## 第一章 语言介绍与开发环境

Sprite类是一种轻型容器，Shape类代表原始矢量图形 p8

## 第二章 语言基本元素

引用的内存消耗一般比值要小很多 p22

对引用类型而言，const只能保证持有的引用不变，并不保证引用对象自身不变。 p23

Object的成员有两种：属性Property，存放各种数据；方法Method，存放函数对象 p30

全等运算符不执行类型转换 p38

## 第三章 流程控制

continue终止当前这一轮循环，跳到下一轮循环，循环体内continue后语句不执行。 p53

switch是，虚拟机用全等运算符===来比较两个表达式 p57

## 第四章 函数及高级使用技巧

函数使用技巧，代理函数对象、建立函数执行队列…… p68

## 第五章 类和对象——从定义到原理

字符串拼方法名：实例名[“方法名”](参数…) p87

静态属性、方法应用：1集中管理数据 2实现枚举(用来定义一个或多个枚举成员表示一组状态) 3工具类 p90

一个枚举类只包含不同状态的属性，在as3中我们所要实现的Enumeration的分工更加纯粹，仅仅管理不同状态的值 p91

当我们使用new关键字加类名时，构造函数就执行了 p93

构造函数中只可以使用return来控制程序流程 p95

有dynamic是动态类，没有是密封类，尽量使用密封类 p96

this关键字只限于实例属性和实例方法。因为this关键字必须要指代一个具体的实例，不能访问静态属性 p98

使用this，1向其他类提供对象自身引用，2与return结合在类方法中返回自身引用，3明确指定使用实例属性 p98

方法重载：指将写多个同名方法，达到同名的函数实现不同的功能，as3不支持方法重载 p99

通过在方法“()”中写入“...参数”实现方法重载 p99

## 第六章 封装——类的成员、类、包与访问控制

即使使用全饰路径，也必须先导入包 p109

时间轴上写代码，flash.\*包下的类都会在编译时自动导入 p110

Flash IDE两个默认类路径，“.”fla文件当前所在目录，另一个是系统类包的路径，包括flash.\*及所有子包等 p114

Flex IDE类路径分为源码路径（源代码）和类路径（第三方类库） p114

编译器导入package的顺序，类路径1：有，继续导入别的类；没有，类路径2…… p115

Private访问控制只是限制了直接访问实例私有成员，却不能阻止用引用来访问私有属性指向的对象（限制private数组，限制不了数组指向的对象） p119

Protected只能被当前类和当前类子类访问，不受是否在同一包限制 p120

代码升级，老的属性弃用，为了和以前代码兼容，使用getter和setter方法对外部虚拟一个属性。 P122

包外类名字和同一包内的类名重复，包外类优先调用 p124

用包外类实现单例模式（只有一个实例） p124

 1http://www.cnitblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifpackage {  
 2http://www.cnitblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public class Singleton {  
 3http://www.cnitblog.com/Images/OutliningIndicators/InBlock.gif        private static  var \_instance:Singleton;  
 4http://www.cnitblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        public function Singleton(singletoner:Singletoner) {  
 5http://www.cnitblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif            if (singletoner == null) {  
 6http://www.cnitblog.com/Images/OutliningIndicators/InBlock.gif                throw new Error("只能用getInstance()来获取实例");  
 7http://www.cnitblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
 8http://www.cnitblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
 9http://www.cnitblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        public static function getInstance():Singleton {  
10http://www.cnitblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif            if (\_instance == null) {  
11http://www.cnitblog.com/Images/OutliningIndicators/InBlock.gif                \_instance = new Singleton(new Singletoner());  
12http://www.cnitblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
13http://www.cnitblog.com/Images/OutliningIndicators/InBlock.gif            return \_instance;  
14http://www.cnitblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
15http://www.cnitblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
16http://www.cnitblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
17http://www.cnitblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifinternal class Singletoner {  
18http://www.cnitblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}

参考http://www.cnitblog.com/flashlizi/archive/2007/11/09/33236.html

## 第七章 命名空间

定义好的命名空间替换已有的public、internal等 p130

命名空间只用在类成员上，不能用在类上 p131

引用命名空间：1 use namespace 空间名；2对象.命名空间::实例属性（方法） p131

编译器判断命名空间是否相同，就是看uri属性 p133

系统生成命名空间的默认uri：包内类外，“包名:命名空间名”；类中，“包名:类名/命名空间的访问控制:命名空间名”；方法中，“包名:类名/访问控制:方法名/包名:命名空间名” p133

定义命名空间：在独立的.as文件中，可以有访问控制；在类中，相当于静态成员，属于整个类，用“类名.命名空间名”访问 p135

public等本质上都是命名空间，所以用了命名空间就不能再加public这些访问控制符 p135

use namespace只能打开，不能关闭 p136

只要任何地方出现use namespace都会被直接提前，不受上下文逻辑影响 p137

限定符“::”，独立在.as文件中的命名空间、类中定义的命名空间 p137

不同自定义命名空间中可以有重名成员，public等四个不能有重名成员 p141

命名空间应用：多版本控制（debug模式和正常模式、不同语言切换） p143

## 第八章 复合与继承

一般在位图应用、网络连接、端口连接时，使用对象时初始化 p150

被复合对象随复合类一起创建消亡，被聚合对象不一定随聚合类一起创建消亡，可能被第三方使用。复合的对象都是在复合类内部创建，聚合的对象往往是由外部传入引用的 p150

没有声明任何继承，那么实际上是直接继承自Object类 p151

子类具有父类所有非private的实例属性和方法 p151

类的静态成员不被子类继承，但在子类中可以访问父类静态成员 p153

只能重写实例方法，不能重写实例属性；静态成员需要重写，可直接在子类中定义同名成员 p155

重写条件：用override标明这是重写、访问控制相同、参数数目类型相同、返回值相同 p156

将super看成一个变量，直接持有对父类的引用，而super()就是父类的构造函数 p157

通过自定义命名空间，使父类成员对子类有选择的可见（父类中使用use namespace打开的独立.as文件中定义的命名空间，才能让其他外部类使用::访问） p159

父类自定义命名空间被子类继承（父类带命名空间的成员能被子类继承） p160

孔雀是鸟的一种，那么孔雀类是鸟类的子类，继承；轮胎是汽车的一部分，不能因为汽车使用轮胎就说汽车是一种特殊轮胎，复合。 P161

## 第九章 多态

as，当对象符合该数据类型时，返回仍是这个对象的引用，编译器会将返回的对象理解成该数据类型的对象；如不符合，返回null。 p165

as和is搭配使用 p165

向下转换：当一个子类对象以父类数据类型使用时，可以将它再还原成子类对象 p170

向下转换方式，1父类类型对象 as 子类类型，类型不合null；2子类构造函数（父类类型对象），类型不合TypeError异常 p171

向下转换例子 p172

## 第十章 抽象类和接口

类分为具体类（可以生成实例）和抽象类（不需要生成实例） p175

当一个父类不需要生成自己的实例，只是用来继承时，这个类就是抽象类 p175

as3不支持抽象类，不提供abstract关键字，但该关键字被保留 p176

使用抽象类原则：应该拥有尽可能多的子类共同代码；抽象类中数据能少则少；抽象类中定义的抽象方法必须在子类中重写 p176

as3实现抽象类抽象方法，加个第三方（http://mimswright.com/blog/?p=46）的类检测，初始化类或调用方法则抛出异常 p178

能够以其他数据类型向上转型，是接口的核心 p180

创建接口：interface；接口中只定义方法（包括静态、实例、getter和setter），不定义属性；所有定义的方法不可以加访问控制符，接口中方法都是public；方法没有实现，直接以“;”结尾；按习惯接口名称以大写I开头 p181

一个类在定义时如果实现（implements）一个接口，就是宣告：它将拥有这个接口定义的所有方法，可以把它当成这种接口的数据类型来使用 p181

导入多个接口时，各接口中方法名不能同名 p182

实现接口：对接口中定义的方法只能用public访问控制；名称必须和接口中的方法名称相同，参数数目、类型及方法的返回类型必须和接口中相同；参数名、默认参数值不必相同；例子 p182

接口与多继承、向上转换、向下转换，例子 p185

接口及所有父接口中方法不能同名 p186

抽象类和接口区别：抽象类中方法可以部分实现，接口中方法必须为空；抽象类只能是单继承，接口则能让一个类实现多个接口；当需要类作为多种类型存在时，优先选择使用接口；为已存在类添加抽象类型，接口比抽象类简单 p188

单纯给已有的具体类添加新的抽象类型，优先考虑接口；需要给已有的具体类和将要设计的具体类中的共同代码集中管理，优先考虑抽象类 p189

接口和抽象类结合使用：抽象类对接口的实现是被所有子类继承的，子类不必再一一实现接口；为系统扩展留余地 p190

默认适配器（Default Adapter）模式，抽象类实现接口，具体类继承抽象类 p190