Java

Абстрактные классы

Лекция #13

Пустовалова О.Г.
доцент. каф. мат.мод.
ИММиКН ЮФУ

Содержание

АБСТРАКТНЫЕ КЛАССЫ

Зачем нужны абстрактные классы

- Идея абстрактного класса заключается в следующем предположении — для работы иногда вам требуются не полностью готовые классы, а «заготовки» (полуфабрикаты).
- Они уже кое-что умеют, но в «сыром виде» их использовать нельзя.

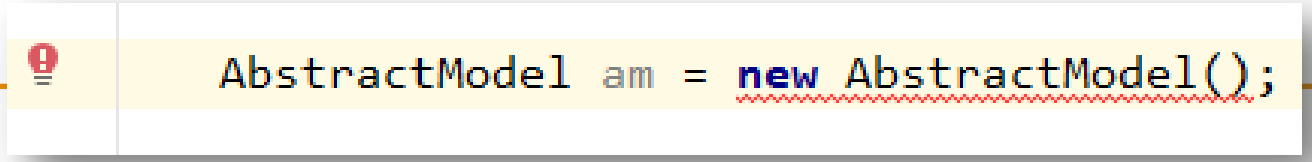
- Создать экземпляр абстрактного класса нельзя.
- Такой класс требует доработки под какие-либо конкретные условия.

Создание абстрактного класса

- Для объявления абстрактного класса достаточно добавить ключевое слово **abstract** в описании класса

```
abstract public class AbstractModel {


}
```

Если вы попробуете создать объект этого класса, то компилятор выдаст сообщение об ошибке.



Абстрактный метод

Можно описать и метод – как абстрактный.

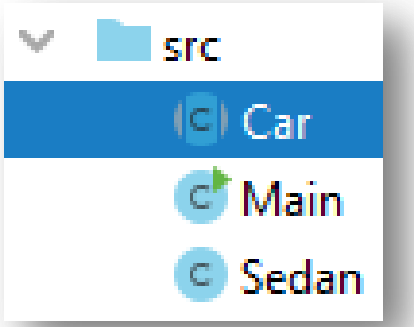
Тогда тело метода должно отсутствовать, сразу за описанием метода ставится точка с запятой.

```
public abstract class AbstractModel
{
    public abstract void processModel();
}
```

ПРИМЕР
АБСТРАКТНЫЙ КЛАСС CAR

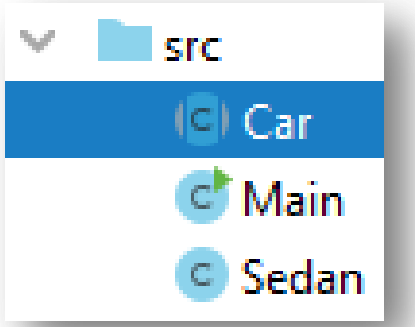
Абстрактный класс. Пример. Часть 1

```
public abstract class Car {

    private String model;
    private String color;
    private int maxSpeed;

    public abstract void gas();

    public abstract void brake();
```



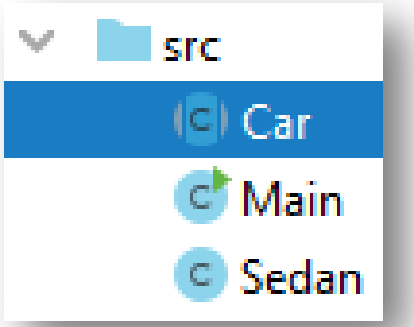
Абстрактный класс. Пример. Часть 2

```
public String getModel() {
    return model;
}

public void setModel(String model) {
    this.model = model;
}

public String getColor() {
    return color;
}
```



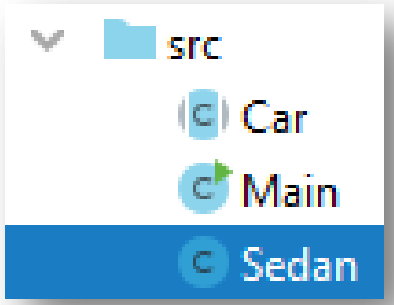
Абстрактный класс. Пример. Часть 3

```
public void setColor(String color) {
        this.color = color;
    }

    public int getMaxSpeed() {
        return maxSpeed;
    }

    public void setMaxSpeed(int maxSpeed) {
        this.maxSpeed = maxSpeed;
    }
}
```



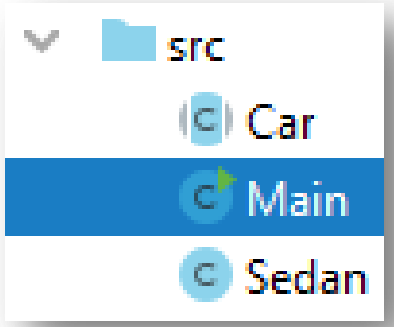
Абстрактный класс. Пример. Часть 4

```
public class Sedan extends Car {


    @Override
    public void gas() {
        System.out.println("Седан газует!");
    }


    @Override
    public void brake() {
        System.out.println("Седан тормозит!");
    }
}
```



Абстрактный класс. Пример. Часть 5

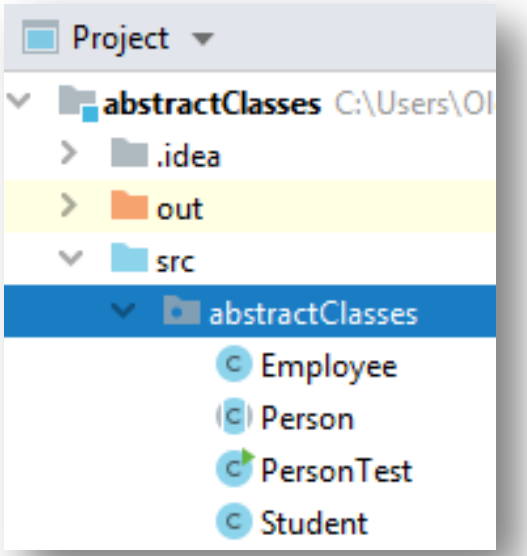
```
public class Main {

    public static void main(String[] args) {



//  Car car = new Car(); // Ошибка! Класс Car является абстрактным!
        Sedan sedan = new Sedan();
        sedan.setColor("red");
        sedan.setModel("KIA");
        sedan.setMaxSpeed(210);

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



ПРИМЕР
АБСТРАКТНЫЙ КЛАСС PERSON

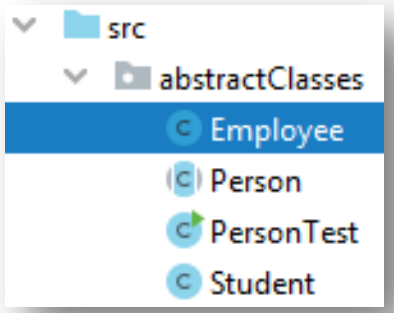
Абстрактные классы. Пример. Часть 1

```
package abstractClasses;

public abstract class Person {

    public abstract String getDescription();

    private String name;

    public Person(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

}
```



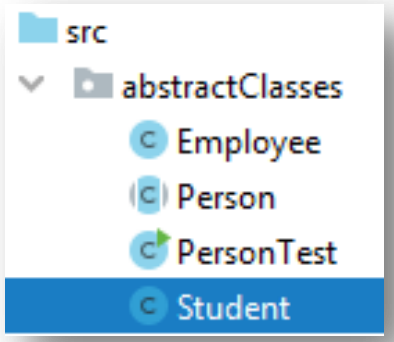
Абстрактные классы. Пример. Часть 2

```
package abstractClasses;

public class Employee extends Person {

    private double salary;

    public Employee(
            String name, double salary) {
        super(name);
        this.salary = salary;
    }

    public double getSalary() {
        return salary;
    }

    public String getDescription() {
        return String.format(
                "работник с зарплатой $%.2f", salary);
    }}
```



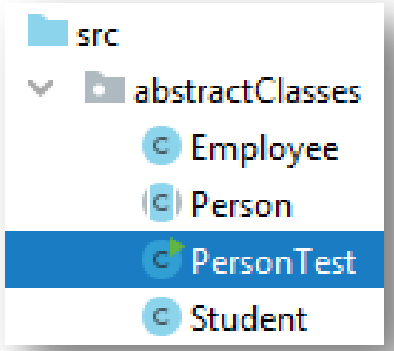
Абстрактные классы. Пример. Часть 2

```
package abstractClasses;

public class Student extends Person {

    private String major;

    /**
     * @param name  Имя студента
     * @param major Специализация студента
     */
    public Student(String name, String major) {
// передать строку name конструктору суперкласса
// в качестве его параметра
        super(name);
        this.major = major;
    }

    public String getDescription() {
        return "студент, область интересов - " + major;
    }}
```



Абстрактные классы. Пример. Часть 2

```
package abstractClasses;

public class PersonTest {

    public static void main(String[] args) {
        Person[] people = new Person[2];

// заполнить массив people объектами типа Student и Employee
        people[0] = new Employee("Иван", 500);
        people[1] = new Student("Савва", "математика");


// вывести имена и описания всех лиц,
// представленных объектами типа Person

        for (int i = 0; i < people.length ; i++) {
            System.out.println(people[i].getName() + ", "
                                        + people[i].getDescription());
        }
}}
```



Абстрактные классы

Создать экземпляры абстрактного класса нельзя!

```
new Person("Vince Vu"); //Ошибка!
```

Для абстрактных классов *можно* создавать объектные переменные, но такие переменные должны ссылаться на объект неабстрактного класса.

```
Person p = new Student("Vince Vu", "Economics");
```

Абстрактные методы являются важным понятием языка Java.

Они в основном применяются при создании *интерфейсов*

Спасибо за внимание!