QA
מסע

אלכס גורבצ'וב

АЛЕКСЕЙ ГОРБАЧЁВ

JAVA

JAVA

Java

Наследование в Java

Наследование в Java позволяет повторно использовать код одного класса в другом классе, то есть вы можете унаследовать новый класс от уже существующего класса.

Главный наследуемый класс в Java называют родительским классам, или суперклассом.

Наследующий класс называют дочерним классом, или подклассом. Подкласс наследует все поля и свойства суперкласса, а также может иметь свои поля и свойства, отсутствующие в классе-родителе.

Наследование в Java

К примеру, у нас есть класс Dog. Что есть у каждой собаки? Четыре лапы, один хвост - и еще они умеют гавкать и вилять хвостом.

Наша задача: создать классы для разных пород - овчарок, бульдогов и болонок. У всех овчарок, бульдогов и болонок тоже будет четыре лапы и хвост, они тоже будут гавкать и вилять хвостом.

Можно просто брать и копировать эти методы и переменные в каждый класс. Но зачем? Мы можем использовать наследование.

Если мы сделаем все классы пород наследниками класса Dog, они будут иметь доступ ко всем его методам и переменным автоматически (кроме тех, что private).

Java

Наследование в Java

```java
public class Person {
    private String name;
    private String email;
    private String phone;
```

```java
public class Child extends Person{

    private String parentName;
```

```java
public class Student extends Person {
    private double avgGrade;
```

Наследование в Java

Как наследовать?
Для того, чтобы унаследовать класс, нужно использовать ключевое слово extends:

```
class Dobermann extends Dog {

}
```

Java

Наследование в Java. Правила наследования.

1. **Правило 1:** Наследуем только один класс.
   - Java не поддерживает наследование нескольких классов. Один класс - один родитель.
   - Обратите внимание - нельзя наследовать самого себя!

2. **Правило 2: Наследуется все кроме приватных переменных и методов.**
   - Выше мы говорили, что класс-наследник будет иметь доступ ко всем переменным и методам родителя. Это не совсем так. На самом деле, все методы и переменные, помеченные модификатором private, не доступны классу-наследнику.

Наследование в Java. Правила наследования.

3. **Правило 3. Переделать метод класса-родителя.**
   - Представим, что мы наследуем класс, но нам нравится не все, что мы унаследовали. Допустим мы хотим, чтобы определенный метод работал не так, как в родителе.
   - Для того, чтобы переопределить метод класса-родителя, пишем над ним @Override:

```
class Dobermann extends Dog {

    @Override
    public void barking ()
    {
        System.out.println("Bark!");
    }

}
```

Наследование в Java. Правила наследования.

4. **Правило 4. Вызываем методы родителя через ключевое слово super.**
   - Представим, что Вы хотите изменить метод родительского класса - дописать пару строк. Тогда в своем методе мы можем вызвать родительский метод с помощью ключевого слова super.
     Например, у нас есть класс Dog с методом voice():

```
public class Dog {
    public void barking(){
        System.out.println("Hello World from Dog class!");
    }
}
```

     Вызовем метод barking () из класса Dobermann:

```
Dobermann d = new Dobermann();
        d. barking();
```

Наследование в Java. Правила наследования.

5. **Правило 5. Запрещаем наследование.**
   - Если мы не хотим, чтобы кто-то наследовал Ваш класс, поставьте перед ним модификатор final. Например:

```
final class Dog {

}
```

Теперь при попытке создания наследника мы получим ошибку.

Java

Наследование в Java. Правила наследования.

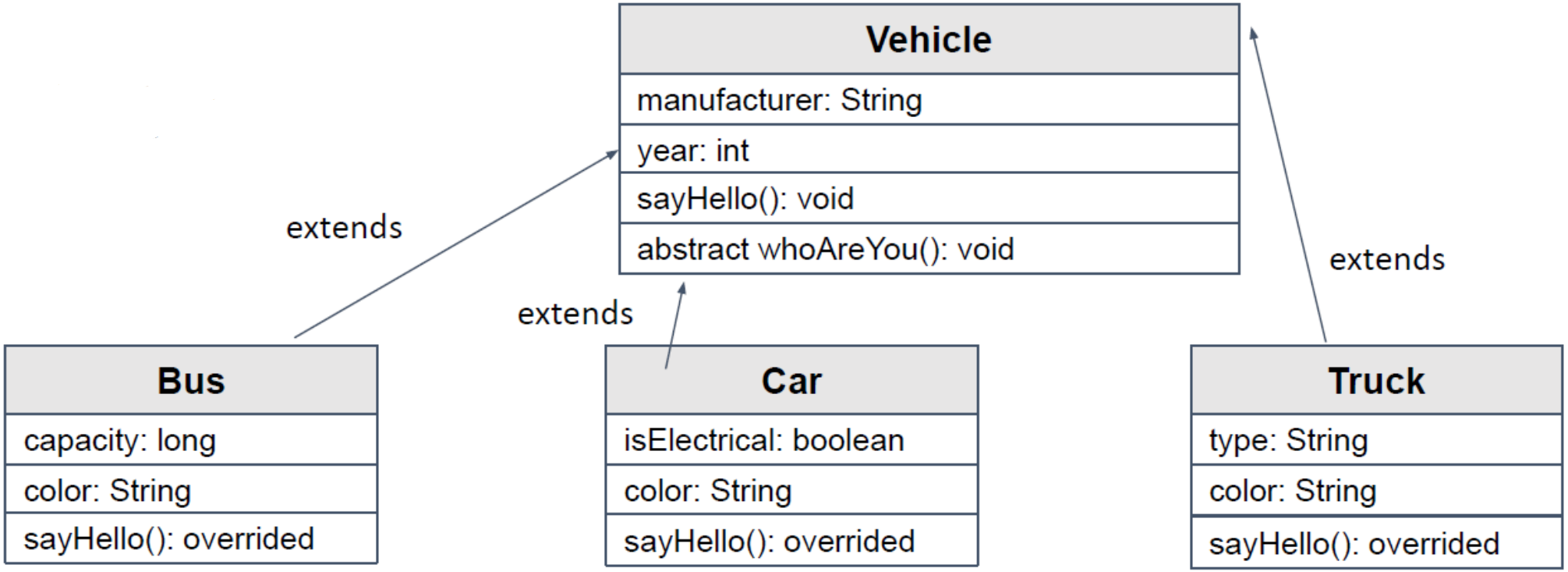
1. **Правило 1:** Наследуем только один класс.
   - Java не поддерживает наследование нескольких классов. Один класс - один родитель.
   - Обратите внимание - нельзя наследовать самого себя!

2. **Правило 2: Наследуется все кроме приватных переменных и методов.**
   - Выше мы говорили, что класс-наследник будет иметь доступ ко всем переменным и методам родителя. Это не совсем так. На самом деле, все методы и переменные, помеченные модификатором private, не доступны классу-наследнику.

Java

Задание.

- Создайте по экземпляру каждого класса.



Java

Java. Абстрактные классы.

Кроме обычных классов в Java есть абстрактные классы. Абстрактный класс похож на обычный класс. В абстрактном классе также можно определить поля и методы, но в то же время нельзя создать объект или экземпляр абстрактного класса.
Абстрактные классы призваны предоставлять базовый функционал для классов-наследников. А производные классы уже реализуют этот функционал.

При определении абстрактных классов используется ключевое слово **abstract**:

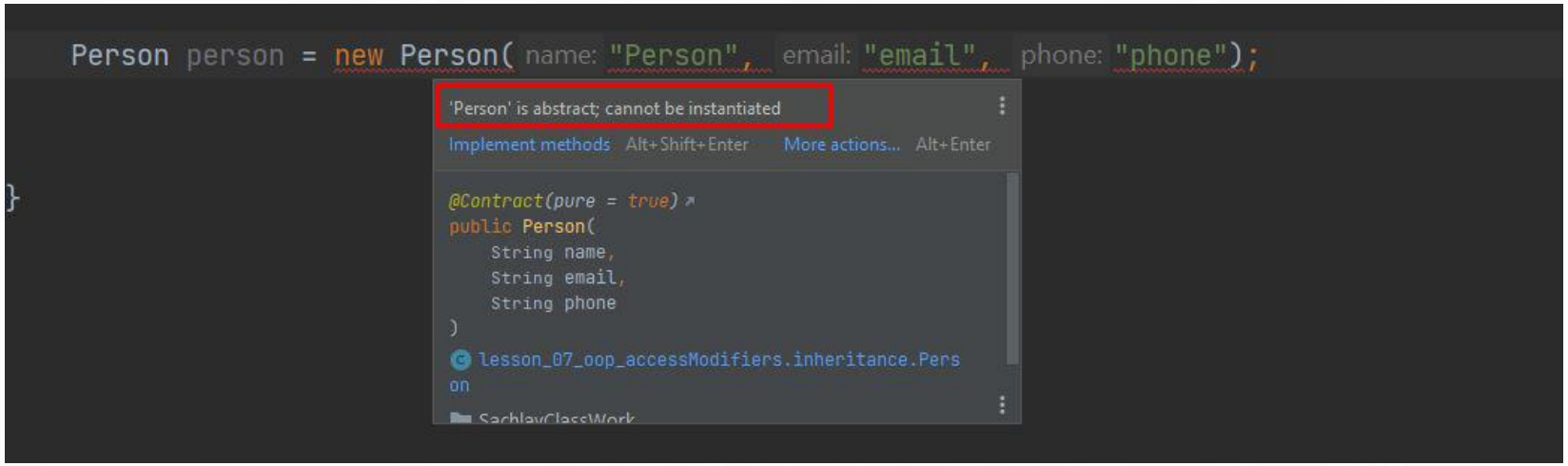
```
public abstract class Human {
    private String name;
    public String getName () {
        return name;
    }
}
```

Java

Java. Абстрактные классы.

Можно создавать класс с ключевым словом abstract даже, если в нем не имеется ни одного абстрактного метода.
Это бывает полезным в ситуациях, где в классе абстрактные методы просто не нужны, но необходимо запретить создание экземпляров этого класса.

Создавать объект на основе абстрактного класса нельзя.



Java

Java. Абстрактные классы.

Если вы хотите, чтобы класс содержал конкретный метод, но вы желаете, чтобы фактическая реализация этого метода определялась дочерними классами, вы можете объявить метод в родительском классе как абстрактный.

```
public abstract class Employee {
   private String name;
   private String address;
   private int number;

   public abstract double computePay();
   // Остаток определения класса
}
```

Java

Задание Геометрические фигуры.

Создайте иерархию классов для представления геометрических фигур: круга и прямоугольника. Каждая фигура должна иметь методы для вычисления площади и периметра. Используйте принципы наследования и полиморфизма.

- Создайте абстрактный класс Shape, который будет служить базовым классом для всех геометрических фигур. В этом классе объявите абстрактные методы calculateArea() и calculatePerimeter().
- От класса Shape унаследуйте два класса: Circle и Rectangle.
- Реализуйте класс Circle, который будет представлять круг. У него должны быть поля для радиуса. Реализуйте методы calculateArea() и calculatePerimeter() для вычисления площади и периметра круга соответственно.

Java

Задание Геометрические фигуры.

- Реализуйте класс Rectangle, представляющий прямоугольник. У него должны быть поля для ширины и высоты. Реализуйте методы calculateArea() и calculatePerimeter() для вычисления площади и периметра прямоугольника.
- Напишите класс Main, в котором создайте несколько объектов классов Circle и Rectangle. Продемонстрируйте использование полиморфизма, вызывая методы calculateArea() и calculatePerimeter() для объектов обоих типов и выводя результаты на экран.

Thanks for your time ☺