**【第一章 Java概述】**

1、Java语言的主要特征：面向对象、跨平台、多线程、安全、健壮、高性能、简单、动态、分布式

2、为什么说java是面向对象的语言？

①java是基于信息隐藏和抽象数据类型概念的纯面向对象的语言

②java中每个对象封装数据和方法，而方法实施对数据的处理

③java中可以通过继承、组合机制实现代码复用

3、为什么说java是跨平台的开发语言？

java跨平台的实现原理：java程序运行在虚拟机之上，java虚拟机中的Java解释器用来解释执行java编译器编译之后的程序

4、Java虚拟机（JVM）主要实现Java字节码的解释和执行，为不同的平台提供统一的接口。

5、java环境中的几个概念：

①JVM（java虚拟机）

②JRE（java运行时环境）

③JDK（java开发工具包/java开发环境）

6、java程序的基本组成是“类”（使用class声明），方法不可以单独存在。

类和方法都是在一对大括号中定义的

7、main方法的形式：public static void main(String[] args){ }

8、java的注释：

①块注释（/\*注释内容\*/）

②行注释（//注释内容）

③文档注释（/\*\*注释内容\*/）

**【第二章 Java基础】**

1、Java中的输入输出：通过流实现，使用System.out表示标准输出设备；使用System.in表示标准输入设备

Java中的输入需要使用Scanner的对象读取来自System.in的输入。

<Scanner input = new Scanner(System.in);>

Scanner类在使用时须在源文件的第一行导入import java.util.Scanner；

Scanner对象中常用的获取用户输入的方法有：

①String nextLine( ) //获取用户输入的一行信息

②boolean nextBoolean( ) //获取用户输入的一个boolean类型的值

③short nextShort( ) //获取用户输入的一个short类型的值

④int nextInt( ) //获取用户输入的一个int类型的值

⑤long nextLong( ) //获取用户输入的一个long类型的值

⑥double nextDouble( ) //获取用户输入的一个double类型的值

2、标识符概念：

Java语言中，对于变量，常量，函数，语句块也有名字，我们统统称之为Java标识符。

标识符是用来给类、对象、方法、变量、接口和自定义数据类型命名的。

3、标识符组成：

①Java标识符由数字，字母和下划线(\_)，美元符号($)组成，只能以字符、“\_”或“$”开头，不能以数字开头。

②标识符是大小写敏感的并且未规定最大长度。

③标识符不能是Java关键字或保留字。

4、Java数据类型分为基本类型和引用类型两大类：

1）基本类型：

①整数类型：byte、short、int、long

②浮点类型：double、float

③字符类型：char

④布尔类型：boolean

2）引用类型：类、接口、数组、枚举

5、byte（1字节）、short（2字节）、int（4字节）、long（8字节）、

float（4字节）、double（8字节）、char（2字节）

6、

①整型类型：用于表示没有小数部分的数值，它允许是负数

②浮点类型：用于表示有小数部分的数值。

③字符类型：用于表示单个字符。通常用来表示字符常量。

采用unicode编码，字符的存储范围在\u0000~\uFFFF

编码中的第一个字节仍与 ASCII 兼容

④布尔类型：有false和true两个值，用来判定逻辑条件。

整数值和布尔值之间不能进行相互转换

7、变量是在程序运行过程中其值可以被改变的量。

变量的定义：①变量的声明②变量的初始化

变量声明的格式：数据类型 变量名1[,变量名2，...];

变量的初始化：声明一个变量时，通过赋值语句对变量进行显示的初始化。

8、常量是在程序运行过程中其值始终保持不变的量。

Java使用关键字final来定义常量。

常量定义的语法格式

final 数据类型 常量名称 = 值 ;

常量被赋值之后，就不能再改变了。

习惯上，常量名使用大写定义。

9、&&表示逻辑“与”；|| 表示逻辑“或”；! 表示逻辑“非”。

&&和 || 是按照“短路”方式求值的。如果第一个操作数已经能够确定表达式的值，第二个操作数就不必计算了。

10、类型转换：基本数据类型之间可以进行相互转换。

①隐式转换：当进行类型加宽转换时可以自动实现，被称为隐式转换。

②强制转换：当进行类型收缩转换时转换必须进行显式转换，被称为强制转换。

11、数据类型按容量大小排序

byte→ short→ (char)→ int→long→float→ double。

byte，short, char之间不会相互转换，他们三者在计算时首先转换为 int 类型。[需要强制类型转换]

12、Java提供了一个更为简洁的循环语句，用于数组或者集合的遍历。

增强型for循环

基本语法：

for ( 类型 变量名: 数组或集合 ) {

循环体

}

**【第三章 数组】**

1、数组是在内存中连续存储一组具有相同数据类型的数据元素的有序集合。

数组的特点：

①在整个生命周期中长度固定不可变。

②数组可以存储基本类型元素和引用类型元素。

③同一个数组中必须存储相同类型的元素。

④数组中的元素有先后顺序，其顺序位置由数组下标决定。

2、数组声明的基本语法：

array\_type array\_Name[ ];

array\_type[ ] array\_Name;

不能在[ ]中指定数组长度，数组使用前需要进行初始化。

3、数组的初始化：

①静态初始化：当数组元素的初始化值直接由大括号“{ }”中的数据给出时，就称为静态初始化

②动态初始化：动态初始化先用new操作符为数组分配内存，然后才为每一个元素赋初值。

其一般格式如下：

array\_Name = new array\_type [ arraySize];

4、二维数组声明的一般格式：（前面括号内的数值表示行数，后面的表示列数）

array\_type array\_Name[ ][ ];

array\_type [ ][ ] array\_Name;

array\_type [ ] array\_Name[ ];

5、Arrays类的常用方法：

①copyOf 实现数组的复制

②fill 实现数组元素的初始化

③sort 实现数组的排序

④binarySearch 实现排序后的数组元素查找

**【第四章 类和对象】**

1、面向过程的缺陷：

①代码的重用性和可维护性差。

②数据和对数据的操作分离，数据安全不能保证。

2、类和对象

对象：现实世界中某个具体的物理实体，每个对象都包含特定的属性和行为。

对象：现实世界中某个具体的物理实体在计算机逻辑中的映射和体现。

类：现实世界中具有相同属性和行为的物理实体的抽象。

类：是一种抽象的数据类型，是同种对象的集合与抽象，是具有共同行为和属性的若干对象的统一描述体。

类是对对象的抽象，通常即包含成员属性和成员方法的抽象数据类型

对象是类的实例，是按照类类型创建的实例

3、面向对象：将属性及其方法封装为类，以类的对象作为程序的基本元素，通过向对象发送消息，从而调用相应方法完成各种功能。

4、面向对象程序设计的特点：

①对象为基本元素

②数据——安全

③程序的维护量较小、代码的重用率高(继承)

④程序 = 对象 + 对象 + 对象 + ... ...

5、面向对象程序设计的三大特性：继承、封装、多态（同一操作作用于不同的类的对象，将产生不同的执行结果）

①继承的优势

提高了代码的重用率，提高了编程效率。

②封装的优势：

封装是面向对象程序设计语言实现信息隐藏的方法。

封装隐藏了具体的实现细节，使某些成员设为私有从而提高了安全性和可靠性。

③多态的优势：

增强了程序的灵活性

6、

class是声明类的关键字，<类名>是标识符，通常以首字母大写开头，用来与对象名，方法和变量名区分。

类体可以包含成员属性和成员方法两部分，成员属性声明必须以分号结束。

定义一个类就相当于定义了一个类型和一个作用域。

7、对象的实例化格式：

<类名> <对象名> = new <类名> ([参数1，参数2,…]) ;

或

<类名> <对象名>；

<对象名> = new <类名> ([参数1，参数2,…]) ;

8、对象的存储方式：每个类的对象分别存储本对象的所有成员属性，一个类的所有对象共用成员方法。

9、类成员的访问：

①访问属性的一般形式：<对象名>.<属性名>;

②访问方法的一般形式：<对象名>.<方法名>([<参数1>，<参数2>，…]);

10、封装的实现：通过访问修饰符来实现信息隐藏

一般将成员属性使用private修饰

定义用来访问成员属性的方法，并使用public修饰

11、

值传递：表示方法接收的是调用者提供的变量的值。

地址传递：表示方法接收的是调用者提供的变量地址。

值传递不能通过形参修改实参的值，地址传递可以通过形参修改实参对应变量的值。

12、

方法传参（基本数据类型）：值传递（方法不可以改变实参的值）。

方法传参（引用类型）：引用传递（方法体内，通过引用改变了实际参数对象的内容）。

当参数是基本数据类型时，方法不能修改实际参数的值。

当参数是引用类型时，方法通过地址引用改变了实际参数对象的内容。

13、方法的重载：同一个类中两个或以上的同名方法，参数类型、个数或顺序不同称为方法的重载。

方法重载的依据：①方法名相同；②参数列表必须不同

[注意] 1>返回值类型可以不同（返回值类型不作为重载的依据）

2>是否是静态的也不作为依据

3>重载的方法之间可以互相调用

14、构造方法：

①功能：初始化对象

②方法名：与类名相同

③参数：可有，可无

④返回值：不指定返回值（不能写void ）

⑥内容：任意，通常只包含成员赋值语句

⑦调用方法：使用new创建对象时自动调用

15、

在一个类中可以定义多个构造方法，但构造方法的参数列表不能相同。

构造方法实现时可以通过this关键字调用已经存在的构造方法。

16、

this关键字：this用于引用本类中的属性和方法 { this.属性、this.方法() }

this调用本类的构造方法：使用this()、this(参数)的形式

this调用构造方法，只能放在重载的构造方法中的第一行

this还可以作为返回值，返回“当前调用方法”的那个对象的引用。

17、垃圾回收机制

1）垃圾收集器受JVM的控制，JVM决定什么时候运行垃圾收集器。

①当JVM感到内存不足时会运行垃圾收集器

②在Java程序中可以请求JVM运行垃圾收集器，但无法保证JVM会答应请求

2）当没有引用能够访问对象时，该对象就是适合进行垃圾收集的。

① 空引用；② 隔离引用

**【第五章 类的继承】**

1、

1）继承就是从已有类（父类、超类、基类）的基础上产生一个新的类（子类、派生类）

派生类通过继承自动拥有基类的属性和方法，继承是实现类的重用、软件复用的重要手段。

2）基类与派生类的关系：

派生类与基类存在是一种(is a kind of)的关系，一个派生类对象可以看作是一个（is a）基类对象

派生类是基类的特化，而基类是派生类的泛化

基类和派生类是相对的

2、继承的语法格式：

class 子类名称 extends 父类名称{

//扩充或修改的属性与方法;

}

3、Java 中的单重继承机制：一个类只能有一个直接基类（父类）;

final修饰的类不能有派生类（子类），String是典型特例；

Object类是所有Java类的顶级基类（父类）。

4、继承过程：

①吸收父类成员：子类把父类的全部成员（不包括构造方法）接收过来

②增加子类成员：对父类功能的扩展，一般在声明子类时还应自己定义子类的构造方法（构造方法是不能被继承的）

③调整父类成员：隐藏同名父类成员

5、子类内部只能访问父类中的非private类型的属性和方法

6、子类和父类中含有同一属性时，父类构造方法先调用，再调用子类方法

7、

super关键字：调用父类的构造方法{ super()、super(参数) }。

super() 、super(参数)调用只能放在子类构造方法的第一行。

super用于引用父类中的属性或方法

子类构造方法中没有显式调用父类构造方法，则默认调用父类的无参构造方法。

super.属性、super.方法()

注意：super只能访问父类中的非private类型的属性或方法。

使用 super.属性名称的形式，位置不固定。

super.方法名([参数]) 调用父类非private类型的普通方法，位置不固定。

8、方法的重写：在继承机制中派生类继承了基类的属性和方法，同时还可以对基类中的属性和方法做进一步的扩充或者修改；派生类对基类中定义的方法进行重新实现称为方法重写。

重写只能出现在继承关系之中。

重写的主要优点是能够定义某个派生类型特有的行为。

9、子类中可以对父类中定义的方法进行改造，但必须遵循一定的规则：

①重写的方法返回值类型是父类方法返回值类型的子类或者相同；

②重写的方法具有相同的方法名；

③重写的方法参数列表必须相同；

④不能重写被标识为final的方法；

⑤重写的方法不能缩小访问权限；

⑥不能重写父类私有的方法。

10、方法重写最大的作用就是在不改变原来代码的基础上可以对其中任意模块进行改造。

11、方法重写的意义：

1）重写方法可以是实现多态，用父类的引用来操纵子类对象，但是在实际运行中对象将调用子类重写的方法。

2）父类的引用可以指向子类的对象：向上转型

3）好处：①可以使用子类强大的功能

②可以抽取父类的共性

4）指向子类对象的父类类型的引用可以调用父类中定义的属性和方法

5）指向子类对象的父类类型的引用无法调用子类中定义而父类中没有的方法

12、如何访问另一个包中的公有类？

第一种方式：在每个类名之前添加完整的包名。

第二种方式：通过import关键字引入包。

import指定类

import整个包

13、访问带包的类：

1）通过全名来访问某包中的类。

2）通过import关键字引入包

import语句的唯一价值是减少键入。

import语句，必须位于package和类声明之间。

默认会引入java.lang包。

14、访问修饰符缺省时，即成员没有任何访问权限修饰符时，说明该成员具有默认权限或者又叫包权限

