ЛАБОРАТОРНАЯ РАБОТА №4
ИНТЕРФЕЙСЫ В JAVA.

ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ:

Цель данной лабораторной работы - изучить понятие интерфейса, научиться создавать интерфейсы в Java и применять их в программах.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ:

Механизм наследования очень удобен, но он имеет свои ограничения. В частности мы можем наследовать только от одного класса, в отличие, например, от языка C++, где имеется множественное наследование.

В языке Java подобную проблему позволяют решить интерфейсы. Интерфейсы определяют некоторый функционал, не имеющий конкретной реализации, который затем реализуют классы, применяющие эти интерфейсы. И один класс может применить множество интерфейсов.

Чтобы определить интерфейс, используется ключевое слово interface.

Определим следующий интерфейс:

```
public interface Printable{

    void print();
}
```

Интерфейс может определять различные методы, которые, так же как и абстрактные методы абстрактных классов не имеют реализации. В данном случае объявлен только один метод.

Все методы интерфейса не имеют модификаторов доступа, но фактически по умолчанию доступ public, так как цель интерфейса - определение функционала для реализации его классом. Поэтому весь функционал должен быть открыт для реализации.

И также при объявлении интерфейса надо учитывать, что только один интерфейс в файле может иметь тип доступа public. А его

название должно совпадать с именем файла. Остальные интерфейсы (если такие имеются в файле java) не должны иметь модификаторов доступа.

Интерфейс может определять различные методы, которые, так же как и абстрактные методы абстрактных классов не имеют реализации. В данном случае объявлен только один метод.

Все методы интерфейса не имеют модификаторов доступа, но фактически по умолчанию доступ public, так как цель интерфейса - определение функционала для реализации его классом. Поэтому весь функционал должен быть открыт для реализации.

И также при объявлении интерфейса надо учитывать, что только один интерфейс в файле может иметь тип доступа public. А его название должно совпадать с именем файла. Остальные интерфейсы (если такие имеются в файле java) не должны иметь модификаторов доступа.

Чтобы класс применил интерфейс, надо использовать ключевое слово implements:

```
class Book implements Printable{

    String name;
    String author;
    int year;

    Book(String name, String author, int year){
        this.name = name;
        this.author = author;
        this.year = year;
    }

    public void print() {

        System.out.printf("Книга '%s' (автор %s) была издана в %d году \n", name, author, year);
    }
}
```

При этом надо учитывать, что если класс применяет интерфейс, то он должен реализовать все методы интерфейса, как в случае выше реализован метод print.

Потом в главном классе мы можем использовать данный класс и его метод print:

```
Book b1 = new Book("Война и мир", "Л. Н. Толстой", 1863);
b1.print();
```

В тоже время мы не можем напрямую создавать объекты интерфейсов, поэтому следующий код не будет работать:

```
Printable pr = new Printable();
pr.print();
```

Одним из преимуществ использоваия интерфейсов является то, что они позволяют добавить в приложение гибкости. Например, в дополнение к классу Book определим еще один класс, который будет реализовывать интерфейс Printable:

```
public class Journal implements Printable {

    private String name;

    String getName(){
        return name;
    }

    Journal(String name){

        this.name = name;
    }
    public void print() {
        System.out.printf("Журнал '%s'\n", name);
    }
}
```

Класс Book и класс Journal связаны тем, что они реализуют интерфейс Printable. Поэтому мы динамически в программе можем создавать объекты Printable как экземпляры обоих классов:

```
Printable printable = new Book("Война и мир", "Л. Н. Толстой", 1863);
```

```
printable.print();
printable = new Journal("Хакер");
printable.print();
```

И также как и в случае с классами, интерфейсы могут использоваться в качестве типа параметров метода или в качестве возвращаемого типа:

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

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

    read(new Book("Отцы и дети", "И. Тургенев", 1862));
    read(new Journal("Хакер"));
}

static void read(Printable p){

    p.print();
}

static Printable createPrintable(String name, boolean option){

    if(option)
        return new Book(name, "неизвестен", 2015);
    else
        return new Journal(name);
```

Метод read() в качестве параметра принимает объект интерфейса Printable, поэтому в этот метод мы можем передать как объект Book, так и объект Journal.

Метод createPrintable() возвращает объект Printable, поэтому также мы вожем возвратить как объект Book, так и Journal.

ВАРИАНТЫ ЗАДАНИЙ

1. Создать интерфейс Nameable, с методом getName(), возвращающим имя объекта, реализующего интерфейс. Проверить работу для различных объектов (например, можно создать классы, описывающие

разные сущности, которые могут иметь имя: планеты, машины, животные и т. д.).

2. Реализовать интерфейс Priceable, имеющий метод getPrice(), возвращающий некоторую цену для объекта. Проверить работу для различных классов, сущности которых могут иметь цену.