Plan

2023-09-26

1. Analysis of homework
2. Introduction to OOP
3. Encapsulation
4. Access modifiers, Getters and setters, Constructor

---

1. Разбор домашнего задания
2. Введение в ООП
3. Инкапсуляция
4. Модификаторы доступа, Геттеры и сеттеры, Конструктор

Theory

▶ **English**

▼ **На русском**

Введение в ООП

ООП: что это такое?

ООП - это парадигма программирования, основанная на представлении программы как совокупности взаимодействующих **объектов**.

- **Объект** - это программная сущность, которая имеет свое **состояние** и **поведение**.
- **Состояние объекта** - это данные, которые он содержит.
- **Поведение объекта** - это действия, которые он может выполнять.

Основные понятия ООП: классы, объекты, методы, атрибуты

- **Класс** - это **шаблон**, описывающий свойства и поведение объекта.
- **Объект** - это экземпляр класса, который имеет свои собственные значения свойств и может выполнять действия, описанные методами.
- **Метод** - это функция, принадлежащая классу. Описывающее поведение объекта, действия, которые он может выполнять
- **Атрибут/поле класса** - это переменная, принадлежащая классу. Чаще называют **полем класса**

Программирование - это как строительство объектов. Каждая программа состоит из множества объектов, и ООП помогает нам лучше управлять этими объектами.

Примеры объектов из жизни:

Автомобиль

Состояние объекта/Атрибут/поле класса: Макра, модель, мощность, тип топлива, цвет ...

Поведение объекта/Метод: К примеру автомобиль может ехать, заправляться, сигналить, поворачивать, тормозить ...

Файл

Состояние объекта/Атрибут/поле класса: Размер, тип, имя, расположение, дата создания, дата изменения ...

Поведение объекта/Метод: Открыть, изменить имя или раположение, прочитать, закрыть ....

Дом Состояние объекта/Атрибут/поле класса: адрес, количество этажей, площадь, количество комнат ...

Поведение объекта/Метод: построить, отремонтировать, продать, заселить, покинуть ....

Более подробнее

Машина

Атрибуты:

- Марка: строка, содержащая марку машины.
- Модель: строка, содержащая модель машины.
- Год выпуска: целое число, указывающее год выпуска машины.
- Цвет: строка, содержащая цвет машины.
- Пробег: целое число, указывающее пробег машины в километрах.

Методы:

- Ехать: метод, который заставляет машину двигаться.
- Поворачивать: метод, который заставляет машину поворачивать.
- Тормозить: метод, который заставляет машину тормозить.
- Парковаться: метод, который заставляет машину парковаться.
- Заправляться: метод, который заправляет машину топливом.
- Ремонтировать: метод, который ремонтирует машину.

В ООП есть четыре основных принципа:

1. **ИНКАПСУЛЯЦИЯ**
2. **НАСЛЕДОВАНИЕ**
3. **ПОЛИМОРФИЗМ**
4. **АБСТРАКЦИЯ**

Сегодня мы сосредоточимся на инкапсуляции.

Инкапсуляция

Что такое инкапсуляция?

Инкапсуляция - это объединение данных и методов в одном классе.

Инкапсуляция - это один из ключевых принципов ООП, который позволяет скрыть детали реализации объекта и предоставить только необходимый интерфейс для взаимодействия с ним. Это подобно упаковке подарка - вы видите только внешний вид, но не знаете, что внутри, но можете узнать что внутри и действовать через методы.

Модификаторы доступа

В Java, для реализации инкапсуляции, мы используем модификаторы доступа. Есть три основных модификатора доступа: `public`, `private` и `protected`.

- `public` - это, как если бы вы дали всем доступ к вашему подарку.
- `private` - это как запереть подарок в сейфе и дать доступ только вам.
- `protected` - это подарок, который вы даете только своей семье и друзьям. *(более подробнее рассмотрим позже)*

Геттеры и сеттеры (Getters/Setters)

Геттеры и сеттеры - это методы, которые позволяют управлять данными объекта, соблюдая инкапсуляцию.

- **Геттеры** - это методы, которые позволяют получать значения данных объекта. С помощью них, как через окно в банке, можно посмотреть баланс на вашем счету.
- **Сеттеры** - это методы, которые позволяют устанавливать значения данных объекта. Они как способ внести или снять деньги со счета.
- **Геттеры и сеттеры** должны быть доступны всем, т.е. иметь модификатор доступа `public`

Конструктор

Помимо геттеров и сеттеров, есть еще один важный аспект инкапсуляции - это конструкторы. Конструктор - это специальный метод, который выполняется при создании объекта класса.

Он используется для установки начальных значений объекта. Например, если у нас есть класс "Пользователь", конструктор может инициализировать имя и пароль пользователя при его создании.

Ключевое слово this в Java

this - это ключевое слово в Java, которое используется внутри методов класса для ссылки на текущий объект.

Оно обозначает, что мы обращаемся к атрибутам или методам объекта, в котором выполняется данный код.

this может быть использовано для разрешения конфликта имён между аргументами метода и полями класса, если они имеют одинаковые имена.

Пример класса/шаблона который описывает планету.

У планеты есть атрибуты название `name` и размер `size`

Данный шаблон позволяет установить значения всех атрибутов при его инициализации, а так же мы можем **изменить** значения через `setters` и **получить** их значения через `getters`

Пример:

```
public class Planet { // класс описывающий планету, шаблон планеты

    // поля класса
    private String name;  // имя планеты
    private long size;    // размер планеты


    public Planet(String name, long size) { // конструтор для установки зна
        this.name = name;   // установка значения для поля класса name
        this.size = size;   // установка значения для поля класса size
    }

    // сеттер для поля name
    public void setName(String name) {
        this.name = name;       // установка/изменение значения для поля кл
    }

    // геттер для поля name
```

```
    public String getName() {
        return this.name;       // вернуть значение поля класса name
    }

    // сеттер для поля size
    public void setSize(long size) {
        this.size = size;       // установка/изменение значения для поля кл
    }


    // геттер для поля name
    public long getSize() {
        return this.size;     // вернуть значение поля класса size
    }
}
```

- теперь если кто то решит изменить имя планеты, он может сделать это вызвав метод `setName` это с новым именем
- так же и с размером, к примеру если изначально размер планеты был не верно расчитан. Только для изменения размера нужно вызвать метод `setSize` и указать нужный размер
- для доступа к текущим значениям имени (`name`) и размеру (`size`) планеты нужно использовать getters. Для имени `getName`, для размера `getSize`

Создание Класса (капсулы)

Объекты класса создают с помощью **конструктора** (стандартного метода класса), значения полей устанавливают и/или изменяются с помощью сеттеров (это стандартные методы класса), значения полей доступны с помощью геттеров (это стандартные методы класса).

- Шаг 1: создание полей
- определяем перечень переменных и их типов, принимаем решения о значениях модификаторов
  - public или private
- при написании кода идем в обратном направлении, сначала пишем:
  - модификатор
  - потом тип переменной
  - потом имя переменной
- Шаг 2: определение методов
- стандартные методы:
  - конструктор (позволяет создавать объекты, экземпляры класса)
  - геттеры (позволяет получить значения полей объекта)
  - сеттеры (позволяет установить значения полей объекта)

- дополнительные методы

Пример: Автомобиль (Car): - новый ТИП данных(!!!)

- **марка** - String brand
- **цвет** - String color
- **мощность двигателя** - double
- **тип топлива** - String fuelType который включает в себя все перечисленные поля.

пример создания объекта типа Car

```
public class Main {
    public static void main(String[] args) {
        // для создания объектов используется ключевое слово new
        // при создании объекта необходимо определить его тип. В вашем случа
        // инициализация объекта, после знака "=" new Car("BMW", "Black", 2
        // при инициализации объекта вызывается конструктор класса Car с ат
        Car bmw = new Car("BMW", "Black", 250, "disel"); // создание объект

        //  нашему автомобилу установленны значения:
        //  марка - "BMW"
        //  цвет - "Black"
        //  мощность двигателя - "250"
        //  тип топлива - "disel"

        // прекрасно, теперь нужно заправить автомобиль и можно ехать в отпу
    }
}
```

Homework

▶ **English**

▼ **На русском**

Задача 1

Создайте класс с именем "Пользователь" (**User**) с приватными полями:

- **name** (имя пользователя, строковое значение)
- **age** (возраст пользователя, целочисленное значение)

- **email** (электронная почта пользователя, строковое значение) Добавьте конструктор класса, который принимает параметры для всех полей и инициализирует объект при его создании.

Создайте геттеры и сеттеры для каждого поля класса "Пользователь".

В методе main создайте объекты класса "Пользователь" и инициализируйте их с помощью конструктора и сеттеров. Затем используйте геттеры для получения информации о каждом пользователе и выведите ее в консоль.

Попробуйте изменить значения полей объектов с использованием сеттеров и выведите обновленную информацию в консоль.

Примечание:

- Убедитесь, что вы соблюдаете принцип инкапсуляции, делая поля класса "Пользователь" приватными и предоставляя доступ к ним через геттеры и сеттеры.
- Практикуйтесь в создании объектов, их инициализации и управлении данными с использованием геттеров и сеттеров.

Задача 2

Создание класса "**Книга**"

Создайте класс с именем "Книга" (Book) с приватными полями:

- **title** (название книги, строковое значение)
- **author** (автор книги, строковое значение)
- **year** (год выпуска книги, целочисленное значение)
- **isbn** (ISBN книги, строковое значение)

Добавьте конструктор класса, который принимает параметры для всех полей и инициализирует объект при его создании.

Создайте геттеры и сеттеры для каждого поля класса "Книга".

В методе main создайте несколько объектов класса "Книга" и инициализируйте их с помощью конструктора и сеттеров. Затем используйте геттеры для получения информации о каждой книге и выведите ее в консоль.

Попробуйте изменить значения полей объектов с использованием сеттеров и выведите обновленную информацию в консоль.

Задача 3*: Учет банковских счетов

Цель: Создать класс "Банковский счет" с использованием инкапсуляции и реализовать методы для внесения и снятия средств.

Создайте класс с именем "БанковскийСчет" (**BankAccount**) с приватными полями:

- **accountNumber** (номер счета, строковое значение)
- **balance** (баланс счета, десятичное число с двумя знаками после запятой)

Добавьте конструктор класса, который принимает параметры для номера счета и начального баланса и инициализирует объект при его создании.

Создайте геттеры и сеттеры для номера счета и баланса.

- Создайте метод **deposit**, который принимает сумму для внесения и увеличивает баланс счета на эту сумму.
- Создайте метод **withdraw**, который принимает сумму для снятия и уменьшает баланс счета на эту сумму, если на счету достаточно средств. Если сумма для снятия больше баланса, выведите сообщение об ошибке.

В методе **main** создайте объекты класса "**BankAccount**", инициализируйте их с помощью конструктора и выведите информацию о счетах, балансах и произведите операции по внесению и снятию средств.

Попробуйте разные операции с внесением и снятием средств, включая случаи, когда на счету недостаточно средств.

Code

code/HwSolution13/src/Task1.java

```
/**
 * @author Andrej Reutow
 * created on 25.09.2023
 * Создать массив из 20-ти случайных целых чисел из интервала от -100 до 10(
 * Попало ли число 0 в этот массив? (выяснить с помощью binary search)
 * Если да, то на какое место (индекс) в массиве? Если нет, то на какой инд(
 */
public class Task1 {
    public static void main(String[] args) {
        int[] arr = new int[5];
        for (int i = 0; i < arr.length; i++) {
            arr[i] = (int) (Math.random() * 201) - 100;
        }
        System.out.println("Массив до сортировки");
        printArray(arr);
        bubbleSort(arr);
        System.out.println("Массив после сортировки");
        printArray(arr);
```

```
        int target = 15;
        int index = binarySearch(arr, target);

        if (index >= 0) {
            System.out.println("Число " + target + " находится на индексе:
        } else {
            int insertionPoint = 0;
            for (int i = 0; i < arr.length; i++) {
                if (arr[i] > target || i == arr.length - 1) {
                    insertionPoint = i;
                    break;
                }
            }
            System.out.println("Число " + target + " отсутствует. Его следу
        }
    }

    public static void printArray(int[] arr) {
        for (int i : arr) {
            System.out.print(i + " ");
        }
        System.out.println();
    }

    private static int binarySearch(int[] source, int target) {
        int left = 0;
        int right = source.length - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;
            int currentValue = source[mid];
            if (currentValue == target) {
                return mid;
            } else if (currentValue < target) {
                left = mid + 1;
            } else if (currentValue > target) {
                right = mid - 1;
            }
        }
        return -1;
    }
```

```
    public static void bubbleSort(int[] arr) {
        for (int indexOut = 0; indexOut < arr.length; indexOut++) {
            for (int indexIn = 0; indexIn < arr.length - 1 - indexOut; inde

                if (arr[indexIn] > arr[indexIn + 1]) {
                    int temp = arr[indexIn];
                    arr[indexIn] = arr[indexIn + 1];
                    arr[indexIn + 1] = temp;
                }

            }
        }
    }
}
```

code/HwSolution13/src/Task2.java

```
/**
 * @author Andrej Reutow
 * created on 25.09.2023
 * <p>
 * Создать массив из 20-ти случайных целых чисел из интервала от -100 до 10
 * который получает на вход исходный массив и возвращает массив, содержащий
 */
public class Task2 {
    public static void main(String[] args) {
        int[] arr = new int[20];
        for (int i = 0; i < arr.length; i++) {
            arr[i] = (int) (Math.random() * 201) - 100;
        }

        printArray(arr);

        int[] positiveArr = filterPositive(arr);
        printArray(positiveArr);
    }

    public static int[] filterPositive(int[] arr) {
        int count = 0;
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] > 0) {
                count++;
            }
        }
```

```
        int[] result = new int[count];
        int index = 0;
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] >= 0) {
                result[index] = arr[i];
                index++;
            }
        }
        return result;
    }

    public static void printArray(int[] arr) {
        for (int j = 0; j < arr.length; j++) {
            int i = arr[j];
            System.out.print(i + " ");
        }
        System.out.println();
    }
}
```

code/HwSolution13/src/Task3.java

```
/**
 * @author Andrej Reutow
 * created on 25.09.2023
 * <p>
 * Реализовать способ обмена значениями двух переменных целого типа, не исп
 * В идеале написать метод swap(a, b).
 */
public class Task3 {

    public static void main(String[] args) {
        int a = 5;
        int b = 10;
        System.out.println("До обмена: a = " + a + ", b = " + b);
        swap(a, b);
    }

    public static void swap(int a, int b) {
        a = a + b;
        b = a - b;
        a = a - b;
        System.out.println("После обмена: a = " + a + ", b = " + b);
```

```
    }
}
```

code/HwSolution13/src/hw13/Task1.java

```
package hw13;

/**
 * @author Andrej Reutow
 * created on 26.09.2023
 * Условие: Напишите программу на Java, которая использует бинарный поиск д
 * <p>
 * Пример:
 * <p>
 * Вход: Отсортированный массив [1, 3, 3, 3, 7, 9, 11, 13, 15, 17, 17, 19]
 */
public class Task1 {

    public static void main(String[] args) {
        int[] arr = {1, 3, 3, 3, 7, 9, 11, 13, 15, 17, 17, 19};
        int target = 3;
        int firstIndex = binnarySearch(arr, target);

        if (firstIndex != -1) {
            System.out.println("Индекс первого вхождения числа " + target +
        } else {
            System.out.println("Число " + target + " не найдено в массиве."
        }
    }

    public static int binnarySearch(int[] arr, int target) {
        int left = 0;
        int right = arr.length - 1;
        int result = -1;

        while (left <= right) {
            int mid = left + (right - left) / 2;

            if (arr[mid] == target) {
                result = mid;
                right = mid - 1; // Двигаемся влево для поиска первого вхож
            } else if (arr[mid] < target) {
                left = mid + 1;
            } else {
```

```
                right = mid - 1;
            }
        }

        return result;
    }
}
```

code/HwSolution13/src/hw13/Task2.java

```
package hw13;

/**
 * @author Andrej Reutow
 * created on 26.09.2023
 * <p>
 * Напишите программу, которая находит n-ное вхождение заданного числа в от
 * <p>
 * Пример:
 * <p>
 * Вход: Массив [1, 3, 3, 3, 7, 9, 11, 13, 15, 17, 17, 19] и число 3 с номе
 * <p>
 * Выход: Индекс 2, поскольку второе вхождение числа 3 находится на этой по
 */
public class Task2 {

    public static void main(String[] args) {
        int[] arr = {1, 3, 3, 3, 7, 9, 11, 13, 15, 17, 17, 19};
        int target = 3;
        int n = 2; // Второе вхождение числа три

        int linearIndex = findNthOccurrenceLinear(arr, target, n);

        if (linearIndex != -1) {
            System.out.println("Индекс " + n + "-го вхождения числа " + tar
        } else {
            System.out.println("Число " + target + " с " + n + "-м вхождени
        }
    }

    public static int findNthOccurrenceLinear(int[] arr, int target, int n)
        int count = 0;
```

```
        int index = -1;

        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == target) {
                count++;
                if (count == n) {
                    index = i;
                    break;
                }
            }
        }

        return index;
    }
}
```

code/HwSolution13/src/hw13/Task3.java

```
package hw13;

/**
 * @author Andrej Reutow
 * created on 26.09.2023
 * <p>
 * Напишите программу, которая сначала сортирует массив целых чисел по возра
 * <p>
 * Пример:
 * <p>
 * Вход: Массив [10, 3, 15, 7, 8, 5, 11, 2].
 * <p>
 * Выход: Два отсортированных массива: [2, 3, 5, 7, 8, 10, 11, 15] и [15, 1
 */
public class Task3 {

    public static void main(String[] args) {
        int[] arr = {10, 3, 15, 7, 8, 5, 11, 2};

        // Сортировка по возрастанию.
        bubbleSortAscending(arr);
        System.out.print("Отсортированный массив по возрастанию: ");
        for (int num : arr) {
            System.out.print(num + " ");
```

```
        }
        System.out.println();

        // Сортировка по убыванию.
        bubbleSortDescending(arr);
        System.out.print("Отсортированный массив по убыванию: ");
        for (int i = 0; i < arr.length; i++) {
            System.out.print(arr[i] + " ");
        }
    }

    // Метод для сортировки массива по возрастанию сортировкой пузырьком.
    public static void bubbleSortAscending(int[] arr) {
        int n = arr.length;

        for (int i = 0; i < n - 1; i++) {

            for (int j = 0; j < n - 1 - i; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }

    // Метод для сортировки массива по убыванию сортировкой пузырьком.
    public static void bubbleSortDescending(int[] arr) {
        int n = arr.length;

        for (int i = 0; i < n - 1; i++) {

            for (int j = 0; j < n - 1 - i; j++) {
                if (arr[j] < arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
```

```
    }
}
```

code/HwSolution13/src/hw13/Task4.java

```
package hw13;

import java.util.Scanner;

/**
 * @author Andrej Reutow
 * created on 26.09.2023
 * Пользователь вводит нескольких слов.
 * Сохраните каждое слово в массиве и выведите все слова в обратном порядке
 */
public class Task4 {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String[] words = new String[4];

        for (int i = 0; i < words.length; i++) {
            System.out.println("Enter word: " + (i + 1));
            String word = scanner.nextLine();
            words[i] = word;
        }

        printReversed(words);
    }

    public static void printReversed(String[] words) {
        for (int i = words.length - 1; i >= 0; i--) {
            System.out.print(words[i] + "\t");
        }
    }
}
```

code/HwSolution13/src/hw13/Task5.java

```
package hw13;
```

```java
/**
 * @author Andrej Reutow
 * created on 26.09.2023
 * С помощью вложенного цикла For (цикл в цикле) написать метод, выводящий
 * 1
 * 12
 * 123
 * 1234
 * 12345
 * 123456
 */
public class Task5 {

    public static void main(String[] args) {
        int n = 6; // Количество строк треугольника
        printTriangle(n);
    }

    public static void printTriangle(int n) {
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print(j);
            }
            System.out.println();
        }
    }
}
```

code/HwSolution13/src/hw13/Pizza.java

```java
package hw13;

/**
 * @author Andrej Reutow
 * created on 25.09.2023
 *
 * <p>
 * <p>
 * Цель задачи: Найти и вывести имя друга, который съел больше всех кусков
 * <p>
 * <p>
 * У вас есть массив из 6 элементов, каждый из которых представляет количес
 * <p>
 * <p>
```

```
 * Создайте массив, который будет содержать количество кусков пиццы, съеден
 * Создайте массива с именами друзей: пример: {"Алекс", "Борис", "Вера", "Г
 * Найдите максимально значение в первом массиве и запомните индекс этого з
 * Найдите друга во втором массиве, который съел больше всех кусков пиццы.
 * </p>
 */


public class Pizza {


    public static void main(String[] args) {
        int[] pizzas = {2, 4, 3, 5, 10, 3};
        String[] friends = {"Алекс", "Борис", "Вера", "Галя", "Дима", "Елен

        int indexOfMaxElement = findIndexOfMaxElement(pizzas);

        System.out.println(friends[indexOfMaxElement] + " съел(а) больше вс
    }

    public static int findIndexOfMaxElement(int[] array) {
        int maxElementIndex = 0;
        int max = array[maxElementIndex];

        for (int i = 0; i < array.length; i++) {
            if (array[i] > max) {
                max = array[i];
                maxElementIndex = i;
            }
        }

        return maxElementIndex;
    }
}
```

code/Lesson16/src/Application.java

```
/**
 * @author Andrej Reutow
 * created on 26.09.2023
 */
public class Application {
```

```
    public static void main(String[] args) {
//        Planet planetMars = new Planet("Mars", 500);
//        Planet planetEarth = new Planet("Earth", 200);
//        Planet planetJupiter = new Planet("Jupiter", 400);
//
//        System.out.println(planetMars.getName()); // Mars
//        System.out.println(planetEarth.getName()); // Earth
//        System.out.println(planetJupiter.getName()); // Jupiter
//
//        planetMars.setName("ABC");
//        System.out.println();
//        System.out.println(planetMars.getName()); // ABC
//        System.out.println(planetEarth.getName()); // Earth
//        System.out.println(planetJupiter.getName()); // Jupiter
//
//        planetMars.setName("Mars");
//        System.out.println();
//        System.out.println(planetMars.getName()); // Mars
//        System.out.println(planetEarth.getName()); // Earth
//        System.out.println(planetJupiter.getName()); // Jupiter
//
//        System.out.println();
//        System.out.println("Имя планеты " + planetJupiter.getName() + ",
//
//        System.out.println();
//        planetJupiter.printDetails();
//        planetEarth.printDetails();
//        planetMars.printDetails();

        // пример с классом Car
        Car bwm = new Car("BMW", "Black", 250.0, "disel"); // создание новг
        bwm.drive(); // начать движение машины BMW
        bwm.stop(); // отсановить BMW

        System.out.println();

        Car vw = new Car("VW", "White", 100.0, "disel"); // создание новго
        vw.drive(); // начать движение VW
        vw.stop(); // отсановить VW

        System.out.println();
```

```
        Car ford = new Car("Ford", "Green", 160.0, "disel"); // создание но
        ford.drive(); // начать движение Frod
        ford.stop(); // отсановить Frod
    }
}
```

code/Lesson16/src/Planet.java

```
/**
 * @author Andrej Reutow
 * created on 26.09.2023
 */
public class Planet {

    // поля класса
    private String name;
    private long size;

    // конструктор
    public Planet(String name, long size) {
        this.name = name;
        this.size = size;
    }

    // геттер для поля имя
    public String getName() {
        return this.name;
    }

    // сеттер для поля имя
    public void setName(String name) {
        this.name = name;
    }

    // геттер для поля size
    public long getSize() {
        return this.size;
    }

    // сеттер для поля size
    public void setSize(long size) {
        this.size = size;
    }
```

```
    public void printDetails() {
        System.out.println("Имя планеты " + this.name + ", размер: " + this
    }
}
```

code/Lesson16/src/Car.java

```
/**
 * @author Andrej Reutow
 * created on 26.09.2023
 * <p>
 * марка - String brand
 * цвет - String color
 * мощность двигателя - double
 * тип топлива - String fuelType который включает в себя все перечисленные
 */
public class Car { // шаблон объекта

    // поля класса
    private String make;
    private String color;
    private double power;
    private String fuelType;


    //конструктор
    public Car(String make, String color, double power, String fuelType) {
        this.make = make;
        this.color = color;
        this.power = power;
        this.fuelType = fuelType;
    }

    // метод который приводит нашу машину (объект) в движение
    public void drive() {
        System.out.println("I am " + this.make + " and i drive now");
    }

    // метод который отсанавливет нашу машину (объект)
    public void stop() {
        System.out.println("I am " + this.make + "  and i stopped");
```

```
        }
    }
```