Практическая работа №7. Реализация интерфейсов

Цель: цель данной практической работы – научится разрабатывать практике пользовательские интерфейсы, и применять их в программах на языке Джава.

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 Джава подобную проблему позволяют решить интерфейсы. Интерфейсы определяют некоторый функционал, не имеющийконкретной реализации, который затем реализуют классы, применяющие эти интерфейсы. И один класс может применитьмножество интерфейсов.

Чтобы определить интерфейс, используется ключевое словоinterface. 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. Вданном 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. Вданном 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 названиедолжно 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:

Листинг 7.1 – Пример 1 реализации интерфейса

```
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 иего метод print:

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

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

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

Одним из преимуществ использования 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:

Листинг 7.2 – Пример 2 реализации интерфейса

```
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();//для одного объекта
    printable = new Journal("Хакер");
    printable.print();//для другого объекта
```

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

Листинг 7.3 – Пример 3 реализации интерфейса

```
    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"));
    }
    //статический метод класса
    static void read(Printable p){
    p.print();
    }
    //статический метод
    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 такжемы можем возвратить как объект Book, так и Journal.

Статические методы интерфейса

Интерфейс Java может иметь статические методы. Статические методы в интерфейсе Java должны иметь реализацию, в отличие от обычных методов. Вот пример статического метода в интерфейсе Java:

Листинг 7.4 – Пример интерфейса со статическим методом

```
public interface MyInterface {

    public static void print (String text) {
        System.out.print (текст);
    }
}
```

Вызов статического метода в интерфейсе выглядит и работает так же, как вызов статического метода в классе. Вот пример вызова статического print() метода из MyInterface интерфейса выше :

```
MyInterface.print ("Привет, статический метод!");
```

Статические методы в интерфейсах могут быть полезны, когда у вас есть некоторые служебные методы, которые вы хотели бы сделать доступными, которые естественным образом вписываются в интерфейс, связанный с той же ответственностью.

Например, Vehicle интерфейс может иметь статический метод `printVehicle(Vehicle v)`.

Задания на практическую работу №7

1. Создайте в draw.io UML диаграмму и напишите по ней реализацию. Диаграмма должна включать в себя следующие элементы: интерфейс Movable, содержащий в себе методы для движения прямоугольника (вверх, вниз, влево, вправо) и класс MovableRectangle (движущийся прямоугольник), реализующий интерфейс Movable.

2. Напишите по диаграмме класс MovableRectangle (движущийся прямоугольник), реализующий интерфейс Movable, класс прямоугольник,

который можно представить как две движущиеся точки MovablePoint (верхняя левая и нижняя правая точки – topLeft и bottomRight), также реализующие интерфейс Movable;

3. Добавьте в класс параметризированные конструкторы, входящие в состав классов; метод в классах для перевода числовых значений в Строку. Убедитесь, что две точки имеют одну и ту же скорость при помощи специального логического метода SpeedTest(), проверяющего это.

4. Разработайте интерфейс MathCalculable, который содержит объявления математических функций: возведения в степень и модуль комплексного числа, также содержит число PI. Напишите класс MathFunc, который реализует, реализует этот интерфейс. Например, вычисления длины окружности, для чего используйте число PI из интерфейса. Протестируйте класс

Замечание:

```
MathCalculable mc1 = new MathFunc(); // правильно
MathCalculable mc2 = new MathCalculable ();
//ошибка - запрещено объявлять экземпляр интерфейса
```

5. Разработайте интерфейс для работы со строками, который содержит а) функции подсчета символов в строке б) функция возвращает строку, которая образовывает строку, состоящую из символов исходной строки s, которые размещены на нечетных позициях: 1, 3, 5, ...в) функцию инвертирования строки

6. Реализуйте интерфейс в классе ProcessStrings и протестируйте работу класса

7. Создать статический метод printMagazines(Printable[] printable) в классе Magazine, который выводит на консоль названия только журналов.

8. Создать статический метод printBooks(Printable[] printable) в классе Book, который выводит на консоль названия только книг. Используем оператор instanceof.

9. Представьте отчет преподавателю на проверку