**01 block**

**main.m**

#import <Foundation/Foundation.h>

typedef int (^MyBlock)(int,int);

int main(int argc, const char \* argv[]) {

//block用来保存一段代码

//block的标志：^

/\*

block跟函数很像

1、可以保存代码

2、有返回值

3、有形参

4、调用方式一样

\*/

//定义block变量，没有返回值，没有形参

//如果block没有形参，可以省略等号右边的形参括号

void(^myblock)()=^{

NSLog(@"-----------");

NSLog(@"-----------");

};

//利用block变量调用block内部的代码

myblock();

//有形参，有返回值

int (^sumblock)(int,int) = ^(int a,int b){

return a+b;

};

int sum = sumblock(10,11);

NSLog(@"sum is %d",sum);

void (^lineblock)(int) =^(int n){

for (int i = 0; i < n; i++) {

NSLog(@"----------");

}

};

lineblock(5);

int (^sumBlock)(int,int);

sumBlock = ^(int a,int b){

return a+b;

};

// MyBlock sumBlock = ^(int a,int b){

// return a+b;

// };

int (^minusBlock)(int,int) = ^(int a,int b){

return a-b;

};

MyBlock multiplyBlock = ^(int a,int b){

return a\*b;

};

NSLog(@"%d-%d-%d",sumBlock(5,6),minusBlock(8,9),multiplyBlock(5,6));

return 0;

}

**02 protocol**

**MyProtocol.h**

#import <Foundation/Foundation.h>

//定义了一个名叫MyProtocol的协议

@protocol MyProtocol <NSObject>

//@required要求实现，不实现就会发出警告

//@optional不要求实现

-(void)test4;

@required

-(void)test;

-(void)test2;

@optional

-(void)test3;

@end

**Person.h**

#import <Foundation/Foundation.h>

//#import "MyProtocol3.h"

//#import "MyProtocol2.h"

@protocol MyProtocol2;

@protocol MyProtocol3;

//只要一个类遵守了某一个协议，就能拥有这份协议中所有方法的声明

// :继承父类

// <>遵守协议

@interface Person : NSObject <MyProtocol3,MyProtocol2>

@property(nonatomic,strong) id<MyProtocol2> obj;

@end

**Person.m**

#import "Person.h"

#import "MyProtocol2.h"

#import "MyProtocol3.h"

@implementation Person

-(void)test

{

}

-(void)test2

{

}

-(void)test4

{

}

-(void)study

{

}

@end

**MyProtocol2.h**

#import <Foundation/Foundation.h>

@protocol MyProtocol2 <NSObject>

-(void)study;

@optional

-(void)study2;

@end

**MyProtocol3.h**

#import <Foundation/Foundation.h>

#import "MyProtocol.h"

//一个协议遵守了另外一个协议，就拥有了另外一个协议的所有方法的声明

@protocol MyProtocol3 <MyProtocol>

-(void)run;

@end

**Dog.h**

#import <Foundation/Foundation.h>

//#import "MyProtocol2.h"

@protocol MyProtocol2;

@interface Dog : NSObject<MyProtocol2>

@end

**Dog.m**

#import "Dog.h"

@implementation Dog

-(void)study

{

NSLog(@"study");

}

@end

**main.m**

/\*

1、协议的定义

@protocol协议名称<NSObject>

// 方法声明列表等

@end

2、如何遵守协议

1>一个类如何遵守协议

@interface 类名：父类名<协议名称1、协议名称2>

@end

2>协议遵守协议

@protocol 协议名称<其他协议名称1、其他协议名称2>

@end

3、协议中方法声明的关键字

1>@required（默认）

要求实现，如果没有实现，会发出警告

2>optional

不要求实现

4、定义一个变量的时候，可以限制这个变量保存的对象遵守某个协议

类名<协议名称> \*变量名

id<MyProtocol> obj;

NSObject<MyProtocol> \*obj2;

如果没有遵守对应的协议，编译器会发出警告

5、@Property中声明的属性也可用作一个声明协议的限制

@property(nonatomic,strong)类名<协议名称> \*属性名

6、协议可以定义在单独.h文件中，也可以定义在某个类中

1>如果这个协议只用在某个类中，应该把协议定义在该类中

2>如果这个协议用在很多类中，就应该定义在单独文件中

7、分类可以定义在单独.h和.m中，也可以定义在原来类中

1>一般情况下，都是定义在单独文件中

2>定义在原来类中的分类，只要能看懂即可

\*/

#import <Foundation/Foundation.h>

#import "MyProtocol.h"

#import "Person.h"

#import "MyProtocol3.h"

#import "Dog.h"

int main(int argc, const char \* argv[]) {

//NSObject \*obj = [[NSObject alloc] init];

//NSObject \*obj2 = @"jadufi";

//要求obj3保存的对象必须是遵守MyProtocol协议的

//NSObject<MyProtocol> \*obj3 = [[NSObject alloc] init];

//NSObject<MyProtocol> \*obj4 = [[Person alloc] init];

//要去obj5遵守MyProtocol3，并且继承了Person

//Person<MyProtocol3> \*obj5;

return 0;

}

**03 协议的应用-代理模式**

**Person.h**

#import <Foundation/Foundation.h>

//#import "Agent.h"

#import "TicketDelegate.h"

@interface Person : NSObject

//拥有一个代理属性

//@property(nonatomic,strong)Agent \*delegate;

//代理的类名随便，必须遵守TicketDelegate协议

@property(nonatomic,strong) id<TicketDelegate> delegate;

-(void)buyTicket;

@end

**Person.m**

#import "Person.h"

@implementation Person

//买电影票

-(void)buyTicket

{

//叫代理去帮自己买票（询问票价、票的剩余张数等）

double price = [\_delegate ticketPrice];

int number = [\_delegate leftTicketNumber];

NSLog(@"通过代理，票价=%f,还剩%d张票",price,number);

}

@end

**Agent.h**

//负责询问电影票情况的代理

#import <Foundation/Foundation.h>

#import "TicketDelegate.h"

@interface Agent : NSObject<TicketDelegate>

////返回票价

//-(double)ticketPrice;

//

////剩余票数

//-(int)leftTicketNumber;

@end

**Agent.m**

#import "Agent.h"

@implementation Agent

//返回剩余的票数

-(int)leftTicketNumber

{

return 1;

}

//返回‘ 每一张票的价格

-(double)ticketPrice

{

return 35;

}

@end

**TicketDelegate.h**

//声明一些方法

#import <Foundation/Foundation.h>

@protocol TicketDelegate <NSObject>

//返回票价

-(double)ticketPrice;

//剩余票数

-(int)leftTicketNumber;

@end

**NextAgent.h**

#import <Foundation/Foundation.h>

#import "TicketDelegate.h"

@interface NextAgent : NSObject<TicketDelegate>

@end

**NextAgent.m**

#import "NextAgent.h"

@implementation NextAgent

//返回剩余的票数

-(int)leftTicketNumber

{

return 2;

}

//返回‘ 每一张票的价格

-(double)ticketPrice

{

return 30;

}

@end

**Main.m**

#import <Foundation/Foundation.h>

#import "Person.h"

#import "Agent.h"

#import "NextAgent.h"

int main(int argc, const char \* argv[]) {

//人

Person \*p = [[Person alloc] init];

//代理

Agent \*a = [[Agent alloc] init];

//设置人的代理

p.delegate = a;

//人计划看电影

[p buyTicket];

Person \*p1 = [[Person alloc] init];

NextAgent \*a1 = [[NextAgent alloc] init];

p1.delegate = a1;

[p1 buyTicket];

return 0;

}

**04 Foundation-结构体**

**main.m**

#import <Foundation/Foundation.h>

int main(int argc, const char \* argv[]) {

/\*

NSRange(location length)

NSPoint\CGPoint

NSSize\CGSize

NSRect\CGRect

\*/

NSRange r1 = NSMakeRange(2, 4);

NSString \*str = @"i love oc";

//查找某个字符串在str中的范围

NSRange range = [str rangeOfString:@"love"];

//如果找不到，length=0，location=NSNotFound

//NSLog(@"%d",NSNotFound);

NSRange range1 = [str rangeOfString:@"java"];

NSLog(@"loc = %ld, length = %ld",range1.location,range1.length);

CGPoint p1 = NSMakePoint(10, 10);

NSString \*str1 = NSStringFromPoint(p1);

NSLog(@"str1 = %@",str1);

NSPoint p2 = CGPointMake(4, 6); //最常用

NSSize s1 = CGSizeMake(34, 56);

NSSize s2 = NSMakeSize(100, 50);

CGSize s3 = NSMakeSize(180, 60);

CGRect rect1 = CGRectMake(0, 0, 100, 50);

CGRect r2 = {{0,0},{100,50}};

CGRect r3 = {p1,s2};

CGRect r4 = {CGPointZero,CGSizeMake(100, 90)};

//表示原点

//CGPointZero == CGPointMake(0, 0);

//CGSizeZero;

//CGRectZero;

//NSLog(@"x=%f,y=%f,width=%f,height=%f",rect1.origin.x,rect1.origin.y,rect1.size.width,rect1.size.height);

NSString \*str2 = NSStringFromRect(rect1);

NSLog(@"rect is %@",str2);

//比较两个点是否相同,还有比较尺寸，比较rect的

CGPointEqualToPoint(CGPointMake(10, 10), CGPointMake(10, 10));

//比较是否包含点

BOOL b = CGRectContainsPoint(rect1, p1);

return 0;

}

CGRect myRect(CGFloat x, CGFloat y, CGFloat width, CGFloat height)

{

CGRect rect;

rect.origin.x = x;

rect.origin.y = y;

rect.size.width = width;

rect.size.height = height;

return rect;

}