

语言基础第六天：

回顾：

1. break：跳出循环
continue：跳过循环体中剩余语句而进入下一次循环
2. 嵌套循环：
 - 循环中套循环，外层循环走一次，内层循环走所有次，嵌套层数越少越好，
 - break只能跳出当前一层循环
3. 数组：
 - 引用数据类型、相同数据类型元素的集合

```
int[] arr = new int[3]; //0,0,0
int[] arr = {2,3,7};
int[] arr = new int[]{2,3,7};
arr[0] = 100;
System.out.println(arr[arr.length-1]);
for(int i=0;i<arr.length;i++){
    arr[i] = (int)(Math.random()*100);
    System.out.println(arr[i]);
}
Arrays.sort(arr); //升序
```

精华笔记：

1. 数组(下):
 - 复制：
 - System.arraycopy(a,1,b,0,4);
 - int[] b = Arrays.copyOf(a,6);
 - a = Arrays.copyOf(a,a.length+1); //数组的扩容
2. 方法：函数、过程
 - 作用：用于封装一段特定的业务逻辑功能
 - 建议：尽可能独立，一个方法只干一件事
 - 方法可以被反复多次调用
 - 好处：可以减少代码重复，有利于代码维护
 - 何时用：只要是一个独立的业务功能，就得封装到一个方法中
3. 方法的定义：五要素

```
修饰词 返回值类型 方法名(参数列表){
    方法体-----具体的业务逻辑功能实现
}
```

4. 方法的调用：
 - 无返回值：方法名(有参传参);
 - 有返回值：数据类型 变量 = 方法名(有参传参);

5. return:

- return 值; //1)结束方法的执行 2)返回结果给调用方-----用在有返回值的方法中
- return; //1)结束方法的执行-----用在无返回值的方法中

6. 方法的重载(overloading):

- 发生在同一类中, 方法名相同, 参数列表不同
- 编译器在编译时会根据方法的签名自动绑定方法

笔记:

1. 数组(下):

- 复制:

```
int[] a = {10,20,30,40,50};
int[] b = new int[6]; //0,0,0,0,0,0
//a:源数组
//1:源数组的起始下标
//b:目标数组
//0:目标数组的起始下标
//4:要复制的元素个数
System.arraycopy(a,1,b,0,4); //灵活性好
for(int i=0;i<b.length;i++){
    System.out.println(b[i]);
}
```

```
int[] a = {10,20,30,40,50};
//a:源数组
//b:目标数组
//6:目标数组的长度
// --若目标数组长度>源数组长度,则末尾补默认值
// --若目标数组长度<源数组长度,则将末尾的截掉
int[] b = Arrays.copyOf(a,6); //灵活性差
for(int i=0;i<b.length;i++){
    System.out.println(b[i]);
}

int[] a = {10,20,30,40,50};
//数组的扩容(创建了一个更大的新的数组,并将数据复制进去了)
a = Arrays.copyOf(a,a.length+1);
for(int i=0;i<a.length;i++){
    System.out.println(a[i]);
}
```

```
package day06;
import java.util.Arrays;
//求数组元素的最大值,并将其存储到数组最后一个元素的下一个位置
public class MaxOfArray {
    public static void main(String[] args) {
        int[] arr = new int[10];
        for(int i=0;i<arr.length;i++){
            arr[i] = (int)(Math.random()*100);
            System.out.println(arr[i]);
        }
    }
}
```

```

    }

    int max = arr[0]; //假设第1个元素为最大值
    for(int i=1;i<arr.length;i++){ //遍历剩余元素
        if(arr[i]>max){ //若剩余元素大于max
            max = arr[i]; //将max修改为较大的
        }
    }
    System.out.println("最大值为:"+max);

    arr = Arrays.copyOf(arr,arr.length+1); //扩容
    arr[arr.length-1] = max; //将最大值max赋值到最后一个元素上
    for(int i=0;i<arr.length;i++){
        System.out.println(arr[i]);
    }

}
}

```

2. 方法：函数、过程

- 作用：用于封装一段特定的业务逻辑功能
- 建议：尽可能独立，一个方法只干一件事
- 方法可以被反复多次调用
- 好处：可以减少代码重复，有利于代码维护
- 何时用：只要是一个独立的业务功能，就得封装到一个方法中

3. 方法的定义：五要素

```

修饰词 返回值类型 方法名(参数列表){
    方法体-----具体的业务逻辑功能实现
}

```

```

//无参无返回值
public static void say(){
    System.out.println("大家好，我叫WKJ，今年38岁了");
}

//有参无返回值
public static void sayHi(String name){ //-----形参
    System.out.println("大家好，我叫"+name+"，今年38岁了");
}

//有参有返回值
public static void sayHello(String name,int age){ //-----形参
    if(age>=60){ //在某种特定条件下，提前结束方法
        return; //结束方法
    }
    System.out.println("大家好，我叫"+name+"，今年"+age+"岁了");
}

//无参有返回值
public static double getNum(){
    //在有返回值的方法中，必须得通过return来返回数据
    //return; //编译错误，return后必须跟一个数据
}

```

```

    //return "abc"; //编译错误, return后数据的类型必须与返回值类型匹配
    return 8.88; //1)结束方法的执行 2)返回结果给调用方
}

//有参有返回值
public static int plus(int num1,int num2){
    int num = num1+num2;
    return num; //返回的是num里面的那个数
    //return num1+num2; //返回的是num1与num2的和
}

//生成一个整型数组,并填充随机数据
public static int[] generateArray(int len,int max){
    Random rand = new Random();
    int[] arr = new int[len];
    for(int i=0;i<arr.length;i++){
        arr[i] = rand.nextInt(max+1);
    }
    return arr;
}

```

4. 方法的调用:

- 无返回值: 方法名(有参传参);

```

public static void main(String[] args) {
    say(); //调用say()方法

    //sayHi(); //编译错误, 有参则必须传参
    //sayHi(250); //编译错误, 参数类型必须匹配
    sayHi("zhangsan"); //String name="zhangsan" //-----实参
    sayHi("lisi"); //String name="lisi" //-----实参

    sayHello("zhangsan",25); //-----实参
    sayHello("lisi",27); //-----实参
}

```

- 有返回值: 数据类型 变量 = 方法名(有参传参);

```

public static void main(String[] args) {
    double a = getNum(); //getNum()的值就是return后的那个数
    System.out.println(a); //8.88---模拟对返回值的后续操作

    int b = plus(5,6);
    System.out.println(b); //11---模拟对返回值的后续操作

    int m=5,n=6;
    int c = plus(m,n); //传的是m,n里面的数
    System.out.println(c); //11---模拟对返回值的后续操作

    int[] d = generateArray(5,100); //模拟第1个人的访问
    System.out.println("数组的长度为:"+d.length); //10---模拟对返回值的后续操作

    for(int i=0;i<d.length;i++){ //---模拟对返回值的后续操作
        System.out.println(d[i]);
    }
}

```

```

    }

    int[] e = generateArray(8,20); //模拟第2个人的访问
    System.out.println("第1个元素的值:"+e[0]); //?---模拟对返回值的后续操作
    for(int i=0;i<e.length;i++){ //---模拟对返回值的后续操作
        System.out.println(e[i]);
    }
}

```

5. return:

- return 值; //1)结束方法的执行 2)返回结果给调用方-----用在有返回值的方法中
- return; //1)结束方法的执行-----用在无返回值的方法中

6. 方法的重载(overloading):

- 发生在同一类中，方法名相同，参数列表不同
- 编译器在编译时会根据方法的签名自动绑定方法

```

package day06;
public class MethodDemo {
    public static void main(String[] args) {
        say(); //自动绑定无参show
        say("zhangsan"); //自动绑定String参的
        say("lisi",25); //自动绑定String+int参的
        //say(3.456); //编译错误，没有double参的say方法
    }

    //无参无返回值
    public static void say(){
        System.out.println("大家好，我叫WKJ，今年39岁了");
    }

    //有参无返回值
    public static void say(String name){ //-----形参
        System.out.println("大家好，我叫"+name+"，今年39岁了");
    }

    //有参无返回值
    public static void say(String name,int age){ //-----形参
        if(age>=50){ //在某种特定条件下，提前结束方法
            return; //结束方法的执行
        }
        System.out.println("大家好，我叫"+name+"，今年"+age+"岁了");
    }

    public static void say(int age) {} //正确的重载
    public static void say(int age,String name){} //正确的重载
    //public static int say(){ return 1; } //编译错误，重载与返回值类型无关
    //public static void say(String address) {} //编译错误，重载与参数名称无
    关
}

```

补充:

1. 形参：形式参数，定义方法时的参数为形参
实参：实际参数，调用方法时的参数为实参
2. 方法的签名：方法名+参数列表
3. 明日单词：

```
1)price: 价格
2)score: 分数
3)total: 总共
4)avg: 平均
5)discount: 折扣
6)generate: 生成
7)index: 下标/索引
```

练习:

```
public static void main(String[] args){
    say();
    say("zhangsan");
    say("zhangsan",25);
    double a = getNum(); //输出a--模拟对返回值的后续操作
    int b = plus(5,6); //输出b--模拟对返回值的后续操作
    int m=50,n=60; int c = plus(m,n); //输出c--模拟对返回值的后续操作
    int[] d = generateArray(5,100); //输出d的长度，输出每个元素的值-模拟对返回值的后续操作
    int[] e = generateArray(3,10); //输出e中每个元素的值-模拟对返回值的后续操作
}

public static void say(){ 固定的问好 }
public static void say(String name){ 问好，名字写活了 }
public static void say(String name,int age){ 问好，名字和年龄都写活了}
public static double getNum(){ return 8.88; }
public static int plus(int num1,int num2){ return num1+num2; }
public static int[] generateArray(int len,int max){
    int[] arr = new ...; ...; return arr;
}
```