字符串

String

1. 字符串不可变，值创建后不能更改，字符串实际上就是字符数组。
2. 直接赋值是将数据放在常量池，如s1=”a”，s2=”a”，s1==s2是true，他们的地址是一样的。
3. char[] chs = {‘a’};s3 = new String(chs);s4 = new String(chs);s3 == s4是false，s1 == s3 也是false。
4. 基本类型是比较数据值是否相同，引用类型是比较地址值是否相同。可以用equals()方法来比较引用类型的值。

StringBuilder

1. stringbuilder是可变的字符串，stringbuilder的内容是可变的。
2. StringBuilder sb = new StringBuilder();添加方法，sb2 = sb.append(值)。反转方法，sb2.reverse();。
3. StringBuilder转换为String的方法，sb.toString()；String转换为StringBuilder的方法，StringBuilder sb = new StringBuilder(s)。

集合(Arraylist)

1. Arraylist其实是个可变的数组，创建对象Arraylist<E> arr = new Arraylist<>();
2. 添加方法为arr.add();修改方法为arr.set(下标,修改值);查询方法为arr.get(下标);删除方法为arr.remove(下标);

继承

1. 使用extend让子类继承父类，当调用子类时，会先调用父类的无参构造方法，如果要调用父类的有参构造方法，可用super（E）调用。this()是调用类中的方法或属性。
2. 重写方法，可用@Override注解来确认重写方法的正确性。父类中private修饰方法不能被子类重写。

修饰符

1. private私有，默认不能再不同包的子类被调用，protected不能被不同包无关类使用，public公有。
2. final是最终的意思，被final修饰的方法不能被重写，被final修饰的属性也不能修改，被final修饰的类不能被继承。被final修饰的引用类型（对象）不能再次创建对象，即地址不能变。
3. static是静态的意思，被static修饰的成员类中的属性，是被所有对象共享的，即创建一次值，即可被所有对象共享。被static修饰的成员方法，只能访问静态的成员方法和成员变量。

多态

1. 同一个对象，在不同时刻表现出的不同形态。例：（父类）Animal a = new （子类）Cat（）;成员变量：编译和执行看左边；成员方法：编译时看左边，执行时看右边。
2. 好处：提高了程序的扩展性，定义方法时，用父类作为参数，在使用时，用具体的子类参与操作。例：public void useAnimal(Animal a ){a.eat();}，cat extend Animal，cat重写了eat方法。弊端：不能使用子类的特有功能。
3. 向上转型：从子到父，Animal a = new Cat();
4. 向下转型：从父到子，Cat c = (Cat)a;这样就能访问子类的特有功能。

抽象

1. 抽象类和方法要用abstract修饰，抽象类可以有抽象方法和正常的方法。抽象方法必须在抽象类中。
2. 抽象类的子类，要么重写抽象类中所有抽象方法，要么也设置为抽象类。

接口

1. 接口中的变量默认被final和静态static修饰。
2. 一个类只能继承一个类，可以实现多个接口。
3. 接口作为形参时，要先实现这个接口再传。

内部类

1. 内部类可以直接访问外部类的成员，包括私有；外部类访问内部类的成员，必须创建对象。
2. 测试类访问内部类时，外部类.内部类 对象名 = new 外部类。内部（）；
3. 匿名内部类：new 类或接口名(){ 重写方法； }，本质是继承或实现了该接口的子类匿名对象。