# 附录二　填空题汇总

## 一、面向对象部分

1.类是对象的模板。它定义对象的属性，并提供用于创建对象的构造方法以及操作对象的普通方法。

2.类也是一种数据类型。可以用它来声明对象引用变量。对象引用变量存放的只是对该对象的引用。

3.对象是类的实例。可以使用 new 操作符创建对象，使用点操作符（.）通过对象的引用变量来访问该对象的成员。

4.实例变量或方法属于类的一个实例。它的使用与各自的实例相关联。静态变量是被同一个类的所有实例所共享的。可以在不使用实例的情况下调用静态方法。

5.类的每个实例都能访问这个类的静态变量和静态方法。然而，为清晰起见，最好使用“ClassName.variable（类名.变量）”和“ClassName.method（类名.方法）”来调用静态变量和静态方法。

6.可见性修饰符指定类、方法和数据是如何被访问的。public 类、方法或数据可以被任何客户程序访问，private 方法或数据只能在本类中访问。

7.可以提供获取（访问器）方法或者设置（修改器）方法使客户程序能够看到或修改数据。

8.获取方法的方法签名为 public returnType getPropertyName()。如果返回值类型（returnType）是 boolean 型，则获取方法应该定义为 public boolean isPropertyName()。设置方法的方法签名为 public void setPropertyName(dataType propertyValue)。

9.所有参数都是以按值传递的方式传递给方法的。对于基本类型的参数，传递的是实际值；而对于引用数据类型的参数，则传递的是对象的引用。

10.Java 数组是一个可以包含基本类型值或对象类型值的对象。在创建一个对象数组时，它的元素被赋予默认值 nu11。

11.一旦被创建，不可变对象就不能被改变了。为了防止用户修改对象，可以定义该对象为不可变类。

12.实例变量和静态变量的作用域是整个类，无论该变量在什么位置定义。实例变量和静态变量可以在类的任何位置定义。

13.this 关键字可以用于引用调用对象。它也可以用于在构造方法中调用同一个类的另外一个构造方法。

14.可以从现有的类定义新的类，这称为类的继承。新类称为子类或继承类，现有的类称为超类、父类或基类。

15.构造方法用来构造类的实例。不同于属性和方法，子类不会继承父类的构造方法。它们只能用关键字 super 从子类的构造方法中调用。

16.构造方法可以调用重载的构造方法或其父类的构造方法。这种调用必须是构造方法的第一条语句。如果没有显式地调用它们中的任何一个，编译器就会把 super()作为构造方法的第一条语句，它调用的是父类的无参构造方法。

17.为了重写一个方法，必须使用与它的父类中的方法一样的签名、一样或者兼容的返回类型来定义子类中的方法。

18.实例方法只有在可访问时才能重写。这样，私有方法是不能重写的，因为它是不能在类本身之外访问的。如果子类中定义的方法在父类中是私有的，那么这两个方法是完全没有关系的。

19.静态方法与实例方法一样可以继承。但是，静态方法不能重写，如果父类中定义的静态方法在子类中重新定义，那么父类中定义的方法被隐藏。

20.Java 中的每个类都继承自 java.lang.Object 类。如果一个类在定义时没有指定继承关系，那么它的父类就是 Object。

21.如果一个方法的参数类型是父类，可以向该方法的参数传递任何子类的对象。这称为多态。

22.因为子类的实例总是其父类的实例，所以，总是可以将一个子类的实例转换成一个父类的变量。当把父类实例转换成它的子类变量时，必须使用转换标记“（子类名）”进行显式转换，向编译器表明你的意图。

23.一个类定义一个类型。子类定义的类型称为子类型，而父类定义的类型称为父类型。

24.当从引用变量调用实例方法时，该变量的实际类型在运行时决定使用该方法的哪个实现。这称为动态绑定。

25.可以使用表达式 obj instanceof AClass 测试一个对象是否是一个类的实例。

26.可以使用 ArrayList 类来创建一个对象，用于存储一个对象列表。

27.可以使用 protected 修饰符来防止方法和数据被不同包的非子类访问。

28.可以使用 fina1 修饰符来表明一个类是最终类，是不能被继承的；也可以表明一个方法是最终的，是不能被重写的。

29.接口 java.lang.Comparable 定义了 compareTo 方法。Java 类库中的许多类都实现了 Comparable。

30.接口 java.lang.Cloneable 是一个标记接口。实现 Cloneable 接口的类的对象是可克隆的。

## 二、方法部分

1.程序模块化和可重用性是软件工程的中心目标之一。Java 提供了很多有助于完成这一目标的有效构造。方法就是一个这样的构造。

2.方法头指定方法的修饰符、返回值类型、方法名和参数。

3.方法可以返回一个值。返回值类型 returnValueType 是方法所返回的值的数据类型。如果方法不返回值，则返回值类型就是关键字 void。

4.参数列表是指方法中参数的类型、次序和数量。方法名和参数列表一起构成方法签名。参数是可选的，也就是说，一个方法可以不包含参数。

5.return 语句也可以用在 void 方法中，用来终止方法并返回到方法的调用者。

6.传递给方法的实际参数应该与方法签名中的形式参数具有相同的数目、类型和顺序。

7 方法抽象是把方法的应用和实现分离。用户可以在不知道方法是如何实现的情况下使用方法。方法的实现细节封装在方法内，对调用该方法的用户隐藏。这称为信息隐藏或封装。

12.方法抽象将程序模块化为整齐、层次分明的形式。将程序写成由简洁的方法构成的集合会比其他方式更容易编写、调试、维护和修改。这种编写风格也会提高方法的可重用性。

13.当实现一个大型程序时，可以使用自顶向下或自底向上的编码方法。不要一次性编写完整个程序。

## 三、数组部分

### （一）一维数组

1.使用语法 elementType[] arrayRefVar（元素类型数组引用变量）或 elementType arrayRefVar[]（元素类型 数组引用变量[]）声明一个数组类型的变量。

2.数组变量不是基本数据类型变量。数组变量包含的是对数组的引用。

3.只有创建数组后才能给数组元素赋值。可以使用 new 操作符创建数组，语法如下：new elementType[arraySize]（数据类型[数组大小]）。

4.数组中的每个元素使用语法 arrayRefVar[index]（数组引用变量[下标]）表示。下标必须是一个整数或一个整数表达式。

5.创建数组之后，它的大小就不能改变，可以使用 arrayRefVar.length 得到数组的大小。由于数组的下标总是从 0 开始，所以，最后一个下标总是 arrayRefVar.length-1。如果试图引用数组下标范围外的元素，就会发生越界错误。

6.程序员经常会错误地用下标 1 访问数组的第一个元素，但是，实际上这个元素的下标应该是 0。这个错误称为下标过 1 错误。

7.当创建一个数组时，如果其中的元素的基本数据类型是数值型，那么赋默认值 0。字符类型的默认值为'\u0000'，布尔类型的默认值为 false。

8.将数组参数传递给方法时，实际上传递的是数组的引用。被调用的方法可以修改调用者中原数组的元素。

9.如果数组是排好序的，对于查找数组中的一个元素而言，二分查找比线性查找更加高效。

### （二）多维数组

1.可以使用二维数组来存储表格。

2.可以使用以下语法来声明二维数组变量：元素类型[][] 数组变量。

3.可以使用以下语法来创建二维数组变量：new 元素类型[行的个数][列的个数]。

4.使用下面的语法表示二维数组中的每个元素：数组变量[行下标][列下标]。

5.可以使用数组初始化简写形式来创建和初始化二维数组：

元素类型[][] 数组变量={{某行的值}，...，{某行的值}}。

## 四、控制结构部分

### （一）顺序结构部分

1.标识符是程序中用于命名诸如变量、常量、方法、类、包等元素的名称。

2.标识符是由字母、数字、下划线（_）和美元符号（$）构成的字符序列。标识符必须以字母或下划线开头，不能以数字开头。标识符不能是保留字。标识符可以为任意长度。

3.变量用于存储程序中的数据。声明变量就是告诉编译器该变量可以存储何种数据类型。

4.有两种类型的 import 语句：明确导入和通配符导入。明确导入是在 import 语句中指定导

入单个类；通配符导入将包中所有的类导入。

5.在 Java 中，等号（=）被用作赋值操作符。

6.命名常量（或简称为常量）表示从不改变的数据。

7.用关键字 final 声明命名常量。

8.可以使用(type)value 这样的表示法显式地将数值从一个类型转换到另一个类型。

9.将一个较小范围类型的变量转换为较大范围类型的变量称为扩展类型。

10.将一个较大范围类型的变量转换为较小范围类型的变量称为缩小类型。

11.扩展类型不需要显式转换，可以自动完成。缩小类型必须显式完成。

12.在计算机科学中，1970 年 1 月 1 日午夜零点称为 UNIX 时间戳。

**（二）选择结构部分**

1.boolean 类型变量可以存储值 true 或 false。

2.关系操作符（<、<=、==、!=、>、>=）产生一个布尔值结果。

3.选择语句用于可选择的动作路径的编程。选择语句有以下几种类型：单分支 if 语句、双分支 if-else 语句、嵌套 if 语句、多分支 if-else 语句、switch 语句和条件操作符。

4.各种 if 语句都是基于布尔表达式来决定控制的。根据表达式的值是 true 或 false，这些语句选择两种可能路径中的一种。

5.当对 p1&&p2 求值时，Java 先求 p1 的值，如果 p1 为 true，再对 p2 求值；如果 p1 为 false，就不再对 p2 求值。当对 p1||p2 求值时，Java 先求 p1 的值，如果 p1 为 false，再对 p2 求值；如果 p1 为 true，就不再对 p2 求值。因此，&&也称为条件与操作符或短路与操作符，而||也称为条件或操作符或短路或操作符。

6.switch 语句根据 char、byte、short、int 或者 String 类型的 switch 表达式来决定控制。

7.表达式中的操作符按照括号、操作符优先级以及操作符结合规则所确定的次序进行求值。

8.括号用于强制将求值的顺序以任何顺序进行。

9.赋值操作符是右结合的。

**（三）循环结构部分**

1.有三类循环语句：while 循环、do-while 循环和 for 循环。

2.循环中包含重复执行的语句的部分称为循环体。

3.循环体执行一次称为循环的一次迭代。

4.在设计循环时，既需要考虑循环控制结构，还需要考虑循环体。

5.for 循环控制由三部分组成。第一部分是初始操作，通常用于初始化控制变量。第二部分是循环继续条件，决定是否执行循环体。第三部分是每次迭代后执行的操作，经常用于调整控制变量。

6.while 循环和 for 循环都称为前测循环。do-while 循环称为后测循环。

## 五、字符与字符串部分

1.Java 提供了在 Math 类中的数学方法，用于执行数学函数。

2.字符类型 char 表示单个字符。

3.转义序列包含反斜杠\以及后面的字符或者数字组合。字符\称转义字符。

4.字符' '、\t、\f、\r 和\n 称为空白字符。

5.字符可以基于它们的 Unicode 应用关系操作符进行比较。

6.Character 类包含方法 isDigit、isLetter、isLetterorDigit、isLowerCase、isUpperCase，用于判断一个字符是否是数字、字母、小写字母还是大写字母。它也包含 toLowerCase 和 toupperCase 方法返回小写或大写字母。

7.字符串是一个字符序列。字符串的值包含在一对匹配的双引号中。字符的值包含在一对匹配的单引号中。

8.字符串在 Java 中是对象。只能通过一个指定对象调用的方法称为实例方法。非实例方法称为静态方法，可以不使用对象来调用。

9.可以调用字符串 length()方法获取它的长度，使用 charAt(index)方法从字符串中提取特定下标位置的字符，使用 indexOf 和 lastIndexOf 方法找出一个字符串中的某个字符或某个子串。

10.可以使用 concat 方法连接两个字符串，或者使用加号（+）连接两个或多个字符串。可以使用 substring 方法从字符串中提取子串。

11.可以使用 equals 和 compareTo 方法比较字符串。如果两个字符串相等，equals 方法返回 true；如果它们不等，则返回 false。compareTo 方法根据一个字符串等于、大于或小于另一个字符串，分别返回 0、正整数或负整数。

12.printf 方法使用格式限定符来显示一个格式化的输出。

## 六、基本程序设计部分

1.标识符是程序中用于命名诸如变量、常量、方法、类、包等元素的名称。

2.标识符是由字母、数字、下划线（_）和美元符号（$）构成的字符序列。标识符必须以字母或下划线开头，不能以数字开头。标识符不能是保留字。标识符可以为任意长度。

3.变量用于存储程序中的数据。声明变量就是告诉编译器该变量可以存储何种数据类型。

4.有两种类型的 import 语句：明确导入和通配符导入。明确导入是在 import 语句中指定导入单个类；通配符导入将包中所有的类导入。

5.在 Java 中，等号（=）被用作赋值操作符。

6.方法中声明的变量必须在使用前被赋值。

7.命名常量（或简称为常量）表示从不改变的数据。

8.用关键字 final 声明命名常量。

9.Java 提供四种整数类型（byte、short、int、1ong）表示四种不同大小范围的整数。

10.Java 提供两种浮点类型（float、double）表示两种不同精度的浮点数。

11.Java 提供操作符完成数值运算:加号（+）、减号（-）、乘号（*）、除号（/）和求余符号（%）。

12.整数运算（/）得到的结果是一个整数。

13.Java 表达式中的数值操作符和算术表达式中的使用方法是完全一致的。

14.Java 提供扩展赋值操作符：+=（加法赋值）、-=（减法赋值）、*=（乘法赋值）、/=（除法赋值）以及%=（求余赋值）。

15.自增操作符（++）和自减操作符（--）分别对变量加 1 或减 1。

16.当计算的表达式中有不同类型的值时，Java 会自动地将操作数转换为恰当的类型。

17.可以使用(type)value 这样的表示法显式地将数值从一个类型转换到另一个类型。

18.将一个较小范围类型的变量转换为较大范围类型的变量称为扩展类型。

19.将一个较大范围类型的变量转换为较小范围类型的变量称为缩小类型。

20.扩展类型不需要显式转换，可以自动完成。缩小类型必须显式完成。

21.在计算机科学中，1970 年 1 月 1 日午夜零点称为 UNIX 时间戳。