# 第一章

1.Java语言有哪些特点？

Java是一种具有“简单、面向对象、平台无关性与可移植性、稳定性和安全性、多线程并且是动态的、高性能、分布式”等特点的语言。

2.虚拟机、垃圾收集器

# 第三章

1.编写程序为什么要注释？

主要目的是为了提高Java程序的可读性，好的程序应首先易于阅读，其次才是效率高低的问题。程序员按照一定的要求书写注释，将来就可以用javadoc自动生成帮助文档，节约了大量宝贵的时间。

# 第五章

4.protected修饰符有何特点？

声明为protected的方法和成员变量能被同一个包里的所有类访问，能被该类的子类所访问，子类可以和父类不在一个包里。

5.Object都有哪些方法？

equals（）、toString（）

# 第六章

1.this和super各有几种方法？

this用法有三种：

(1).this.域变量、this.成员方法：在方法内借助this来明确表示用的是类的域变量；

(2).this(参数)——引用重载的构造方法；

(3).this代指当前对象。

super用法有两种：

(1).super.域变量、super,成员方法(参数)；

(2).super(参数)，构造方法的继承

5.接口有什么作用？自己定义一个接口，并给出实现类和使用类。

·（1）.通过接口可以实现不相关类的相同行为，而不需要了解对象所对应的类；

（2）.通过接口可以指明多个类需要实现的方法；

（3）.通过接口可以了解对象的交互界面，而不需要了解对象对应的类

//定义接口：

interface Human

{

public static string sex=man;

public abstract void eat();

public abstract void sleep();

public abstract void communicate();

}

//实现接口：

class Chinese implements Human

{

public void eat(){ System.out.println(“Chinese eat rice!”);}

public void sleep(){ System.out.println(“Chinese sleep for 8 hours!”);}

public void communicate(){ System.out.println(“Chinese communicate in Chinese!”);}

}

//使用接口：

public class Test

{

public static void main(String args[])

{

Human h=new Chinese();

h.eat();

h.sleep();

h.communicate();

}

}

6.抽象类与接口的异同点是什么？

|  |  |  |  |
| --- | --- | --- | --- |
|  | | 抽象类 | 接口 |
|
| 相同点 | | 二者都有抽象方法，都不能实例化。都有自己的声明，并能引用具体子类或实现类对象 | |
|
| 不同点 | 属性 | 可以有域变量 | 不能有域变量，只能是静态变量 |
| 成员方法 | 可以有具体方法，而且具体方法可以调用抽象方法 | 如果有方法，则全部是抽象方法 |
|
| 实现策略 | 必须有子类继承 | 必须有实现类实现 |
| 扩展性 | 弱 | 强 |

7.引用比较方法有哪些？

（1）.equals方法是object的方法，因此所有类对象都可以利用他进行比较，判断是否指向同一对象。

（2）.如果“==”两边是对象引用，则比较的是它们的引用是否相同；如果两边是数值，则比较的是它们的值（如果值类型不同，有可能发生类型转换）；如果一边是引用，一边是值，则编译错误。

（3）.instanceof运算符的格式：“a instanceof A”，其中a为对象的引用，A为类。如果a为A的实例或A子类的实例，则返回true；如果a为A父类的实例，则返回false；如果a对象的类和A没有任何关系，则编译不会通过。

8.内部类的作用是什么？什么情况下使用匿名内部类？

内部类就是在某个类的内部又定义了一个类，被内部类嵌入的类称为外部类。

作用:

（1）.内部类可以很好的实现隐藏。一般的非内部类，是不允许有private和protected权限的，但内部类可以；

（2）.内部类拥有外围类的所有元素的访问权限；

（3）.可以实现多重继承；

（4）.可以避免修改接口而实现同一个类中两种同名方法的调用。

匿名内部类就是在类中需要实例化这个类的地方（通常为方法内），定义一个没有名称得类。使用条件：

（1）.只用到类的一个实例；

（2）.类在定义后马上用到；

（3）.类非常小（Sun推荐是在四行代码以下）。

# 第七章

1.“程序中凡是可能出现异常的地方必须进行捕获或抛出”，这句话对吗？

不对，runtime异常不进行try/catch编译不会出错,但程序会因此停止；而非runtime异常在程序里不进行try/catch或throw，编译就会出错，所以必须捕获或抛出。

2.自定义一个异常类，并在程序中主动产生这个异常类对象。

public class SelfGenerateException extends Exception

{

SelfGenerateException(String msg)

{

super(msg);

}

static void throwtest()throws SelfGenerateException

{

int a=1;

if(a==1)

{

throw new SelfGenerateException("a为1!");

}

}

public static void main(String args[])

{

try

{

throwtest();

}catch(SelfGenerateException e)

{

e.printStackTrace();

}

}

}

# 第八章

5.String与StringBuffer的区别是什么?如何相互转换？

String是对象不是原始类型；为不可变对象，一旦创建就不能修改它的值；对于已经存在的String对象的修改都是重新创建一个新的对象，然后把新的值保存进去。String是final类，不能被继承。

StringBuffer是一个可变对象，当对它进行修改的时候不会像String那样重新建立对象，它只能通过构造函数来建立。不能通过赋值符号对它进行赋值。对象被建立以后，在内存中就会分配内存空间，并初始保存一个NULL，向其赋值的时候可以通过它的append方法。

StringBuffer的构造方法可以将一个String对象转化为StringBuffer，如：

String s；

StringBuffer sb=new StringBuffer（s）；

而其方法toString()可以将一个StringBuffer转换成一个String对象，如：

String s=new StringBuffer().append(“a”).toString（）；

# 第九章

1.线程和进程的联系和区别是什么？

线程是隶属于操作系统的概念，是程序执行中的单个顺序流程。

进程就是一个执行中的程序，是操作系统对其资源（内存和CPU时间等）进行分配的基本单位。

多线程则是指一个进程中可以同时运行多个不同的线程，执行不同的任务。

区别：同类的多个线程共享一块内存空间和一组系统资源，而线程本身的数据通常只有微处理器的寄存器数据，以及一个供程序执行时使用的堆栈。系统在产生一个线程，或者在各个线程之间切换时，负担要比进程小得多，线程被称为轻负荷进程。

3.创建线程有几种方法？他们之间的区别是什么?

Java产生线程有两种方法：一种是继承Thread类，且覆盖其run方法；另一种就是实现Runnable接口，并将实现类对象作为参数传递给Thread类的构造方法。

区别：(1).直接继承线程Thread类：该方法编写简单，可以直接操作线程。由于已经继承了Thread，不能再继承其他类了。

(2).实现Runnable接口：当一个类已继承了另一个类时，就只能用实现Runnable接口的方式来创建线程。另外，使用此方法的更多原因是多个线程可共享实现类对象的资源。

4.线程的生命周期有哪些状态？哪些方法可以改变这些形态？

五个状态：

创建状态；

可运行（就绪）状态；

运行中的状态；

阻塞状态；

死亡状态。

线程状态的改变：

线程睡眠:

public static void sleep (long millis) throws InterruptedException

暂停线程:

public static void yield()

连接线程join()的三种用法：

1. 等待线程结束:

public final void join() throws InterruptedExpection

1. 最多为线程等待millis毫秒：

public final void join(long millis) throws InterruptedExpection

1. 最多为线程等待millis毫秒+nanos纳秒：

public final void join(long millis,int nanos) throws InterruptedExpection

5.什么是线程安全？为什么会产生线程安全问题？如何解决线程安全问题？

一个类或者程序所提供的接口对于线程来说是原子操作或者多个线程之间的切换不会导致该接口的执行结果存在二义性，叫做线程安全。

线程安全问题都是由全局变量及静态变量引起的。若每个线程中对全局变量、静态变量只有读操作，而无写操作，这个全局变量是线程安全的；若有多个线程同时执行写操作，一般都需要考虑线程同步，否则就可能影响线程安全。

为解决线程安全问题，Java引入监视器来保证共享数据操作的同步性。用关键字synchronized修饰某个对象后该对象就成了监视器。

Synchronized有如下三种用法：

（1）.synchronized代码块：监视器就是指定的对象；

（2）.synchronized方法：监视器就是this对象；

（3）.synchronized静态方法：监视器就是相应的类。

# 第十二章

1. AWT常用容器有哪些？如何将一个组件加入到容器当中？

面板（Panel）、框架（Frame）、对话框类（Dialog）、滚动面板（ScrollPanel）

通过容器的add（）方法将组件对象加入到容器当中。

9.Java委托事件模型比层次型模型有哪些优点？

（1）.一个事件源对象中的同一事件可以被多个监听者进行处理；

（2）.一个事件源中的多个事件可以分别被不同的监听者进行处理；

（3）.一个监听者可以注册到多个事件源中，对同类事件进行处理。

这些都增加了Java事件处理上的灵活性，是层次型模型不要具备的。

10.事件对象在编程中的作用是什么？

可以利用事件对象提供的方法得到事件源对象信息：

ActionEvent e；

e.getSource（）；