5.3.3.2. Сохранение данных в файл

Для сохранения измененных данных необходимо нажать на кнопку «Сохранить»  , в появившемся диалоговом окне выбрать файл, в который нужно внести изменения или выбрать директорию и ввести название нового файла, в который будут записаны данные. Если пользователь не введет в название файла расширение .xml, программа сделает это автоматически – рис. 3.6.7.1 - 3.6.7.3.

Price list

Search by manufacturer Enter information Search

Manufacturer	Model	Price, USD	Quantity
Asrock	GeForce RTX 4090	5200	17
Gigabyte	GeForce RTX 4090	5000	12
Asus ROG	GeForce RTX 4080	4800	16
Asus ROG	GeForce RTX 4080	4800	15
Asus ROG	GeForce RTX 4080	4800	12
Palit	GeForce RTX 4080	4800	1
Palit	GeForce RTX 4080	4800	10
Palit	Radeon RX 7900 XTX	4600	17
MSI	Radeon RX 7900 XTX	4600	13
XFX	Radeon RX 7900 XTX	4600	9
MSI	Radeon RX 7900 XTX	4600	1
Palit	GeForce RTX 3090 Ti	4400	6
Asrock	GeForce RTX 3090 Ti	4400	8
Asrock	GeForce RTX 4070 Ti	4200	15
XFX	GeForce RTX 4070 Ti	4200	4
MSI	GeForce RTX 4070 Ti	4200	3
XFX	GeForce RTX 3080 Ti	3800	11
XFX	GeForce RTX 3080 Ti	3800	0
ASUS	GeForce RTX 3080 Ti	3800	8
Gigabyte	GeForce RTX 3080 Ti	3800	4
EVGA	GeForce RTX 3080	3400	18
Asrock	GeForce RTX 3080	3400	8
MSI	Radeon RX 7900 XT	3200	13
EVGA	Radeon RX 7900 XT	3200	7
Palit	Radeon RX 6900 XT	2800	14
MSI	Radeon RX 6900 XT	2800	15
MSI	GeForce Titan RTX	2600	19
EVGA	GeForce Titan RTX	2600	3
EVGA	GeForce Titan RTX	2600	4
XFX	GeForce RTX 3070 Ti	2400	15

Рис. 3.6.7.1. – внесение изменений в таблицу

C:\Users\neko\Desktop\Lab\Edited.xml

Search...

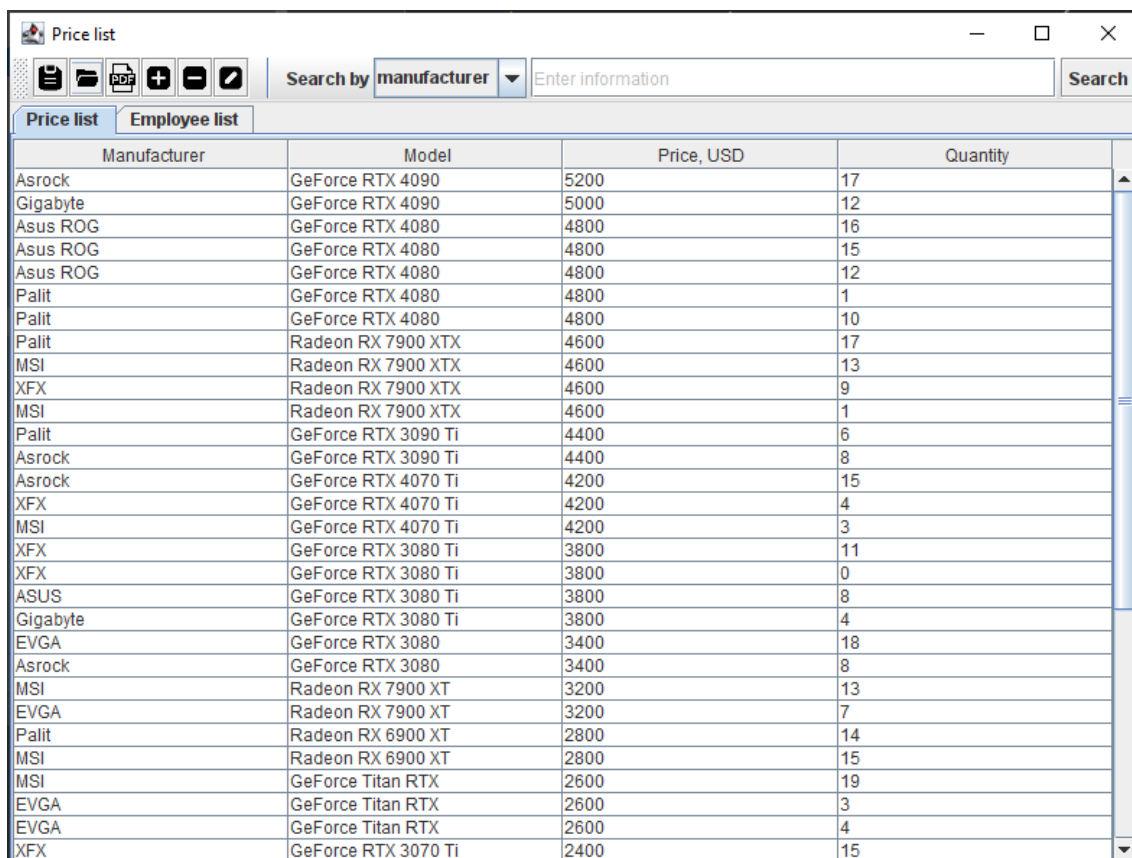
Яндекс Маркет Яндекс

```

<?xml version="1.0" encoding="UTF-8"?>
<pricelist>
  <GPU quantity="17" price="5200" model="GeForce RTX 4090" manufacturer="Asrock"/>
  <GPU quantity="12" price="5000" model="GeForce RTX 4090" manufacturer="Gigabyte"/>
  <GPU quantity="16" price="4800" model="GeForce RTX 4080" manufacturer="Asus ROG"/>
  <GPU quantity="15" price="4800" model="GeForce RTX 4080" manufacturer="Asus ROG"/>
  <GPU quantity="12" price="4800" model="GeForce RTX 4080" manufacturer="Asus ROG"/>
  <GPU quantity="1" price="4800" model="GeForce RTX 4080" manufacturer="Palit"/>
  <GPU quantity="10" price="4800" model="GeForce RTX 4080" manufacturer="Palit"/>
  <GPU quantity="17" price="4600" model="Radeon RX 7900 XTX" manufacturer="Palit"/>
  <GPU quantity="13" price="4600" model="Radeon RX 7900 XTX" manufacturer="MSI"/>
  <GPU quantity="9" price="4600" model="Radeon RX 7900 XTX" manufacturer="XFX"/>
  <GPU quantity="1" price="4600" model="Radeon RX 7900 XTX" manufacturer="MSI"/>
  <GPU quantity="6" price="4400" model="GeForce RTX 3090 Ti" manufacturer="Palit"/>
  <GPU quantity="8" price="4400" model="GeForce RTX 3090 Ti" manufacturer="Asrock"/>
  <GPU quantity="15" price="4200" model="GeForce RTX 4070 Ti" manufacturer="Asrock"/>
  <GPU quantity="4" price="4200" model="GeForce RTX 4070 Ti" manufacturer="XFX"/>
  <GPU quantity="3" price="4200" model="GeForce RTX 4070 Ti" manufacturer="MSI"/>
  <GPU quantity="11" price="3800" model="GeForce RTX 3080 Ti" manufacturer="XFX"/>
  <GPU quantity="0" price="3800" model="GeForce RTX 3080 Ti" manufacturer="XFX"/>
  <GPU quantity="8" price="3800" model="GeForce RTX 3080 Ti" manufacturer="ASUS"/>
  <GPU quantity="4" price="3800" model="GeForce RTX 3080 Ti" manufacturer="Gigabyte"/>
  <GPU quantity="18" price="3400" model="GeForce RTX 3080" manufacturer="EVGA"/>
  <GPU quantity="8" price="3400" model="GeForce RTX 3080" manufacturer="Asrock"/>
  <GPU quantity="13" price="3200" model="Radeon RX 7900 XT" manufacturer="MSI"/>
  <GPU quantity="7" price="3200" model="Radeon RX 7900 XT" manufacturer="EVGA"/>
  <GPU quantity="14" price="2800" model="Radeon RX 6900 XT" manufacturer="Palit"/>
  <GPU quantity="15" price="2800" model="Radeon RX 6900 XT" manufacturer="MSI"/>
  <GPU quantity="19" price="2600" model="GeForce Titan RTX" manufacturer="MSI"/>
  <GPU quantity="3" price="2600" model="GeForce Titan RTX" manufacturer="EVGA"/>
  <GPU quantity="4" price="2600" model="GeForce Titan RTX" manufacturer="EVGA"/>
  <GPU quantity="15" price="2400" model="GeForce RTX 3070 Ti" manufacturer="XFX"/>

```

Рис. 3.6.7.2. – Сохраненный файл



The screenshot shows a window titled "Price list" with a toolbar containing icons for file operations and a search bar. Below the toolbar are two tabs: "Price list" (selected) and "Employee list". The main area displays a table with the following data:

Manufacturer	Model	Price, USD	Quantity
Asrock	GeForce RTX 4090	5200	17
Gigabyte	GeForce RTX 4090	5000	12
Asus ROG	GeForce RTX 4080	4800	16
Asus ROG	GeForce RTX 4080	4800	15
Asus ROG	GeForce RTX 4080	4800	12
Palit	GeForce RTX 4080	4800	1
Palit	GeForce RTX 4080	4800	10
Palit	Radeon RX 7900 XTX	4600	17
MSI	Radeon RX 7900 XTX	4600	13
XFX	Radeon RX 7900 XTX	4600	9
MSI	Radeon RX 7900 XTX	4600	1
Palit	GeForce RTX 3090 Ti	4400	6
Asrock	GeForce RTX 3090 Ti	4400	8
Asrock	GeForce RTX 4070 Ti	4200	15
XFX	GeForce RTX 4070 Ti	4200	4
MSI	GeForce RTX 4070 Ti	4200	3
XFX	GeForce RTX 3080 Ti	3800	11
XFX	GeForce RTX 3080 Ti	3800	0
ASUS	GeForce RTX 3080 Ti	3800	8
Gigabyte	GeForce RTX 3080 Ti	3800	4
EVGA	GeForce RTX 3080	3400	18
Asrock	GeForce RTX 3080	3400	8
MSI	Radeon RX 7900 XT	3200	13
EVGA	Radeon RX 7900 XT	3200	7
Palit	Radeon RX 6900 XT	2800	14
MSI	Radeon RX 6900 XT	2800	15
MSI	GeForce Titan RTX	2600	19
EVGA	GeForce Titan RTX	2600	3
EVGA	GeForce Titan RTX	2600	4
XFX	GeForce RTX 3070 Ti	2400	15

Рис.3.6.7.3 – открытие сохраненного файла

4. Исходные тексты ПК представлены в приложении А.

Заключение

В результате проделанной работы разработан ПК «Учет прейскуранта товаров и списка работников магазина графических процессоров», предназначенный для учета прейскуранта товаров и списка работников магазина.

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

Курсовой проект удовлетворяет поставленным требованиям.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЕ ТЕКСТЫ ПК

main.class

```
package org.example;

import org.apache.log4j.Logger;

public class Main {
    private static final Logger log = Logger.getLogger("Main.class");
    // Создание и отображение экранной формы
    public static void main(String[] args) {
        log.info("Запуск приложения");
        new GPUShopAccounting().show();
    }
}
```

GPU.class

```
package org.example;

public class GPU {
    private String m_manufacturer;
    private String m_model;
    private String m_price;
    private String m_quantity;
    private int m_id;

    public GPU(String manufacturer, String model, String price, String quantity, int id){
        m_manufacturer = manufacturer;
        m_model = model;
        m_price = price;
        m_quantity = quantity;
        m_id = id;
    }

    public String getManufacturer() {
        return m_manufacturer;
    }

    public String getModel() {
        return m_model;
    }
}
```



```

    }

    public String getPrice() {
        return m_price;
    }

    public String getQuantity() {
        return m_quantity;
    }


    public int getId() {
        return m_id;
    }

    public void setManufacturer(String manufacturer) {
        m_manufacturer = manufacturer;
    }

    public void setModel(String model) {
        m_model = model;
    }

    public void setPrice(String price) {
        m_price = price;
    }

    public void setQuantity(String quantity) {
        m_quantity = quantity;
    }

    public void setId(int id) {
        m_id = id;
    }
}

```

Employee.class

```

package org.example;

public class Employee {
    private String m_name;
    private String m_surname;
    private String m_title;
    private String m_phoneNumber;
    private int m_id;

    public Employee(String name, String surname, String title, String phoneNumber, int id){

```

```

        m_name = name;
        m_surname = surname;
        m_title = title;
        m_phoneNumber = phoneNumber;
        m_id = id;
    }
    public String getName() {
        return m_name;
    }
    public String getPhoneNumber() {
        return m_phoneNumber;
    }
    public String getSurname() {
        return m_surname;
    }
    public String getTitle() {
        return m_title;
    }

    public int getId() {
        return m_id;
    }
    public void setName(String name){
        m_name = name;
    }
    public void setSurname(String surname){
        m_surname = surname;
    }
    public void setTitle(String title){
        m_title = title;
    }
    public void setPhoneNumber(String phoneNumber){
        m_phoneNumber = phoneNumber;
    }
    public void setId(int id){
        m_id = id;
    }
}

```

Sale.class

```
package org.example;

public class Sale {
    private String sellerId;
    private String soldItemId;
    private String date;

    public Sale(String sellerId, String soldItemId, String date){
        this.sellerId = sellerId;
        this.soldItemId = soldItemId;
        this.date = date;
    }

    public String getDate() {
        return date;
    }

    public String getSellerId() {
        return sellerId;
    }

    public String getSoldItemId() {
        return soldItemId;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public void setSellerId(String sellerId) {
        this.sellerId = sellerId;
    }

    public void setSoldItemId(String soldItemId) {
        this.soldItemId = soldItemId;
    }
}
```

GPUShopAccounting.class

```

package org.example;

/* Подключение библиотек */

import javax.management.RuntimeErrorException;
import javax.swing.*;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import javax.swing.event.TableModelEvent;
import javax.swing.event.TableModelListener;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableCellEditor;
import javax.swing.table.TableRowSorter;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import net.sf.jasperreports.engine.*;
import net.sf.jasperreports.engine.data.JRXmlDataSource;
import net.sf.jasperreports.engine.export.JRPdfExporter;
import org.w3c.dom.*;
import org.xml.sax.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import org.apache.log4j.Logger;

public class GPUShopAccounting {
    private static final Logger log = Logger.getLogger("Interface.class");
    /* Объявления графических компонентов */
    public JFrame priceList;
    private DefaultTableModel priceListModel;
    private DefaultTableModel employeesListModel;
    private DefaultTableModel salesListModel;
    private JButton saveButton;
    private JButton openButton;
    private JButton addButton;

```

```

private JButton removeButton;
private JButton editButton;
private JButton generateReportButton;
private JLabel searchByLabel;
private JComboBox searchBy;
private JTextField searchInformation;
private JButton searchButton;
private JToolBar toolBar;
private JTabbedPane tabsPane;
private JTable graphicProcessorUnits;
private TableRowSorter gpuRowSorter;
private JScrollPane gpuTableScrollPane;
private JTable employees;
private TableRowSorter employeesRowSorter;
private JScrollPane employeeTableScrollPane;
private JTable sales;
private TableRowSorter salesRowSorter;
private JScrollPane salesScrollPane;
private JTextArea additionalInformation;
private String sourceFileName;
private boolean isTableEditable;
public static final Object mutexPriceList = new Object();
private List<GPU> gpuList;
private List<Employee> employeeList;
private List<Sale> saleList;

/**
 * Creates a JFrame window and sets its size
 * Creates save, open, add, remove, edit buttons and adds them to a toolbar
 * Creates search menu and puts adds it to a toolbar
 * Places a toolbar to the top of a window
 * Creates a table with the information about GPUs and puts it to the center of a window
 * Creates an additional information text field and puts it to the right of a window
 */
public void show() {
    /* Создание окна */
    log.info("Создание окна");
    priceList = new JFrame("Price list");

```

```

priceList.setSize(800, 600);
priceList.setLocation(300, 300);
priceList.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

/* Скейлинг изображений для правильного отображения в кнопках */
ImageIcon imageIconSave = new ImageIcon(new
ImageIcon("src/main/resources/save.png").getImage().getScaledInstance(20,
20,
Image.SCALE_DEFAULT));
ImageIcon imageIconOpen = new ImageIcon(new
ImageIcon("src/main/resources/open.png").getImage().getScaledInstance(20,
20,
Image.SCALE_DEFAULT));
ImageIcon imageIconReport = new ImageIcon(new
ImageIcon("src/main/resources/report.png").getImage().getScaledInstance(20,
20,
Image.SCALE_DEFAULT));
ImageIcon imageIconAdd = new ImageIcon(new
ImageIcon("src/main/resources/add.png").getImage().getScaledInstance(20,
20,
Image.SCALE_DEFAULT));
ImageIcon imageIconRemove = new ImageIcon(new
ImageIcon("src/main/resources/remove.png").getImage().getScaledInstance(20,
20,
Image.SCALE_DEFAULT));
ImageIcon imageIconEdit = new ImageIcon(new
ImageIcon("src/main/resources/edit.png").getImage().getScaledInstance(20,
20,
Image.SCALE_DEFAULT));

/* Создание кнопок и прикрепление иконок */
log.info("Создание кнопок");
saveButton = new JButton(imageIconSave);
openButton = new JButton(imageIconOpen);
generateReportButton = new JButton(imageIconReport);
addButton = new JButton(imageIconAdd);
removeButton = new JButton(imageIconRemove);
editButton = new JButton(imageIconEdit);

/* Настройка подсказок для кнопок */
log.info("Настройка подсказок для кнопок");
saveButton.setToolTipText("Save");
openButton.setToolTipText("Open");
generateReportButton.setToolTipText("Generate report");

```

```

addButton.setToolTipText("Add row");
removeButton.setToolTipText("Remove row");
editButton.setToolTipText("Edit mode");

/* Настройка размера кнопок */
log.info("Настройка размера кнопок");
saveButton.setPreferredSize(new Dimension(25, 25));
openButton.setPreferredSize(new Dimension(25, 25));
generateReportButton.setPreferredSize(new Dimension(25, 25));
addButton.setPreferredSize(new Dimension(25, 25));
removeButton.setPreferredSize(new Dimension(25, 25));
editButton.setPreferredSize(new Dimension(25, 25));

/* Создание поиска */
log.info("Создание поиска");
searchByLabel = new JLabel("Search by");
searchBy = new JComboBox(new String[]{"manufacturer", "model"});
searchInformation = new JTextField("Enter information");
searchInformation.setForeground(Color.LIGHT_GRAY);
searchButton = new JButton("Search");

/* Добавление кнопок и поиска на панель инструментов */
log.info("Добавление кнопок и поиска на панель инструментов");
toolBar = new JToolBar("Toolbar");
toolBar.add(saveButton);
toolBar.add(openButton);
toolBar.add(generateReportButton);
toolBar.add(addButton);
toolBar.add(removeButton);
toolBar.add(editButton);
toolBar.add(Box.createHorizontalStrut(12));
toolBar.add(new JSeparator(SwingConstants.VERTICAL));
toolBar.add(Box.createHorizontalStrut(12));
toolBar.add(searchByLabel);
toolBar.add(Box.createHorizontalStrut(3));
toolBar.add(searchBy);
toolBar.add(Box.createHorizontalStrut(3));
toolBar.add(searchInformation);

```

```

toolBar.add(Box.createHorizontalStrut(3));
toolBar.add(searchButton);

/* Размещение панели инструментов */
log.info("Размещение панели инструментов");
priceList.setLayout(new BorderLayout());
priceList.add(toolBar, BorderLayout.NORTH);

/* Создание таблицы с данными */
log.info("Создание таблицы с данными");
String[] gpuCharacteristics = {"Manufacturer", "Model", "Price, USD", "Quantity", "ID"};
String[][] gpuData = {/*{"MSI", "GeForce GTX 1080Ti", "30000", "Yes"}, {"GIGABYTE",
"GeForce RTX 2060Ti", "40000", "Yes"},
        {"Palit", "GeForce RTX 2070", "50000", "Yes"}, {"Asus", "GeForce RTX 2070Ti",
"55000", "No"},
        {"Asus ROG", "GeForce RTX 2080Ti", "60000", "No"}*/};
isTableEditable = false;
priceListModel = new DefaultTableModel(gpuData, gpuCharacteristics) {
    @Override
    public boolean isCellEditable(int row, int column) {
        return isTableEditable;
    }
};

graphicProcessorUnits = new JTable(priceListModel);
graphicProcessorUnits.setAutoCreateRowSorter(true);
gpuRowSorter = new TableRowSorter<DefaultTableModel>(priceListModel);
graphicProcessorUnits.setRowSorter(gpuRowSorter);
gpuTableScrollPane = new JScrollPane(graphicProcessorUnits);
graphicProcessorUnits.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
gpuList = new ArrayList<GPU>();
graphicProcessorUnits.getModel().addTableModelListener(new TableModelListener() {
    public void tableChanged(TableModelEvent e) {
        if (isTableEditable){
            if (tabsPane.getSelectedIndex() == 0){
                switch (e.getColumn()){

```



```

        case 0:

gpuList.get(e.getFirstRow()).setManufacturer((String)graphicProcessorUnits.getValueAt(e.getFirstRow(),
e.getColumn()));

        break;
        case 1:

gpuList.get(e.getFirstRow()).setModel((String)graphicProcessorUnits.getValueAt(e.getFirstRow(),
e.getColumn()));

        break;
        case 2:

gpuList.get(e.getFirstRow()).setPrice((String)graphicProcessorUnits.getValueAt(e.getFirstRow(),
e.getColumn()));

        break;
        case 3:

gpuList.get(e.getFirstRow()).setQuantity((String)graphicProcessorUnits.getValueAt(e.getFirstRow(),
e.getColumn()));

        break;
        case 4:

gpuList.get(e.getFirstRow()).setId(Integer.parseInt((String)graphicProcessorUnits.getValueAt(e.getFirstRow
(), e.getColumn())));

        break;
    }
}
else if (tabsPane.getSelectedIndex() == 1){
    switch (e.getColumn()){
        case 0:

employeeList.get(e.getFirstRow()).setName((String)graphicProcessorUnits.getValueAt(e.getFirstRow(),
e.getColumn()));

        break;
        case 1:

employeeList.get(e.getFirstRow()).setSurname((String)graphicProcessorUnits.getValueAt(e.getFirstRow(),
e.getColumn()));

```

```

        break;
    case 2:

employeeList.get(e.getFirstRow()).setTitle((String)graphicProcessorUnits.getValueAt(e.getFirstRow(),
e.getColumn()));

        break;
    case 3:

employeeList.get(e.getFirstRow()).setPhoneNumber((String)graphicProcessorUnits.getValueAt(e.getFirstRow(), e.getColumn()));

        break;
    case 4:

employeeList.get(e.getFirstRow()).setId(Integer.parseInt((String)graphicProcessorUnits.getValueAt(e.getFirstRow(), e.getColumn())));

        break;
    }
}
}
}
});

/* Размещение таблицы с данными */
//log.info("Размещение таблицы с данными");
//priceList.add(gpuTableScrollBar, BorderLayout.CENTER);

/* Создание таблицы с сотрудниками */
log.info("Создание таблицы с сотрудниками");
String[] employeesCharacteristics = {"Name", "Surname", "Title", "Phone number", "ID"};
String[][] employeesData = {/* {"Michael", "De Santa", "CEO", "+1-202-555-0180"}, {"Trevor",
"Phillips", "Co-CEO", "+1-202-555-0129"},
{"Juan", "Williams", "Salesman", "+1-202-555-0183"}, {"Dennis", "Garcia", "Shop
assistant", "+1-202-555-0165"},
{"Morgan", "Davies", "Cleaner", "+1-202-555-3478"}, {"Keith", "Martins", "Janitor", "+1-
202-555-3421"} */};

employeesListModel = new DefaultTableModel(employeesData, employeesCharacteristics) {
    @Override
    public boolean isCellEditable(int row, int column) {

```

```

        return isTableEditable;
    }
};

employees = new JTable(employeesListModel);
employees.setAutoCreateRowSorter(true);
employeesRowSorter = new TableRowSorter<DefaultTableModel>(employeesListModel);
employees.setRowSorter(employeesRowSorter);
employeeTableScrollPane = new JScrollPane(employees);
employees.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
employeeList = new ArrayList<Employee>();

/* Создание таблицы с данными */
log.info("Создание таблицы с продажами");
String[] salesCharacteristics = {"Seller's name", "Seller's ID", "Sold item", "Sold item ID",
    "Date"};

String[][] salesData = { };
salesListModel = new DefaultTableModel(salesData, salesCharacteristics) {
    @Override
    public boolean isCellEditable(int row, int column) {
        return false;
    }
};

sales = new JTable(salesListModel);
sales.setAutoCreateRowSorter(true);
salesRowSorter = new TableRowSorter<DefaultTableModel>(salesListModel);
sales.setRowSorter(salesRowSorter);
salesScrollPane = new JScrollPane(sales);
sales.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
saleList = new ArrayList<Sale>();

/* Создание окна с доп. информацией
log.info("Создание окна с доп. информацией");
additionalInformation = new JTextArea("Space for additional information");

/* Размещение окна с доп. информацией
log.info("Размещение окна с доп. информацией");
priceList.add(additionalInformation, BorderLayout.EAST);

```

```

priceList.add(new JScrollPane(additionalInformation), BorderLayout.EAST);*/

/* Добавление слушателей */
log.info("Добавление слушателей");
OpenButtonListener openButtonListener = new OpenButtonListener();
openButton.addMouseListener(openButtonListener);

SaveButtonListener saveButtonListener = new SaveButtonListener();
saveButton.addMouseListener(saveButtonListener);

GenerateReportButtonListener generateReportButtonListener = new
GenerateReportButtonListener();
generateReportButton.addMouseListener(generateReportButtonListener);

EditButtonListener editButtonListener = new EditButtonListener();
editButton.addMouseListener(editButtonListener);

FocusListener searchInformationFocusListener = new SearchInformationFocusListener();
SearchInformationKeyListener searchInformationKeyListener = new
SearchInformationKeyListener();
searchInformation.addFocusListener(searchInformationFocusListener);
searchInformation.addKeyListener(searchInformationKeyListener);

SearchButtonListener searchButtonListener = new SearchButtonListener();
searchButton.addMouseListener(searchButtonListener);

AddButtonListener addButtonListener = new AddButtonListener();
addButton.addMouseListener(addButtonListener);

RemoveButtonListener removeButtonListener = new RemoveButtonListener();
removeButton.addMouseListener(removeButtonListener);

tabsPane = new JTabbedPane(JTabbedPane.TOP,
        JTabbedPane.SCROLL_TAB_LAYOUT);
tabsPane.addTab("Price list", gpuTableScrollPane);
tabsPane.addTab("Employee list", employeeTableScrollPane);
tabsPane.addTab("Sales list", salesScrollPane);
tabsPane.setMnemonicAt(0, String.valueOf(1).charAt(0));

```

```

tabsPane.setMnemonicAt(1, String.valueOf(2).charAt(0));
tabsPane.setMnemonicAt(2, String.valueOf(3).charAt(0));

ChangeListener changeListener = new ChangeListener() {
    public void stateChanged(ChangeEvent changeEvent) {
        updateSearchByBox(tabsPane.getSelectedIndex());
    }
};
tabsPane.addChangeListener(changeListener);

priceList.add(tabsPane);

try {
    openFile("PriceList.xml");
}
catch (IOException | ParserConfigurationException | SAXException e){ }

priceList.setVisible(true);
}

/**
 * Creates a frame for selecting a CSV file and writes GPUPTable information to it.
 *
 * @throws IOException if there was a problem writing information.
 * @throws ParserConfigurationException if there was a problem configuring a parser.
 * @throws TransformerConfigurationException if there was a problem configuring a transformer.
 * @throws TransformerException if there was a problem with a transformer.
 */
private void saveFile() throws IOException, ParserConfigurationException,
TransformerConfigurationException, TransformerException {
    log.info("Сохранение данных в файл");
    String fileFormat = "*.xml";
    FileDialog save = new FileDialog(priceList, "Save file", FileDialog.SAVE);
    save.setFile(fileFormat);
    save.setVisible(true);
    String fileName = save.getDirectory() + save.getFile();
    if (!fileName.endsWith(".xml")) {
        fileName += ".xml";
    }
}

```

```

    }
    if (save.getFile() == null) return;
    sourceFileName = fileName;
    Thread saveToXmlThread = new Thread(new SaveToFileThread(fileName, priceListModel,
employeesListModel, salesListModel));
    saveToXmlThread.start();
}

/**
 * Creates a frame for selecting a CSV file and writes its information to GPUPTable.
 *
 * @throws ParserConfigurationException if there was a problem configuring a parser.
 * @throws SAXException if there was a problem parsing a file.
 * @throws IOException if there was a problem reading a file.
 */
private void openFile() throws ParserConfigurationException, SAXException, IOException {
    log.info("Считывание данных из файла");
    String fileFormat = "*.xml";
    FileDialog load = new FileDialog(priceList, "Open file", FileDialog.LOAD);
    load.setFile(fileFormat);
    load.setDirectory(System.getProperty("user.dir"));
    load.setVisible(true);
    String fileName = load.getDirectory() + load.getFile();
    if (load.getFile() == null) return;
    openFile(fileName);
}

private void openFile(String fileName) throws ParserConfigurationException, SAXException,
IOException {
    sourceFileName = fileName;
    DocumentBuilder dBuilder =
        DocumentBuilderFactory.newInstance().newDocumentBuilder();
    Document document = dBuilder.parse(new File(fileName));
    document.getDocumentElement().normalize();
    NodeList gpuNodeList = document.getElementsByTagName("GPU");
    int priceListTableRowCount = priceListModel.getRowCount();
    for (int i = 0; i < priceListTableRowCount; i++) priceListModel.removeRow(0);
    for (int i = 0; i < gpuNodeList.getLength(); i++) {
        Node element = gpuNodeList.item(i);

```

```

NamedNodeMap attrs = element.getAttributes();
String manufacturer = attrs.getNamedItem("manufacturer").getNodeValue();
String model = attrs.getNamedItem("model").getNodeValue();
String price = attrs.getNamedItem("price").getNodeValue();
String quantity = attrs.getNamedItem("quantity").getNodeValue();
int id = Integer.parseInt(attrs.getNamedItem("id").getNodeValue());
gpuList.add(new GPU(manufacturer, model, price, quantity, id));
priceListModel.addRow(new Object[]{ manufacturer, model, price, quantity, id });
}

NodeList employeesNodeList = document.getElementsByTagName("employee");
int employeeListTableRowCount = employeesListModel.getRowCount();
for (int i = 0; i < employeeListTableRowCount; i++) employeesListModel.removeRow(0);
for (int i = 0; i < employeesNodeList.getLength(); i++) {
    Node element = employeesNodeList.item(i);
    NamedNodeMap attrs = element.getAttributes();
    String name = attrs.getNamedItem("name").getNodeValue();
    String surname = attrs.getNamedItem("surname").getNodeValue();
    String title = attrs.getNamedItem("title").getNodeValue();
    String number = attrs.getNamedItem("number").getNodeValue();
    int id = Integer.parseInt(attrs.getNamedItem("id").getNodeValue());
    employeeList.add(new Employee(name, surname, title, number, id));
    employeesListModel.addRow(new Object[]{ name, surname, title, number, id });
}

NodeList salesNodeList = document.getElementsByTagName("sale");
int salesListTableRowCount = salesListModel.getRowCount();
for (int i = 0; i < salesListTableRowCount; i++) salesListModel.removeRow(0);
for (int i = 0; i < salesNodeList.getLength(); i++) {
    Node element = salesNodeList.item(i);
    NamedNodeMap attrs = element.getAttributes();
    String sellerName = attrs.getNamedItem("sellerName").getNodeValue();
    String sellerId = attrs.getNamedItem("sellerId").getNodeValue();
    String soldItemName = attrs.getNamedItem("soldItemName").getNodeValue();
    String soldItemId = attrs.getNamedItem("soldItemId").getNodeValue();
    String date = attrs.getNamedItem("date").getNodeValue();
    saleList.add(new Sale(sellerId, soldItemId, date));
    salesListModel.addRow(new Object[]{ sellerName, sellerId, soldItemName, soldItemId,
date});
}

```

```

    }

    public void addRow() {
        log.info("Добавление строки");
        if (tabsPane.getSelectedIndex() == 0){
            if (graphicProcessorUnits.getSelectedRow() == -1) {
                priceListModel.addRow(new Object[]{null,null,null,null});
            }
            else {
                gpuList.add(graphicProcessorUnits.getSelectedRow() + 1, new GPU(null, null, null, null, -
1));
                priceListModel.insertRow(graphicProcessorUnits.getSelectedRow() + 1, new
Object[]{null,null,null,null});
            }
        }
        else if (tabsPane.getSelectedIndex() == 1){
            if (employees.getSelectedRow() == -1){
                employeeList.add(new Employee(null, null, null, null, -1));
                employeesListModel.addRow(new Object[]{null,null,null,null});
            }
            else {
                employeeList.add(graphicProcessorUnits.getSelectedRow() + 1, new Employee(null, null,
null, null, -1));
                employeesListModel.insertRow(employees.getSelectedRow() + 1, new
Object[]{null,null,null,null});
            }
        }
        else if (tabsPane.getSelectedIndex() == 2){
            addSale();
        }
    }

    private void addSale(){
        JTextField sellerId = new JTextField(5);
        JTextField soldItemId = new JTextField(5);
        JTextField date = new JTextField(5);

        JPanel saleSetupPanel = new JPanel();
        saleSetupPanel.add(new JLabel("Seller's id:"));
    }

```



```

saleSetupPanel.add(sellerId);
saleSetupPanel.add(Box.createHorizontalStrut(15)); // a spacer
saleSetupPanel.add(new JLabel("Sold item id:"));
saleSetupPanel.add(soldItemId);
saleSetupPanel.add(Box.createHorizontalStrut(15)); // a spacer
saleSetupPanel.add(new JLabel("Sale date:"));
saleSetupPanel.add(date);

int result = JOptionPane.showConfirmDialog(null, saleSetupPanel,
    "Please, enter sale data", JOptionPane.OK_CANCEL_OPTION);
if (result != JOptionPane.OK_OPTION) {
    return;
}
String soldItemName = "!ERROR";
String sellerName = "!ERROR";
int quantityLeft = 0;
for (int i = 0; i < gpuList.size(); ++i){
    if (gpuList.get(i).getId() == Integer.parseInt(soldItemId.getText())){
        soldItemName = gpuList.get(i).getManufacturer();
        soldItemName += " ";
        soldItemName += gpuList.get(i).getModel();
        quantityLeft = Integer.parseInt(gpuList.get(i).getQuantity());
    }
}
for (int i = 0; i < employeeList.size(); ++i){
    if (employeeList.get(i).getId() == Integer.parseInt(sellerId.getText())){
        sellerName = employeeList.get(i).getName();
        sellerName += " ";
        sellerName += employeeList.get(i).getSurname();
    }
}
if (quantityLeft == 0){
    JOptionPane.showMessageDialog(null, "This GPU is already out of stock", "Error",
JOptionPane.ERROR_MESSAGE);
    return;
}
JPanel saleConfirmationPanel = new JPanel();

```

```

        saleConfirmationPanel.setLayout(new
BoxLayout(saleConfirmationPanel,
BoxLayout.Y_AXIS));
        saleConfirmationPanel.add(new JLabel("Seller's name:"));
        saleConfirmationPanel.add(new JLabel(sellerName));
        saleConfirmationPanel.add(Box.createVerticalStrut(10)); // a spacer
        saleConfirmationPanel.add(new JLabel("Sold item name:"));
        saleConfirmationPanel.add(new JLabel(soldItemName));

        if (sellerName.equals("!ERROR") | soldItemName.equals("!ERROR")){
            JOptionPane.showMessageDialog(null, saleConfirmationPanel, "Invalid data. Please, try
again.", JOptionPane.OK_OPTION);
            return;
        }
        else {
            result = JOptionPane.showConfirmDialog(null, saleConfirmationPanel, "Please, confirm the
data.", JOptionPane.OK_CANCEL_OPTION);
            if (result == JOptionPane.CANCEL_OPTION){
                return;
            }
        }
        Sale newSale = new Sale(sellerId.getText(), soldItemId.getText(), date.getText());
        saleList.add(newSale);
        salesListModel.addRow(new Object[]{sellerName, sellerId.getText(), soldItemName,
soldItemId.getText(), date.getText()});
        for (int i = 0; i < gpuList.size(); ++i){
            if (gpuList.get(i).getId() == Integer.parseInt(newSale.getSoldItemId())){
                gpuList.get(i).setQuantity(String.valueOf(Integer.parseInt(gpuList.get(i).getQuantity()
1)));
                priceListModel.setValueAt(gpuList.get(i).getQuantity(), i, 3);
                break;
            }
        }
    }

/**
 * Removes a selected row
 *
 * @throws NoRowSelectedException if no row is selected

```

```

        */
private void removeRow() throws NoRowSelectedException {
    log.info("Удаление строки");
    if (tabsPane.getSelectedIndex() == 0){
        if (graphicProcessorUnits.getSelectedRow() == -1) {
            throw new NoRowSelectedException();
        } else {

gpuList.remove(gpuRowSorter.convertRowIndexToModel(graphicProcessorUnits.getSelectedRow()));

priceListModel.removeRow(gpuRowSorter.convertRowIndexToModel(graphicProcessorUnits.getSelectedRow()));

        }
    }
    else if (tabsPane.getSelectedIndex() == 1){
        if (employees.getSelectedRow() == -1) {
            throw new NoRowSelectedException();
        } else {

employeeList.remove(employeesRowSorter.convertRowIndexToModel(employees.getSelectedRow()));

employeesListModel.removeRow(employeesRowSorter.convertRowIndexToModel(employees.getSelectedRow()));

        }
    }
    else if (tabsPane.getSelectedIndex() == 2){
        if (sales.getSelectedRow() == -1) {
            throw new NoRowSelectedException();
        } else {
            for (int i = 0; i<gpuList.size(); ++i){
                if (gpuList.get(i).getId() ==
Integer.parseInt(saleList.get(salesRowSorter.convertRowIndexToModel(sales.getSelectedRow())).getSoldItemId())){

gpuList.get(i).setQuantity(String.valueOf(Integer.parseInt(gpuList.get(i).getQuantity()) + 1));
                priceListModel.setValueAt(gpuList.get(i).getQuantity(), i, 3);
                break;
            }
        }
    }
}

```

```

    }

    saleList.remove(salesRowSorter.convertRowIndexToModel(sales.getSelectedRow()));

salesListModel.removeRow(salesRowSorter.convertRowIndexToModel(sales.getSelectedRow()));
    }
    }
}

/**
 * Performs search in a table with DefaultTableModel in a column selected in JComboBox searchBy
 ignoring letter register.
 *
 * @throws SearchFailedException if search did not yield results.
 */
public void performSearch() throws SearchFailedException {
    log.info("Выполнение поиска");
    if (!searchInformation.getText().equals("Enter information")) {
        if (tabsPane.getSelectedIndex() == 0){
            RowFilter<DefaultTableModel, Object> rowFilter = null;
            rowFilter = RowFilter.regexFilter("(?i)" + searchInformation.getText(),
searchBy.getSelectedIndex());

            gpuRowSorter.setRowFilter(rowFilter);
            if (gpuRowSorter.getViewRowCount() == 0) {
                throw new SearchFailedException();
            }
        }
        else if (tabsPane.getSelectedIndex() == 1) {
            RowFilter<DefaultTableModel, Object> rowFilter = null;
            rowFilter = RowFilter.regexFilter("(?i)" + searchInformation.getText(),
searchBy.getSelectedIndex());

            employeesRowSorter.setRowFilter(rowFilter);
            if (employeesRowSorter.getViewRowCount() == 0) {
                throw new SearchFailedException();
            }
        }
        else if (tabsPane.getSelectedIndex() == 2) {
            RowFilter<DefaultTableModel, Object> rowFilter = null;

```

```

        rowFilter      =      RowFilter.regexFilter("(?i)"      +      searchInformation.getText(),
searchBy.getSelectedIndex() == 0 ? 1 : 3);
        salesRowSorter.setRowFilter(rowFilter);
        if (salesRowSorter.getViewRowCount() == 0) {
            throw new SearchFailedException();
        }
    }
}
}
}

```

/* Класс слушателя для кнопки записи информации в файл */

```

public class SaveButtonListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        try {
            saveFile();
        } catch (IOException IOex) {
            IOex.printStackTrace();
        } catch (ParserConfigurationException PCex) {
            PCex.printStackTrace();
        } catch (TransformerConfigurationException TCex) {
            TCex.printStackTrace();
        } catch (TransformerException Tex) {
            Tex.printStackTrace();
        }
    }
}
}

```

/* Класс слушателя для кнопки открытия файла */

```

public class OpenButtonListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        try {
            openFile();
        } catch (FileNotFoundException FNFex) {
            FNFex.printStackTrace();
        } catch (IOException IOex) {
            IOex.printStackTrace();
        } catch (ParserConfigurationException PCEex) {
            PCEex.printStackTrace();
        }
    }
}

```

```

    } catch (SAXException SAXex) {
        SAXex.printStackTrace();
    }
}
}

```

```

public class GenerateReportButtonListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        String reportName = JOptionPane.showInputDialog("Please, enter report name");
        if (reportName == null) return;
        if (!reportName.endsWith(".pdf")) reportName+=" .pdf";
        String fileFormat = "*.xml";
        FileDialog load = new FileDialog(priceList, "XML file to get report from", FileDialog.LOAD);
        load.setFile(fileFormat);
        load.setDirectory(System.getProperty("user.dir"));
        load.setVisible(true);
        String fileName = load.getDirectory() + load.getFile();
        if (load.getFile() == null) return;
        Thread generateReportThread = new Thread(new CreatePDFReportThread(fileName,
reportName));
        generateReportThread.start();
    }
}

```

```

/* Класс слушателя для кнопки удаления строки */
public class AddButtonListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        addRow();
    }
}

```

```

/* Класс слушателя для кнопки удаления строки */
public class RemoveButtonListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        try {
            removeRow();
        } catch (NoRowSelectedException NRWex) {

```

```

        JOptionPane.showMessageDialog(null,        NRWex.getMessage(),        "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

/* Класс слушателя для кнопки изменения */
public class EditButtonListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        ImageIcon editInactive = new ImageIcon(new
ImageIcon("src/main/resources/edit.png").getImage().getScaledInstance(20,
Image.SCALE_DEFAULT));
        ImageIcon editActive = new ImageIcon(new
ImageIcon("src/main/resources/editActive.png").getImage().getScaledInstance(20,
Image.SCALE_DEFAULT));
        isTableEditable = !isTableEditable;
        if (isTableEditable) {
            editButton.setBackground(Color.BLACK);
            editButton.setIcon(editActive);
        } else {
            editButton.setBackground(Color.WHITE);
            editButton.setIcon(editInactive);
        }
    }
}

/* Класс слушателя для строки поиска */
private class SearchInformationFocusListener implements FocusListener {
    public void focusGained(FocusEvent e) {
        if (searchInformation.getText().equals("Enter information")) {
            searchInformation.setText("");
            searchInformation.setForeground(Color.BLACK);
        }
    }

    public void focusLost(FocusEvent e) {
        if (searchInformation.getText().length() == 0) {

```

```

        gpuRowSorter.setRowFilter(null);
        employeesRowSorter.setRowFilter(null);
        salesRowSorter.setRowFilter(null);
        searchInformation.setText("Enter information");
        searchInformation.setForeground(Color.LIGHT_GRAY);
    }
}

```

```

/* Класс слушателя нажатий клавиш для строки поиска */
private class SearchInformationKeyListener extends KeyAdapter {
    public void keyPressed(KeyEvent e) {
        if (e.getKeyCode() == KeyEvent.VK_ENTER) {
            try {
                performSearch();
            } catch (SearchFailedException SFex) {
                handleSearchFailedException(SFex);
            }
        }
    }
}

```

```

/* Класс слушателя для кнопки поиска */
public class SearchButtonListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        try {
            performSearch();
        } catch (SearchFailedException SFex) {
            handleSearchFailedException(SFex);
        }
    }
}

```

/* Класс исключения в случае нажатия кнопки, требующей выбранной строки, при отсутствии таковой */

```

private class NoRowSelectedException extends Exception {
    public NoRowSelectedException() {
        super("Please, select a row");
    }
}

```



```

    }
}

/**
 * Handles SearchFailedException
 *
 * @param SFex an object of SearchFailedException class
 */
private void handleSearchFailedException(SearchFailedException SFex) {
    gpuRowSorter.setRowFilter(null);
    employeesRowSorter.setRowFilter(null);
    salesRowSorter.setRowFilter(null);
    JOptionPane.showMessageDialog(null, SFex.getMessage(), "Search failed",
JOptionPane.ERROR_MESSAGE);
}

private void updateSearchByBox(int tabIndex){
    if (tabIndex == 0){
        searchBy.removeAllItems();
        searchBy.addItem("manufacturer");
        searchBy.addItem("model");
    } else if (tabIndex == 1) {
        searchBy.removeAllItems();
        searchBy.addItem("name");
        searchBy.addItem("surname");
        searchBy.addItem("title");
    }
    else if (tabIndex == 2) {
        searchBy.removeAllItems();
        searchBy.addItem("Seller's ID");
        searchBy.addItem("Sold item ID");
    }
}
}

```

CreatePDFReportThread.class

```
package org.example;
```

```

import net.sf.jasperreports.engine.*;
import net.sf.jasperreports.engine.data.JRXmlDataSource;
import net.sf.jasperreports.engine.export.JRPdfExporter;

import java.util.HashMap;

public class CreatePDFReportThread implements Runnable {
    private final ClassLoader classLoader = getClass().getClassLoader();
    private final String xmlFilePath;
    private final String htmlOutPath;

    public CreatePDFReportThread(String xmlFilePath, String htmlOutPath) {
        this.xmlFilePath = xmlFilePath;
        this.htmlOutPath = htmlOutPath;
    }

    @Override
    public void run() {
        synchronized (GPUShopAccounting.mutexPriceList) {
            try {
                Thread.sleep(10); // imitates delay between joining mutex
                System.out.println("CreateHTMLReportThread is running");
                System.out.println(xmlFilePath);
                createPriceListReport(xmlFilePath,    "/pricelist/GPU",    "ReportTemplate.jrxml",
htmlOutPath);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    public static void createPriceListReport(String datasource, String xpath, String template,
String resultpath) {
        try {
            // Указание источника XML-данных
            JRXmlDataSource dataSource = new JRXmlDataSource(datasource, xpath);
            // Создание отчета на базе шаблона
            JasperReport jasperReport = JasperCompileManager.compileReport(template);

```

```

// Заполнение отчета данными
    JasperPrint print = JasperFillManager.fillReport(jasperReport, new HashMap(),
dataSource);

    JRExporter exporter = null;
    if (resultpath.toLowerCase().endsWith(".pdf"))
        exporter = new JRPdfExporter(); // Генерация отчета в формате PDF
// Задание имени файла для загрузки отчета
    exporter.setParameter(JRExporterParameter.OUTPUT_FILE_NAME, resultpath);
// Подключение данных к отчету
    exporter.setParameter(JRExporterParameter.JASPER_PRINT, print);
// Вывозка отчета в заданном формате
    exporter.exportReport();
} catch (JRException e) {
    e.printStackTrace();
}
}
}

```

SaveToFileThread.class

```
package org.example;
```

```
import org.w3c.dom.Document;
```

```
import org.w3c.dom.Element;
```

```
import org.w3c.dom.Node;
```

```
import javax.swing.table.DefaultTableModel;
```

```
import javax.xml.parsers.DocumentBuilder;
```

```
import javax.xml.parsers.DocumentBuilderFactory;
```

```
import javax.xml.parsers.ParserConfigurationException;
```

```
import javax.xml.transform.Transformer;
```

```
import javax.xml.transform.TransformerException;
```

```
import javax.xml.transform.TransformerFactory;
```

```
import javax.xml.transform.dom.DOMSource;
```

```
import javax.xml.transform.stream.StreamResult;
```

```
import java.io.FileWriter;
```

```
import java.io.IOException;
```

```
public class SaveToFileThread implements Runnable {
```

```

private final String xmlFilename;
private final DefaultTableModel priceListModel;
private final DefaultTableModel employeeListModel;
private final DefaultTableModel salesListModel;

public SaveToFileThread(String xmlFilename, DefaultTableModel model1,
DefaultTableModel model2, DefaultTableModel model3) {
    if (!xmlFilename.endsWith(".xml")) {
        throw new RuntimeException("The .xml file is expected!");
    }

    this.xmlFilename = xmlFilename;
    this.priceListModel = model1;
    this.employeeListModel = model2;
    this.salesListModel = model3;
}

@Override
public void run() {
    synchronized (GPUShopAccounting.mutexPriceList) {
        try {
            Thread.sleep(10);
            System.out.println("SaveToFileThread is running");
            DocumentBuilder builder =
                DocumentBuilderFactory.newInstance().newDocumentBuilder();
            Document document = builder.newDocument();
            Node pricelistNode = document.createElement("pricelist");
            document.appendChild(pricelistNode);
            for (int i = 0; i < priceListModel.getRowCount(); i++) {
                Element GPU = document.createElement("GPU");
                pricelistNode.appendChild(GPU);
                GPU.setAttribute("manufacturer", (String) priceListModel.getValueAt(i, 0));
                GPU.setAttribute("model", (String) priceListModel.getValueAt(i, 1));
                GPU.setAttribute("price", (String) priceListModel.getValueAt(i, 2));
                GPU.setAttribute("quantity", (String) priceListModel.getValueAt(i, 3));
                GPU.setAttribute("id", String.valueOf(i));
            }
            for (int i = 0; i < employeeListModel.getRowCount(); i++) {

```

```

        Element employee = document.createElement("employee");
        pricelistNode.appendChild(employee);
        employee.setAttribute("name", (String) employeeListModel.getValueAt(i, 0));
        employee.setAttribute("surname", (String) employeeListModel.getValueAt(i, 1));
        employee.setAttribute("title", (String) employeeListModel.getValueAt(i, 2));
        employee.setAttribute("number", (String) employeeListModel.getValueAt(i, 3));
        employee.setAttribute("id", String.valueOf(i));
    }
    for (int i = 0; i < salesListModel.getRowCount(); i++) {
        Element sale = document.createElement("sale");
        pricelistNode.appendChild(sale);
        sale.setAttribute("sellerName", (String) salesListModel.getValueAt(i, 0));
        sale.setAttribute("sellerId", (String) salesListModel.getValueAt(i, 1));
        sale.setAttribute("soldItemName", (String) salesListModel.getValueAt(i, 2));
        sale.setAttribute("soldItemId", (String) salesListModel.getValueAt(i, 3));
        sale.setAttribute("date", String.valueOf(i));
    }
    Transformer transformer = TransformerFactory.newInstance().newTransformer();
    java.io.FileWriter fileWriter = new FileWriter(xmlFilename);
    transformer.transform(new DOMSource(document), new StreamResult(fileWriter));
} catch (ParserConfigurationException | IOException | TransformerException |
InterruptedException e) {
    e.printStackTrace();
}
}
}
}

```

TableFillerThread.class

```
package org.example;
```

```

import org.w3c.dom.Document;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import javax.swing.table.DefaultTableModel;
import javax.xml.parsers.DocumentBuilder;

```

```

import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

public class TableFillerThread implements Runnable {
    private final String fileName;
    private final DefaultTableModel priceListModel;
    private final DefaultTableModel employeeListModel;

    public TableFillerThread(String fileName, DefaultTableModel model1, DefaultTableModel
model2) {
        if (!fileName.endsWith(".xml")) {
            throw new RuntimeException("The .xml file expected!");
        }

        if (!Files.exists(Paths.get(fileName))) {
            throw new RuntimeException("The xml file doesn't exists!");
        }

        this.fileName = fileName;
        this.priceListModel = model1;
        this.employeeListModel = model2;
    }

    @Override
    public void run() {
        synchronized (GPUShopAccounting.mutexPriceList) {
            try {
                Thread.sleep(10);
                //System.out.println("TableFillerThread is running");
                DocumentBuilder builder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
                Document document = builder.parse(new File(fileName));
                document.getDocumentElement().normalize();
                NodeList GPUList = document.getElementsByTagName("GPU");
            }
        }
    }
}

```

```

        for (int i = 0; i < GPUList.getLength(); ++i) {
            Node element = GPUList.item(i);
            NamedNodeMap attributes = element.getAttributes();
            Object[] rowsData = {
                attributes.getNamedItem("manufacturer").getNodeValue(),
                attributes.getNamedItem("model").getNodeValue(),
                attributes.getNamedItem("price").getNodeValue(),
                attributes.getNamedItem("quantity").getNodeValue(),
            };
            priceListModel.addRow(rowsData);
        }
        NodeList employeesNodeList = document.getElementsByTagName("employee");
        for (int i = 0; i < employeesNodeList.getLength(); i++) {
            Node element = employeesNodeList.item(i);
            NamedNodeMap attrs = element.getAttributes();
            String name = attrs.getNamedItem("name").getNodeValue();
            String surname = attrs.getNamedItem("surname").getNodeValue();
            String title = attrs.getNamedItem("title").getNodeValue();
            String number = attrs.getNamedItem("number").getNodeValue();
            employeeListModel.addRow(new String[]{name, surname, title, number});
        }
    } catch (ParserConfigurationException | SAXException | IOException |
InterruptedException e) {
        e.printStackTrace();
    }
}

SearchFailedException.class
package org.example;

/* Класс исключения в случае отсутствия результатов поиска */
public class SearchFailedException extends Exception {
    public SearchFailedException() {
        super("No results");
    }
}

PriceList.xml

```

```

<?xml version="1.0" encoding="UTF-8"?><pricelist><GPU id="0" manufacturer="Asrock"
model="GeForce RTX 4090" price="5000" quantity="17"/><GPU id="1" manufacturer="Gigabyte"
model="GeForce RTX 4090" price="5000" quantity="12"/><GPU id="2" manufacturer="Asus
ROG" model="GeForce RTX 4080" price="4800" quantity="11"/><GPU id="3"
manufacturer="Palit" model="GeForce RTX 4080" price="4800" quantity="10"/><GPU id="4"
manufacturer="Palit" model="Radeon RX 7900 XTX" price="4600" quantity="17"/><GPU id="5"
manufacturer="XFX" model="Radeon RX 7900 XTX" price="4600" quantity="9"/><GPU id="6"
manufacturer="MSI" model="Radeon RX 7900 XTX" price="4600" quantity="1"/><GPU id="7"
manufacturer="Palit" model="GeForce RTX 3090 Ti" price="4400" quantity="6"/><GPU id="8"
manufacturer="Asrock" model="GeForce RTX 3090 Ti" price="4400" quantity="8"/><GPU id="9"
manufacturer="Asrock" model="GeForce RTX 4070 Ti" price="4200" quantity="15"/><GPU
id="10" manufacturer="XFX" model="GeForce RTX 4070 Ti" price="4200" quantity="4"/><GPU
id="11" manufacturer="MSI" model="GeForce RTX 4070 Ti" price="4200" quantity="3"/><GPU
id="12" manufacturer="XFX" model="GeForce RTX 3080 Ti" price="3800" quantity="11"/><GPU
id="13" manufacturer="ASUS" model="GeForce RTX 3080 Ti" price="3800" quantity="8"/><GPU
id="14" manufacturer="Gigabyte" model="GeForce RTX 3080 Ti" price="3800"
quantity="4"/><GPU id="15" manufacturer="EVGA" model="GeForce RTX 3080" price="3400"
quantity="18"/><GPU id="16" manufacturer="Asrock" model="GeForce RTX 3080" price="3400"
quantity="8"/><GPU id="17" manufacturer="MSI" model="Radeon RX 7900 XT" price="3200"
quantity="13"/><GPU id="18" manufacturer="EVGA" model="Radeon RX 7900 XT" price="3200"
quantity="7"/><GPU id="19" manufacturer="Palit" model="Radeon RX 6900 XT" price="2800"
quantity="14"/><GPU id="20" manufacturer="MSI" model="Radeon RX 6900 XT" price="2800"
quantity="15"/><GPU id="21" manufacturer="MSI" model="GeForce Titan RTX" price="2600"
quantity="19"/><GPU id="22" manufacturer="EVGA" model="GeForce Titan RTX" price="2600"
quantity="4"/><GPU id="23" manufacturer="XFX" model="GeForce RTX 3070 Ti" price="2400"
quantity="15"/><GPU id="24" manufacturer="Asrock" model="GeForce RTX 3070" price="2200"
quantity="3"/><GPU id="25" manufacturer="EVGA" model="GeForce RTX 3070" price="2200"
quantity="3"/><GPU id="26" manufacturer="ASUS" model="GeForce RTX 3070" price="2200"
quantity="18"/><GPU id="27" manufacturer="XFX" model="GeForce RTX 2080 Ti" price="2000"
quantity="18"/><GPU id="28" manufacturer="Asrock" model="GeForce RTX 4060 Ti"
price="1800" quantity="2"/><GPU id="29" manufacturer="Palit" model="GeForce RTX 4060 Ti"
price="1800" quantity="18"/><GPU id="30" manufacturer="EVGA" model="GeForce RTX 4060
Ti" price="1800" quantity="17"/><GPU id="31" manufacturer="Asrock" model="GeForce Titan V"
price="1600" quantity="8"/><GPU id="32" manufacturer="Asus ROG" model="GeForce Titan V"
price="1600" quantity="15"/><GPU id="33" manufacturer="MSI" model="GeForce Titan V"
price="1600" quantity="13"/><GPU id="34" manufacturer="ASUS" model="GeForce Titan V"
price="1600" quantity="4"/><GPU id="35" manufacturer="Gigabyte" model="Radeon RX 6800
XT" price="1400" quantity="2"/><GPU id="36" manufacturer="ZOTAC" model="Radeon RX 6800

```


XT" price="1400" quantity="19"/><employee id="0" name="Michael" number="+12025550180"
 surname="De Santa" title="CEO"/><employee id="1" name="Trevor" number="+12025550145"
 surname="Phillips" title="Co-CEO"/><employee id="2" name="Isaac" number="+12021006334"
 surname="Bowman" title="Computer technician"/><employee id="3" name="Dominick"
 number="+12021015724" surname="Little" title="Computer technician"/><employee id="4"
 name="Kieran" number="+12021026962" surname="Lowe" title="Computer
 technician"/><employee id="5" name="Lewis" number="+12021028145" surname="Graham"
 title="Computer technician"/><employee id="6" name="Martin" number="+12021009961"
 surname="Allen" title="Computer technician"/><employee id="7" name="Bo"
 number="+12021011942" surname="Johnston" title="Web-programmer"/><employee id="8"
 name="Everest" number="+12021032391" surname="Payne" title="Web-programmer"/><employee
 id="9" name="Lionel" number="+12021000153" surname="Stewart" title="Web-
 programmer"/><employee id="10" name="Colin" number="+12021017421" surname="Olson"
 title="Web-programmer"/><employee id="11" name="Legacy" number="+12021019895"
 surname="Gibson" title="Web-programmer"/><employee id="12" name="Santino"
 number="+12021014771" surname="Robinson" title="Salesman"/><employee id="13" name="Julio"
 number="+12021019912" surname="Stevens" title="Salesman"/><employee id="14" name="Keith"
 number="+12021017035" surname="Edwards" title="Salesman"/><employee id="15" name="Jrue"
 number="+12021023811" surname="Brown" title="Salesman"/><employee id="16" name="Hector"
 number="+12021017673" surname="Austin" title="Salesman"/><employee id="17" name="Ryland"
 number="+12021007711" surname="Barnes" title="Salesman"/><employee id="18" name="Finley"
 number="+12021025547" surname="Garza" title="Salesman"/><employee id="19" name="Eliam"
 number="+12021032757" surname="Ortiz" title="Salesman"/><employee id="20" name="Isaiah"
 number="+12021008723" surname="Walters" title="Salesman"/><employee id="21"
 name="Kaiser" number="+12021000778" surname="Jacobs" title="Salesman"/><employee id="22"
 name="Ari" number="+12021022190" surname="Dean" title="Salesman"/><employee id="23"
 name="Ira" number="+12021030106" surname="Barnett" title="Salesman"/><employee id="24"
 name="Charles" number="+12021019264" surname="Elliott" title="Salesman"/><employee id="25"
 name="Chris" number="+12021023805" surname="Mason" title="Salesman"/><employee id="26"
 name="Yousef" number="+12021024370" surname="Hughes" title="Salesman"/><employee id="27"
 name="Mario" number="+12021031101" surname="Delgado" title="Salesman"/><employee id="28"
 name="Edgar" number="+12021019629" surname="Parker" title="Salesman"/><employee id="29"
 name="Gustavo" number="+12021019954" surname="Pena" title="Salesman"/><employee id="30"
 name="Ramon" number="+12021004966" surname="Powers" title="Salesman"/><employee
 id="31" name="Marshall" number="+12021026308" surname="Douglas"
 title="Salesman"/><employee id="32" name="Gary" number="+12021024626"
 surname="Williamson" title="Assistant"/><employee id="33" name="Warren"
 number="+12021021538" surname="Becker" title="Assistant"/><employee id="34" name="Jonah"

number="+12021022929" surname="Zimmerman" title="Assistant"/><employee id="35"
name="Jonas" number="+12021031115" surname="Wade" title="Janitor"/><employee id="36"
name="Tru" number="+12021022704" surname="Day" title="Janitor"/><employee id="37"
name="Dangelo" number="+12021002306" surname="Santos" title="Janitor"/><employee id="38"
name="Riggs" number="+12021005021" surname="Sandoval" title="Janitor"/><employee id="39"
name="Jimmy" number="+12021019072" surname="Vaughn" title="Janitor"/><sale date="0"
sellerId="10" sellerName="Colin Olson" soldItemId="2" soldItemName="Asus ROG GeForce RTX
4080"/></pricelist>