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](#__RefHeading___Toc131_1832763852)

[​ ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 4](#__RefHeading___Toc133_1832763852)

[​ ПРАКТИЧЕСКАЯ ЧАСТЬ 8](#__RefHeading___Toc135_1832763852)

[​ ВЫВОД 16](#__RefHeading___Toc137_1832763852)

[​ ЛИТЕРАТУРА 17](#__RefHeading___Toc139_2247490813)

# 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 А 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:

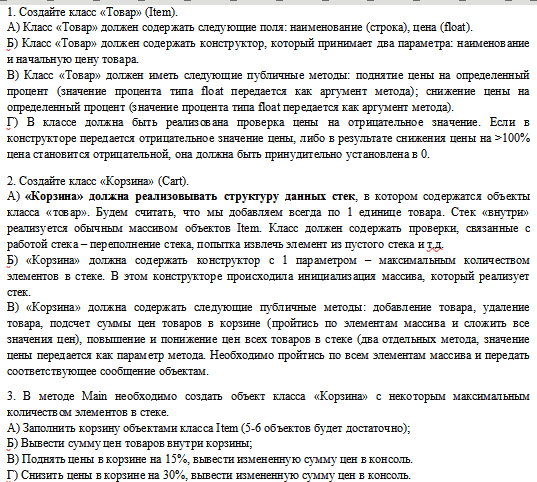
* ***private*** – члены класса доступны только членам данного класса;
* по умолчанию (package-private) – члены класса доступны классам, которые находятся в этом же пакете;
* ***protected*** – члены класса доступны классам, находящимся в том же пакете, и подклассам – в других пакетах;
* ***public*** –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

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

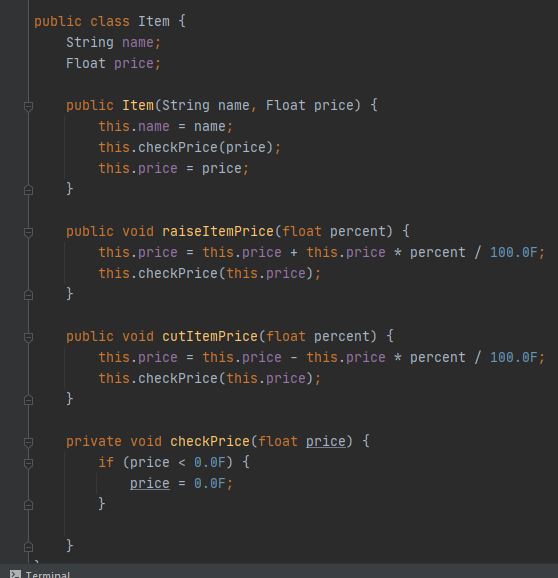
Всё что объявлено ***private***, доступно только конструкторам и методам внутри класса и нигде больше. Они выполняют служебную или вспомогательную роль в пределах класса и их функциональность не предназначена для внешнего пользования. Закрытие (***private***) полей обеспечивает инкапсуляцию.

# ПРАКТИЧЕСКАЯ ЧАСТЬ

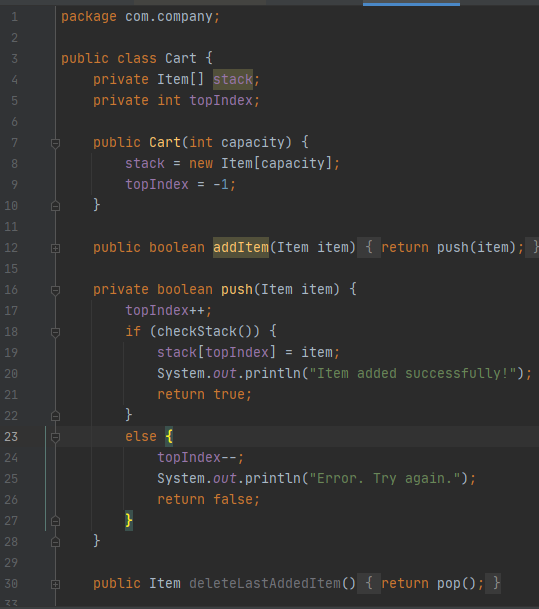
1. Условие

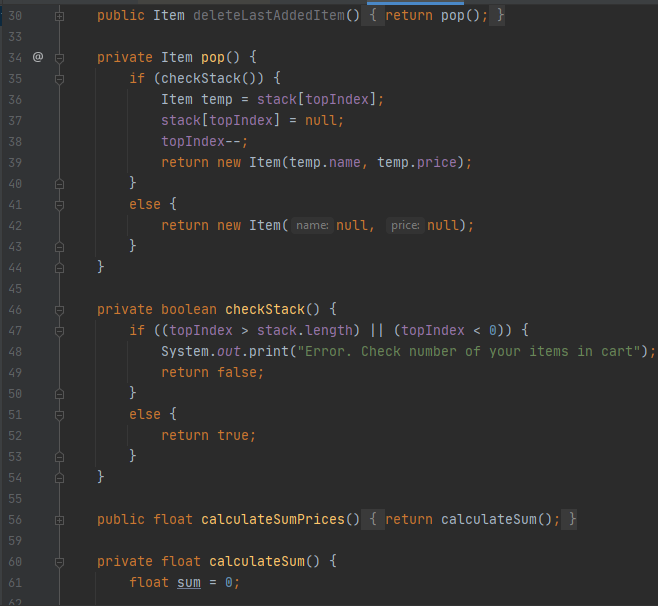
Рисунок 1: Условие задания

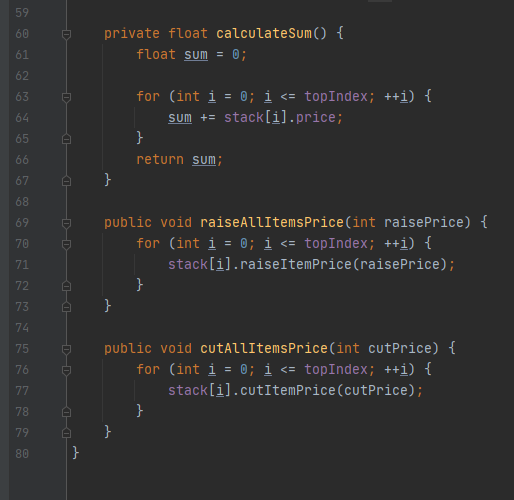
1. Реализация класса Item

Рисунок 2: Конструктор и интерфейс класса Item

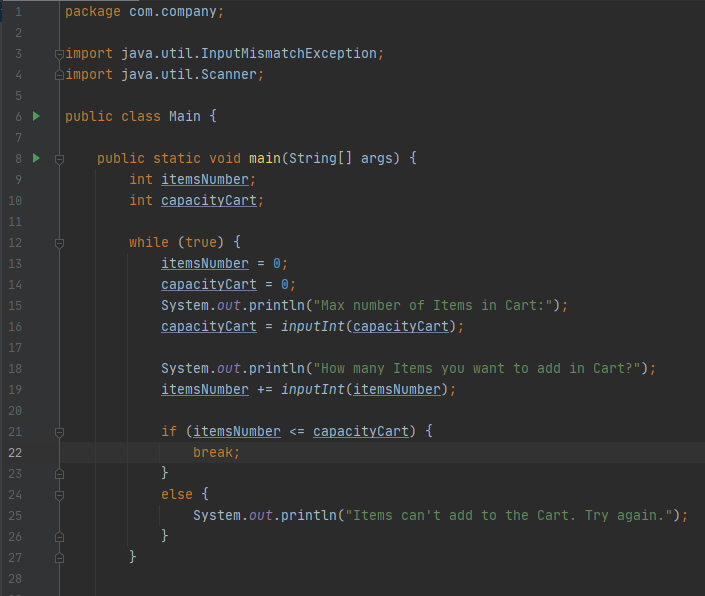
1. Реализация класса Cart:

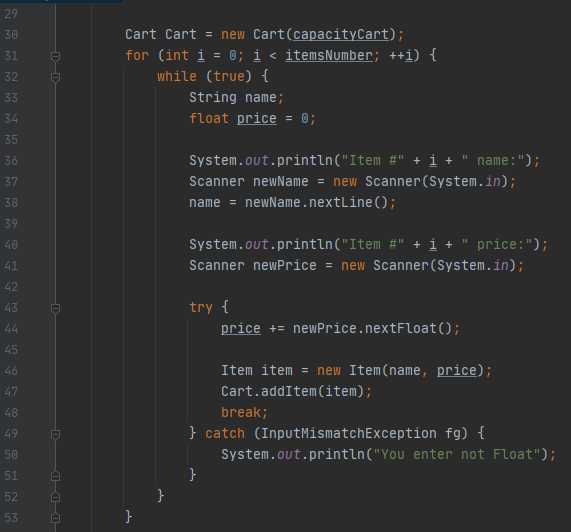
Рисунок 3: Класс Cart: конструктор, методы добавить\удалить Item

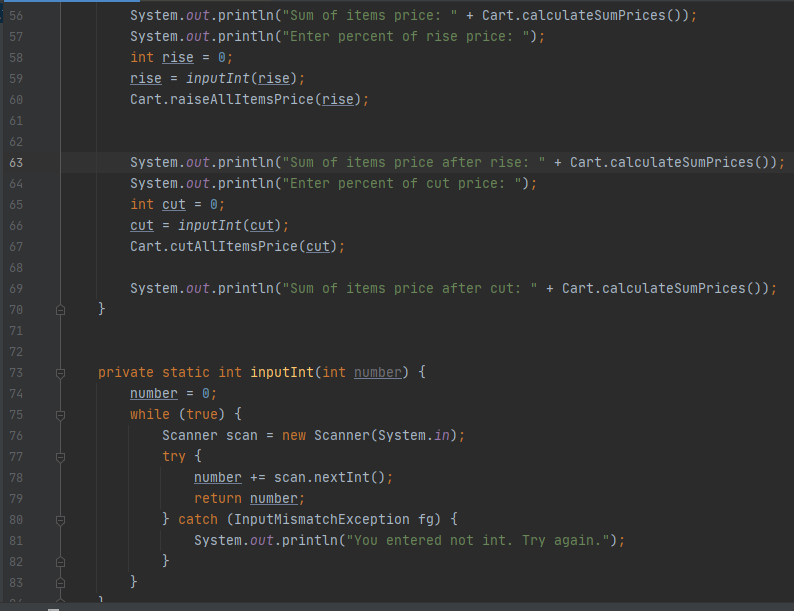
Рисунок 4: Класс Cart: Удаление Item'a, проверка Stack'a, подсчёт цены

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

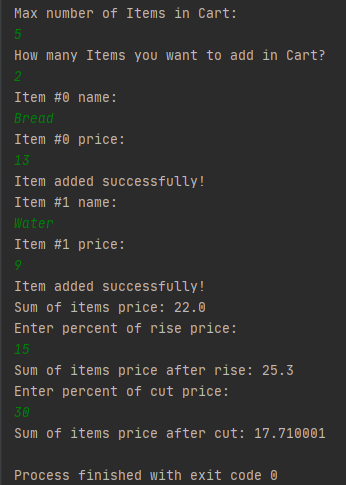
1. Реализация класса Main:

Рисунок 6: Ввод и проверка исходных данных.

Рисунок 7: Создание и заполнение объекта класса Cart

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

# ВЫВОД

Рисунок 9: Запуск и тестовые данные

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>