ОДЕССКИЙ НАЦИОНАЛЬНЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИНСТИТУТ КОМПЬЮТЕРНЫХ СИСТЕМ
Кафедра «Информационные технологии»

Лабораторная работа №2

по дисциплине: «Объектно-ориентированное программирование»

на тему: «Основа работы с классами и объектами. Инкапсуляция Конструктор. Модификаторы доступа»

Вариант 13

Выполнил:
ст. гр. НАД-191
Краковский В.А.
Проверили:
д. Рудниченко М.Д.
ст. пр. Павлов О.А.

Одесса 2020

Оглавление

ВВЕДЕНИЕ.....3
ТЕОРЕТИЧЕСКАЯ ЧАСТЬ.....4
ПРАКТИЧЕСКАЯ ЧАСТЬ.....8
ВЫВОД.....16
ЛИТЕРАТУРА.....17

ВВЕДЕНИЕ

Цель лабораторной работы:

- Ознакомиться с основами написания классов;
- Освоить инкапсуляцию классов;
- Разобраться с использованием конструктора;
- Научиться использовать модификаторы доступа.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Объект – структура, которая объединяет данные и методы, которые эти данные обрабатывают. Это позволят разграничить область применения методов. **Объект** – это строительный блок объектно-ориентированных программ. Объектно-ориентированная программа является, по сути, набором объектов.

Объект состоит из трех частей:

- имя объекта;
- состояние (данные объекта, переменные состояния). Состояние объекта характеризуется перечнем всех свойств данного объекта и текущими значениями каждого из этих свойств;
- методы (операции).

Данные объектов. Данные, содержащиеся в объекте, представляют его состояние. В терминологии ООП, эти данные называются атрибутами. Например, атрибутами работника могут быть имя, фамилия, пол, дата рождения, номер телефона и т.д. В разных объектах атрибуты имеют разное значение.

Поведение объектов. Поведение объекта – то, что он может сделать (в структурном программировании это реализовывалось функциями, процедурами, подпрограммами).

Сообщения – механизм коммуникации между объектами. Например, когда объект A вызывает метод объекта B, объект A отправляет сообщение объекту B. Ответ объекта B определяется его возвращаемым значением. Только «открытые» методы могут вызываться другим объектом.

Каждый объект определяется общим шаблоном, который называется классом. В рамках класса задается общий шаблон, структура, на основе которой затем создаются объекты. Данные, относящиеся к классу, называются полями

класса, а программный код для их обработки – методами класса. Поля и методы иногда называют общим термином – члены класса.

В классе описываются, какого типа данные относятся к классу, а также то, какие методы применяются к этим данным. Затем, в программе на основе того или иного класса создается экземпляр класса (объект), в котором указываются конкретные значения полей и выполняются необходимые действия над ними.

Согласно конвенциям кода Java:

- каждый класс должен содержаться в своем отдельном файле с расширением .java;
- название файла должно совпадать с названием класса;
- класс должен быть именем существительным;
- имя класса должно его описывать;
- имя класса начинается с большой буквы;
- если имя состоит из нескольких слов, то каждое слово начинается с большой буквы.

Конструктор – это специальный метод, который вызывается при создании нового объекта. Синтаксис конструктора отличается от синтаксиса обычного метода. Его имя совпадает с именем класса, в котором он находится, и он не имеет возвращаемого типа.

Инкапсуляция – один из основополагающих принципов ООП. Инкапсуляция – это одна из причин, почему так широко используется ООП.

Инкапсуляцию можно считать защитной оболочкой, которая предохраняет код и данные от произвольного доступа со стороны другого кода, находящегося снаружи оболочки. Доступ к коду и данным, находящимся внутри оболочки, строго контролируется тщательно определенным интерфейсом (набором общедоступных, публичных методов).

В любом классе присутствуют две части: интерфейс и реализация.

Интерфейс отражает внешнее поведение объектов этого класса. Внутренняя реализация описывает представления и механизмы достижения желаемого поведения объекта.

В интерфейсе собрано все, что касается взаимодействия данного объекта с другими объектами, а реализация скрывает от других объектов все детали, не имеющие отношения к процессу взаимодействия объектов.

Инкапсуляция – это процесс отделения друг от друга элементов объекта, определяющих его устройство и поведения; инкапсуляция служит для того, чтобы изолировать контрактные обязательства от их реализации.

Инкапсуляция позволяет локализовать части реализации системы, которые могут подвергнуться изменениям. По мере развития программы, разработчики могут принять решение изменить внутреннее устройство тех или иных объектов с целью улучшения производительности или экономии памяти. Но интерфейс будет нетронутым и позволит другим объектам таким же способом взаимодействовать с этим объектом. (Пример автомобиля – педали, руль, приборная панель и внутренняя начинка).

Инкапсуляция в Java реализована с помощью использования модификаторов доступа.

Язык Java предоставляет несколько уровней защиты, которые позволяет настраивать область видимости данных и методов. В Java имеется четыре категории видимости элементов класса:

- ***private*** – члены класса доступны только членам данного класса;
- по умолчанию (package-private) – члены класса доступны классам, которые находятся в этом же пакете;
- ***protected*** – члены класса доступны классам, находящимся в том же пакете, и подклассам – в других пакетах;
- ***public*** – члены класса доступны для всех классов в этом и других пакетах.

Член класса (переменная, конструктор, методы), объявленный ***public***, доступен из любого метода вне класса.

Всё что объявлено ***private***, доступно только конструкторам и методам внутри класса и нигде больше. Они выполняют служебную или вспомогательную роль в пределах класса и их функциональность не предназначена для внешнего пользования. Закрытие (***private***) полей обеспечивает инкапсуляцию.

ПРАКТИЧЕСКАЯ ЧАСТЬ

1. Условие

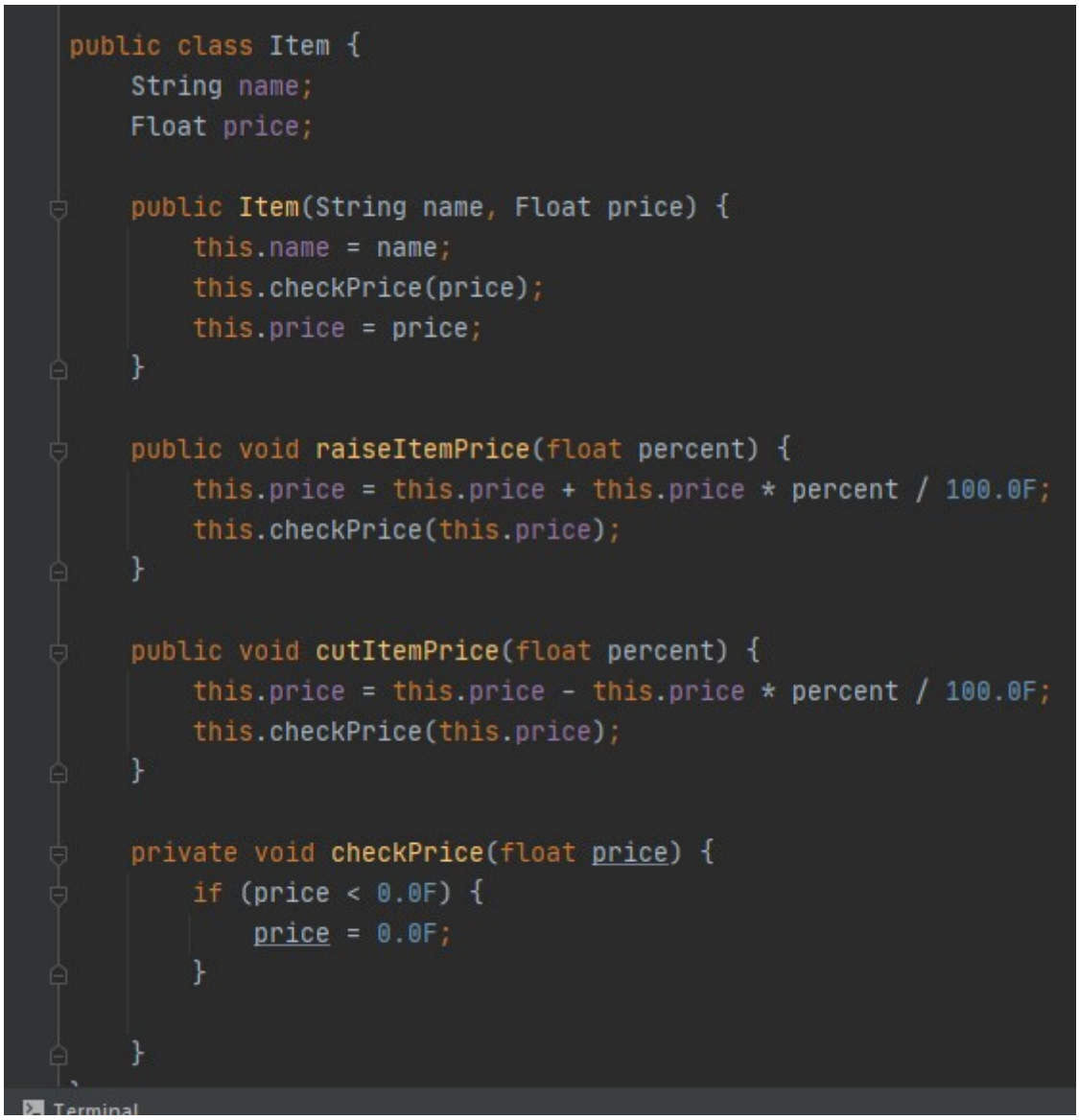
1. Создайте класс «Товар» (Item).
А) Класс «Товар» должен содержать следующие поля: наименование (строка), цена (float).
Б) Класс «Товар» должен содержать конструктор, который принимает два параметра: наименование и начальную цену товара.
В) Класс «Товар» должен иметь следующие публичные методы: поднятие цены на определенный процент (значение процента типа float передается как аргумент метода); снижение цены на определенный процент (значение процента типа float передается как аргумент метода).
Г) В классе должна быть реализована проверка цены на отрицательное значение. Если в конструкторе передается отрицательное значение цены, либо в результате снижения цены на >100% цена становится отрицательной, она должна быть принудительно установлена в 0.

2. Создайте класс «Корзина» (Cart).
А) **«Корзина» должна реализовывать структуру данных стек**, в котором содержатся объекты класса «товар». Будем считать, что мы добавляем всегда по 1 единице товара. Стек «внутри» реализуется обычным массивом объектов Item. Класс должен содержать проверки, связанные с работой стека – переполнение стека, попытка извлечь элемент из пустого стека и т.д.
Б) «Корзина» должна содержать конструктор с 1 параметром – максимальным количеством элементов в стеке. В этом конструкторе происходила инициализация массива, который реализует стек.
В) «Корзина» должна содержать следующие публичные методы: добавление товара, удаление товара, подсчет суммы цен товаров в корзине (пройтись по элементам массива и сложить все значения цен), повышение и понижение цен всех товаров в стеке (два отдельных метода, значение цены передается как параметр метода. Необходимо пройтись по всем элементам массива и передать соответствующее сообщение объектам.

3. В методе Main необходимо создать объект класса «Корзина» с некоторым максимальным количеством элементов в стеке.
А) Заполнить корзину объектами класса Item (5-6 объектов будет достаточно);
Б) Вывести сумму цен товаров внутри корзины;
В) Поднять цены в корзине на 15%, вывести измененную сумму цен в консоль.
Г) Снизить цены в корзине на 30%, вывести измененную сумму цен в консоль.

Рисунок 1: Условие задания

2. Реализация класса Item



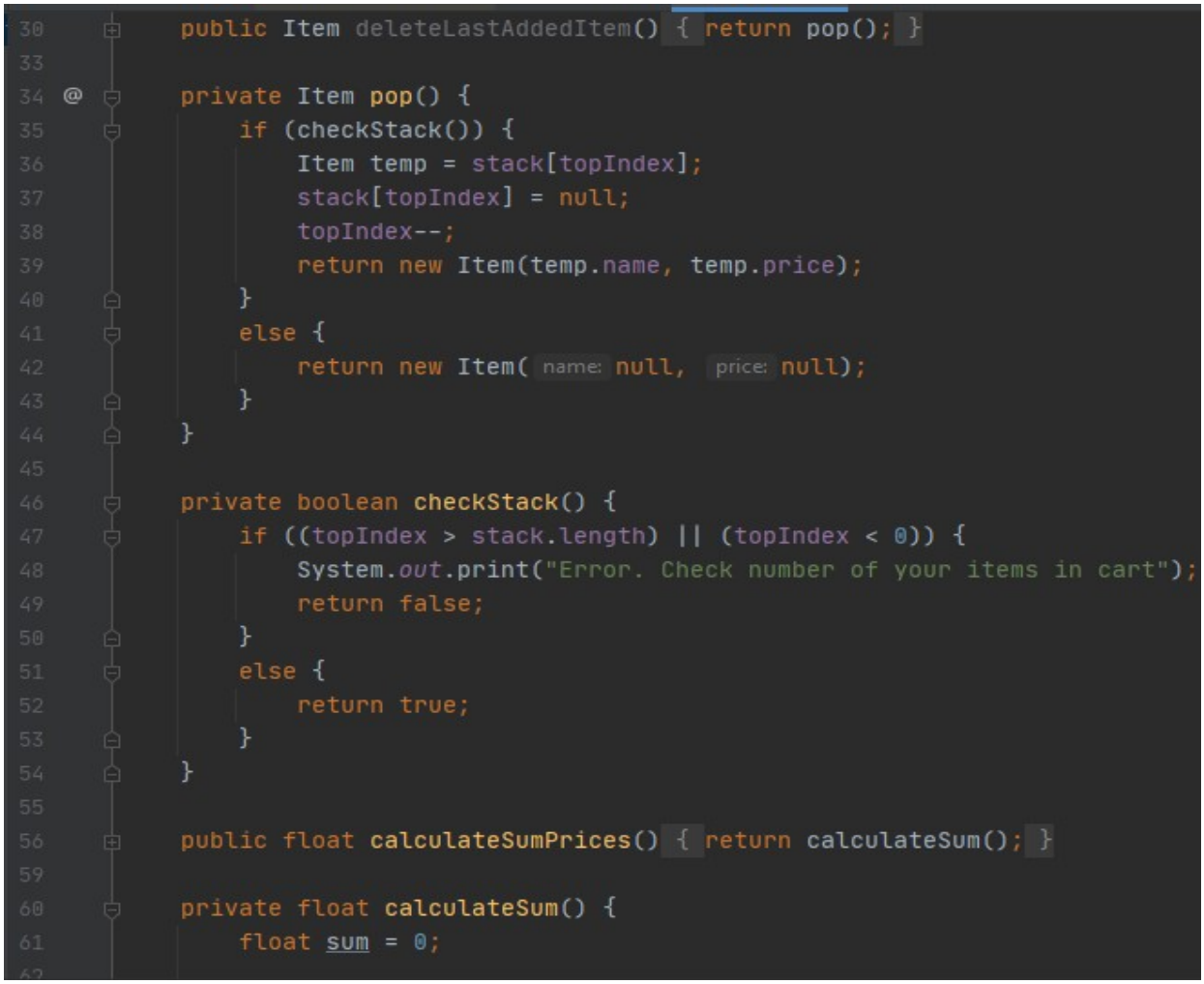
```
public class Item {
    String name;
    Float price;

    public Item(String name, Float price) {
        this.name = name;
        this.checkPrice(price);
        this.price = price;
    }

    public void raiseItemPrice(float percent) {
        this.price = this.price + this.price * percent / 100.0F;
        this.checkPrice(this.price);
    }

    public void cutItemPrice(float percent) {
        this.price = this.price - this.price * percent / 100.0F;
        this.checkPrice(this.price);
    }

    private void checkPrice(float price) {
        if (price < 0.0F) {
            price = 0.0F;
        }

    }
```

Рисунок 2: Конструктор и интерфейс класса Item

3. Реализация класса Cart:

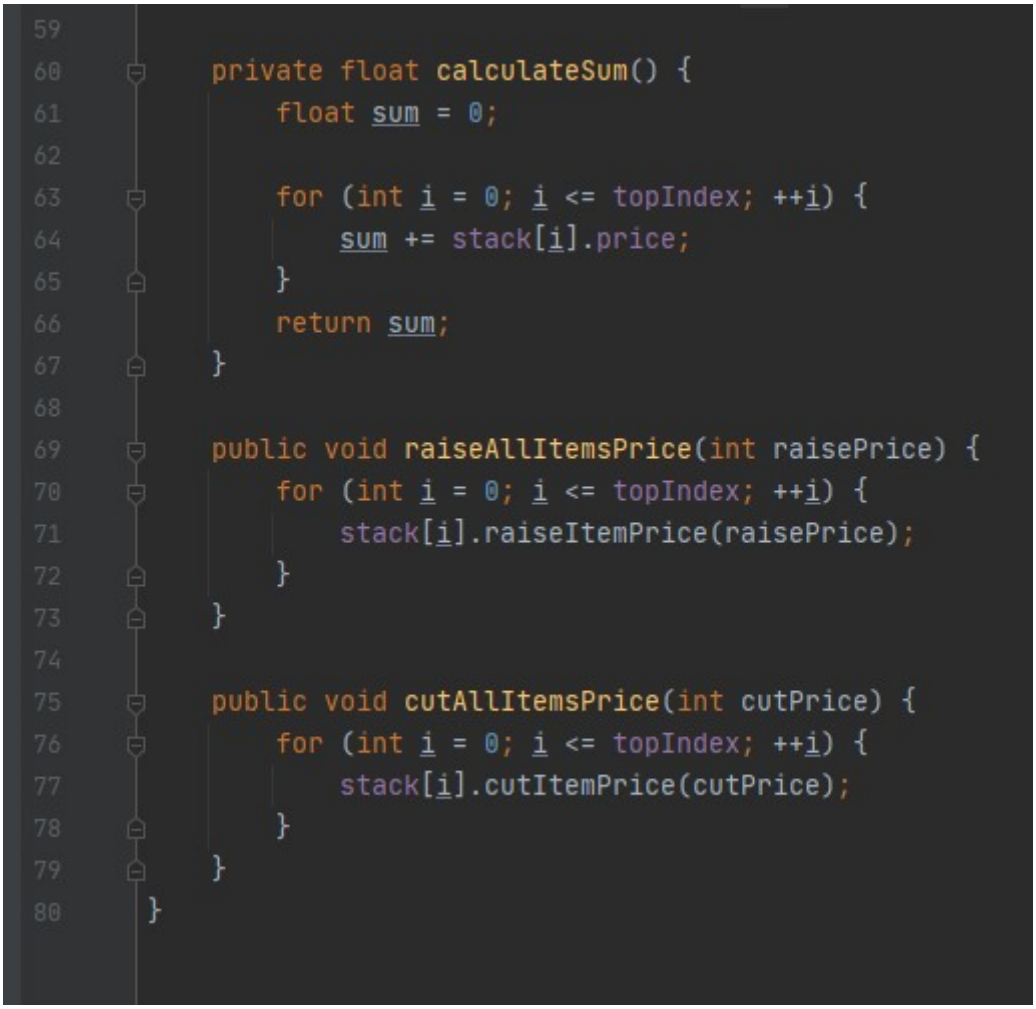
```java
package com.company;

public class Cart {
    private Item[] stack;
    private int topIndex;

    public Cart(int capacity) {
        stack = new Item[capacity];
        topIndex = -1;
    }

    public boolean addItem(Item item) { return push(item); }

    private boolean push(Item item) {
        topIndex++;
        if (checkStack()) {
            stack[topIndex] = item;
            System.out.println("Item added successfully!");
            return true;
        }
        else {
            topIndex--;
            System.out.println("Error. Try again.");
            return false;
        }
    }

    public Item deleteLastAddedItem() { return pop(); }
```

Рисунок 3: Класс Cart: конструктор, методы добавить\удалить Item



```
30      public Item deleteLastAddedItem() { return pop(); }
33
34  @   private Item pop() {
35          if (checkStack()) {
36              Item temp = stack[topIndex];
37              stack[topIndex] = null;
38              topIndex--;
39              return new Item(temp.name, temp.price);
40          }
41          else {
42              return new Item( name: null,  price: null);
43          }
44      }
45
46      private boolean checkStack() {
47          if ((topIndex > stack.length) || (topIndex < 0)) {
48              System.out.print("Error. Check number of your items in cart");
49              return false;
50          }
51          else {
52              return true;
53          }
54      }
55
56      public float calculateSumPrices() { return calculateSum(); }
59
60      private float calculateSum() {
61          float sum = 0;
```

Рисунок 4: Класс Cart: Удаление Item'а, проверка Stack'а, подсчёт цены



```
59
60      private float calculateSum() {
61          float sum = 0;
62
63          for (int i = 0; i <= topIndex; ++i) {
64              sum += stack[i].price;
65          }
66          return sum;
67      }
68
69      public void raiseAllItemsPrice(int raisePrice) {
70          for (int i = 0; i <= topIndex; ++i) {
71              stack[i].raiseItemPrice(raisePrice);
72          }
73      }
74
75      public void cutAllItemsPrice(int cutPrice) {
76          for (int i = 0; i <= topIndex; ++i) {
77              stack[i].cutItemPrice(cutPrice);
78          }
79      }
80  }
```

Рисунок 5: Класс Cart: Подсчёт суммы, Повышение, уменьшение цены.

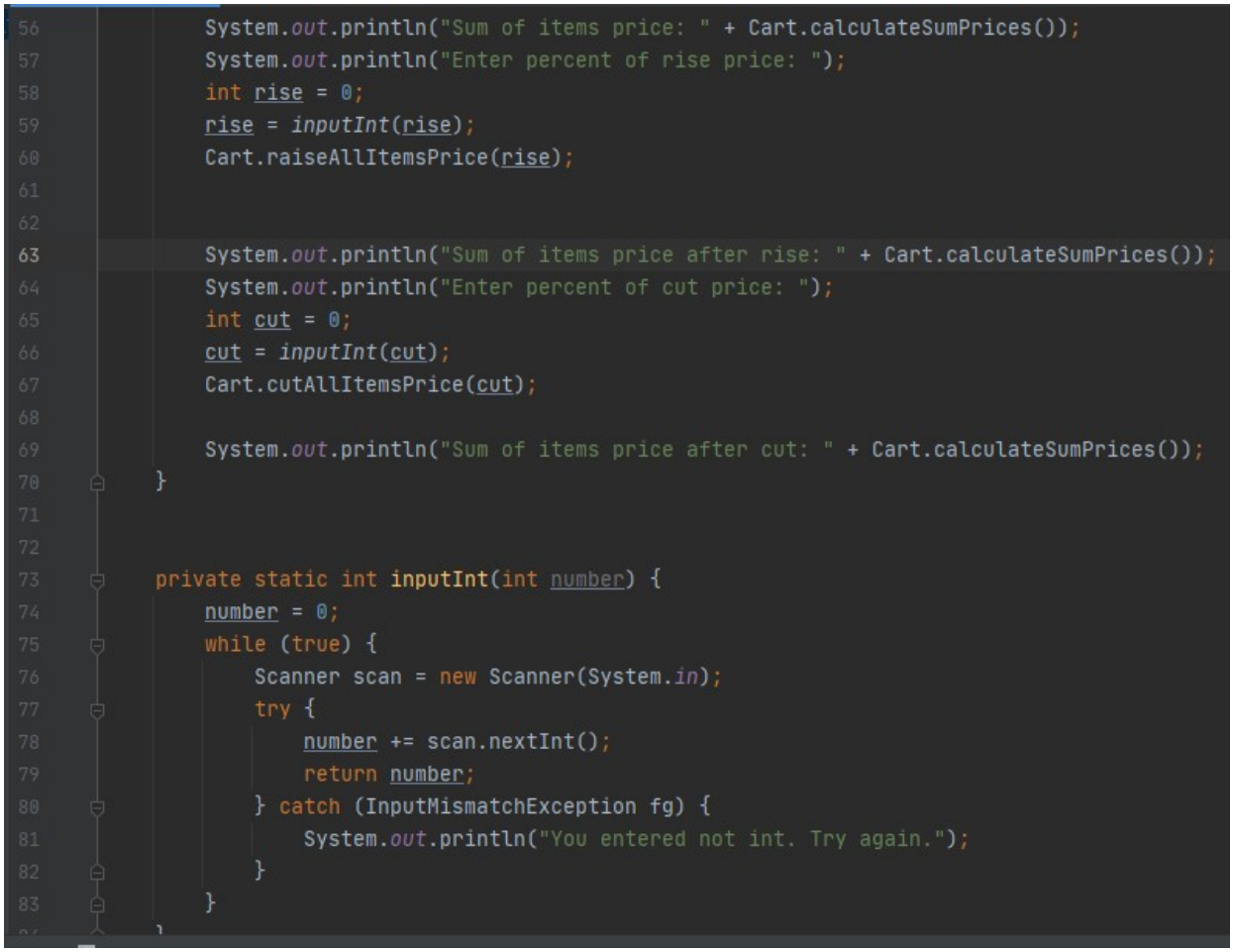
4. Реализация класса Main:

```java
package com.company;

import java.util.InputMismatchException;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        int itemsNumber;
        int capacityCart;

        while (true) {
            itemsNumber = 0;
            capacityCart = 0;
            System.out.println("Max number of Items in Cart:");
            capacityCart = inputInt(capacityCart);

            System.out.println("How many Items you want to add in Cart?");
            itemsNumber += inputInt(itemsNumber);

            if (itemsNumber <= capacityCart) {
                break;
            }
            else {
                System.out.println("Items can't add to the Cart. Try again.");
            }
        }

```

Рисунок 6: Ввод и проверка исходных данных.

```
29
30          Cart Cart = new Cart(capacityCart);
31          for (int i = 0; i < itemsNumber; ++i) {
32              while (true) {
33                  String name;
34                  float price = 0;
35
36                  System.out.println("Item #" + i + " name:");
37                  Scanner newName = new Scanner(System.in);
38                  name = newName.nextLine();
39
40                  System.out.println("Item #" + i + " price:");
41                  Scanner newPrice = new Scanner(System.in);
42
43                  try {
44                      price += newPrice.nextFloat();
45
46                      Item item = new Item(name, price);
47                      Cart.addItem(item);
48                      break;
49                  } catch (InputMismatchException fg) {
50                      System.out.println("You enter not Float");
51                  }
52              }
53          }
```

Рисунок 7: Создание и заполнение объекта класса Cart



```
        System.out.println("Sum of items price: " + Cart.calculateSumPrices());
        System.out.println("Enter percent of rise price: ");
        int rise = 0;
        rise = inputInt(rise);
        Cart.raiseAllItemsPrice(rise);


        System.out.println("Sum of items price after rise: " + Cart.calculateSumPrices());
        System.out.println("Enter percent of cut price: ");
        int cut = 0;
        cut = inputInt(cut);
        Cart.cutAllItemsPrice(cut);

        System.out.println("Sum of items price after cut: " + Cart.calculateSumPrices());
    }


    private static int inputInt(int number) {
        number = 0;
        while (true) {
            Scanner scan = new Scanner(System.in);
            try {
                number += scan.nextInt();
                return number;
            } catch (InputMismatchException fg) {
                System.out.println("You entered not int. Try again.");
            }
        }
    }
```

Рисунок 8: Вызов методов класса Cart и реализация метода проверки вводимых целочисленных значений

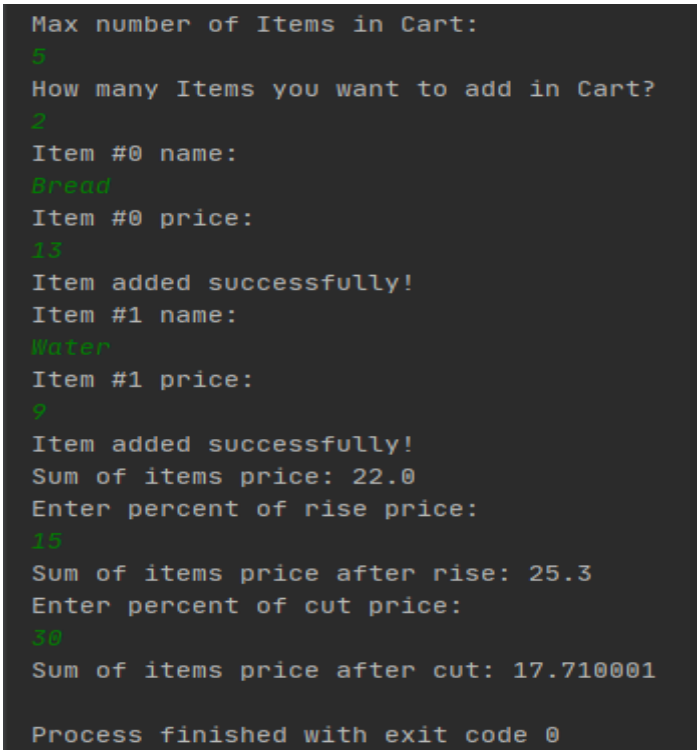


Рисунок 9: Запуск и тестовые данные

ВЫВОД

На этой лабораторной работе я узнал каким образом создавать объекты и их прототипы при помощи классов и конструкторов, а также работать с их свойствами, благодаря реализованным методам. Рассмотрено и изучено основное понятие ООП — Инкапсуляция.

ЛИТЕРАТУРА

1. Рудниченко Н.Д. - Учебное пособие по ООП — [Электронный доступ] - https://drive.google.com/drive/folders/1a-K_UOYEY-UiDUm3MSUFdN5k2-Zrb7yt