1.方法内不能定义方法

2.引用类型有堆内存 - 都是值传递。

- 基本类型的参数传输存储的数据值。

- 引用类型的参数传输存储的地址值。

3.OUT为外部循环打标签，后续用continut

4.在Java语言中，一个Java源文件（ .java ）中最多只能有一个 public 修饰的类，并且该源文件的文件名必须与这个 public 类的类名完全相同（包括大小写）。

但可以有多个非 public 的类，它们的访问权限仅限于当前源文件内被访问。

如果希望不同包中的（非public）类能够引用，需要将类声明为 public ，或者通过其他访问控制机制（如内部类、接口等）来实现间接访问。

5.this用来指当前对象

6.创建对象时根据小括号内括号参数情况选择构造器，写构造器时写了有参数构造器也要写无参数构造器，别人可能要用

7.setter和getter，构造器右键生成

8.什么是实体类？有啥特点？

- 成员变量必须私有，且要为他们提供get、set方法；必须有无参数构造器。

- 仅仅只是一个用来保存数据的java类，可以用它创建对象，保存某个事物的数据。

实体类有啥应用场景？

- 实体类对应的是软件开发里现在比较流行的开发方式，数据和数据的业务处理相分离

9.java中存在对象数组

10.只要是以“...”方式写出的字符串对象，会存储到

字符串常量池，且相同内容的字符串只存储一份；

但通过new方式创建字符串对象，每new一次都会

产生一个新的对象放在堆内存中。

11.ArrayList<Sting>，只能加String，有相同内容时，remove默认删除第一次出现的

12.- next() 方法读取输入中的下一个单词，以空白字符（空格、制表符、换行符等）作为分隔，读取到空白字符就停止。

- nextLine() 方法读取当前行的剩余内容，包括行中的空格，直到遇到换行符为止，换行符会被读取并丢弃。

java类型分为基本类型和引用类型

变量分为局部变量（方法内）和成员变量

1.有static修饰，属于类，在计算机里只有一份，会被类的全部对象共享。

实例变量（对象的变量）：无static修饰，属于每个对象的。

2.类变量推荐用类名.类变量去访问

3.static修饰的方法属于类，无修饰属于对象

类方法中可以直接访问类的成员，不可以直接访问实例成员。

实例方法中既可以直接访问类成员，也可以直接访问实例成员。

实例方法中可以出现this关键字，类方法中不可以出现this关键字的

5.static代码块用于类的初始化，在构造器前执行

6.static懒汉式，饿汉式单例

7.protected修饰，可以在子类访问，但不能在子类创建的对象中访问

8.一个类不能继承多个类，但可以多层继承，所有类默认是object类的子类

9.重写：方法名合参数列表都相同

10. 重写小技巧：使用Override注解，他可以指定java编译器，检查我们方法重写的格式是否正确，代码可读性也会更好。

子类重写父类方法时，访问权限必须大于或者等于父类该方法的权限（public > protected > 缺省 ）。

重写的方法返回值类型，必须与被重写方法的返回值类型一样，或者范围更小。

私有方法、静态方法不能被重写，如果重写会报错的。

11.在子类方法中访问其他成员（成员变量、成员方法），是依照就近原则的。

12.指定访问子类成员变量（成员方法）和父类成员变量分别使用this和super

13.子类构造器的特点：

子类的全部构造器，都会先调用父类的构造器，再执行自己。

子类构造器是如何实现调用父类构造器的： 默认情况下，子类全部构造器的第一行代码都是super()（写不写都有），它会调用父类的无参数构造器。

如果父类没有无参数构造器，则我们必须在子类构造器的第一行手写super(....)，指定去调用父类的有参数构造器。

14.补充知识：this(...)调用兄弟构造器

this(...) 、super(...) 都只能放在构造器的第一行，因此，有了this(...)就不能写super(...)了，反之亦然。

15.多态的前提

有继承/实现关系；存在父类引用子类对象；存在方法重写。

多态的一个注意事项

多态是对象、行为的多态，Java中的属性(成员变量)不谈多态。

16.在多态形式下，右边对象是解耦合的，更便于扩展和维护。

定义方法时，使用父类类型的形参，可以接收一切子类对象，扩展性更强、更便利。

多态下会产生的一个问题

● 多态下不能使用子类的独有功能。

17.类型转换有几种形式？能解决什么问题？

● 自动类型转换、强制类型转换。

● 可以把对象转换成其真正的类型，从而解决了多态下不能调用子类独有方法的问题。

强制类型转换需要注意什么？

● 存在继承/实现时，就可以进行强制类型转换，编译阶段不会报错。

● 但是，运行时，如果发现对象的真实类型与强转后的类型不同会报错（ClassCastException）

强制类型转换前？Java建议我们做什么事情？

● 使用instanceof判断当前对象的真实类型：对象 instanceof 类型。

18.常量：public static final修饰的成员变量，建议名称全部大写，多个单词下划线连接，用于记录系统配置信息，“宏替换”，出现的常量会被替换成字面量

19.final

● final关键字是最终的意思，可以修饰（类、方法、变量）

● 修饰类：该类被称为最终类，特点是不能被继承了。

● 修饰方法：该方法被称为最终方法，特点是不能被重写了。

● 修饰变量：该变量只能被赋值一次。

final修饰变量的注意

● final修饰基本类型的变量，变量存储的数据不能被改变。

● final修饰引用类型的变量，变量存储的地址不能被改变，但地址所指向对象的内容是可以被改变的。他指的是地址不能改变

20.抽象类的注意事项、特点

抽象方法只能有抽象签名

● 抽象类中不一定有抽象方法，有抽象方法的类一定是抽象类。

● 类该有的成员（成员变量、方法、构造器）抽象类都可以有。

● 抽象类最主要的特点：抽象类不能创建对象，仅作为一种特殊的父类，让子类继承并实现。

● 一个类继承抽象类，必须重写完抽象类的全部抽象方法，否则这个类也必须定义成抽象类。

抽象类更好支持多态

抽象类用来写模板方法比较好，模板方法建议用final来修饰，让其不能被重写

21. public interface 接口名 {

// 成员变量（常量）

// 成员方法（抽象方法）

● 注意：接口不能创建对象；接口是用来被类实现（implements）的，实现接口的类称为实现类。

修饰符 class 实现类 implements 接口1, 接口2, 接口3,... {}

● 一个类可以实现多个接口（接口可以理解成干爹），实现类实现多个接口，必须重写完全部接口的全部抽象方法，否则实现类需要定义成抽象类。

它必须重写完全部接口里面的全部抽样方法

接口多继承实现扩展

接口有保障

面向接口编程

1.1、JDK8开始，接口中新增了方法

- 默认方法：使用default修饰，使用实现类的对象调用。

- 静态方法：static修饰，必须用当前接口名调用

- 私有方法：private修饰，jdk9开始才有的，只能在接口内部被调用。

- 他们都会默认被public修饰。

JDK8开始，接口中为啥要新增这些方法？

- 增强了接口的能力，更便于项目的扩展和维护。

2.接口可以多继承，一个接口继承多个继承，便于实现类去实现

3.一个接口实现多个接口，如果多个接口中存在方法签名冲突，，则此时不支持多实现

4.一个类继承了父类，又同时实现了接口，父类中和接口中有同名的默认方法，实现类会优先用父类的。

5.一个类实现了多个接口，多个接口中存在同名的默认方法，可以不冲突，这个类重写该方法即可。

1.1、JDK8开始，接口中新增了方法

- 默认方法：使用default修饰，使用实现类的对象调用。

- 静态方法：static修饰，必须用当前接口名调用

- 私有方法：private修饰，jdk9开始才有的，只能在接口内部被调用。

- 他们都会默认被public修饰。

JDK8开始，接口中为啥要新增这些方法？

- 增强了接口的能力，更便于项目的扩展和维护。

2.接口可以多继承，一个接口继承多个继承，便于实现类去实现

3.一个接口实现多个接口，如果多个接口中存在方法签名冲突，，则此时不支持多实现

4.一个类继承了父类，又同时实现了接口，父类中和接口中有同名的默认方法，实现类会优先用父类的。

5.一个类实现了多个接口，多个接口中存在同名的默认方法，可以不冲突，这个类重写该方法即可。

1. 成员内部类是什么？如何创建其对象？

- 就是类中的一个普通成员，类似前面我们学过的普通成员变量、成员方法

- 外部类名.内部类名 对象名 = new 外部类(...).new 内部类(...);

成员内部类的实例方法中，访问其他成员有啥特点？

- 可以直接访问外部类的实例成员、静态成员

- 可以拿到当前外部类对象，格式是：外部类名.this。

2.什么是静态内部类？如何创建对象？有啥特点？

- 有static修饰的内部类。

- 外部类名.内部类名 对象名 = new 外部类.内部类(...);

- 可以直接访问外部类的静态成员，不能直接访问外部类的实例成员。

局部内部类是定义在方法等里面的

3.匿名内部类的书写格式是什么样的？

new 类或接口(参数值...) {

类体(一般是方法重写);

};

new Animal() {

@Override

public void cry() {

}};

匿名内部类有啥特点？

● 匿名内部类本质就是一个子类，并会立即创建出一个子类对象

匿名内部类有啥作用、应用场景？

● 可以更方便的创建出一个子类对象。

● 匿名内部类通常作为一个参数传输给方法。

4.public static A[] values();

public static A valueOf(java.lang.String);

}

- 枚举类的第一行只能罗列一些名称，这些名称都是常量，并且每个常量记住的都是枚举类的一个对象。

- 枚举类的构造器都是私有的（写不写都只能是私有的），因此，枚举类对外不能创建对象。

- 枚举都是最终类，不可以被继承。

- 枚举类中，从第二行开始，可以定义类的其他各种成员。

- 编译器为枚举类新增了几个方法，并且枚举类都是继承：java.lang.Enum类的，从enum类也会继承到一些方法。

5.抽象枚举，里面的方法是抽象方法，罗列对象时要重写方法

6.枚举用于限定参数，常量也可以（比较方便，但是不能限定，可能被随意填写）

7.- 定义类、接口、方法时，同时声明了一个或者多个类型变量（如： ），称为泛型类、泛型接口，泛型方法，它们统称为泛型。

java

public class ArrayList<E>{

...

}

- 作用：泛型提供了在编译阶段约束所能操作的数据类型，并自动进行检查的能力！这样可以避免强制类型转换，及其可能出现的异常。

- 泛型的本质：把具体的数据类型作为参数传给类型变量。

8.泛型可以定义多个类型<A，B>，里面也可以是<extends Animal>

9.修饰符 interface 接口名<类型变量，类型变量，...> 泛型接口

1. 泛型方法

1.泛型方法

修饰符 <类型变量，类型变量，...> 返回值类型 方法名(形参列表){

}

public static void test(T t){

}

通配符

● 就是 “?”，可以在 “使用泛型” 的时候代表一切类型；E T K V 是在定义泛型的时候使用。

泛型的上下限：

● 泛型上限：? extends Car:? 能接收的必须是Car或者其子类 。

● 泛型下限：? super Car：? 能接收的必须是Car或者其父类。

2.泛型的擦除问题和注意事项

● 泛型是工作在编译阶段的，一旦程序编译成class文件，class文件中就不存在泛型了，这就是泛型擦除。

● 泛型不支持基本数据类型，只能支持对象类型（引用数据类型）。

1.泛型方法

修饰符 <类型变量，类型变量，...> 返回值类型 方法名(形参列表){

}

public static void test(T t){

}

通配符

● 就是 “?”，可以在 “使用泛型” 的时候代表一切类型；E T K V 是在定义泛型的时候使用。

泛型的上下限：

● 泛型上限：? extends Car:? 能接收的必须是Car或者其子类 。

● 泛型下限：? super Car：? 能接收的必须是Car或者其父类。

2.泛型的擦除问题和注意事项

● 泛型是工作在编译阶段的，一旦程序编译成class文件，class文件中就不存在泛型了，这就是泛型擦除。

● 泛型不支持基本数据类型，只能支持对象类型（引用数据类型）。

