JAVA 学习笔记

1、面对对象编程的特点

2、private public protected, 的区别

3、简述 Java this关键字

4、简述 Java super关键字

5、类的实例化过程

6、static ， final

7、接口(interface)和抽象类(abstactclass)的区别

8、什么是多态

9、“==” 和 equals() 的区别

10、hashCode()

11、ArrayList, HashSet, TreeSet

12、compareTo(Object o)

13、Iterator 迭代器

14、foreach 遍历集合

15、toString()

16、throw 与 throws

17、String类

<1>String[] split(String regex)

<2>String[] split(String regex, int limit)

<3>String substring

<4>indexOf

18、Math.random()

19、关于输入输出流

<1> BufferedReader（Reader in）

<2>BufferedWriter(Writer out);

20、定制排序方法 compare(Object o1, Object o2)

1、面对对象编程的特点

继承 封装 多态

2、private public protected, 的区别

private：私有的，仅当前对像内部可以访问。

public： 共有的，当前对象，包内，保间均可以访问。

protected：受保护的，继承对象可以访问，包内可以访问，包间无法访问。
（friendly 不写修饰符默认为此： 当前对象，包内可以访问，子内包间无法访问）

作用域	当前类	同一package	子孙类	其他package
public	√	√	√	√
protected	√	√	√	×
friendly	√	√	×	×
private	√	×	×	×

3、简述 Java this关键字

this：Java关键字，指向当前对象的引用，认为是当前对像即可，‘.’可以访问或者是调用方法，属性。
例： this(); ------------调用当前对象的构造方法。
this(1)； ----------调用当前对象有参数的构造方法并传递参数为1。
this.value = 1;-----将当前对象的value属性赋值为1.
this.output();------调用当前对象的无参方法output()。
.....

4、简述 Java super关键字

super：用于子类中指向父类的引用，认为是当前子类对象的父类对象即可，‘.’操作与this一样。
例： super(); ------------调用父类对象的构造方法。
super(1)； ----------调用父类对象有参数的构造方法并传递参数为1。
super.value = 1;-----将父类对象的value属性赋值为1.
super.output();------调用父类对象的无参方法output()。
.....

5、类的实例化过程

例： ListArray<> L = new ListArray<int>();
实例化过程： 1、在方法区加载LiatArray类。
2、在栈区创建一个对象的引用 L。
3、在堆区创建对象的实例，并将该实例与栈区引用L实现动态绑定。

6、static ， final

static ： 静态的，该关键字定义的属性，方法，类，都会直接加载到方法区，不需要被包含对象的的实例即可直接使用。

final：定义常量，final定义的属性，方法，类都会以常量形式加载到常量池里面，属性不可以修改，方法不可以重写，类不可以被继承，不能被实例化。

7、接口(interface)和抽象类(abstactclass)的区别

抽象类里面含有抽象方法，也就是说也可以含有非抽象方法。抽象方法需要子类去实现，而非抽象方法不需要去实现，但是可以重写。

接口里面全部是方法，这些方法为子类提供了大致的方向，而接口的实现类需要实现全部的接口方法。

8、什么是多态

多态，见名知意，即同一种对象的多种形态。打个比方--动物是一个对象，猫，猪，狗，还有你就是动物这个对象的不同表现形式，我们可以通过用动物的引用来指定对象的实例。

例： 已知接口（interface）animal， 猫（cat），猪（pig），你（you）都实现了animal接口，或则animal为父类对象，cat， pig，you 都继承于animal

```
1 //有上面的已知条件可以这样写：
2     animal a = new cat();  // 创建一个猫实例的动物对象
3     animal[] b = { new cat(), new pig(), new you()};//创建一个动物数组，
  里面有猫 猪 狗 还有你的实例
```

以上关系自能自上而下，不能同层，或者像上一层使用。

9、“==” 和 equals() 的区别

“==” 当作基础类型比较时比较的是两个值是否相等，当两个对象比较时比较的是两个对象的引用是否相等，即地址是否相等。

equals()方法默认继承与Object，当对象没有改写equals() 方法时调用equals()，方法其作用和“==”一样，对象改写equals()之后按照equals()给定的规则判断对象是否相等。

10、hashCode()

与equals方法一样，hashCode()默认继承于Object，通常为区别两个对象我们用哈希码来判断他们是否相等，哈希码不一定是对象的地址，改写之后有利于使用equals方法来判断两个对象是否一样。从而提高效率。

11、ArrayList, HashSet, TreeSet

ArrayList：允许相同元素存在，顺序与插入顺序一致。

HashSet： 不允许相同元素存在，存贮形式为散列是一种无须集合。

TreeSet： 不允许相同元素存在，会将插入元素按指定方法排序，实质上是一棵二叉排序树。

12、compareTo(Object o)

compareTo(Object o), 是Comparable<T> 接口下的一个方法，Comparable接口在Java.lang 包里面不需要导包，而且该接口里面就只有compareTo方法，想要使TreeSet听从自己的指挥，按照什么方法来排序就得实现Comparable接口。

compareTo方法返回值是一个int类型，当返回值：

< 0 ------ 对象 o 大于 当前对象

= 0 ------ 对象 o 等于当前对象

>0 ------ 对象 o 小于当前对象

想让当前对象按照什么方式排序就必须实现compareTo方法。

例：

```
class INT implements Comparable<INT>{
    public int value;
    public int compareTo(INT o){
        return this.value - o.value;
    }
}//  这样在调用工具类排序的时候，就会按照从小到大的规则排序，若想从小将第四行改成： return -(this.value - o.value);
```

13、Iterator 迭代器

Interator是一个工具类，与Commectios不同， Intreator是一个迭代器，使用它来遍历集合。

```
List<> a = new ArrayLIst<String>();
Iterator iterator = a.iterator();
while(iterator.hasNext()){
    System.out.println(iterator.next());
}
```

调用ArrayList a 的iterator() 方法，获得迭代器对象然后通过迭代器对象来遍历集合a。

14、foreach 遍历集合

foreach遍历集合比较方便所以通常会使用 。

```
String a[] = { "a" , "b", "c"};
for(String i : a){
    System.out.println(i);
}
```

foreach只能遍历结合，不能对集合中的元素进行修改，删除。

15、toString()

toString(); 方法默认继承Object， 用于返回对象的字符串表示形式， 一般也就是说想要对象你能够用println进行指定的输出就必须重写toString(); 方法。

16、throw 与 throws

throw： 抛出一个异常，通常是当程序出现某种情况时主动抛出一个异常：

```
double kk = 3.1415;
if(kk.equals("3.1415")){
    throw NumberFormatException();
}else System.out.println(kk);
```

throws：可能出现异常， 表示方法在调用时很可能会出现异常该异常交由上级处理

```
public static void function() throws NumberFormatException{
        String s = "abc";
        System.out.println(Double.parseDouble(s));
    }

    public static void main(String[] args) {
        try {
            function();
        } catch (NumberFormatException e) {
            System.err.println("非数据类型不能转换。");
            //e.printStackTrace();
        }
}
```

17、String类

<1>String[] split(String regex)

该方法是String类下面的一个方法，用于将当前串以串 regex 为界分割到一个字符串数组里面。

几点注意： 1、当连续碰到regex 则碰到几次分几个空串。

2、如果表达式不匹配输入的任何内容，返回的数组只具有一个元素，即此字符串。（尤其注意空字符串这种情况，他也是一个字符串）

3、可以匹配的情况下，每一个字符串都由另一个匹配给定表达式的子字符串终止，或者由此字符串末尾终止（数组中的字符串按照他们在此字符串出现的顺序排列）

```
String a = "abccafg";
String[] b = a.split("c");
System.out.println(b.length); // 重复为空串 输出3

String a = "abccbd";
String[] b = a.split(",");
System.out.println(b.length); // 无匹配b为原字符串  结果输出1

String a = "";
String[] b = a.split(",");
System.out.println(b.length); // 无匹配b为原字符串  结果输出1

String a = "";
String[] b = a.split("");
System.out.println(b.length); // 都为空串 匹配切割  结果输出1

String a = ",";
String[] b = a.split(",");
System.out.println(b.length); // 仅有一次切割 再无  输出结果为0
```

<2>String[] split(String regex, int limit)

与<1>方法相同，int limit 用于控制模式匹配的次数。

1.limit>0:

模式匹配将被最多应用n-1次，数组的长度将不会大于n，数组的最后一项将包含所有超出最后匹配的定界符的输入。

2.limit<0:

模式匹配将应用尽可能多的次数，而且数组的长度是任何长度。

3.lilmit=0:

模式匹配将被应用尽可能多的次数，数组可以是任何长度，并且结尾空字符串将被丢弃。

<3>String substring

该方法用于返回子串

String substring(int beginIndex) : 从第beginIndex个位置开始的子串 , 最后开始的话为空串

String substring(int beginIndex, int endIndex): 从第beginIndex 开始 到endIndex结束的子串，若开始为3结束为4则输出第4个位置的字符，若开始结束位置相等输出空串。

```
1         String a = "abcd";
2         System.out.println(a.substring(1)); // bcd
3         System.out.println(a.substring(2,3));//c
4         System.out.println(a.substring(2,2));//""
5         System.out.println(a.substring(3));//d
6         System.out.println(a.substring(4)); // ""
```

<4>indexOf

indexOf 用于返回指定子串位置的检索，具体参见API

18、Math.random()

位于java.lang 包里面的Math类里面的random()，方法返回的是一个double类型，且值大于等于0 ，小于1。

19、关于输入输出流

<1> BufferedReader（Reader in）

用于向文件读取数据流，创建一个句柄， 失败会返回异常。

readLine(), 读取一行字符串 返回其引用。

skip("")；跳过字符

<2>BufferedWriter(Writer out);

用于像文件输入数据流，创建一个写入句柄，失败会返回异常。

write()，重载，可以写入一个字符，一个字符串，一个字符串的一部分。

20、定制排序方法 compare(Object o1, Object o2)

compare方法是接口Comparator的唯一方法，他与compareTo这种自然排序方法不一样， 他可以定制两个对象的排序方式，重写该方法即可定制排序，在创建容器的时候就要传入比较器，如下：

```
1  SortedSet<Student> a = new TreeSet<>(new ScoreComparator());
```