Интерфейсы

Чем похожи класс и интерфейс

Интерфейс схож с классом следующим образом:

- Интерфейс может содержать любое количество методов.
- Интерфейс записан в файле с расширением .java, и имя интерфейса совпадает с именем файла.
- Байт-код интерфейса находится в .class файле.
- Интерфейсы появляются в пакетах, и их соответствующий файл байт-кода должен быть в структуре каталогов, которая совпадает с именем пакета.

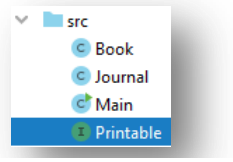
Чем отличается класс от интерфейса?

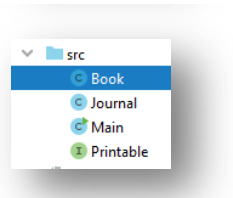
Отличие интерфейса от класса в Java:

- Вы не можете создать экземпляр интерфейса.
- В интерфейсе не содержатся конструкторы.
- Интерфейс не может содержать поля экземпляров. Поля, которые могут появиться в интерфейсе, обязаны быть объявлены и статическими, и final.
- Интерфейс не расширяется классом, он реализуется классом.
- Интерфейс может расширить множество интерфейсов.

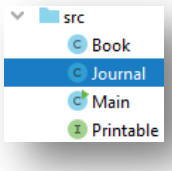
Пример 1. Несколько классов, Один интерфейс

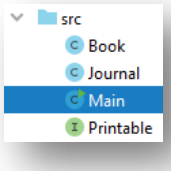

```
interface Printable{

    void print();
}



class Book implements Printable{

    String name;
    String author;

    Book(String name, String author){

        this.name = name;
        this.author = author;
    }

    public void print() {

        System.out.printf("%s (%s) \n", name, author);
    }
}
```





```
class Journal implements Printable {

    private String name;

    String getName(){
        return name;
    }

    Journal(String name){

        this.name = name;
    }
    public void print() {
        System.out.println(name);
    }
}


public class Main{

    public static void main(String[] args) {

        Printable printable = new Book("Java. Complete Reference", "H. Shildt");
        printable.print();


        printable = new Journal("Foreign Policy");
        printable.print();

    }
}
```





Пример 2. Приватные методы

По умолчанию все методы в интерфейсе фактически имеют модификатор public.

Однако начиная с Java 9 можно определять в интерфейсе методы с модификатором private.

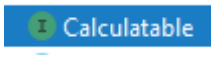
Они могут быть статическими и нестатическими, но они не могут иметь реализации по умолчанию.

Подобные методы могут использоваться только внутри самого интерфейса, в котором они определены.

Если необходимо выполнять в интерфейсе некоторые повторяющиеся действия, и в этом случае такие действия можно выделить в приватные методы.

```
public class Main{

    public static void main(String[] args) {

        Calculatable c = new Calculation();
        System.out.println(c.sum(1, 2));
        System.out.println(c.sum(1, 2, 4));
    }
}

class Calculation implements Calculatable{
```

```
}


interface Calculatable{

    default int sum(int a, int b){
        return sumAll(a, b);
    }
    default int sum(int a, int b, int c){
        return sumAll(a, b, c);
    }

    private int sumAll(int... values){
        int result = 0;
        for(int n : values){
            result += n;
        }
        return result;
    }
}
```

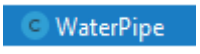


Пример 3. Статические константы

Кроме методов в интерфейсах могут быть определены статические константы.

Хотя такие константы также не имеют модификаторов, но по умолчанию они имеют модификатор доступа public static final, и поэтому их значение доступно из любого места программы.

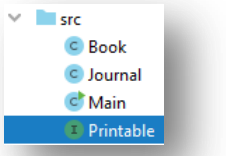

```
interface Stateable{

    int OPEN = 1;
    int CLOSED = 0;

    void printState(int n);
}


class WaterPipe implements Stateable{

    public void printState(int n){
        if(n==OPEN)
            System.out.println("Water is opened");
        else if(n==CLOSED)
            System.out.println("Water is closed");
        else
            System.out.println("State is invalid");
    }
}


public class Main{

    public static void main(String[] args) {
```



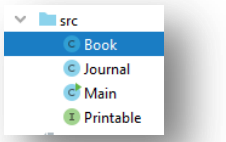
```
        WaterPipe pipe = new WaterPipe();
        pipe.printState(1);
    }
}
```

Пример 4.
Интерфейсы как параметры и результаты методов

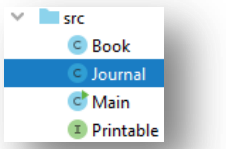
```
interface Printable{

    void print();
}
```



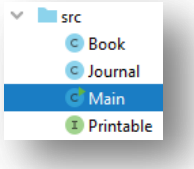

```
class Book implements Printable{

    String name;
    String author;

    Book(String name, String author){

        this.name = name;
        this.author = author;
    }

    public void print() {

        System.out.printf("%s (%s) \n", name, author);
    }
}
```




```
class Journal implements Printable {

    private String name;

    String getName(){
        return name;
    }

    Journal(String name){

        this.name = name;
    }
    public void print() {
        System.out.println(name);
    }
}
```




```
public class Main{
    static void read(Printable p){

        p.print();
    }
```



```
    static Printable createPrintable(String name, boolean option){

        if(option)
            return new Book(name, "Undefined");
        else
            return new Journal(name);
    }

    public static void main(String[] args) {

        Printable printable = createPrintable("Foreign Affairs",false);
        printable.print();

        read(new Book("Java for impatients", "Cay Horstmann"));
        read(new Journal("Java Dayly News"));
    }


}
```

Задания

1. Создать интерфейс **Voiseable** и реализующие его
   классы Кот, Собака и Корова. Интерфейс **Voiseable** содержит метод voise(). Вызвать метод voise() для каждого животного.
2. Создать интерфейс **Музыкальный Инструмент** и реализующие его
   классы Гитара, Барабан и Труба. Интерфейс **Музыкальный Инструмент** содержит метод play() и переменную String KEY ="До мажор". Гитара содержит переменные класса **количество Струн**, Барабан - **размер**, Труба - **диаметр**. Создать массив типа **Музыкальный Инструмент**, содержащий инструменты разного типа. В цикле вызвать метод play() для каждого инструмента, который должен выводить строку "Играет такой-то инструмент с такими то характеристиками".
3. Определить интерфейс Printable, содержащий метод void print().
   Определить класс Book, реализующий интерфейс Printable.
   Определить класс Magazine, реализующий интерфейс Printable.
   Создать массив типа Printable, который будет содержать книги и журналы.
   В цикле пройти по массиву и вызвать метод print() для каждого объекта.