Введение в ООП. Часть 2

AIT
Hi-tech team

НАШИ ПРАВИЛА

Включенная камера

Вопросы по поднятой руке

Не перебиваем друг друга

Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору

Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

ЦЕЛЬ

Научиться реализовывать и применять методы

ПЛАН ЗАНЯТИЯ

- МЕТОДЫ ЭКЗЕМПЛЯРА
- МЕТОДЫ С ВОЗВРАТОМ ЗНАЧЕНИЯ
- МЕТОДЫ И СОСТОЯНИЕ ОБЪЕКТОВ
- ПЕРЕГРУЖЕННЫЕ МЕТОДЫ

МЕТОДЫ ЭКЗЕМПЛЯРА

Помимо состояния, объект также обладает некоторым поведением. Поведение определяется методами, описанными в классе.

В предыдущих уроках мы рассмотрели статические методы, использующие модификатор static. В этом уроке наш фокус сместится на нестатические методы — те, которые вызываются на уровне экземпляра класса.

```
Car polo = new Car(45, 6);
polo.refuel(10); // вызов метода объекта
```

```
class Car {
    double fuelVolume = 0; // объем топлива в литрах
    double maxFuelVolume; // объем бака
    double mileage = 0; // пробег в километрах
    double fuelConsumption; // расход топлива на 100 км

    Car(double maxFuelVolume, double fuelConsumption) {
        this.maxFuelVolume = maxFuelVolume;
        this.fuelConsumption = fuelConsumption;
    }

    // метод для заправки автомобиля
    void refuel(double liters) {
        if (fuelVolume + liters > maxFuelVolume) {
            fuelVolume = maxFuelVolume;
        } else {
            fuelVolume += liters;
        }
    }
}
```

СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО

Метод - поведение объекта

Вызов метода

МЕТОДЫ С ВОЗВРАТОМ ЗНАЧЕНИЯ

Очевидно, методы могут возвращать какие-либо значения в вызывающий код

```
Car polo = new Car(45, 6);
polo.refuel(10);

if (polo.go(5)) { // вызов метода объекта
    System.out.println("Успешная поездка");
} else {
    System.out.println("Что-то пошло не так…");
}
```

```
class Car {
    // …
    boolean go(double kilometers) {
    // Рассчитываем, сколько топлива потребуется
        double fuelNeeded =
            (kilometers * fuelConsumption) / 100;

    // Проверяем, хватит ли топлива для поездки
        if (fuelVolume >= fuelNeeded) {
            fuelVolume -= fuelNeeded; // Уменьшаем объем
            mileage += kilometers; // Увеличиваем пробег
            return true; // Поездка завершилась успешно
        } else {
            return false; // Не хватило топлива для поездки
        }
    }
}
```

СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО

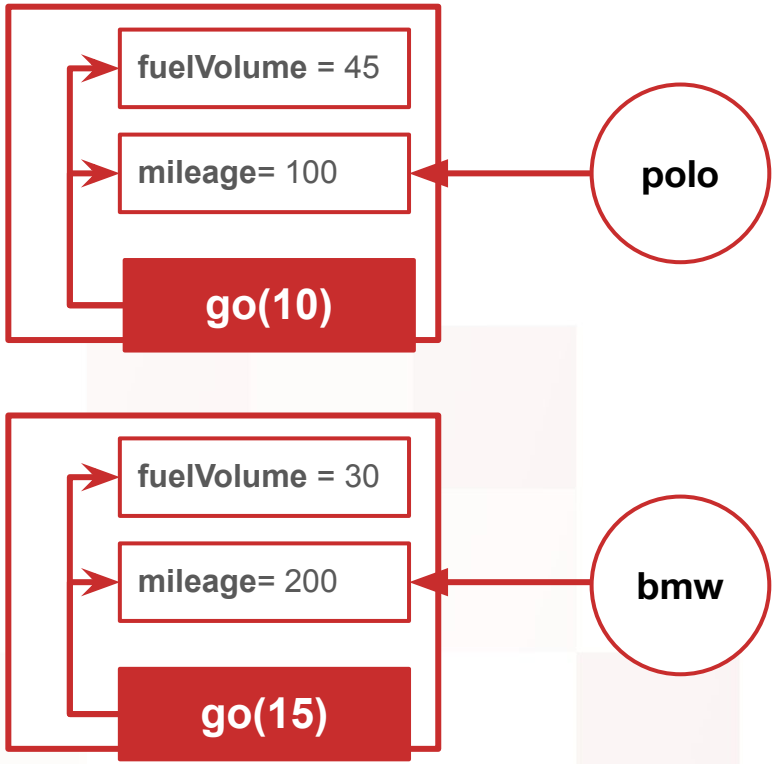
Возврат значения из метода

МЕТОДЫ И СОСТОЯНИЕ ОБЪЕКТОВ

Методы, вызываемые у объекта, взаимодействуют с состоянием только этого конкретного объекта и влияют на его поля:

```
Car polo = new Car(45, 6);
Car bmw = new Car(68, 6.5);

polo.go(10);
bmw.go(15);
```



СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО

Взаимодействие методов с полями

ПЕРЕГРУЖЕННЫЕ МЕТОДЫ

Также есть возможность использования перегруженных (отличающихся только формальными параметрами) методов:

```
Car polo = new Car(45, 6);
polo.refuel(30);
polo.go(100);

// уменьшить пробег на 50 км
polo.adjustOdometer(50);

// обнулить пробег
polo.adjustOdometer();
```

```
class Car {
    // перегруженные методы для изменения
    // показаний пробега
   boolean adjustOdometer(int km) {
       if (mileage >= km) {
           mileage -= km;
           return true;
       } else {
           return false;
       }
    }

    boolean adjustOdometer() {
       mileage = 0;
       return true;
    }

}
```

СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО

Перегрузка методов

ПОИГРАЕМ ;)

Методы объекта и поля

Перегрузка методов

ДОМАШНЕЕ ЗАДАНИЕ

Написать класс Pokemon. В полях определите минимум 2-3 свойства:

- очки здоровья (HP)
- атака (attack)
- защита (defense)
- специальная атака (special attack)
- специальная защита (special defense)
- скорость (speed)
- имя (name)

Реализуйте методы fight и sleep

Заставьте их драться и спать. Покемоны должны иметь имя.

В самой простой версии методы fight(), sleep() должны выводить информацию о происходящем.

В более продуманной версии: используя массивы, надо создать 2 команды покемонов (каждый покемон должен находиться либо в одном либо в другом массиве), посчитать итоговые результаты сражения.
При сражении надо менять состояние (например уменьшать hp).
При сне надо увеличивать hp на какую то долю.

AIT
Hi-tech team

Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в
немецкой школе AIT TR GmbH

AIT
Hi-tech team