**《Java 基础+Java高级》课程考试答案**

**总分：100分　 时间：120钟**

1. **简答题(本大题共40小题，每题2分，共80分)**
2. 请列出Java中的8种基本数据类型。

boolean、char、byte、short、int、long、float、double。

1. 请简述Java程序为什么能实现“一次编译，到处运行” 。

编写好java的源程序，通过编译器把.java文件编译成.class文件，接下来由java虚拟机装载各种运行.class文件。 正是因为java程序是java虚拟机解释执行的，所以才看“一次编译，到处运行”

1. Java中一个char型变量是否可以存储一个中文汉字，为什么？

可以存储。因为在java中以unicode编码，一个char型变量在内存中占2个字节，所以可以存储。

1. 请描述Java中基本数据类型的自动转换规则。

最好直接参考课件上内容。  
boolean类型不参与类型转换；

有两种情况会发生自动类型转换

* + - 1. 占用内存小的原始数据类型向占用

内存大地原始数据类型转换的时候是自动类型转换

* + - 1. 在占用内存大小一致的情况下整数向实数转换的时候是自动类型转换

1. 5%3、-5%3、-5%-3、5%-3的值分别是多少？

5%3=2、-5%3=-2、-5%-3=-2、5%-3=2。

1. int i=5;int j=5;System.out.println(i++);System.out.println(++j);分别输出多少？

5、6。

1. short s1 = 1; s1 = s1 + 1;有什么错? short s1 = 1; s1 += 1;有什么错?

1）对于short s1 = 1; s1 = s1 + 1; 由于s1+1运算时会自动提升表达式的类型，所以结果是int型，再赋值给short类型s1时，编译器将报告需要强制转换类型的错误。

2）对于short s1 = 1; s1 += 1;由于 += 是java语言中的赋值运算符，例如 s1 = 1一样可以赋值，Jvm对于short类型赋值时判断是否超出short的范围如果没有超出范围则默认1为short类型可以直接赋值。所以该表达式正确。

1. int i = 5; if(i < 6 || (i++) > 5) {System.out.println(“i == ”+i); };请问上述代码输出什么结果？

i==5。

1. byte a = 10;byte b = 20;byte c = a + b;System.out.println(c);请问上述代码输出什么结果？

编译错误。

1. 请问在使用switch分支语句进行条件判断时，swtich(expression) 中的expression都可以是哪些数据类型？

JDK6.0及以下版本中expression可以是byte、short、char、int.;JDK7.0版本中expression可以是byte、short、char、int.、String。

1. 请描述面向对象的3个基本特征，每个特征的目的是什么，如何实现每个基本特征？

面向对象的三个特征分别是封装、继承和多态。

面向对象的编程语言有封装、继承 、抽象、多态等4个主要的特征。

封装：

封装是保证软件部件具有优良的模块性的基础，封装的目标就是要实现软件部件的“高内聚、低耦合”，防止程序相互依赖性而带来的变动影响。面向对象的封装就是把描述一个对象的属性和行为的代码封装在一个类中，属性用变量定义，行为用方法进行定义，方法可以直接访问同一个对象中的属性。通常情况下，只要记住让变量和访问这个变量的方法放在一起，将一个类中的成员变量全部定义成私有的，只有这个类自己的方法才可以访问到这些成员变量，这就基本上实现对象的封装。

继承：

在定义和实现一个类的时候，可以在一个已经存在的类的基础之上来进行，把这个已经存在的类所定义的内容作为自己的内容，并可以加入若干新的内容，或修改原来的方法使之更适合特殊的需要，这就是继承。继承是子类自动共享父类数据和方法的机制，这是类之间的一种关系，提高了软件的可重用性和可扩展性。具体语法：

[修饰符] class 子类名 extends 父类{

…

}

多态：

一种行为在不同对象调用时有多种操作

多态存在的三个必要条件

* + - 要有继承，或实现；
    - 要有重写；
    - 父类引用指向子类对象。

一旦满足以上3个条件，当调用父类中被重写的方法后，运行时创建的是哪个子类的对象，就调用该子类中重写的那个方法。

1. 请描述this()和super()的含义？

this()是在该类的构造方法中调用其他重载的构造方法。

super()是在该类的构造方法中调用父类的构造方法。

1. 当一个对象被当做参数传递到一个方法后，此方法可改变这个对象的属性，并可返回变化后的结果，那么这里到底是值传递还是引用传递？

值传递。Java 编程语言只有值传递参数。当一个对象实例作为一个参数被传递到方法中时，参数的值就是对该对象的引用。对象的内容可以在被调用的方法中改变，但对象的引用是永远不会改变的。

1. 请描述构造器的作用以及构造器的调用方式？

构造方法是用来创建类的对象的，实现对类的实例化。在JAVA中使用new关键字来调用类的构造方法。

1. 请描述什么情况下编译器会自动创建一个无参构造器？

在定义类时，如果在类体内没有显式的定义一个构造方法，那么编译器在编译期间，会默认的在类体添加一个访问权限是public，参数列表为空，方法体为空的默认的构造方法。

1. 请描述堆内存和栈内存分别存储哪些变量？

栈内存中存储方法中定义的基本类型的变量和对象的引用。

堆内存用来存放由new创建的对象和数组。

1. 请描述什么是类变量、实例变量以及局部变量？

类变量，是归属类的变量，它是通过在定义类的属性的时，增加static修饰符实现的，所以又称为静态变量。

实例变量，是归属于类的实例的变量，又称为成员变量，没有经过static修饰。

局部变量就是在类中方法中定义的变量，不管是方法的形参，还是方法体内所定义的变量都是局部变量。局部变量只在本方法中有效。

1. 请描述static关键字都能修饰哪些元素，修饰不同元素的含义是什么？

static关键字可以修饰类中的属性、方法和代码块。

修饰属性时，表示该变量为类变量（静态变量），在内存中只有一个拷贝，在类的所有实例中共享。

静态方法和静态变量都可以通过类名直接被访问；静态方法中不能直接访问实例方法和实例变量；在静态方法中不能使用this和super关键字；静态方法不能被abstract修饰。

静态代码块在类被加载时，静态代码块只被执行一次，类中不同的静态代码块按它们在类中出现的顺序被依次执行。

1. 构造器是否可以被重写？

子类不会继承父类的构造器，所以在子类中无法对父类中的构造器进行重写。

1. 请分别解释overload和override的含义，并说明两者的区别？

方法的重载（overload）指在同一个类中同时存在一个以上的同名，但参数列表不同的方法。重载的方法，它们的参数列表必须不同，返回类型可以不同。

方法的覆盖（也叫方法的重写，override）指子类可以修改从父类继承的方法。覆盖方法和被覆盖方法必须具有相同的方法名、参数列表和返回类型。

1. 请描述在Java中什么时候使用接口，什么时候使用抽象类？

接口是一种协定，抽象类则相当于类模板。

如果一个类型必须实现多个协定，或者协定适用于多种类型，使用接口。

如果定义层次结构并且有部分的功能实现建议使用抽象类

**程序设计时优先考虑定义抽象类，而不是接口。**

1. 分别简述final, finally和finalize的含义。

final 用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不可继承。

finally是异常处理语句结构的一部分，表示总是执行。

finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收，JVM不保证此方法总被调用。

1. 请描述instanceof关键字的作用。

判断一个对象是不是某个类(子类)或接口(子接口)的实例。

1. 请描述在Java中实现多态应该具备的3个条件。

实现多态的三个必要条件：

1）要有继承或实现；

2）要有方法的覆盖；

3）要有父类的引用指向子类的对象。

1. 请描述final都可以修饰哪些元素，修饰不同元素的含义是什么？

final可以修饰 class、变量、成员方法。

1）final的class：不可以被继承，即没有子类。

2）final的成员方法：不可以被覆盖。

3）final的变量：

final既可以修饰全局变量，也可以修饰局部变量；final修饰的变量，是常量（具名常量）。

1. abstract的方法是否可以是static的，为什么？

static 方法是类方法,它是不允许子类覆盖（override），而abstract方法是没有实现的方法，是必须由子类来覆盖的，两者矛盾。

1. 接口是否可以继承接口？抽象类是否可以实现接口？抽象类是否可以具体类？

接口可以继承接口。

抽象类可以实现(implements)接口。

抽象类可以继承具体类。

1. 请描述什么是检查性异常和非检查性异常。

检查性异常（非运行时异常，不是RuntimeException类的子类）：开发人员必须显式地写出检查性异常的处理代码，否则无法通过编译。

非检查性异常（运行时异常，是RuntimeException类的子类）：对于非检查性异常，异常处理的代码可写、可不写，程序都能通过编译。

1. 请描述throw和throws的作用。

throws子句声明可能会出现的异常。如果一个方法可能会出现异常，但是没有能力来处理这种异常，可以在方法的生命处用throws子句来声明抛出异常。  
而throw语句用于抛出异常。throw语句用于抛出的对象必须是java.lang.Throwable类或其子类的实例。

1. try{}里有一个return语句，那么紧跟在这个try后面的finally{}里的code会不会被执行，什么时候被执行，在return前还是后？

会执行，在return执行之前。

1. 请描述Java中流按照方向和处理数据的单位都分为哪几类，每种流的基类是什么？

按方向分为输入流和输出流。

按处理数据的单位分为字节流和字符流。

字节输入流的基类为InputStream。

字节输出流的基类为OutputStream。

字符输入流的基类为Reader。

字符输出流的基类为Writer。

1. Java中基本数据类型所对应的包装类型都是什么？并说明基本数据类型和包装类之间的区别。

Java 提供两种不同的类型：引用类型和基本数据类型。Java为每个基本数据类型提供了包装类。基本数据类型对应包装类有：boolean--Boolean, char--Character, byte--Byte, short--Short, int--Integer, long--Long, float--Float, double—Double。

引用类型和基本数据类型具有不同的特征和用法，它们包括：大小和速度问题、数据结构存储问题。当引用类型和基本数据类型用作某个类的实例数据时所指定的缺省值，对象引用实例变量的缺省值为 null，而基本数据类型实例变量的缺省值与它们的类型有关。

1. 请说明String、StringBuffer、StringBuilder的区别。

String的长度是不可变的，StringBuffer和StringBuilider的长度是可变的。如果你对字符串中的内容经常进行操作，特别是内容要修改时，那么使用StringBuffer，如果最后需要String，那么使用StringBuffer的toString()方法。

StringBuilder类是一个可变的字符序列。

JDK自从5.0版本后引入。 StringBuilder类被设计用作 StringBuffer 的一个简易替换，用在字符串缓冲区被单个线程使用的时候 。

1. 请描述ArrayList、LinkedList的区别。

ArrayList和Vector都是使用数组方式存储数据，此数组元素数大于实际存储的数据以便增加和插入元素，它们都允许直接按序号索引元素，但是插入元素要涉及数组元素移动等内存操作，所以索引数据快而插入数据慢，Vector由于使用了synchronized方法（线程安全），通常性能上较ArrayList差，而LinkedList使用双向链表实现存储，按序号索引数据需要进行前向或后向遍历，但是插入数据时只需要记录本项的前后项即可，所以插入速度较快。

LinkedList也是线程不安全的，LinkedList提供了一些方法，使得LinkedList可以被当作堆栈和队列来使用。

1. 请描述Iteartor接口的3个常用API方法，并说明方法含义。

next()：返回游标右边的元素并将游标移动到下一个位置

hasNext()：判断游标右边是否有元素

remove()：删除游标左边的元素，在执行完next之后，该操作只能执行一次。

1. 请列出Map接口的两个实现类，并描述区别？

HashMap和TreeMap。

HashMap基于哈希表实现。TreeMap基于树实现。

HashMap可以通过调优初始容量和负载因子，优化HashMap空间的使用。

TreeMap没有调优选项，因为该树总处于平衡状态

HashMap性能优于TreeMap。

1. 简述Collection和Collections的区别？

Collection是集合类的上级接口，继承与他的接口主要有Set 和List.

Collections是针对集合类的一个帮助类，他提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作。

1. 简述数组和容器的区别。

数组是放同一数据类型的集合。容器中元素的类型可以不同。

数组是定长的，容器的长度可变。

1. String s=new String(“abc”)创建了几个String对象。

两个，”abc”是一个对象，这个对象放在字符串常量缓冲区。new String()每写一遍，就在堆中创建一个新的对象，它依据那个常量”xyz”对象的内容来创建出一个新String对象。

1. ==和equals有什么区别？

没有区别，Object中equals方法就是用==来实现的

1. **编程题(本大题共3小题，共20分)**

1、请使用JDBC编写连接本地数据库emp 表的实现查询功能的代码。（10分）

String driver = "oracle.jdbc.driver.OracleDriver";

String database = "DEMOJDBC";

String url = "jdbc:oracle:thin:@localhost:1521:" + database;

String userName = "SYSMAN";

String password = "ADMIN";

Connection con = null;

Statement st = null;

ResultSet rs = null;

String sql = null;

try {

Class.forName(driver); ---------（1分）

con = DriverManager.getConnection(url, userName, password); ---------（1分）

System.out.println("Database connected successfully.");

st = con.createStatement();---------（1分）

String sql = "SELECT \* FROM EMP "; ---------（1分）

rs = st.executeQuery(sql); --------（1分）

while (rs.next()) {--------（1分）

System.out.println("---------------------------------------------");

System.out.print("EMPNO:" + rs.getInt("id") + "\t");--------（1分）

System.out.print("ENAME:" + rs.getString("NAME") + "\t");--------（1分）

System.out.print("JOB:" + rs.getString("sale") + "\t");--------（1分）

}

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

} finally {

if (rs != null) {

try {

rs.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

if (st != null) {

try {

st.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

if (con != null) {

try {

con.close();--------（1分）

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

2、请输入一组随意的数字组成数组，然后用冒泡法进行数组排序（5分）

int arr[]={10,9,8,7,6,5,4,3,2,1};

int i=0,j=0;

for(i=0;i<10;i++)-------（1分）

{

int temp=0;

for(j=i+1;j<10-i;j++)-------（1分）

{

if(arr[i]>arr[i+1]) -------（1分）

{

temp=arr[i]; arr[i]=arr[i+1]; arr[i+1]=temp; -------（2分）

}

}

System.out.print(arr[i]);

}

3、设计一个计时器，计算函数query()运行所用时间（提示使用多线程实现）（5分）

public static query() throws Exception

{

for(int i =0;i<10;i++)

{

Thread.sleep(1000);

}

}

public static void main(String[] args) {

// TODO Auto-generated method stub

int a=0 ;

Emp emp = new Emp();

System.out.println("start");

//计时

new Thread(new TestThreadTimer().new TestRunnable(emp)).start();// 构造函数 初始化 +

query(emp);// 执行时间

System.out.println("end");

}

class TestRunnable implements Runnable{

Emp emp ;

public TestRunnable(Emp a) {

// TODO Auto-generated constructor stub

emp = a;

}

@Override

public void run() {

// TODO Auto-generated method stubC

int i = 0;

//while(TestMain1.toCLose)

while(emp.getToClose())

{

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

i++;

}

System.out.println("运行了"+i);

}

}