Интерфейсы

Цели: понять идею интерфейсов

План урока:
- методы классов
- как читать сигнатуру методов
- как передаются объекты
- интерфейсы и их реализации

Класс и объект

Класс – тип данных. Мы можем сами описывать сложные сущности, которые состоят из других *типов данных*.

```
public class Car {
    String color;
    String brand;
    String model;
    int hp;
    String plateNumber;
}
```

Класс и объект

Класс – тип данных. Мы можем сами описывать сложные сущности, которые состоят из других *типов данных*.

```
public class Car {
    String color;
    String brand;
    String model;
    int hp;
    String plateNumber;
}
```

Объект (экземпляр класса) – конкретный представитель класса

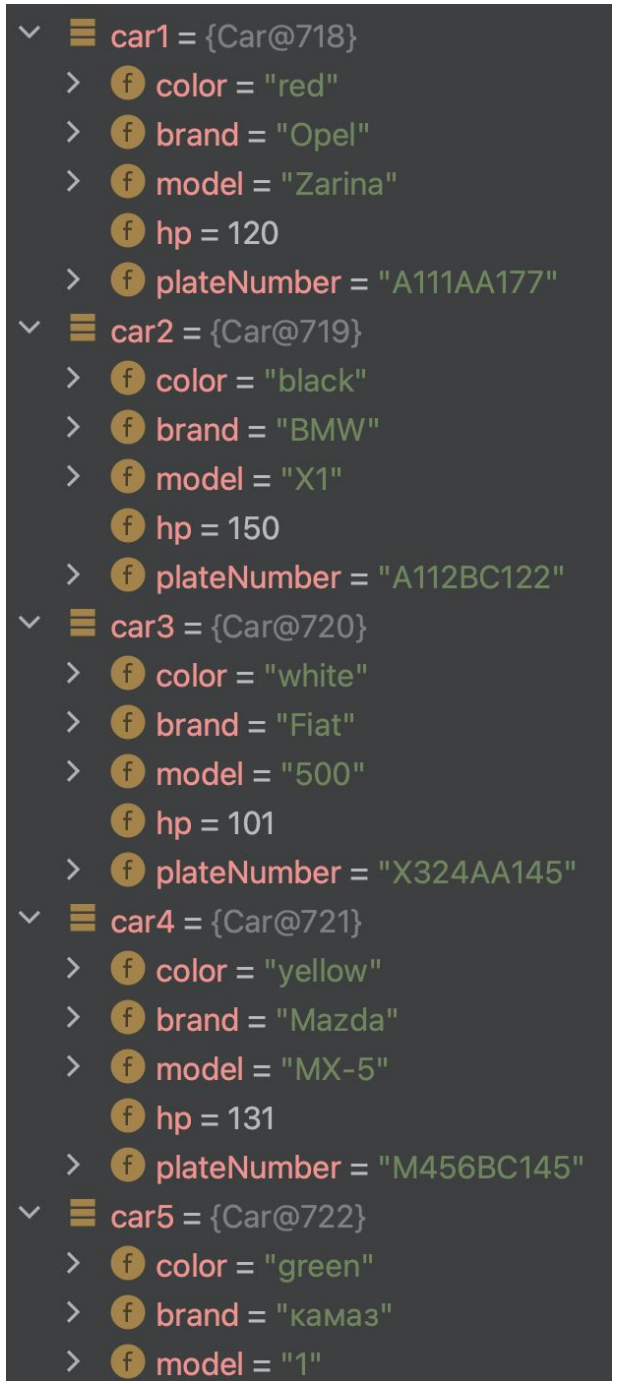
```
Car car1 = new Car( color: "red",  brand: "Opel",  model: "Zarina",  hp: 120,  carPlateNumber: "A111AA177");
Car car2 = new Car( color: "black",  brand: "BMW",  model: "X1",  hp: 150,  carPlateNumber: "A112BC122");
Car car3 = new Car( color: "white",  brand: "Fiat",  model: "500",  hp: 101,  carPlateNumber: "X324AA145");
Car car4 = new Car( color: "yellow",  brand: "Mazda",  model: "MX-5",  hp: 131,  carPlateNumber: "M456BC145");
Car car5 = new Car( color: "green",  brand: "камаз",  model: "1",  hp: 10000,  carPlateNumber: "P234ET02");
```

Класс и объект

Состояние объекта – значение полей объекта.

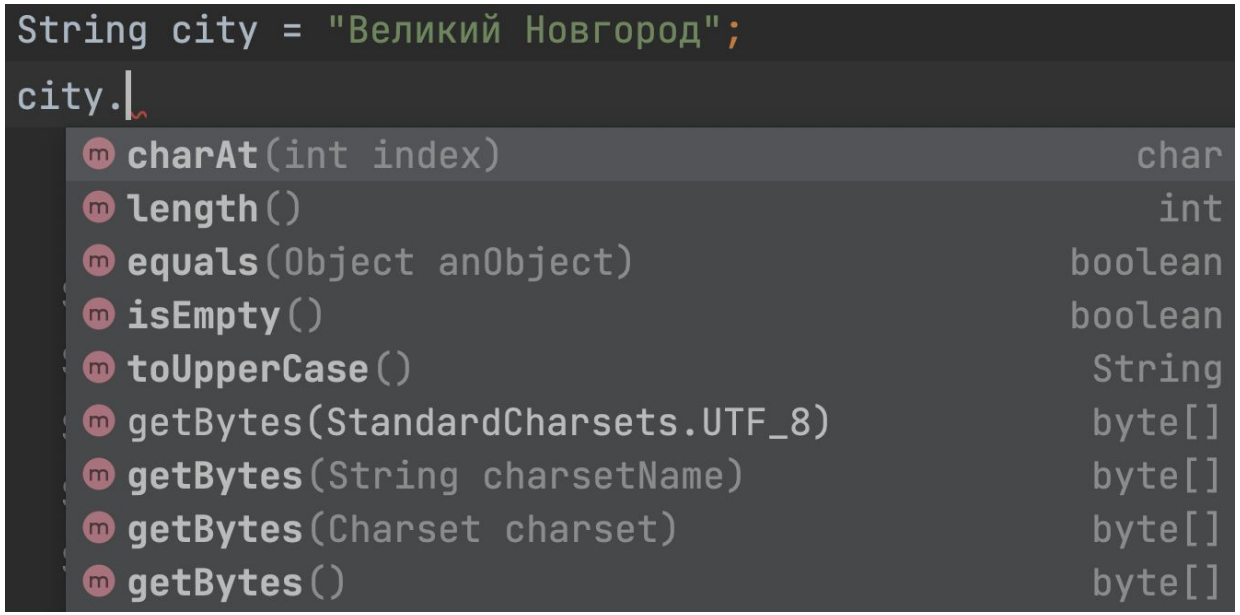
На рисунке мы видим 5 объектов. У них разные состояния.

Да, можно создавать объекты с одинаковыми состояниями.



Класс и объект

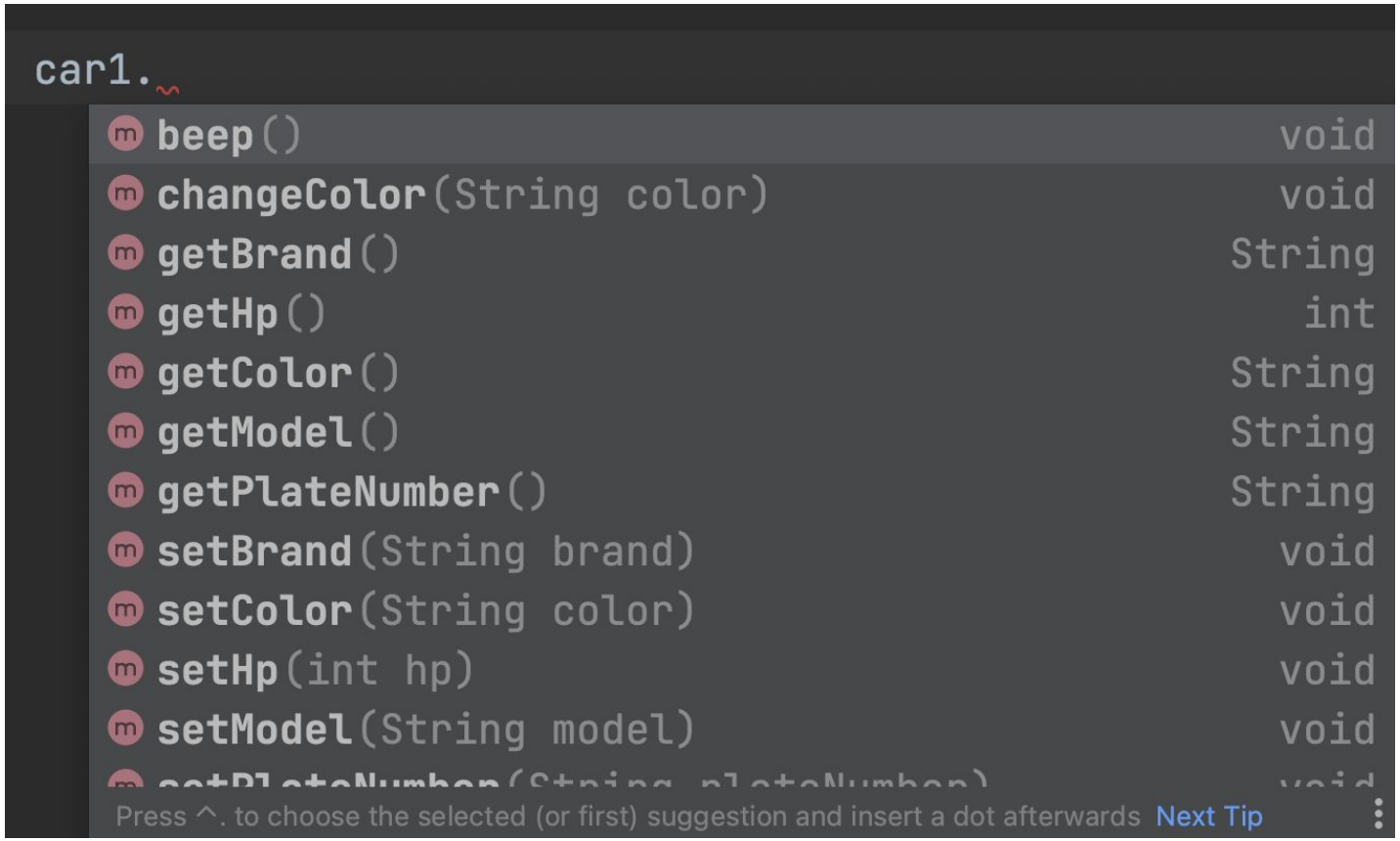
Поведение объекта – набор методов. Ответ на вопрос, что умеет делать объект



Класс и объект

Поведение объекта – набор методов. Ответ на вопрос, что умеет делать объект.

На примере машины



Класс и объект

Состояние объекта принято скрывать. Для этого, всем полям присваивается *модификатор видимости* (модификатор доступа) private.

К таким полям можно обратиться только из кода самого класса.

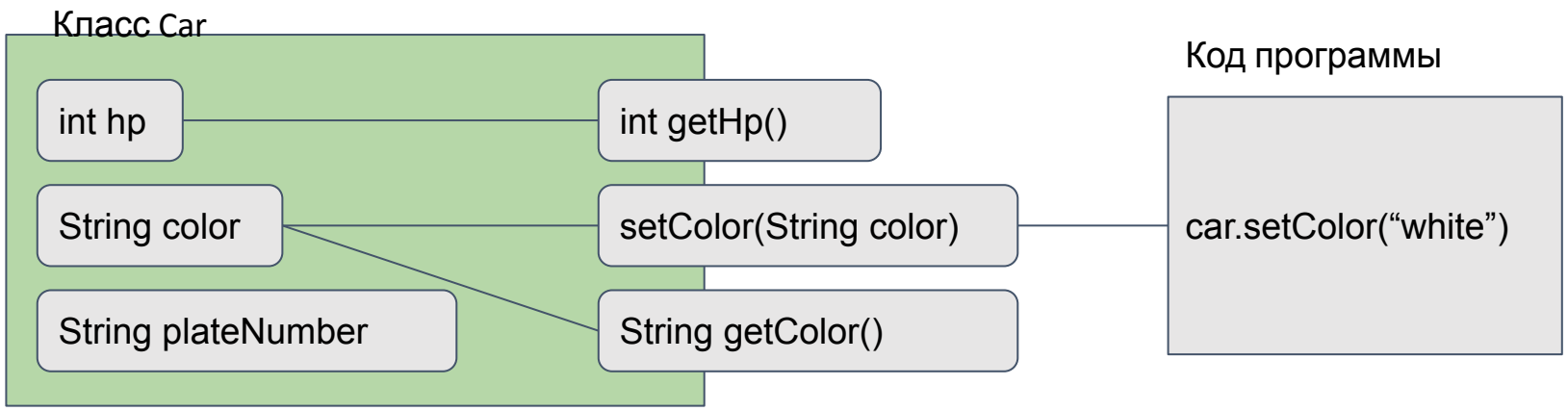
Доступ к полям из “внешнего мира” мы предоставляем через **методы**. Если захотим.

```
4 usages
private String color;
3 usages
private String brand;
3 usages
private String model;
3 usages
private int hp;
3 usages
private String plateNumber;
```

Инкапсуляция

Один из принципов ООП – скрывай состояние объектов и детали реализации.

Вместо сложного алгоритма, предоставь метод, который выполнит этот алгоритм.



Как читать методы

Можно открыть класс и посмотреть на его методы.

- У каждого метода есть имя
- Метод может не принимать параметры, а может принимать много
- Метод может возвращать одно значение указанного типа данных. Нужно указать тип и написать с конце метода return

```
        Имя метода
public void beep(){
    System.out.println("Beeeeeeep");
}

  Метод может принимать много параметров
public void changeColor(String color){
    this.color = color;
}

  Метод может возвращать значение.
public int getHp() {
    return hp;
}
```

```
car1.
    beep()                              void
    changeColor(String color)           void
    getBrand()                          String
```

Задача банковскую карту

Как *можно было* решить

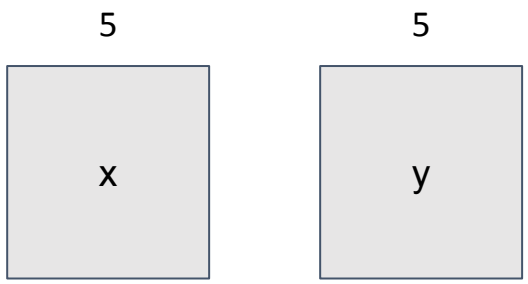
- Создайте класс Card
- У класса должны быть поля:
  - номер
  - pinCode
  - поля класса должны задаваться через конструктор
- Опишите в классе метод, который печатает номер карты в формате "**** **** **** 3456"
- Опишите в классе метод, который принимает на вход пин-код. Если переданный пин-код совпал с пин-кодом карты, напечатать в консоль номер карты без маски (все цифры)
-
- Создайте класс MyProgram + метод psvm
- В методе создайте новую карту
- Вызовите сначала метод печати с маской
- Потом метод печати с пин-кодом
- Убедитесь, что нельзя получить никакие данные карты напрямую через поля (используйте private)

Задача банковскую карту. Решение

```java
public class Card {
    3 usages
    private String number;
    2 usages
    private String pinCode;
    2 usages
    private String maskedNumber;

    2 usages  Dmitry Eremin *
    public Card(String number, String pinCode) {
        this.number = number;
        this.pinCode = pinCode;
        maskNumber();
    }

    1 usage  Dmitry Eremin *
    public String getNumber(){
        return maskedNumber;
    }
```

```java
    public String getNumber(String pinCode){
        if (pinCode.equals(this.pinCode)) {
            return this.number;
        } else {
            return "Not authorized.";
        }
    }

    1 usage  new *
    private void maskNumber(){
        String replace = number.replace( target: " ",  replacement: "");
        String lastFourDigits = replace.substring( beginIndex: replace.length()-4);
        this.maskedNumber = "**** ".repeat( count: 3).concat(lastFourDigits);
    }
}
```

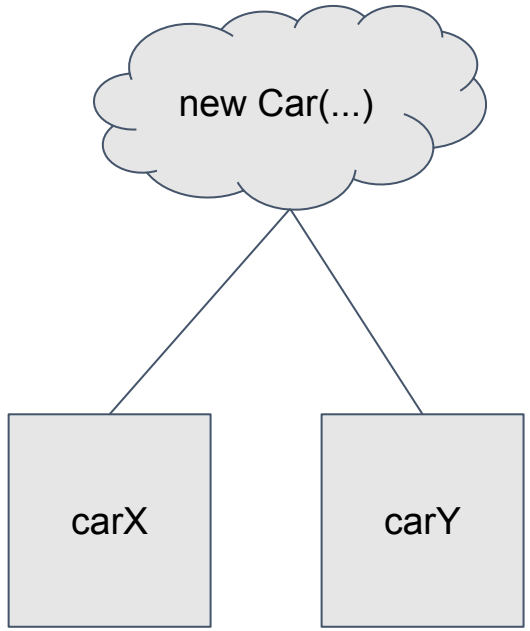
Примитивы передаются по значению.
Объекты передаются по ссылке.




```
int x = 5;
int y = x;
x = 6;

System.out.println("x = " + x);
System.out.println("y = " + y);
```

Примитивы передаются по значению. Объекты передаются по ссылке.

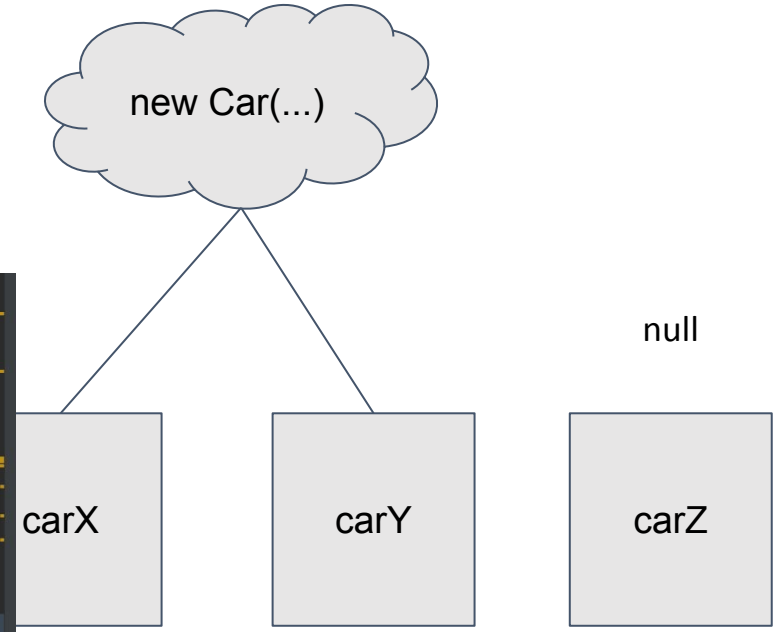
```
Car carX = new Car( color: "red",  brand: "Opel",  model: "Zarina",  hp: 120,
Car carY = carX;

carX.changeColor("black");
System.out.println("carY color = " + carY.getColor());
```



Примитивы передаются по значению. Объекты передаются по ссылке.

```
116         Car carX = new Car( color: "red",  brand: "Opel",  model: "Zarina",  hp: 120,  carPlateNumber: "A111AA177");
117         Car carY = carX;
118
119         carX.changeColor("black");
120         System.out.println("carY color = " + carY.getColor());
121
122         Car carZ = null;
123         System.out.println(carZ.getBrand());
124
```

```
Run:  ArrayGames
carY color = black
Exception in thread "main" java.lang.NullPointerException Create breakpoint : Cannot invoke "ru.inno.course.arrays.Car
 .getBrand()" because "carZ" is null
    at ru.inno.course.arrays.ArrayGames.main(ArrayGames.java:123)
```



Что такое интерфейс?

Общее определение: **Интерфейс — это совокупность методов и правил взаимодействия элементов системы.**

Другими словами, интерфейс определяет как элементы будут взаимодействовать между собой.

- Интерфейс двери — наличие ручки;
- Интерфейс автомобиля — наличие руля, педалей, рычага коробки передач;
- Интерфейс дискового телефона — трубка + дисковый набиратель номера.

Когда вы используете эти "объекты", вы уверены в том, что вы сможете использовать их подобным образом.

Благодаря тому, что вы знакомы с их интерфейсом.

В программировании что-то похожее. Почему графическую часть программы часто называют интерфейсом? Потому, что она определяет каким образом вы сможете использовать основной функционал, заложенный в программе.

Что такое интерфейс?

Интерфейс – ответ на вопрос: **что** умеет делать объект.

Как он это делает, мы не знаем.

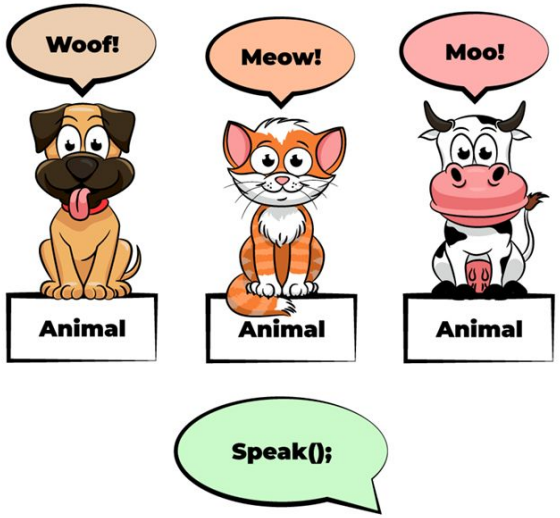
У автомобиля есть интерфейс – педали и руль.

Мы *не должны* знать, **как** работает автомобиль.

Мы знаем, **что** делает педаль и руль – можем пользоваться.

Полиморфизм

- В более общем смысле, концепцией полиморфизма является идея **“*один интерфейс, множество методов*“.**
- Это означает, что можно создать общий интерфейс для группы близких по смыслу действий. Преимуществом полиморфизма является то, что он помогает снижать сложность программ, разрешая использование того же интерфейса для задания единого класса действий. Выбор же конкретного действия, в зависимости от ситуации, возлагается на компилятор.
- Вам, как программисту, не нужно делать этот выбор самому. Нужно только помнить и использовать общий интерфейс.



Вопросы?