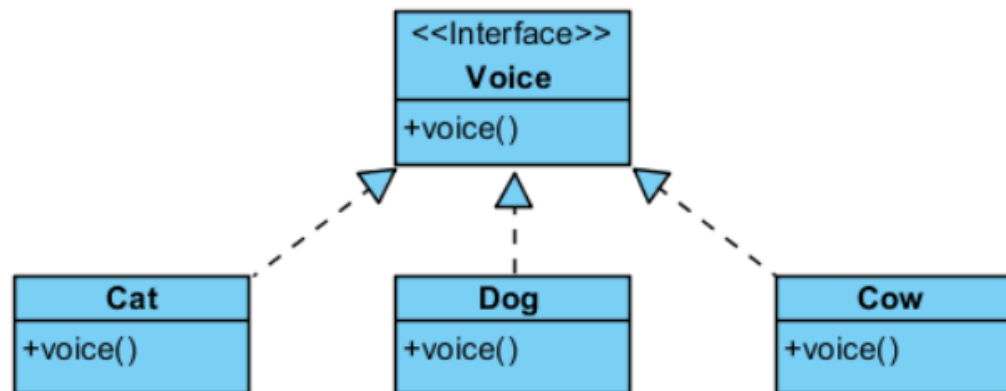


Задача 1. Напишите программу, реализующую следующую диаграмму классов:



Переопределяемые методы `voice()` выводят соответствующую строку на консоль.

Код программы:

```
package interfacevoice;
public class InterfaceVoice {

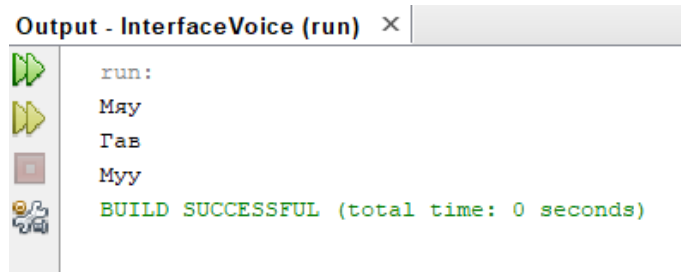
    public static void main(String[] args) {
        //Создание объектов классов и вызов переопределенных
методов voice()
        Cat cat = new Cat();
        cat.voice();
        Dog dog = new Dog();
        dog.voice();
        Cow cow = new Cow();
        cow.voice();
    }
}

//Создание интерфейса
interface Voice {
    public void voice();
}

class Cat implements Voice {
    @Override //Переопределение метода voice()
    public void voice() {
        System.out.println("Мяу");
    }
}
class Dog implements Voice {
    @Override //Переопределение метода voice()
    public void voice() {
        System.out.println("Гав");
    }
}
}
```

```
class Cow implements Voice {
    @Override //Переопределение метода voice()
    public void voice() {
        System.out.println("Муу");
    }
}
```

Вывод:



Переопределение методов позволяет подклассу или дочернему классу обеспечивать специфическую реализацию метода, уже реализованного в одном из суперклассов или родительских классов.

2. Переработайте задачу о игре в кости с использованием интерфейсов. В игре участвуют N игроков, где компьютер находится последним в списке. Одновременно подбрасываются K кубиков, и выигрывает тот, у кого больше сумма очков. Победитель начинает бросать в следующем раунде, а игра продолжается до достижения 7 побед. Игра начинается с вас.

В код предыдущей работы по игре в кости был добавлен интерфейс, методы которого реализованы в обновленном классе игрока.

Код программы:

Интерфейс игрока

//интерфейс игрока

```
public interface PlayerInterface{
    public void setsum(int s);
    public int getsum();
    public void setwin(int w);
    public int getwin();
    public void printPlayer();
}
```

Класс игрока реализующий методы интерфейса PlayerInterface

```
//класс игрока реализующий методы интерфейса PlayerInterface
static class Player implements PlayerInterface{

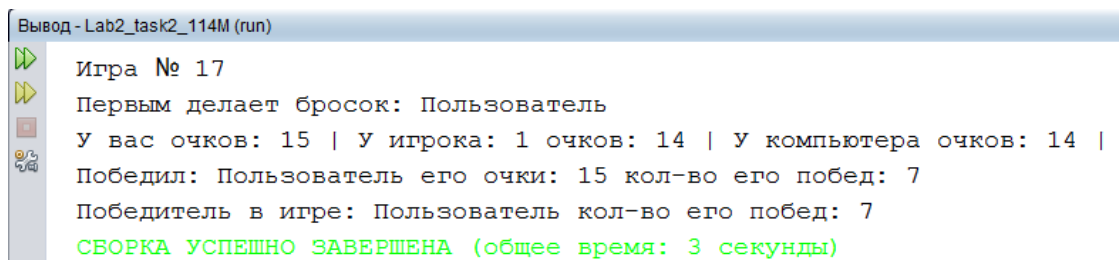
    private int win; //кол-во побед игрока
    private int sumscore; //сумма очков за раунд
```

```

Player(int win, int sumscore) {
    this.win = win;
    this.sumscore = sumscore;
}
public void setsum(int sumscore) {
    this.sumscore = sumscore;
}
public int getsum() {
    return sumscore;
}
public void setwin(int win) {
    this.win += win;
}
public int getwin() {
    return win;
}
public void printPlayer(){
    System.out.println("кол-во побед: "+this.win+",
сумма очков: "+this.sumscore);
}

```

Вывод:

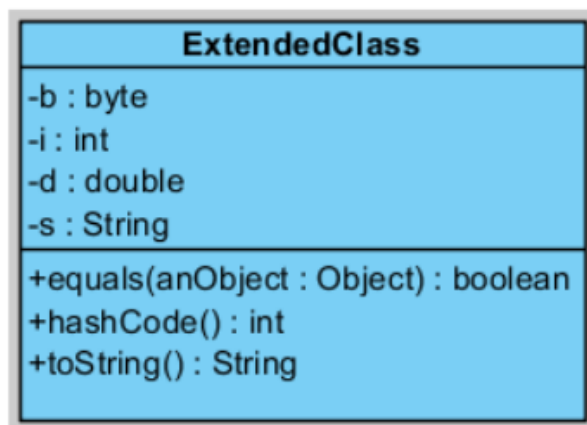


```

Вывод - Lab2_task2_114M (run)
Игра № 17
Первым делает бросок: Пользователь
У вас очков: 15 | У игрока: 1 очков: 14 | У компьютера очков: 14 |
Победил: Пользователь его очки: 15 кол-во его побед: 7
Победитель в игре: Пользователь кол-во его побед: 7
СБОРКА УСПЕШНО ЗАВЕРШЕНА (общее время: 3 секунды)

```

3. Напишите программу, реализующую класс:



Код программы:

Главный метод main

```

public static void main(String[] args) {
    ExtendedClass ex1 = new ExtendedClass((byte)3, 303,
                                           33.33, "stroka");
}

```

```

ExtendedClass ex2 = new ExtendedClass((byte)5, 505,
                                       55.55, "stroka");
ExtendedClass ex3 = new ExtendedClass((byte)3, 303,
                                       33.33, "stroka");

//сравнение экземпляров класса
System.out.println(ex1.equals(ex3)); //true
System.out.println(ex1.equals(ex2)); //false
System.out.println(ex2.equals(ex3)); //false
//Вывод хэш-кодов
System.out.println(ex1.hashCode());
System.out.println(ex2.hashCode());
System.out.println(ex3.hashCode());
//сравнение хэш-кодов
System.out.println(ex1.hashCode() == ex3.hashCode());
//true
System.out.println(ex1.hashCode() ==
                   ex2.hashCode()); //false
System.out.println(ex2.hashCode() == ex3.hashCode());
//false

//Вывод объектов
System.out.println(ex1.toString()); //3, 303, 33.33,
                                     stroka
System.out.println(ex2.toString()); //5, 505, 55.55,
                                     stroka
System.out.println(ex3.toString()); //3, 303, 33.33,
                                     stroka
}

```

Создание класса с переопределенными методами equals, hashCode и toString

```

//класс с переопределенными методами equals, hashCode и toString
byte b;
int i;
double d;
String s;

public ExtendedClass(byte b, int i, double d, String s) {
    this.b=b;
    this.i = i;
    this.d = d;
    this.s = s;
}

```

Переопределение метода сравнения

```

//переопределение метода сравнения
@Override
public boolean equals(Object obj) {
    if (obj == this) {
        return true;
    }
    if (obj == null || obj.getClass() != getClass()) {

```

```

        return false;
    }
    ExtendedClass ex = (ExtendedClass) obj;
    return ex.b==b
           && ex.i == i
           && ex.d == d
           && ex.s.equals(s);
}

```

Переопределение метода получения хэш-кода

```

//переопределение метода получения хэш-кода
@Override
public int hashCode() {
    int result = 17;
    // 31 используется, поскольку это простое число, а
    // также обеспечивает более высокую производительность, т.к.:
    // 31 * result == (result << 5) - result
    result = 31 * result + b;
    result = 31 * result + i;
    result = (int) (31 * result + d);
    result = 31 * result + (s == null ? 0 :
s.hashCode());
    return result;
}

```

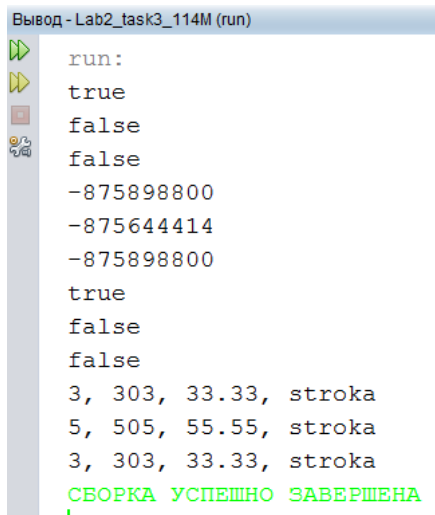
Переопределение метода преобразования в строку

```

//переопределение метода преобразования в строку
@Override
public String toString() {
    return this.b+", "+this.i+", "+this.d+", "+s;
}
}

```

Вывод:



```

Вывод - Lab2_task3_114M (run)
run:
true
false
false
-875898800
-875644414
-875898800
true
false
false
3, 303, 33.33, stroka
5, 505, 55.55, stroka
3, 303, 33.33, stroka
СБОРКА УСПЕШНО ЗАВЕРШЕНА

```

4. Классы, которые требуется реализовать в задании, не содержат метода `main()` и должны быть instantiated (объект класса) в методе `main()` основного класса.

Создайте интерфейс Sleepy с методами sleep(), wakeUp() и ask(). Реализуйте этот интерфейс в классе SleepyImpl. Метод sleep() устанавливает флаг awake в значение false, метод wakeUp() — в значение true. Метод ask() выводит на консоль «BOO!», если флаг равен true, и «zzz...» в противном случае.

Код программы:

Главный метод main

```
public static void main(String[] args) {
    //создание экземпляра класса SleepyImpl,
    //который реализует методы интерфейса Sleepyc
    //по умолчанию флаг awake=true
    SleepyImpl sleep = new SleepyImpl(true);
    //вызов метода вывода информации
    sleep.ask(); //вывод: "BOO!"
    sleep.sleep(); //вызов метода спать
    sleep.ask(); //вывод: "zzz..."
    sleep.wakeUp(); //вызов метода проснуться
    sleep.ask(); //вывод: "BOO!"
}
```

Создание интерфейса Sleepyc

```
//создан интерфейс и методы в нем
public interface Sleepyc{
    void sleep(); //метод спать
    void wakeUp(); //метод проснуться
    void ask(); //метод вывода информации
}
```

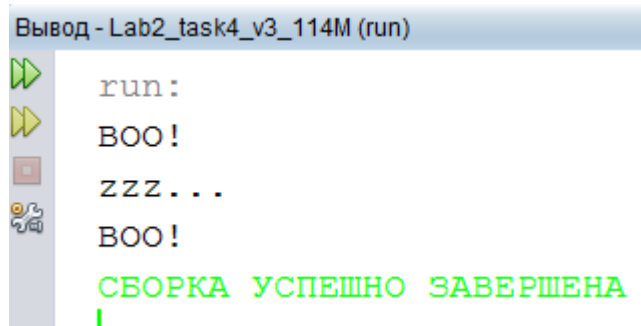
Создание класса, реализующего методы интерфейса Sleepyc

```
//класс, реализующий методы интерфейса Sleepyc
public static class SleepyImpl implements Sleepyc{
    boolean awake; //флаг: спит или нет

    SleepyImpl(boolean awake){ //конструктор класса
        this.awake=awake;
    }
    //реализация методов интерфейса Sleepyc
    //метод спать
    public void sleep() {
        this.awake=false;
    }
    //метод проснуться
    @Override
    public void wakeUp() {
        this.awake=true;
    }
    //метод вывода информации
}
```

```
@Override
public void ask() {
    if (this.awake==true){
        System.out.println("BOO!");
    }else{
        System.out.println("zzz...");
    }
}
```

Вывод:



```
Вывод - Lab2_task4_v3_114M (run)
run:
BOO!
ZZZ...
BOO!
СБОРКА УСПЕШНО ЗАВЕРШЕНА
|
```