**第1章 Java概述**

1、java的特点

1)**跨平台性**: 一个应用程序能够运行与不同的操作系统。

利用**虚拟机(JVM)**技术支持跨平台特性。

2）**完全面向对象**

所有的设计都是在类中实现的，一个java程序就是多个类的集合。

3）可靠性

4）安全性

采用**域管理方式**的安全模型

5）多线程

6）支持分布式网络应用

2、Java应用程序

1）Application

2）Applet

Java**源程序文件（\*.java）**通过编译生成字**节码文件(\*.class)**,再由解释器执行。

Java编译器：javac.exe

Java解释器：java.exe

Java调试器：jdb.exe

3、Java环境变量的设置

path：提供可执行文件（.exe）

classpath：提供类文件（.class）的路径

4、包

包（package）是类的集合。包是**Java区别类名字空间的机制**。一个包中的多个类之间不能重名，不同包中的类名则可以相同。

导入包：import

声明类所在的包：package

5、Java源程序结构的理解：

packageimportclass

声明包的语句只有一个，并且第一条语句；

声明public的类或接口只能有一个；

**Java对源程序文件中的声明语句及文件其命名规则有什么要求？**

在一个Java源程序文件（\*.java）中，可以使用package语句声明包，使用import语句导入包，之后使用class或interface声明多个类或接口。其中，声明为public权限的类或接口只能有一个，且文件名必须与该类名相同。

**程序中的错误有哪几种？分别在什么时刻被发现？**

**语法错、语义错、逻辑错**。编译时能够发现语法错，运行时能够发现语义错，运行时不能发现逻辑错。

**第2章 Java语言基础**

1、Java语言的命名规则

2、数据类型：一个类型和定义在这个类型上的操作集合。

3、Java分为：**基本数据类型**和**引用数据类型**

1）**基本数据类型：整数类型、浮点类型、字符类型和布尔类型**

2）**引用数据类型：数组、类和接口**

4、java是**强类型语言**，遵循**“先声明，后使用”**的原则。

5、变量的4个基本要素：**名字、类型、值和作用域**，变量声明的位置决定变量的作用域。

最终变量：final声明，只能赋值一次

常量：**直接常量**和**符号常量**

6、运算符与操作数，运算分为**单目运算**、**双目运算**和**三目运算**

7、运算符分类：

1）算术运算算符 2）关系运算符

3）位运算符 4）逻辑运算符

5）赋值运算符 6）类型强制转换符

7）条件运算符 ？ ：

8）括号运算符 9）字符串连接运算符

10）点运算符 11）对象运算符 instanceof

12)new

8、运算符的优先级和结合性

9、运算的类型兼容性

1）类型相容 2）赋值相容

10、流程控制语句

switch语句中的，表达式和常量表达式的数据据类型必须是**整数或字符类型**。

11、数组

1） Java的数组都是**动态数组**，声明数组后，使用**new**运算符申请空间。

2）Java声明数组和数组赋值的规则和形式（判断）

3）for语句的**逐元循环： for(类型 变量：数组)**

**4）数组的引用模型**

**5）二维数组的引用模型**

**6）不规则的二维数组**

**12、Java没有全局函数，**函数都是声明在类中，**成员方法，分为静态方法和实例方法。**

13、一个类只有包含main()方法，才可以被Java虚拟机执行。

**publicstaticvoid main(String args[]){ }**

14、参数传递

1）分为**传递值和引用**

2）实参向形参传递的原则

3）常量式形参

4）可变式形参

15、方法重载

多个同名方法但带有不同的参数列表，成为方法重载；

**Java的方法没有参数默认值。**

16.递归方法

17、字符串常量：由**双引号括起来的字符序列**。

18、字符串的运算(判断正误、指出运算结果)

1）赋值

2）连接运算

19、字符串与字符数组不同

字符串不是字符数组。

20、字符串的特性

String是一个类，有很多成员方法。

第3章类的封装、继承和多态

1、类的概念和对象的概念

类是描述对象的数据类型，刻画一组对象的属性和行为，类具有封装性、继承性、多态性和抽象性。

对象是类的变量，实例是类的取值。

2、Java程序设计的基本单位是类

[修饰符] class 类<泛型>[extends父类][implements 接口列表]

{

成员变量的声明；

成员方法的声明及实现；

}

3、对象的使用过程

先声明对象所属的类，动态申请创建一个指定类的实例，并使对象引用该实例，再访问对象的成员变量，调用对象的成员方法，使用完后释放对象。

MyDate d1=new MyDate();

d1.set(2017,11,27);

MyDate d2=d1;//对象引用赋值

4、类封装的两层含义：

1）将数据和对数据的操作包装成一个对象模型；

2）实现信息隐藏，类既要提供对外联系的方法，又要尽可能隐藏类中数据和实现细节。

5、类的构造方法的作用：

用于创建类的一个实例并对实例的成员变量进行初始化。

6、类的构造方法的特点：

1）构造方法与类同名；

2）构造方法通过new运算符调用；

3) 构造方法不需要写返回值类型，因为它的返回值就是该类的一个实例；

4）构造方法可以重载；

7、默认构造方法

1)当一个类没有声明构造方法时，java会自动提供一个无参数的默认构造方法，对成员变量进行初始化。

2)当一个类声明了构造方法时，Java就不再提供默认构造方法了。

3)一个类需要声明一个无参数的构造方法为成员变量提供默认值，同时也为子类提供可用的父类的默认构造方法。

8、拷贝构造方法

1)Java不提供默认的拷贝构造方法；

2)参数为该类对象的构造方法称为拷贝构造方法，实现对象的复制功能；

3)拷贝构造方法分为浅拷贝和深拷贝：

使用已知实例对新创建实例的成员变量逐个赋值，称为浅拷贝；

当一个类包含引用数据类型的成员变量时，该类的拷贝构造方法，不仅要复制对象的所有非引用成员变量的值，还要为引用数据成员的成员变量创建新的实例，并初始化为形式参数实例值，这种称为深拷贝。

9、析构方法的作用:

释放实例并提供特定操作。

10、一个类只用一个finalize()方法，不能重载。

11.Java每个成员方法都可以用this引用该方法的调用对象。

12. 运算符instanceof

13、对象的关系运算符

1)类使用==和!=运算符比较两个对象是否引用同一个实例，不比较对象的地址；

2)equals()方法用于比较两个对象是否相等，比较对象的地址

15、访问控制

详见教材P69

16、Java的成员分为；实例成员和静态成员

1)实例成员属于对象，只有创建了实例，才能访问；

2)使用static声明的成员，称为静态成员(或类方法)，属于类，可以直接用类名访问；

3)静态成员变量的赋值：声明时赋值，或使用static块初始化静态成员变量，不能再构造方法中初始化静态成员。

4)静态成员方法中不能访问实例成员，也不能使用this引用。

17、类的继承性：父类(超类)与子类(派生类)

18、继承的原则

1)子类继承父类所有的成员变量

2)子类继承父类除构造方法以外的成员方法，包括实例方法、静态方法。

3)子类不能继承父类的构造方法，子类必须声明并实现自己的构造方法。

4)子类可以增加成员，，也可以重新定义从父类继承的成员，但不能上出它们。

19、Java只支持单继承，即一个子类只有一个父类。

20、继承是实现软件复用的重要措施。通常祖先类用于通用功能的设计，后代类用于特定功能的设计。

21、当一个类没有声明父类时，默认的父类是Object。

22、子类对父类成员的访问权限：

1)子类不能访问父类的私有成员；

2)子类能够访问父类的公有成员和受保护成员；

3)子类可以访问当前包中父类的缺省权限成员，不能访问其他包中父类的缺省权限成员。

23、类中成员的访问控制原则：

1)仅限本类使用，声明为private;

2)可被子类使用，声明为protected;

3)可被所有类使用，声明为public.

24、子类的构造方法

1)子类的构造方法需要调用父类的构造方法对父类声明的成员变量进行初始化；

2)如果子类没有显式调用父类构造方法，Java将默认调用父类无参数构造方法；

3)子类使用super(参数列表)调用父类的构造方法，并且必须是第一条语句；

4）默认执行super()的情况：

①当一个类没有声明构造方法时，Java为该类提供默认构造方法，调用super()执行父类无参数的构造方法。

②如果子类的构造方法没有调用super()或this()，Java将默认执行super().

25、Java构造方法执行的顺序：

先执行父类的构造方法，再执行子类的构造方法；

26、Java析构方法执行的顺序

先执行子类的析构方法，再执行父类的析构方法。

27、类的多态性：

方法的多态和类型的多态，方法的多态包括方法的重载和覆盖，类型的多态表现为子类是一种父类类型。

28、子类重定义父类成员包括：

1)重定义父类的成员变量，则隐藏父类的成员；

2)重定义父类的成员方法，如果参数列表和返回值类型均相同，则覆盖父类方法；如果参数列表不同，则重载父类的成员方法。

29、子类的实例方法中，使用“super引用”访问被子类隐藏的父类同名成员变量，调用父类被子类覆盖的同名成员方法。静态方法不能用super引用。

30、子类对象即是父类对象，父类对象不是子类对象

31、父类对象可以引用子类实例，子类对象不能引用父类实例。

32、编译时多态和运行时多态，方法重载都是编译时多态。

33、父类对象只能执行在父类中声明、被子类覆盖的子类方法，不能执行子类增加的成员方法。

34、抽象类和抽象方法

抽象类不能被实例化，抽象类通常含有抽象方法，也可以不含抽象方法，但是包含抽象方法的类一定是抽象类。构造方法、静态方法不能被声明为抽象方法。

35、最终类与最终方法：

最终类不能被继承，最终方法不能被子类覆盖。

最终类中包含的都是最终方法，非最终类中也可以含有最终方法。

第4章 接口、内部类和JavaAPI基础

1、接口提供方法声明和方法实现相分离的机制，接口机制使Java具有实现多继承的能力。

2、接口是一组抽象方法、常量和内嵌类型的集合，接口也是一种数据类型。

3、接口可以被多个类实现，接口声明的抽象方法在实现接口的多个类中表现多态性。

4、接口声明格式：

**[public] interface 接口 <泛型> [extends 父接口列表]**

**{**

**[public][static][final] 数据类型 成员变量=常量值；**

**[public][abstract] 返回值类型 成员方法(参数列表)；**

**}**

1)接口中不能有构造方法，声明的都是抽象的实例方法，不能是静态方法；

2）接口访问权限是public 或缺省。

3）接口中成员变量都是常量，不能声明为实例成员变量。

5、声明实现接口的类

**[修饰符] class类<泛型> [extends 父类] [implements 接口列表]**

6、接口的继承是多继承，一个接口可以继承多个父接口。

7、一个类C实现一个接口I，则类C及其子类的实例既属于类C的类型，也属于接口I的类型。

8、抽象类与接口的区别：

1）抽象类为子类约定方法声明，可以部分给出实现，类是单继承；

2）接口是为多个互不相关的类约定某些方法声明，没有实现部分，是多继承。

9、内部类和内部接口：

10、JavaAPI基础

1）Java通过Comparable接口的compareTo()方法约定两个对象比较的规则。

**public interface Comparable<T> //泛型T是实现该接口的类**

**{**

**public abstract int compareTo(T cobj);**

**}**

2）基本数据类型的包装类：

Byte、Short、Integer、Long、Float、Double、Character、Boolean

3）字符串类String

两个字符串常量可用==和！=比较是否相等；

字符串变量需调用String类的equals()方法比较其引用的字符串常量是否相等。

字符串比较大小，利用String类的compareTo()方法。

4）java.util.Comparator比较器接口，可以比较两个对象的大小

**public interface Comparator<T> //泛型T是实现该接口的类**

**{**

**public abstract booleanequals(Object obj);**

**public abstract int compareTo(T cobj1，T cobj2);**

**}**

11、泛型

泛型是指类、接口及方法声明的类型参数。泛型通过为类、接口及方法设置类型参数，使一个类或一个方法可在多种类型的对象上进行操作。

12、泛型声明的格式：

[修饰符] class 类<类型参数列表> [extends 父类][implements 接口列表]

[public] interface 接口<类型参数列表>[extends 父接口列表]

[public] static<类型参数列表>返回值类型 方法([参数列表])[throws异常类列表]

13、泛型的继承性与通配符

? extends T //？表示T及其任意一种子类型，T为？的上界

? super T //？表示T及其任意一种父类型，T为？的下界

第5章 异常处理

1、错误是指程序运行时遇到的硬件错误或，或操作系统、虚拟机等系统软件错误或操作错误。

2、异常是指硬件、操作系统或虚拟机等系统软件运行正常时，程序产生的运行错误。Exception是Java所有异常类的根类。

3、Java的异常类分为运行异常类和非运行异常类。运行异常类是指程序本身错误或数据错误所引发的异常(程序设计时大多可以避免)；非运行异常是指有程序运行环境错误引发的异常(这类异常必须捕获并处理)。

4、异常处理的措施

1）try-catch-final语句

**try{ 潜在异常的语句 }**

**catch(异常类 异常对象){ 捕获异常对象并进行处理的语句 }**

**final{ 最后必须执行的语句，无论是否有异常 }**

5、利用throw语句抛出异常

**throw异常对象**

由throw语句抛出的异常对象必须由try语句捕获并处理。

6、如果一个方法产生异常，而方法不想处理或不能处理，可以在方法声明时，采用throws子句声明该方法将抛出异常。

**[修饰符] 返回值类型 方法([参数列表]) [throws 异常类列表]**

throws子句的作用：声明方法抛出异常，则方法的调用者必须捕获并处理该异常。

7、当Java提供的异常类不能满足需要时，程序可以自定义异常类，由try-catch子句捕获并处理。

第6章 图形用户界面

1、Java图像用户界面的组件和事件由java.awt和javax.swing包共同提供。

2、java.awt是抽象窗口工具集，javax.swing是AWT组件的扩展。

3、组件是具有以下特性的对象：

1）运行时可见

2）具有坐标位置、尺寸、字体、颜色等属性

3）可以拥有并管理其他组件，可以获得输入焦点，可以响应事件。

4、Component组件是所有组件的根类。

5、Container是Component的子类，可以容纳其他组件。Window窗口类和Panel面板类是Container的两个子类。

4、Window窗口的子类：Frame框架和Dialog对话框。

5、java.awt布局管理器主要有：FlowLayout、BorderLayout、GridLayout、GridBagLayout。

6、事件处理

**publicclass含有组件的类 implements事件监听器接口**

**{**

**组件注册事件监听器对象；**

**}**

**publicviod事件处理方法(参数列表)**

**{**

**}**

java.awt.event包中提供AWT事件监听器接口，详见教材P145

AWT组件类中注册事件监听器的方法，详见教材P146

7、Swing组件是AWT组件的扩展，属于轻型组件，可实现Java的跨平台特性。

8、Swing组件在javax.swing包中，Swing事件在javax.swing.event中。

9、Swing组件以“J”开头，都是容器。

10、JFrame和JDialog是Swing顶层容器，Swing组件不能直接添加到顶层容器中，而是加到顶层容器包含的一个成为内容窗格容器中。

11、Swing组件新增两种布局：BoxLayout和Spring Layout

12、各种Swing组件的使用

13、java.awt.Graphics图形类定义设置图形颜色、字体等属性方法以及绘制各种图形的方法。

14、java.awt.Image图像类是抽象类，提供获得绘图对象、图像缩放等功能。

第7章 多线程

1、操作系统中的进程和线程

1）线程和进程的基本概念

2）线程和进程的特性

3）线程的状态和调度

4）并发程序设计的特征

2、Java内置多线程机制。

3、Runnable接口约定线程的执行方法，Thread类创建、管理和控制线程对象。

4、Runnable接口

**public interface Runnable**

**{ public abstract void run();}**

一个线程对象必须实现run()方法才能描述该线程的所有活动及操作。

5、Thread类声明实现Runnable接口。线程对象执行run()方法，该run()方法既可以由该线程自身提供，也可以由实现了Runnable接口的对象提供。

6、Java提供的两种多线程设计的方法：

1）继承Thread类

声明一个线程对象继承Thread类，并且覆盖Thread类的run()方法。

2）实现Runnable接口

一个实现Runnable接口的对象本身不是线程对象，它可以作为一个线程对象的目标对象使用，因此需要同时声明一个Thread线程对象。

7、线程对象的优先级

8、线程的状态和状态改变

9、交互线程：并发执行的需要共享资源或交换数据的多个线程

10、交互线程间的关系：竞争关系和协作关系

11、线程互斥：指若干线程要使用同一共享资源时，任何时刻最多允许一个线程使用，其他要使用该资源的线程必须等待，直到占有资源的线程释放该资源。

12、临界资源与临界区

把共享变量代表的资源称为临界资源，并发线程中与共享变量有关的程序段称为临界区。

13、临界区调度的原则：无空等待、有空让进、择一而入、算法可行

14、操作系统采用“互斥锁”机制实现并发线程互斥进入临界区，对共享资源进行操作。

15、Java利用synchronized声明一段程序为临界区，可以声明一条语句，也可以声明一个方法。

1） **synchronized (对象){语句或语句块}**

2） **sychronized 方法声明{ }**

16、线程同步是指两个以上线程基于某个条件来协调他们的活动，是解决线程协作的手段。

17、操作系统实现线程同步的工具是信号量和PV操作。

18、信号量：共享变量

19、测试信号量状态的操作称为P操作，改变信号量的操作称为V操作。

20、能够利用“互斥锁”和PV操作，来实现线程的互斥和同步。

第8章 输入/输出流和文件操作

1、文件系统提供目录机制实现文件的“按名存取”。

2、按照文件的逻辑结构，分为流式文件和记录文件。

3、文件的存取方式：顺序存取、随机存取和索引存取。

4、流是指一组有顺序的、有起点和终点的字节集合，是对数据传输的总称。

5、流根据方向分为输入流和输出流，基本操作有读操作和写操作，从流中取得数据的操作称为读操作，向流中添加数据的操作称为写操作。对输入流只能进行读操作，对输出流只能进行写操作。

6、设计流的目的是使数据的传输操作独立于相关设备。

7、按流中元素的基本单位，流分为字节流和字符流。

8、抽象字节流

1）InputStream 2)OutputStream

9、文件字节流

1）FileInputStream 2）FileOutputStream

10、数据字节流

1）DataInputStream 2）DataOutputStream

11、对象字节流

1）ObjectInputStream 2）ObjectOutputStream

12、管道字节流

1）PipedInputStream 2）PipedOutputStream

13、抽象字符流

1）Reader 2）Writer

14、字节/字符转换流

1）InputStreamReader 2）OutputStreamWriter

15、文件字符流

1）FileReader 2）FileWriter

16、缓冲字符流

1）BufferedReader 2）BufferedWriter

17、格式化字符输出流

PrintWriter

18、Java标准输入输出

1）System.in System.out System.err

2) PrintStream

19、文件类及其过滤器

1）File类

2）FileFilter接口

Java程序中，可以使用文件过滤器接口和文件类，指定文件过滤条件来实现获取部分文件的功能。

20、文件选择对话框JFileChooser及其文件过滤器

21、随机存取文件类RandomAccessFile，实现对一个文件同时进行既读又写的操作。