数据类型：8种基本类型，4种整型，2种浮点类型，1种用于表示Unicode编码的字符单元的字符类型char（单个字符），1种boolean类型。

基本类型：变量在栈 引用类型：变量引用到堆

Java中将String类对象称为不可变字符串。可以通过截取、拼接的方式

字符串：equals()检验两个字符串是否相等；equalsIgnoreCase()检测是否相等不区分大小写；

一定不能使用“==”运算符检测两个字符串是否相等，这个运算符只能确定两个字符串是否放在同一个位置上。

空串与Null串：空串“ ”是长度为0的字符串。Null表示没有任何对象与该变量关联

final定义常量。某个常量可以在一个类的多个方法中使用，通常称为类常量。类常量的定义位于main方法外部。使用

public static final double PI=3.14

x < y ? x：y

类：修饰符 class 类名 extends 父类 implements 接口

方法：修饰符 返回值类型 方法名 参数列表 （返回值类型可以是基本类型、数组、类等）

属性：修饰符 数据类型 变量名

对象：

（1）定义对象：new创建的时候，分配内存空间，并返回该对象的一个引用。

（2）this的使用：用于代表当前正在使用或访问的对象

(3)消除对象：

**类的封装：**

类的修饰符 public abstract final friendly(缺省方式为friendly)

abstract类：不能直接产生属于这个类的对象，即不能实例化

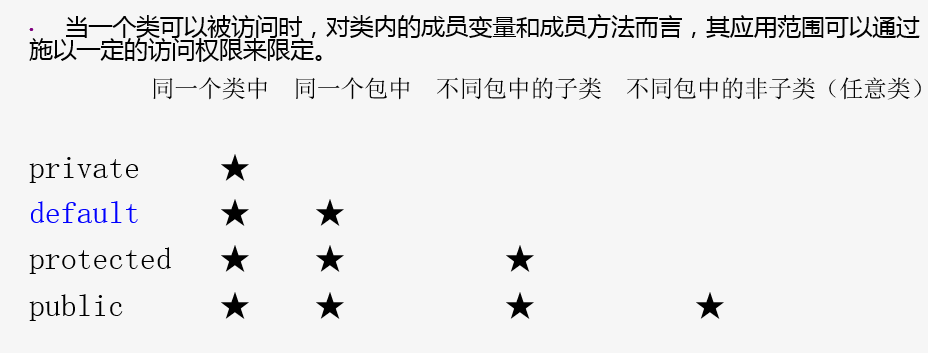
final类：不能被其他任何类所继承

public类：不但可以被同一程序包中的其他类使用，别的程序包中的类也可以使用

friendly类：（缺省，包访问类）只能被本程序包中的其他类使用，不能被别的程序包中的类使用

变量修饰符 public protected private static final transient volatile

方法修饰符 public protected private static final abstract native synchronized



加强：

public: 任何其它类、对象只要可以看到这个类的话，那么它就可以存取变量的数据，或使用方法。

protected：同一类，同一包可以使用。不同包的类要使用，必须是该类的子类。

private：不允许任何其他类存取和调用。

friendly(default，前边没有修饰字的情况)：在同一程序包中出现的类才可以直接使用它的数据和方法.

继承的实现：public class A extends B

java不支持多重继承。子类可以继承父类中设为public protected default的成员变量和方法，不能访问private的变量和方法

类方法的覆盖：即指在子类中重新定义父类中已有的方法

1. 构造函数无法通过继承获得。 需要重写构造函数
2. 方法重写：要求父类与自雷的方法名相同、参数列表、返回值类型也要相同

方法覆盖时应遵循的原则

（1）覆盖后的方法不能比被覆盖的方法有更严格的访问权限。

（2）覆盖后的方法不能比被覆盖的方法产生更多的例外。

如果想在子类中访问父类中被覆盖的成员，可以用super实现，super用来引用当前对象的父类

访问父类成员： super.成员变量 或super.成员方法

访问父类构造方法：super([参数列表])

类方法的重载：即指多个方法可以享有相同的名字。但是这些方法的参数必须不同，或者是参数个数不同，或者是参数类型不同。构造方法也可以被重载。 方法重载不能通过方法的返回值区分。

派生对象的初始化：

1. 利用子类的缺省构造方法初始化
2. 用重载的子类构造方法初始化：

对象间的类型转换：

支持父类和子类之间的类型转换。显示转换格式为：（类名） 类对象

（1）如果是子类对象转换为父类，可进行显示转换或隐式转换

（2）如果父类转换成子类是，编译器首先检查这种是否可以，而且要显示转换

向上转型： Person p = new Man() ; //向上转型不需要强制类型转化  
向下转型： Man man = (Man)new Person() ; //必须强制类型转化

类的多态：类方法的重载是一种多态性。除此之外，多态性还可以是指在程序中需要使用父类对象的地方，都可以用子类对象来代替

1. 编译多态：如方法重载
2. 运行多态：主要由继承和方法重写引起的。凡是父类对象可以使用的地方，子类对象也可以代替使用。如方法重写

调用原则：对子类的实例，如果子类重写了父类的方法，则系统调用子类的方法。如果子类没有重写父类的方法，则调用父类的方法。

其他修饰符：

final:到此为止

* + final在类之前
  + 表示该类是最终类，不能再被继承。
  + final在方法之前
  + 表示该方法是最终方法，该方法不能被任何派生的子类覆盖。
  + final在变量之前
  + 表示变量的值在初始化之后就不能再改变；相当于定义了一个常量。

Static:类变量和类方法

static 在变量或方法之前，表明它们是属于类的，称为类方法（静态方法）或类变量（静态变量）。若无static修饰，则是实例方法和实例变量。类变量在各实例间共享

* + 类变量的生存期不依赖于对象，相当于C语言中全局变量的作用。其它类可以不用通过创建该类的实例，直接通过类名访问它们。
  + 类方法则相当于C语言中的全局函数，其他的类也可以直接通过类名来访问类方法。（Java程序的入口方法main()就是一个类方法）
  + 同一个类的实例方法可以访问该类的类变量和类方法；
  + 而类方法只能访问该类的类变量和类方法，不能直接访问实例的变量和方法。

Abstract 抽象类与抽象方法

* 抽象类：用abstract关键字来修饰一个类时，该类叫做抽象类；

抽象类必须被继承。

抽象类不能被直接实例化。它只能作为其它类的超类，这一点与最终类（final类）正好相反。

* 抽象方法：用abstract来修饰一个方法时，该方法叫做抽象方法。

抽象方法必须被重写

抽象方法只有声明，不能有实现。

定义了抽象方法的类必须是抽象类。 抽象方法不能用private或static关键字修饰

* abstract returnType abstractMethod ( [paramlist] );

instanceof()方法判断对象是否属于某个类的实例

包：

Package

类名有可能相同，用包解决。不同的包允许同名的类

Import(导入非本包中的其他类)

Import java.util.\*

也可以不用import，但要在每个被引入的类前，给出该类所在的包名

接口（interface）: [public] interface interfaceName [extends SuperInterfaceList]

是方法定义和常量值的集合。Java支持单继承，一个类可以实现多个接口。

从本质上讲，接口是一种特殊的抽象类，这种抽象类中只包含常量和方法的定义，而没有方法的实现。接口中的所有方法都是抽象的（abstract可缺省）；接口中的所有数据都是静态常量，即具有public，static,final属性，可以缺省不写修饰符。

接口的实现：

Class 类名 implements 接口名 extends 类名。 实现接口的子类必须重写这些方法

* + 通过接口可以实现不相关类的相同行为，而不需要考虑这些类之间的层次关系。
  + 通过接口可以指明多个类需要实现的方法。
  + 通过接口可以了解对象的交互界面，而不需了解对象所对应的类。
  + 与类继承不同，一个接口可以继承多个父接口
  + 一个public接口只能定义在同名的.java文件中
* 用implements子句来表示一个类使用某个接口。
* 在类体中可以使用接口中定义的常量，而且必须实现接口中定义的所有方法。
* 利用接口可实现多重 继承，即一个类可以实现多个接口，在implements子句中用逗号分隔。
* 接口的作用和抽象类相似，只定义原型，不直接定义方法的内容。
* 接口中的方法和变量必须是public的。

匿名类与内部类：

内部类是指在一个外部类的内部再定义一个类。作为外部类的一个成员。

1. 成员内部类：作为外部类的一个成员，与外部类属性、方法并列。
2. 局部内部类：在外部类的方法体中定义的类；
3. 静态内部类：类似于静态成员，没有外部类对象时，也可以访问它
4. 匿名内部类：没有名字的内部类；所以没办法引用他们，必须在创建时，作为new语句的一部分来声明： new 《类或接口》 《类的主体》

声明一个匿名类（对一个给定的类进行扩展，或者试验一个给定的接口）同时创建该类的一个实例

内部类可以随意使用外部类的成员方法及变量，private的也可以。内部类的实例一定要绑定在外部类的实例上，如果在外部类初始化一个内部类对象，那么内部类对象就会绑定在外部类对象上。

对象的构造方法(Constructor)：

是一种特殊的方法。Java中的每个类都有构造方法，用来初始化该类的一个新的对象。构造方法具有和类名相同的名称，而且不返回任何数据类型。系统在产生对象时会自动执行

This：指向自己的引用，它的一个主要作用是要将自己这个对象当作参数，传送给别的对象中的方法。 调用成员变量和方法 调用构造方法 作为方法的返回值

Super：指这个对象的父类。super可以用来引用父类中的(被覆盖的)方法、(被隐藏的)变量及构造方法

数组：

二维数组：

如果数组中每个元素又是一个数组，就构成多维数组。

For(i=0;i<=arr.length;i++)

For(j=0;j<=arr[i].length;j++)

数组类Arrays:

Java.util.Arrays类中的方法

1. binarySearch 使用二分搜索法在不同类型的数组中搜索特定值，需数组已经排序
2. equals() 比较两个数组是否相等；
3. fill() 用一个指定的值填充数组；
4. sort() 对不同类型的数组排序
5. asList() 返回数组对应的List

字符串处理：

生成字符串：

String类：初始化后不允许改变

StringBuffer类：表示内容可变的字符串对象

访问字符串：

String类：length() startsWith() endsWith()

IndexOf() getChars() chatrAt()

StringBuffer类：同上。

Capacity() 该方法返回StringBuffer缓冲区的容量StringBuffer s1 =new StringBuffer(15) s1.capacity() =15

修改字符串：

String类：不变字符串，不能直接修改它的内容。但可以生成该类的拷贝，对拷贝进行修改

Concat(string str) substring()取子串 replace(char oldchar,char newchar) trim() 去掉两边的空格 static String valueOf()如 String.valueOf()转换成字符串

StringBuffer类：

提供字符串对象直接修改的能力

Append()添加末尾 insert() 插入任何位置 toString() stringBuffer对象转换为String

setCharAt(int index,char ch)

字符串其他操作：

Equal() 比较当前字符串与给定的内容是否相同 equalsIgnoreCase()

S1=”abcd” s2=”adcd” s1=s2 ==比较的是地址空间是否相等，判断是否同一引用

CompareTo() compareToIgnoreCase() 相等为0 当前大于，返回正直，否则负值

正则表达式：

1. 符合体系：
2. 实现方法：java.util.regex

构造一个模式：Pattern p = Pattern.compile(“[a-z]”)

获得一个匹配器：Matcher m = p.matcher(str)

进行判断，得到结果 Boolean b = m.matches();

(3) Matcher类匹配方法：

Matches() 只有整个字符串都匹配了才返回true

lookingAt() 从开头第一个字符进行匹配，匹配成功了不再继续匹配，从第一个字符开始匹配失败了，也不继续匹配

find() 对字符串进行匹配，匹配到的字符串可以在任何位置（找到一个匹配的子串，还会继续找下一个子串）

1. Pattern类方法

Split() 用于分割字符串，并返回一个String[]

第八章 java类库

一、类库概述

Java.lang 语音支持包：字符串、线程等

Java.util工具包：正则表达式，日期等

Java.io输入输出包 文件处理等

Java.awt核心包 gui/图像处理等

Javax.swing扩展包：轻量级GUI组件等

Java.net 网络支持包

Java.applet 小程序applet包

使用java类库的两种方式（1）直接使用系统类：如String 前提是import（java.lang除外）

继承系统类：在用户程序里创建该类的子类。例如每个用户Applet都是继承java.applet包中Applet类的子类。

二、基本数据类型封装

1.数字类

2.

三、Java集合框架

数组不是一个类，存储对象不方便。所以采用集合框架。

1. Java集合概述

集合类的主要作用在于保存多个数据/对象，并提供比数组更灵活的方式，如长度可动态增加的变化。

数组元素既可以是基本类型的值，也可以是对象；而集合只能保存对象。

集合类分为三种：Set List和Map

Set表示不允许容纳重复元素的集合；

List表示可以容纳重复元素的集合；

Map表示存储键/值对的集合，每个键/值对称为一项

1. List接口

List接口的子类包括ArrayList、LinkedList、Vector、Stack。重点掌握ArrayList

ArrayList支持可随需要而增长的动态数组

ArrayList的方法：

Add(Object o) 指定的元素追加到列表的尾部

Add(int index,Object o)将指定的元素插入列表的指定位置

Clear()移除此列表的所有元素

Contains(Object o)包含

Get() 返回此列表中指定位置上的元素

isEmpty()

remove(int index)移除列表中指定位置上的元素

remove(Object o)移除列表指定元素的实例

int size()

LinkedList有点像链表，每一个节点都包含前一个节点的引用，后一个节点的引用和节点存储的值。（双向链表结构）

Vector和ArrayList类似，Vector是同步的，在多线程中安全性更好，但性能稍差

Stack是一种先进先出的堆栈。

1. set接口
2. HashSet

HashSet按照哈希算法来存取对象。向集合中加入一个新对象时，会调用对象的HashCode()方法得到对象的哈希码，然后依据该码计算对象在集合中的存储位置

提供的方法：

Add()

Clear()

Contains() 如果此集合不包含指定元素，则返回true

isEmpty() 如果此集合不包含任何元素，则返回true

remov(Object o) 指定元素存在于此集合中，则将其移除

int size()

1. TreeSet

使用树结构存储元素的类集合，对象按升序存储。为了对有序元素访问，提供以下方法：

First() 返回set中当前第一个（最低）元素

Last() 返回set中当前最后一个（最高）元素

Floor(Object e) 返回set中小于等于给定元素的最大元素

Higher(Objet e) 返回set中严格大于给定元素的最小元素

Lower(Object e) 返回set中严格小于给定元素的最大元素

1. Iterator接口

迭代器，可以遍历集合中的元素，从而获得或删除元素。ListIterator继承Itertor,允许双向遍历列表。

Itertor的主要方法包括：

hasNext() 如果还有未访问元素，返回true,否则返回false

next() 返回下一个元素

romove() 从集合中删除当前元素

使用迭代器循环遍历类集的内容：

1. 通过调用集合类的iterator()方法获得集合类的迭代器
2. 建立一个含有hasNext()方法的循环，只要hasNext()返回true，就进行循环迭代
3. 在循环内部，通过调用next()方法来得到每个元素
4. Map接口

Map是一个存储关键字/值对的集合。关键字和值都是对象。关键字必须唯一，值可以重复

提供的方法：

Clear() 从映射中删除所有的关键字/值对

ContainsKey(Object k) 判断映射中是否包含了关键字K

ContainsVaule(Object v) 判断映射中是否包含了值V

Get(Object k) 返回与关键字k相关联的值

Remove(Object k) 删除关键字等于k的项

IsEmpty() 如果映射是空的，返回true

Set keySet() 返回一个包含映射中关键字的集合

Put(Object o,Object v) 将一个关键字/值对加入映射

Size() 返回映射中关键字/值对的个数

1. HashMap

HashMap类使用散列表实现Map接口，不保证元素的顺序。

1. TreeMap

TreeMap类使用树实现Map接口。TreeMap提供了按顺序存储关键字/值对的有效手段。

备注：List集合与Set集合都实现了Collection接口。

List接口的常用方法：元素可以重复

List<String> list = new LinkedList<String>()

或者 List<String> list = new ArrayList<String>()创建List集合

List.add()

ListIterator<String> firstIterator = list.listIterator() 创建List集合的迭代器

firstIterator.hasNext() 遍历集合中元素

firstIterator.next()

firstIterator.remove()

Set接口的常用方法：不允许存在重复值

Set<String> set = new HashSet<String>() 创建Set集合

Iterator<String> it = set.iterator() 创建Set集合的迭代器

Map接口的常用方法：

Map<String,String> map = new HashMap<String,String>() 创建Map集合

集合类接口的实现类：

List接口的实现类:

List<String> list = new LinkedList<String>() 或者 List<String> list = new ArrayList<String>()

Set接口的实现类：

Set<String> set = new HashSet<String>() 或者 Set<String> set = new TreeSet<String>()

Map接口的实现类：

异常：可控制异常和运行时异常

处理异常三种：

使用try…..catch处理异常:

try

{

需要正常执行但又可能发生异常的语句

}catch(程序需要处理的异常类型)

{

程序发生的异常进行处理的语句

}

使用try……catch……finally处理异常:

try

{

需要执行的语句

}catch()

{ 对异常进行处理的语句

}finally

{

一定会被处理的语句

}

使用try……finally处理异常:

try

{

} finally

{

}

抛出异常：

使用throws声明抛出异常：throws通常用于方法声明。调用该方法的其他方法中对异常要进行处理

Public void method() throws Exception{

}

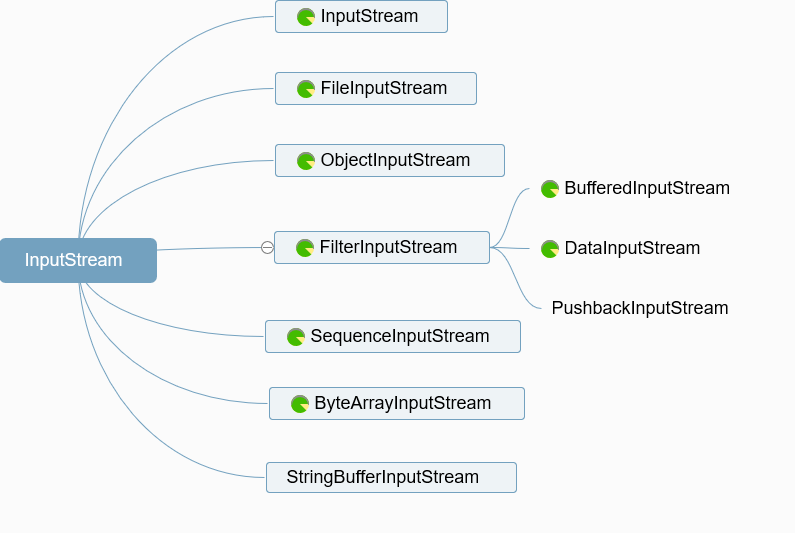
使用throw语句抛出异常：throw语句通常用在方法中，在程序中自行抛出异常。使用throw语句抛出的是异常类的实例，通常与if语句一起使用

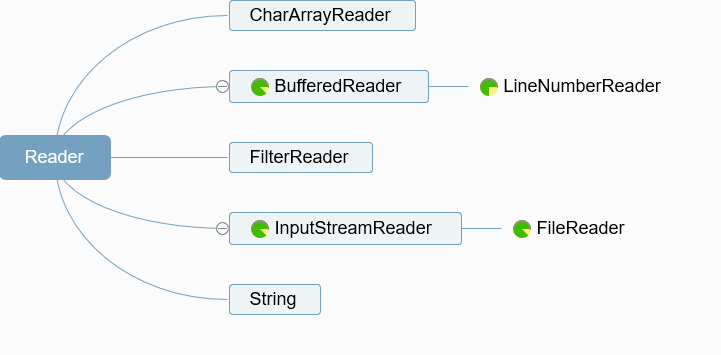
throw new Exception(“对异常的说明”)

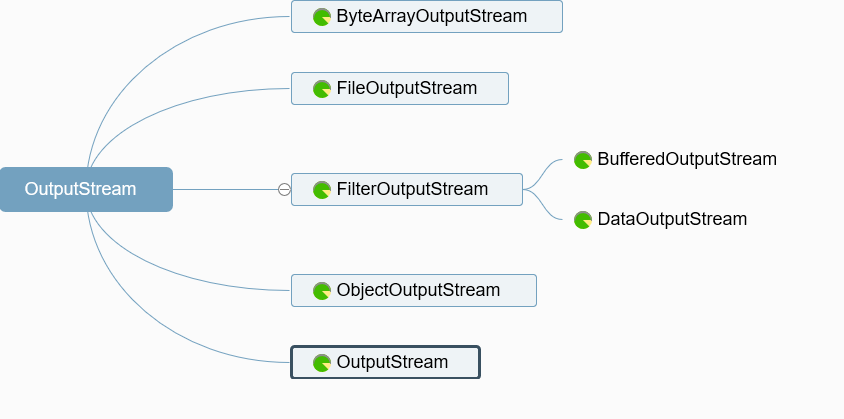
I/O流

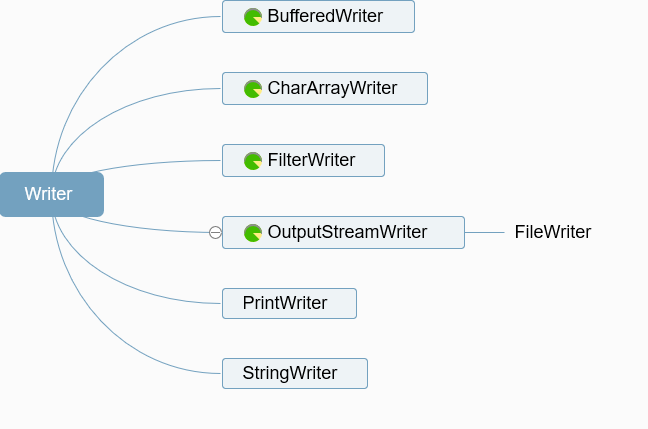
字节输入流：InputStream 字符输入流：Reader

字节输出流：OutputStream 字符输出流：Writer









File类 FileInputStream FileOutputStream FileReader FileWriter

第十二章Java图像界面设计

1. 容器与组件

容器是组件的管理。添加组件等

1. 常用容器

框架 面板 Swing容器：JFrame JPanel

Frame：setSize() setBackground() setvisiable()

Paner add()

1. 布局管理器

SetLayout()

FlowLayout

第二节 awt事件处理机制

1. 事件处理模型

Swing程序设计

Swing常用窗体：

JFrame框架窗体

事件：

多线程