Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

|  |
| --- |
| Институт информационных технологий  и анализа данных |

наименование института

**Отчёт**

к лабораторной работе №4

по дисциплине “ООП”:

|  |
| --- |
| «Разработка классов с использованием механизмов наследования, полиморфизма и инкапсуляции» |

наименование темы

Вариант №9

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент | ИСТб 19-2 |  |  |  | Козлов М.В. |
|  | шифр группы |  | подпись |  | Фамилия ИО |
| Проверил | | |  |  | Маланова Т.В |
|  | | | подпись |  | Фамилия ИО |
| Работа защищена с оценкой | | |  | | |

Иркутск 2020

Содержание

[1 Постановка задачи 3](#_Toc57999074)

[2 Проектирование классов 4](#_Toc57999075)

[3 Описание структуры пользовательского меню 6](#_Toc57999076)

[4 Спецификации 8](#_Toc57999077)

[5 Таблица тестов 15](#_Toc57999078)

[6 Листинг исходного кода 20](#_Toc57999079)

[7 Заключение 21](#_Toc57999080)

[Список использованных источников 22](#_Toc57999081)

1 Постановка задачи

В соответствии с индивидуальным заданием описать иерархию классов, для каждого класса описать поля и соответствующие методы доступа к ним. В зависимости от задания некоторые из этих классов(как минимум один) являют-ся абстрактными и служат для выделения общих данных и поведения для дру-гих классов. Абстрактный класс должен содержать как минимум один абст-рактный метод, реализация которого у его наследников должна различаться. Помимо этого в общую часть задания входит разработка класса группирующего объекты описанных в соответствии с заданием классов. Для первого варианта индивидуального задания это может быть класс «кафедра». Для выполнения за-дания необходимо создать некоторое количество объектов, добавить их в груп-пу используя предусмотренные методы класса-«группы» и для каждого из них вызвать унаследованный метод.

**Вариант индивидуального задания**

9. хлебобулочное изделие, товар, молочный продукт, магазин;

2 Проектирование классов

На рисунке 1 представлена UML диаграмма классов.

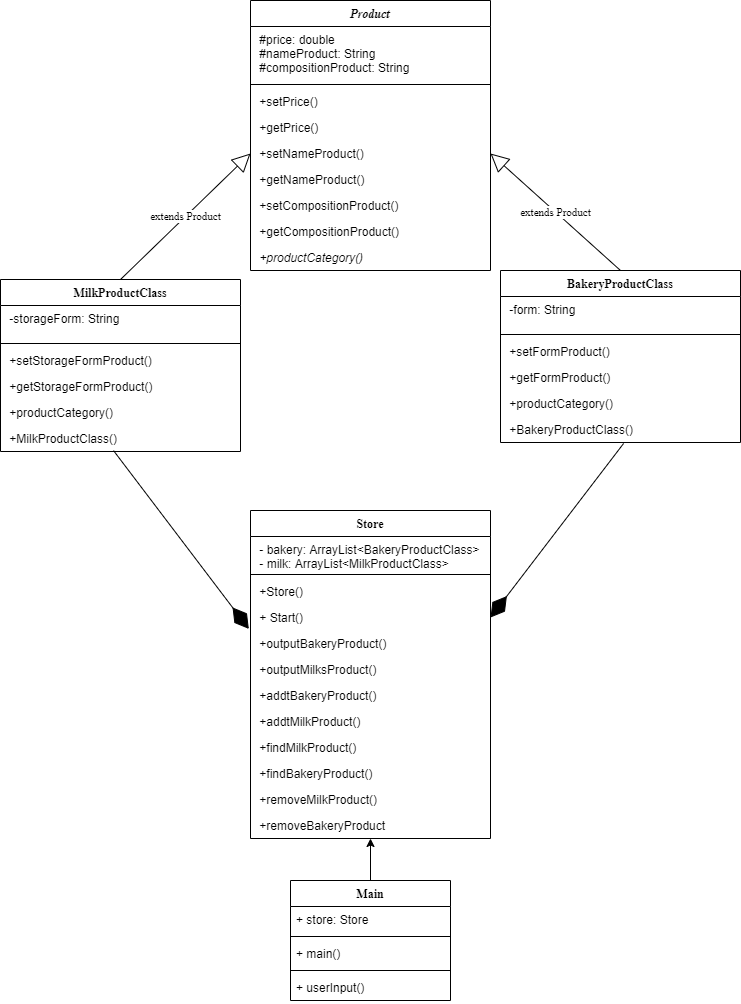


Рисунок 1 – Диаграмма классов UML

3 Описание структуры пользовательского меню

Пользовательское меню реализует класс Store. Весь процесс взаимодействия пользователя с меню осуществляется с помощью функции Start(). Пользовательское меню содержит 10 пунктов:

1. Вывод всех продуктов
2. Вывод хлебо-булочных продуктов
3. Вывод молочных продуктов
4. Добавить хлебо-булочный продукт
5. Добавить молочный продукт
6. Найти молочный продукт.
7. Найти хлебобулочный продукт
8. Удалить молочный продукт
9. Удалить хлебобулочный продукт
10. Выход

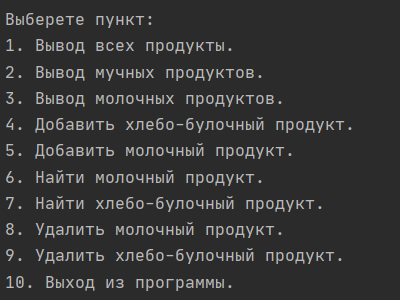


Рисунок 2 –Пользовательское меню

В случае, если пользователь ввёл число не попадающее в диапазон [1; 10], программа предупреждает об этом пользователя и пользователь снова вводит число.



Рисунок 3 – Введен не корректный номер пункта

T

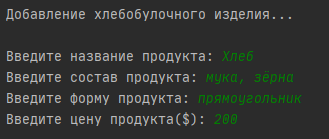


Рисунок 4 – Пример добавления хлебобулочного изделия

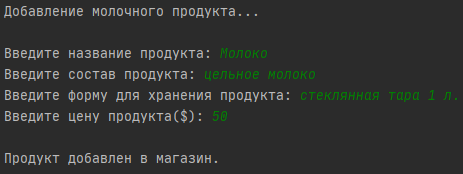


Рисунок 5 – Пример добавления молочного продукта

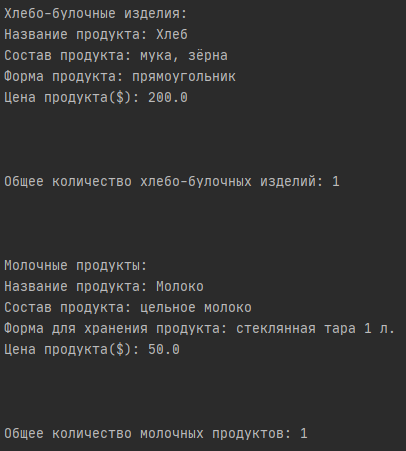


Рисунок 6 – Пример вывода информации о всех продуктах в магазине

4 Спецификации

**4.1** **Абстрактный** **класс Product (супер класс)**

Этот класс реализует базовый класс для классов-наследников.

Таблица 4.1.1 – Таблица полей

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Тип | Уровень доступа | Назначение |
| price | double | protected | Цена продукта |
| nameProduct | String | protected | Название продукта |
| compositionProduct | String | protected | Состав продукта |

Таблица 4.1.2 – Таблица конструкторов

|  |  |  |
| --- | --- | --- |
| Уровень доступа | Аргументы | Описание |
| public | - | Конструктор по умолчанию |

Таблица 4.1.3 – Таблица методов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя | Тип | Статический | Уровень доступа | Аргументы | Назначение |
| getPrice | double | Нет | private | - | Метод для получения цены продукта |
| setNameProduct | String | Нет | public | String n | Метод, для изменения названия продукта |
| getNameProduct |  |  |  | - | Метод для получения имени продукта |
| setCompositionProduct | void |  |  | String c | Метод для изменения состава продукта |
| getCompositionProduct | String |  |  | - | Метод для получения состава продукта |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Продолжение таблицы 4.1.3 | | | | | |
| productCategory | String |  |  | - | Абстрактный метод для обозначения типа продукта |

**4.2** **Класс Store**

Этот класс реализует пользовательское меню.

Таблица 4.2.1 – Таблица полей

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Тип | Уровень доступа | Назначение |
| bakery | ArrayList <BakeryProductClass> | private | Стэк хлебобулочных изделий |
| milk | ArrayList <MilkProductClass> | private | Стэк молочных изделий |

Таблица 4.2.2 – Таблица конструкторов

|  |  |  |
| --- | --- | --- |
| Уровень доступа | Аргументы | Описание |
| public | - | Конструктор по умолчанию |

Таблица 4.2.3 – Таблица методов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя | Тип | Статический | Уровень доступа | Аргументы | Назначение |
| Start | void | Нет | public | - | Метод, реализующий пользовательское меню |
| findMilkProduct | void | Нет | public | int productNum | Метод, реализующий поиск молочного продукта по введённому пользователем номеру в стэке |
| findBakeryProduct | void | Нет | public | int productNum | Метод, реализующий поиск хлебобулочного изделия по введённому пользователем номеру в стэке |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| removeMilkProduct | void | Нет | public | int productNum | Метод, реализующий удаление молочного продукта по введённому пользователем номеру в стэке |
| removeBakeryProduct | void | Нет | public | int productNum | Метод, реализующий удаление хлебобулочного изделия по введённому пользователем номеру в стэке |
| outputBakeryProduct | void | Нет | public | - | Метод, реализующий вывод всех хлебобулочных изделий на экран |
| outputMilksProduct | void | Нет | public | - | Метод, реализующий вывод всех молочных продуктов на экран |
| addBakeryProduct | void | Нет | public | - | Метод, реализующий добавление хлебобулочного изделия в стэк |
| addMilkProduct | void | Нет | public |  | Метод, реализующий добавление молочного продукта в стэк |

**4.3 Класс - наследник MilkProductClass**

Этот класс реализует молочный продукт.

Таблица 4.3.1 – Таблица полей

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Тип | Уровень доступа | Назначение |
| storageForm | String | private | Форма для хранения молочных продуктов |

Таблица 4.3.2 – Таблица конструкторов

|  |  |  |
| --- | --- | --- |
| Уровень доступа | Аргументы | Описание |
| public | - | Конструктор по умолчанию |

Таблица 4.3.3 – Таблица методов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя | Тип | Статический | Уровень доступа | Аргументы | Назначение |
| setStorageFormProduct | void | Нет | private | String f | Метод, изменяющий форму, в которой хранится продукт |
| getStorageFormProduct | void | Нет | public | - | Метод, получающий форму, в которой хранится продукт |
| productCategory | String | Нет | public | - | Метод, переопределяющий категорию продукта |

**4.4 Класс - наследник BakeryProductClass**

Этот класс реализует хлебобулочный продукт.

Таблица 4.4.1 – Таблица полей

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Тип | Уровень доступа | Назначение |
| form | String | private | Форма продукта |

Таблица 4.4.2 – Таблица конструкторов

|  |  |  |
| --- | --- | --- |
| Уровень доступа | Аргументы | Описание |
| public | - | Конструктор по умолчанию |

Таблица 4.4.3 – Таблица методов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя | Тип | Статический | Уровень доступа | Аргументы | Назначение |
| setFormProduct | String | Нет | public | - | Метод, изменяющий форму изделия |
| getFormProduct | String | Нет | public | - | Метод, получающий форму изделия |
| productCategory | String | Нет | public | - | Метод, переопределяющий категорию продукта |

**4.5 Класс - Main**

Этот класс реализует вход в программу.

Таблица 4.5.1 – Таблица методов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя | Тип | Статический | Уровень доступа | Аргументы | Назначение |
| main | String | Нет | public | - | Метод, решающий задачу A |

5 Таблица тестов

Таблица 1 – Таблица тестов

|  |  |  |  |
| --- | --- | --- | --- |
| **N теста** | **Назначение теста** | **Входные данные для теста** | **Выходные данные для теста** |
| 1 | Добавление молочного продукта и вывод молочных продуктов | (рис. 7) | (рис. 8) |
| 2 | Добавление хлебобулочного продукта и вывод хлебо-булочных продуктов | (рис. 9) | (рис. 10) |
| 3 | Добавление хлебобулочного продукта и вывод молочных продуктов(когда их не было добавлено) | (рис. 11) | (рис. 12) |
| 4 | Добавление молочного продукта и вывод всех продуктов | (рис. 13) | (рис. 14) |
| 5 | Добавление хлебобулочного и молочного продукта, с выводом всех продуктов | (рис. 15-16) | (рис. 17) |
| 6 | Поиск молочного продукта | (рис. 18) | (рис. 19) |
| 7 | Поиск хлебобулочного изделия | (рис. 20) | (рис. 21) |
| 8 | Удаление молочного изделия | (рис. 22) | (рис. 23) |
| 9 | Удаление хлебобулочного изделия | (рис. 24) | (рис. 25) |
| 10 | Поиск молочного продукта (продукта под данным номером нет) | (рис. 26) | (рис. 27) |
| 11 | Поиск хлебобулочного изделия (изделия под данным номером нет) | (рис. 28) | (рис. 29) |
| 12 | Удаление молочного изделия (продукта под данным номером нет) | (рис. 30) | (рис. 31) |
| 13 | Удаление хлебобулочного изделия (изделия под данным номером нет) | (рис. 32) | (рис. 33) |

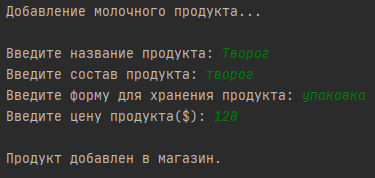


Рисунок 7– Добавление молочного продукта(тест 1 из таблицы тестов 1)

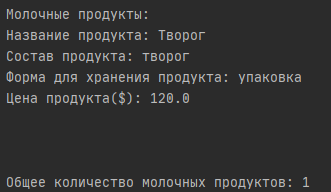


Рисунок 8 – Результат теста 1 из таблицы тестов 1

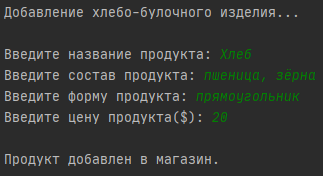


Рисунок 9 - Добавление хлебо-булочного продукта(тест 2 из таблицы тестов 1)

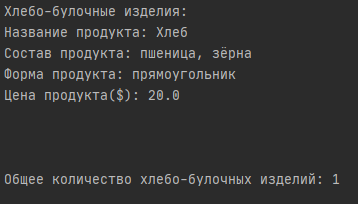


Рисунок 10 – Результат теста 2 из таблицы тестов 1

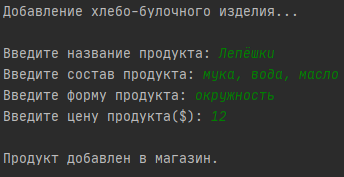


Рисунок 11 - Добавление хлебобулочного продукта(тест 3 из таблицы тестов 1)



Рисунок 12 – Результат теста 3 из таблицы тестов 1

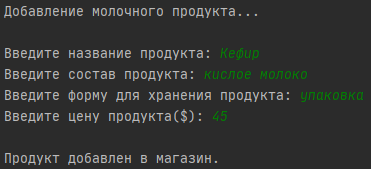


Рисунок 13 - Добавление молочного продукта(тест 4 из таблицы тестов 1)

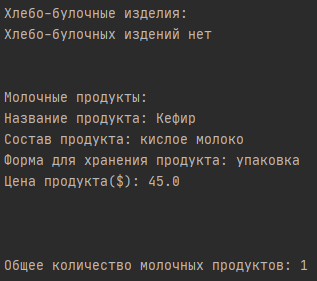


Рисунок 14 – Результат теста 4 из таблицы тестов 1

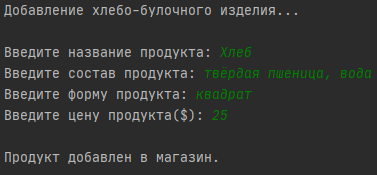


Рисунок 15 - Добавление хлебобулочного продукта(тест 5 из таблицы тестов 1)

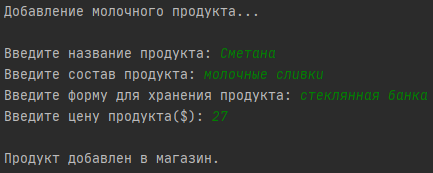


Рисунок 16 - Добавление молочного продукта(тест 5 из таблицы тестов 1)

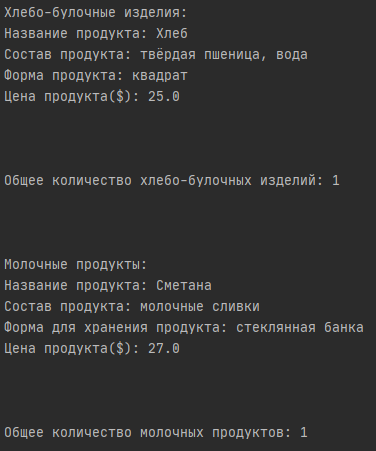


Рисунок 17 – Результат теста 5 из таблицы тестов 1



Рисунок 18 – Результат теста 6 из таблицы тестов 1

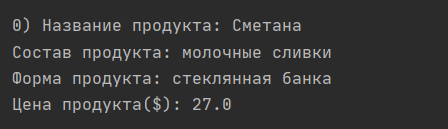


Рисунок 19 – Результат теста 6 из таблицы тестов 1



Рисунок 20 – Результат теста 7 из таблицы тестов 1

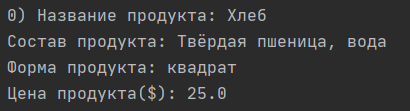


Рисунок 21 – Результат теста 7 из таблицы тестов 1



Рисунок 22 – Результат теста 8 из таблицы тестов 1



Рисунок 23 – Результат теста 8 из таблицы тестов 1



Рисунок 24 – Результат теста 9 из таблицы тестов 1



Рисунок 25 – Результат теста 9 из таблицы тестов 1



Рисунок 26 – Результат теста 10 из таблицы тестов 1



Рисунок 27 – Результат теста 10 из таблицы тестов 1



Рисунок 28 – Результат теста 11 из таблицы тестов 1



Рисунок 29 – Результат теста 11 из таблицы тестов 1



Рисунок 30 – Результат теста 12 из таблицы тестов 1



Рисунок 31 – Результат теста 12 из таблицы тестов 1



Рисунок 32 – Результат теста 13 из таблицы тестов 1



Рисунок 33 – Результат теста 13 из таблицы тестов 1

6 Исходный код

Класс Product:

package com.company;  
  
public abstract class Product {  
 protected double price; //цена продукта  
 protected String nameProduct; //название продукта  
 protected String compositionProduct; //состав продукта  
  
 public Product(){  
 price = 0;  
 nameProduct = "";  
 compositionProduct = "";  
 }  
 public boolean setPrice(double val) {  
 if(val <= 0)  
 return false;  
 this.price = val;  
  
 return false;  
 }  
  
 public double getPrice() {  
 return this.price;  
 }  
  
 public void setNameProduct(String n) {  
 this.nameProduct = n;  
 }  
  
 public String getNameProduct() {  
 return this.nameProduct;  
 }  
  
 public void setCompositionProduct(String c) {  
 this.compositionProduct = c;  
 }  
  
 public String getCompositionProduct() {  
 return compositionProduct;  
 }  
 public abstract String productCategory();{}  
}

Класс BakeryProductClass:

package com.company;  
  
public class BakeryProductClass extends Product{  
 private String form; //форма мучного продукта  
  
 public BakeryProductClass(){  
 super(); //вызов базового конструктора  
 form = "";  
 }  
 public void setFormProduct(String f){  
 this.form = f;  
 }  
  
 public String getFormProduct(){  
 return this.form;  
 }  
  
 @Override  
 public String productCategory() {  
 return "\nХлебо-булочные изделия: \n";  
 }  
}

Класс MilkProductClass:

package com.company;  
  
public class MilkProductClass extends Product{  
  
 private String storageForm; //форма для хранения молочных продуктов  
  
 public MilkProductClass(){  
 super();  
 storageForm = "";  
 }  
 public void setStorageFormProduct(String f){  
 this.storageForm = f;  
 }  
  
 public String getStorageFormProduct(){  
 return this.storageForm;  
 }  
  
 @Override  
 public String productCategory() {  
 return "\nМолочные продукты: \n";  
 }  
}

Класс Store:

package com.company;  
  
import java.util.ArrayList;  
import java.util.Scanner;  
  
//Магазин  
public class Store {  
 private ArrayList<BakeryProductClass> bakery = null; //хлебо-булочные изделия  
 private ArrayList<MilkProductClass> milk = null; //молочные продуктов  
  
 public Store(){  
 bakery = new ArrayList<>();  
 milk = new ArrayList <>();  
 }  
  
 public void Start(){  
 Scanner sc = new Scanner(System.in);  
 int number = 0;  
 int productNum = 0;  
  
 do{  
 System.out.print("\n\nВыберете пункт:\n");  
 System.out.print("1. Вывод всех продукты.\n");  
 System.out.print("2. Вывод мучных продуктов.\n");  
 System.out.print("3. Вывод молочных продуктов.\n");  
 System.out.print("4. Добавить хлебобулочный продукт.\n");  
 System.out.print("5. Добавить молочный продукт.\n");  
 System.out.print("6. Найти молочный продукт.\n");  
 System.out.print("7. Найти хлебобулочный продукт.\n");  
 System.out.print("8. Удалить молочный продукт.\n");  
 System.out.print("9. Удалить хлебобулочный продукт.\n");  
 System.out.print("10. Выход из программы.\n");  
  
 number = Main.userInput("Введите число: ");  
 while((number < 1) || (number > 10)){  
 number = Main.userInput("Введите число из предложенного списка ");  
 }  
  
 switch(number){  
 case 1:{  
 this.outputBakeryProduct();  
 this.outputMilksProduct();  
 break;  
 }  
 case 2:{  
 this.outputBakeryProduct();  
 break;  
 }  
 case 3:{  
 this.outputMilksProduct();  
 break;  
 }  
 case 4:{  
 this.addBakeryProduct();  
 break;  
 }  
 case 5:{  
 this.addMilkProduct();  
 break;  
 }  
 case 6:{  
 productNum = Main.userInput("Введите номер молочного продукта в списке: ");  
 this.findMilkProduct(productNum);  
 break;  
 }  
 case 7:{  
 productNum = Main.userInput("Введите номер хлебобулочного продукта в списке: ");  
 this.findBakeryProduct(productNum);  
 break;  
 }  
 case 8:{  
 productNum = Main.userInput("Введите номер молочного продукта в списке: ");  
 this.removeMilkProduct(productNum);  
 break;  
 }  
 case 9:{  
 productNum = Main.userInput("Введите номер хлебобулочного продукта в списке: ");  
 this.removeBakeryProduct(productNum);  
 break;  
 }  
 }  
  
 }while(number != 10);  
 }  
  
  
 public void findMilkProduct(int productNum) {  
 if(milk.size() == 0){  
 System.out.print("Молочных продуктов нет\n\n");  
 return;  
 }  
 System.out.print("\n");  
 if (milk.size()>=productNum) {  
 System.out.println(productNum + ")" + " Название продукта: " + milk.get(productNum).getNameProduct());  
 System.out.println("Состав продукта: " + milk.get(productNum).getCompositionProduct());  
 System.out.println("Форма продукта: " + milk.get(productNum).getStorageFormProduct());  
 System.out.println("Цена продукта($): " + milk.get(productNum).getPrice());  
 System.out.print("\n\n");  
 }  
 else {System.out.print("Молочного продукта под номером "+(productNum)+" не существует!"); Start();}  
  
 }  
 public void findBakeryProduct(int productNum) {  
 if(bakery.size() == 0){  
 System.out.print("Хлебо-булочных издений нет\n\n");  
 return;  
 }  
 System.out.print("\n");  
 if (bakery.size()>=productNum) {  
 System.out.println(productNum + ")" + " Название продукта: " + bakery.get(productNum).getNameProduct());  
 System.out.println("Состав продукта: " + bakery.get(productNum).getCompositionProduct());  
 System.out.println("Форма продукта: " + bakery.get(productNum).getFormProduct());  
 System.out.println("Цена продукта($): " + bakery.get(productNum).getPrice());  
 System.out.print("\n\n");  
 }  
 else {System.out.print("Хлебобулочного изделия под номером "+(productNum)+" не существует!");Start();}  
  
 }  
 public void removeMilkProduct(int productNum) {  
 if(milk.size() == 0){  
 System.out.print("Молочных продуктов нет\n\n");  
 return;  
 }  
 if (milk.size()>=productNum){  
 System.out.println("Запись под номером "+productNum+"'"+milk.get(productNum).getNameProduct()+"'"+" успешно удалена");  
 milk.remove(productNum);  
 }  
 else {System.out.print("Молочного продукта под номером "+(productNum)+" не существует!");Start();}  
 }  
  
 public void removeBakeryProduct(int productNum) {  
 if(bakery.size() == 0){  
 System.out.print("Хлебобулочных издений нет\n\n");  
 return;  
 }  
 if (bakery.size()>=productNum){  
 System.out.println("Запись под номером "+productNum+"'"+bakery.get(productNum).getNameProduct()+"'"+" успешно удалена");  
 bakery.remove(productNum);  
 }  
 else {System.out.print("Хлебобулочного изделия под номером "+(productNum)+" не существует!");Start();}  
 }  
  
  
  
 public void outputBakeryProduct(){  
 if(bakery.size() == 0){  
 System.out.print("Хлебо-булочных издений нет\n\n");  
 return;  
 }  
  
 for(int i = 0; i < bakery.size(); i++){  
 System.out.println(bakery.get(i).productCategory());  
 System.out.println(i+ ")"+" Название продукта: " + bakery.get(i).getNameProduct());  
 System.out.println("Состав продукта: " + bakery.get(i).getCompositionProduct());  
 System.out.println("Форма продукта: " + bakery.get(i).getFormProduct());  
 System.out.println("Цена продукта($): " + bakery.get(i).getPrice());  
 System.out.print("\n\n");  
 }  
  
 System.out.println("\nОбщее количество хлебо-булочных изделий: " + bakery.size() + "\n\n");  
 }  
  
 public void outputMilksProduct(){  
 if(milk.size() == 0){  
 System.out.print("Молочных продуктов нет\n\n");  
 return;  
 }  
  
 for(int i = 0; i < milk.size(); i++){  
 System.out.println(milk.get(i).productCategory());  
 System.out.println(i+ ")"+" Название продукта: " + milk.get(i).getNameProduct());  
 System.out.println("Состав продукта: " + milk.get(i).getCompositionProduct());  
 System.out.println("Форма для хранения продукта: " + milk.get(i).getStorageFormProduct());  
 System.out.println("Цена продукта($): " + milk.get(i).getPrice());  
 System.out.print("\n\n");  
 }  
  
 System.out.println("\nОбщее количество молочных продуктов: " + milk.size() + "\n\n");  
 }  
  
 public void addBakeryProduct(){  
 System.out.print("\n\nДобавление мучного изделия... \n\n");  
 BakeryProductClass product = new BakeryProductClass();  
  
 Scanner sc = new Scanner(System.in);  
 System.out.print("Введите название продукта: ");  
 product.setNameProduct(sc.nextLine());  
  
 System.out.print("Введите состав продукта: ");  
 product.setCompositionProduct(sc.nextLine());  
  
 System.out.print("Введите форму продукта: ");  
 product.setFormProduct(sc.nextLine());  
  
 System.out.print("Введите цену продукта($): ");  
 product.setPrice(sc.nextDouble());  
  
 bakery.add(product);  
 System.out.print("\nПродукт добавлен в магазин.");  
 }  
  
 public void addMilkProduct(){  
 System.out.print("\n\nДобавление молочного продукта... \n\n");  
  
 MilkProductClass product = new MilkProductClass();  
  
 Scanner sc = new Scanner(System.in);  
 System.out.print("Введите название продукта: ");  
 product.setNameProduct(sc.nextLine());  
  
 System.out.print("Введите состав продукта: ");  
 product.setCompositionProduct(sc.nextLine());  
  
 System.out.print("Введите форму для хранения продукта: ");  
 product.setStorageFormProduct(sc.nextLine());  
  
 System.out.print("Введите цену продукта($): ");  
 product.setPrice(sc.nextDouble());  
  
 milk.add(product);  
 System.out.print("\nПродукт добавлен в магазин.");  
 }  
}

Класс Main:

package com.company;  
  
  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Store store = new Store();  
 store.Start();  
 }  
 public static int userInput(String message) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print(message);  
 int i;  
 while (true) {  
 try {  
 i = Integer.*parseInt*(sc.next());  
 if (i<0)  
 {  
 throw new NumberFormatException();  
 }  
 break;  
 } catch (NumberFormatException e) {  
 System.*out*.println("Введите целое положительное число!");  
 }  
 }  
 return i;  
 }  
  
}

7 Заключение

Были изучены механизмы наследования, полиморфизма, инкапсуляции на практике. Были получены практические навыки описания иерархии классов и организации работы с объектами-членам классов-“групп”. Получены практические навыки использования абстрактных классов и переопределения унаследованных методов.

Список использованных источников

1. Объектно-ориентированное программирование. Метод. указания по

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

ИРНИТУ, 2017. – 24 c.