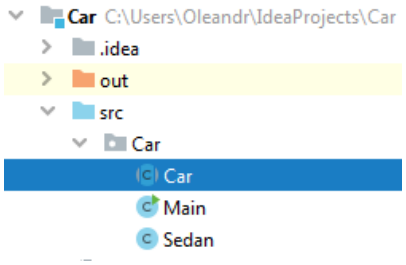
Абстрактные классы

- Идея абстрактного класса заключается в следующем предположении — для работы иногда 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.

Пример 1. Класс Car

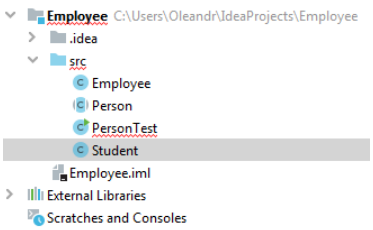



```
package Car;

public abstract class Car {
    private String model;
    private String color;
    private int maxSpeed;

    public String getModel() {
        return model;
    }

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

    public String getColor() {
        return color;
    }

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

    public int getMaxSpeed() {
        return maxSpeed;
    }

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

    public Car(String model, String color, int maxSpeed) {
        this.model = model;
        this.color = color;
        this.maxSpeed = maxSpeed;

    }
    public abstract void gas();

    public abstract void brake();
}


package Car;

public class Sedan extends Car {

    public Sedan(String model, String color, int maxSpeed) {
```

```
        super(model, color, maxSpeed);
    }

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

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


package Car;

public class Main {
    public static void main(String[] args) {
        Sedan sedan= new Sedan("Kia","red",240);

        sedan.setModel("F1");
        sedan.setColor("green");
        sedan.setMaxSpeed(1250);

        System.out.println(sedan.getModel()+" "+sedan.getColor()+"
"+sedan.getMaxSpeed());

    }
}
```


Пример 2. Абстрактный класс в пакете abstractClasses

```
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;
    }
}
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 getDescriptions(){
        return String.format("служащий с зарплатой в %2.2f ",salary);
    }
}



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 getDescriptions(){
        return String.format("студент по специализации  ",major);
    }
}

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, 2000, 10, 1);
        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());
        }


}}
```


Задания

Создать абстрактный класс **animal**. На основе класса **animal** создать классы **wildAnimal** и **domesticAnimal**. На основе классов **wildAnimal** и **domesticAnimal** создать классы **Bear**, **Fox**, **Hare**, **Cow**, **Pig**, **Goose**. Создать методы, определяющие пищу, голоса и поведение животных (Корова ест траву и дает молоко).