Министерство цифрового развития, связи и массовых коммуникаций

Российской Федерации

Ордена Трудового Красного Знамени

Федерально государственное бюджетное образовательное учреждение

высшего образования

МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ

Кафедра «Математической кибернетики и информационных технологий»

Лабораторная работа №2

По дисциплине

«Информационные технологии и программирование»

Выполнил:

Студент группы БВТ2203

Бородин К.Н.

Москва 2023

ВВЕДЕНИЕ

Абстракция – процесс выделения и описания ключевых характеристик объекта, игнорируя детали, не являющиеся существенными для конкретной задачи.

Инкапсуляция – принцип ограничение доступа к внутренним данными и методам объекта, предоставляя контролируемый интерфейс для взаимодействия с ним.

Полиморфизм – принцип, позволяющий объектам разных классов реагировать на одни и те же вызовы методом специфичным для себя образом.

Наследование – механизм, позволяющий создавать новый класс на основе предыдущего.

Абстрактный класс – класс, объект которого не может быть создан напрямую, но может содержать абстрактные методы, которые могут использоваться в его подклассах.

Поле – переменная, хранящая данные или состояние объекта одного класса.

Метод – функция, определённая внутри класса и выполняющая действия над данными этого класса.

Модификаторы доступа определяют уровень доступа к классам, полям и методам внутри класса. Существуют следующие модификаторы: public, private, protected, default.

Public – член класса, доступный из любой части программы. (геттеры и сеттеры)

Private – член класса, доступный только внутри самого класса. (внутренние детали реализации класса)

Protected – член класса, доступный внутри класса и внутри его подклассов. (когда нужен доступ только для подклассов и классов в том же пакете)

Default – член класса, область видимости которого ограничено областью видимости внутри того же пакте.

Перегрузка – создание нескольких методов с одинаковыми именами внутри одного класса, но с разными параметрами.

Переопределение – создание новой реализации метода в подклассе, который уже определен в его суперклассе.

Статическая переменная – переменная, являющаяся общей для всех экзамеляров класса.К ней можно обратиться через имя класса, без создания объекта этого класса.

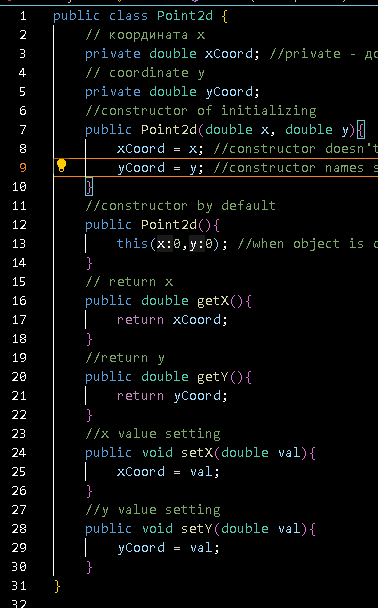
Цель работы: познакомиться с основными концепциями и принципами ООП на языке программирования Java.

Задания:

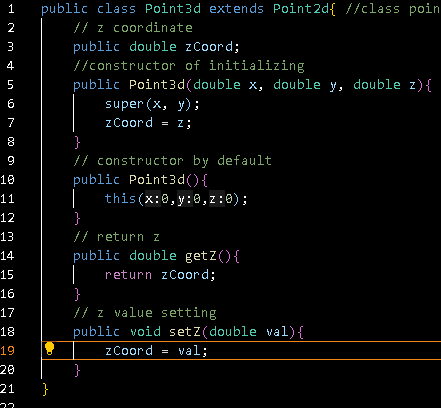
1. Создать код просто класса, представляющего двумерную точку.
2. Создать экземпляр класса.
3. Реализовать класс трёхмерной точки, используя класс двумерной точки.
4. Создать иерархию класса, где базовым классом является транспортное средство, а дочерними являются легковой автомобиль, грузовой автомобиль и мотоцикл. Иерархия должна содержать:
   * 1. Абстрактный класс.
     2. 2 уровня наследуемых классов (классы должны содержать в себе минимум 3 поля и 2 метода, описывающих поведения объекта).
     3. Демонстрацию реализации всех принципов ООП (абстракция, модификаторы доступа, перегрузка, переопределение)
     4. Наличие конструкторов (в том числе по умолчанию)
     5. Наличие геттеров и сеттеров.
     6. Ввод/вывод информации о создаваемых объектах
     7. Предусмотрение в одном из классов создания счётчика созданных объектов с использованием статической переменной
     8. Демонстрация работы.

ХОД РАБОТЫ.

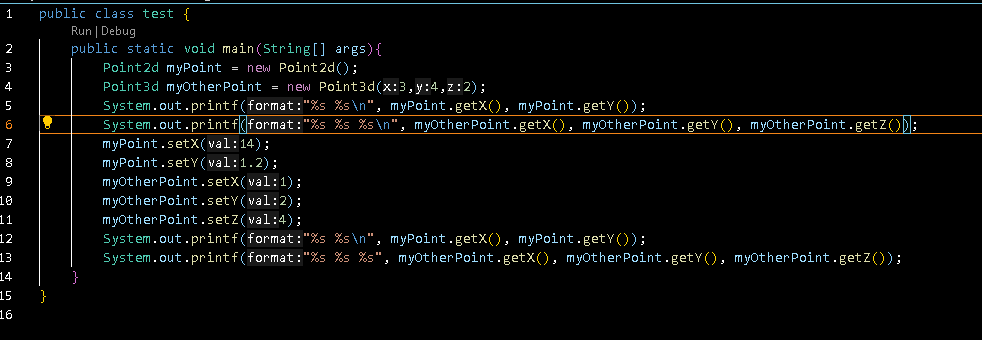
Создаю класс двумерной точки, в котором есть две переменные, перегрузка инициализаторами, два сеттера и две геттера, рисунок 1.

  
Рисунок 1 – создание класса двумерной точки.

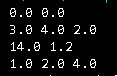
Создаю дочерний класс, у которого будет на одну переменную больше. Инициализация двух переменных будет происходить с помощью наследования от родительского класса при задании этих точек. Инициализаторы перегружены. Для новой переменной добавляю геттер и сеттер, рисунок 2.

  
Рисунок 2 – создание дочернего класса.

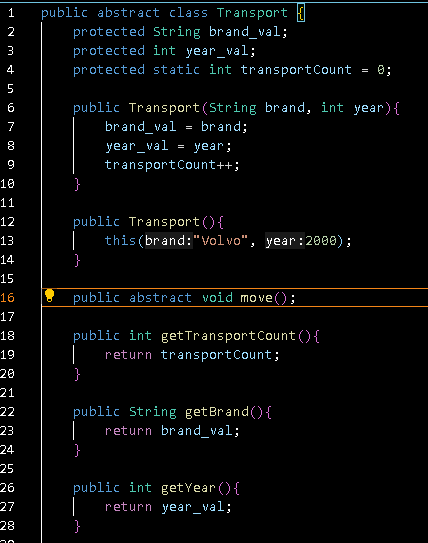
Демонстрируют работы. Для этого создаю родительский класса с не заданными параметрами и дочерний с заданными, с помощью геттеров получаю значения полей этого класса, затем с помощью сеттеров меняю их и вновь с помощью геттеров получаю эти значения, рисунок 3.

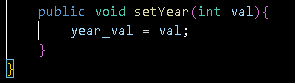
  
Рисунок 3 – демонстрация работы.

Вывод программы представлен на рисунке 4.

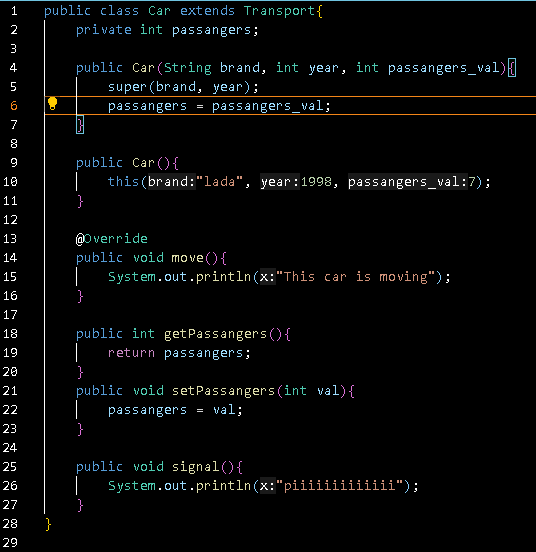
  
Рисунок 4 – вывод программы.

Создаю абстрактный класс транспорта. В нём будут два поля для бренда и года, а также статическая переменная для числа созданных объектов. Инициализаторы перегружены для разных вариантов создания экземпляра класса. Созданы метод движения, выводящий характерное для этого класса сообщение, геттеры для вывода значения, сеттера для ввода, рисунки 5-6.

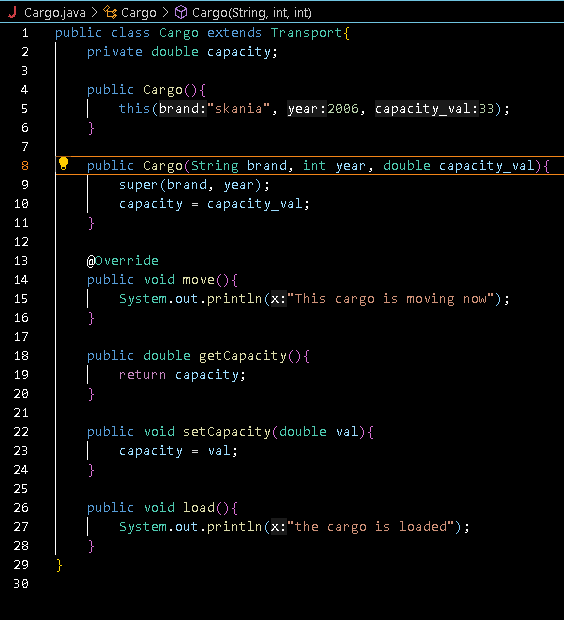
  
Рисунок 5 – создание абстрактного класса.

  
Рисунок 6 – создание абстрактного класса.

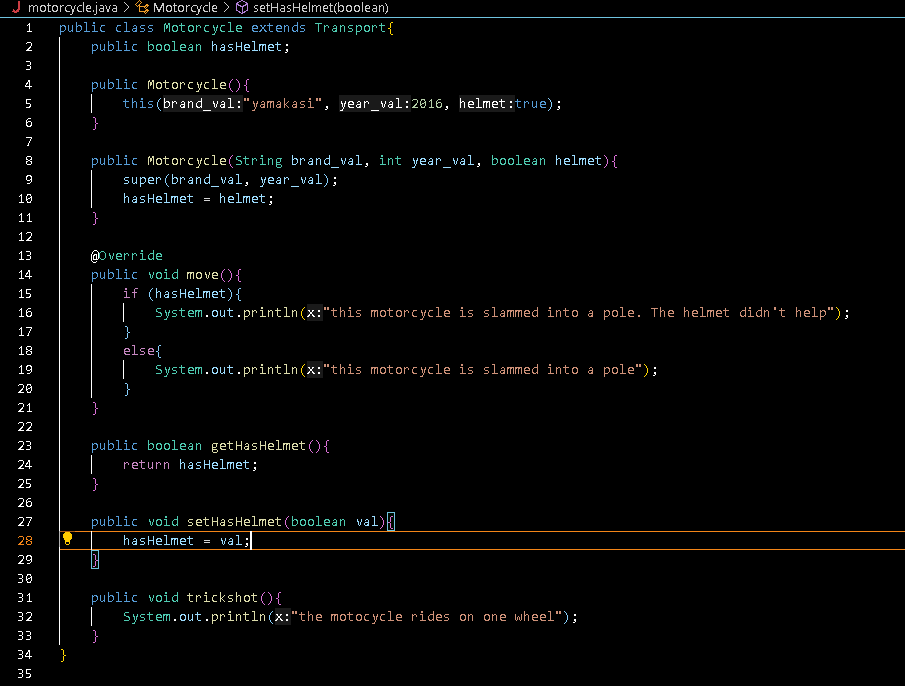
Создаю дочерний класс для легкового автомобиля. Дочерний класс будет иметь дополнительное поле для количества пассажиров. Перегружаю инициализаторы для разных вариантов создания экземпляр, переопределяю метод движения для дочернего класса, для нового поля класса создаю геттер и сеттер, а также создаю характерно поле для класса, рисунок 7.

  
Рисунок 7 – создание дочернего класса.

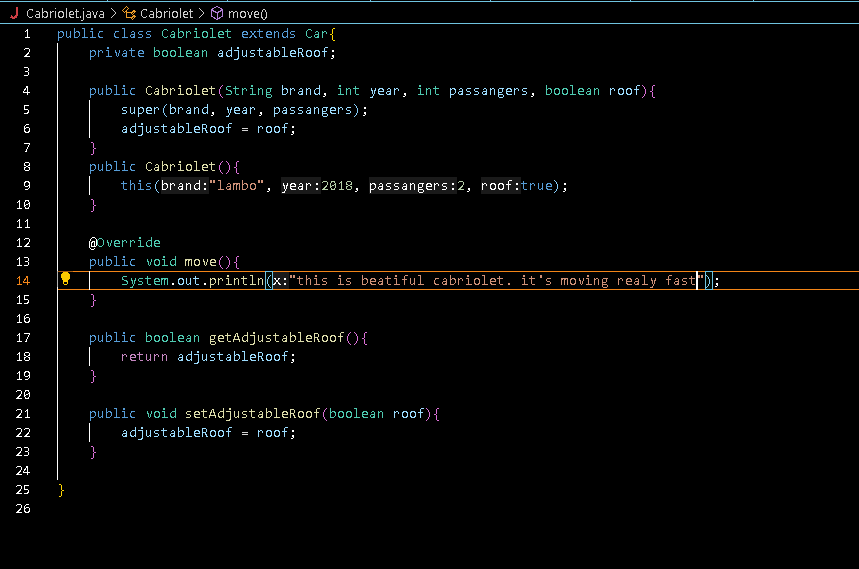
Создаю дочерний класс для грузового транспорта, в нём будет своё поле вместимости, два инициализаторы для разных вариантов создания, переопределение метода движения, геттер и сеттер для поля этого класса и метод, обозначающий загрузку грузовика, рисунок 8.

  
Рисунок 8 – создание второго дочернего класса.

Создаю третий дочерний класс, в котором будут поле наличия шлема, перегруженные инициализаторы, переопределённое поле, геттер и сеттер для характерного поля, метод для этого подкласса, рисунок 9.

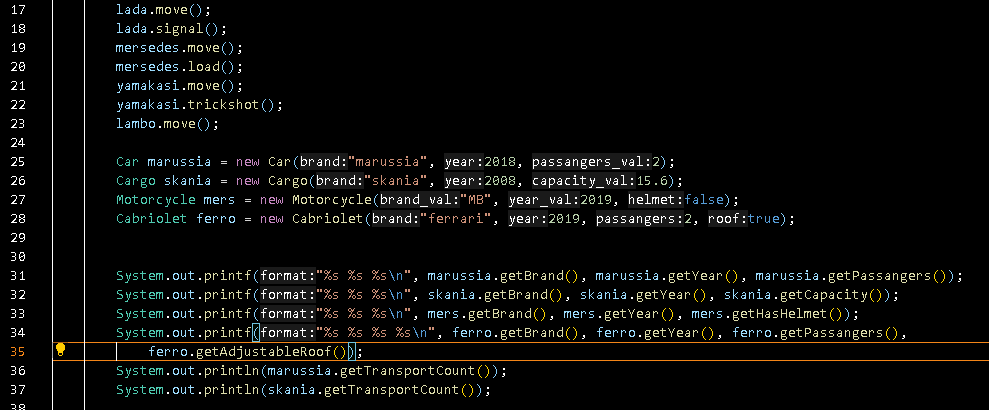
  
Рисунок 9 – создание третьего дочернего класса.

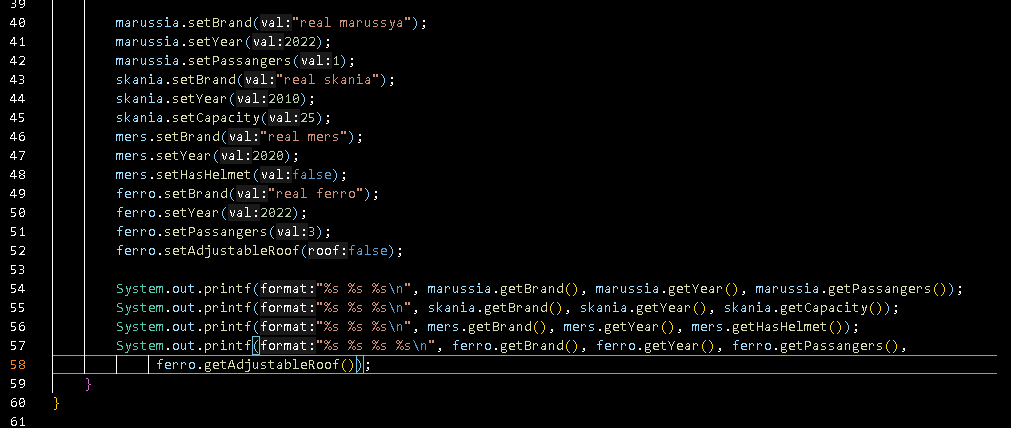
Создаю дочерний класс, наследуюемый от класса car, рисунок 10.

  
Рисунок 10 – создание класс второго уровня наследования.

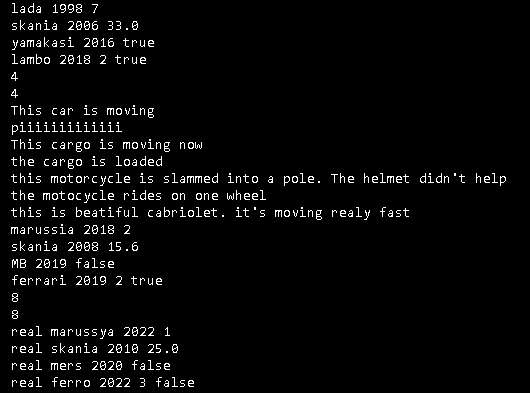
Проверяю работы программы. Сначала инициализирую переменный для каждого экземпляра класса без параметров, с помощью геттеров получаю параметры, затем для каждого класса применяю их методы, после инициализирую переменных с каждым экземпляром класса, но уже с разными параметрами, с помощью геттеров вывожу, затем меняю с помощью сеттеров и вновь с помощью геттеров вывожу, рисунки 11-13.

  
Рисунок 11 – демонстрация работы программы.

  
Рисунок 12 – демонстрация работы программы.

  
Рисунок 13 – демонстрация раоты программы.

Вывод программы представлен на рисунке 14.

  
Рисунок 14 – вывод программы.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Руководство по языку программирования Java // metanit URL: <https://metanit.com/java/tutorial/>
2. Learn Java Programming // programiz URL: <https://www.programiz.com/java-programming>
3. Java Introduction // w3school URL: https://www.w3schools.com/java/java\_intro.asp