MICROOH 麦可网

Android-从程序员到架构师之路

出品人：Sundy
讲师：高焕堂（台湾）

http://www.microoh.com

E02_a

HAL框架与Stub开发 (a)

By 高煥堂

内容

1. 复习：C语言
2. 认识HAL的架构
3. Client如何使用HAL框架呢?
4. HAL插件(Stub)的代码范例
5. JNI Native Client的代码范例
6. 观摩Android的实际HAL-Stub范例

1、复习：C语言

前言

Struct → Class

1970年代，

C语言就有struct结构

1980年代，

C++的class结构就是从C的struct扩充而来

1990年代，

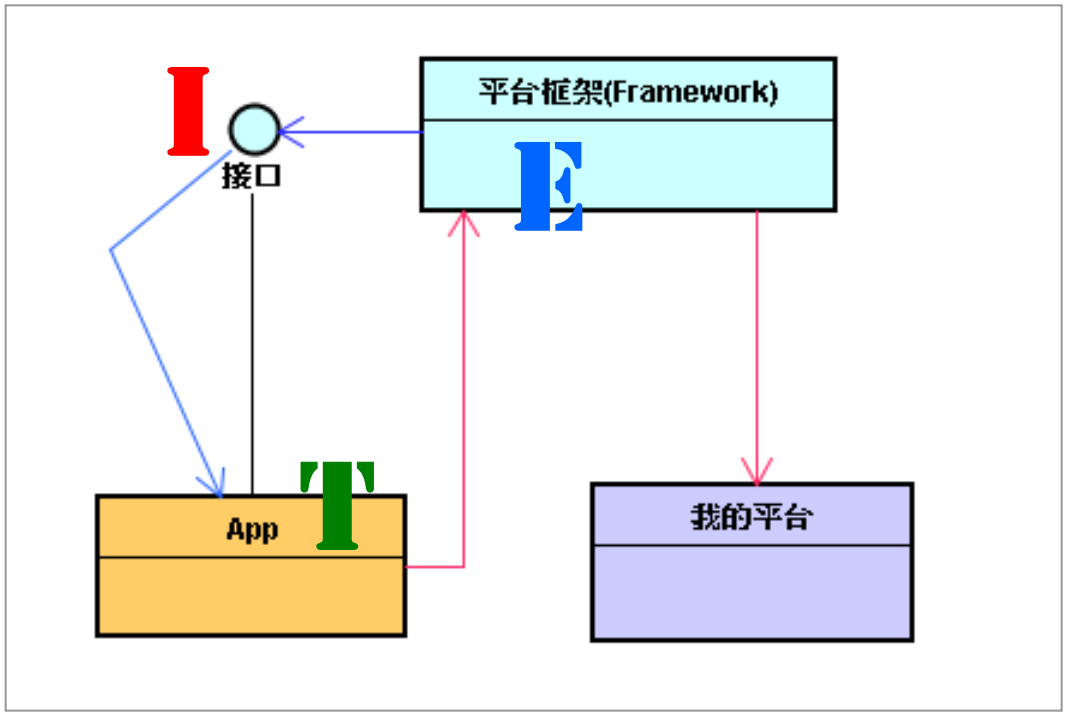
Java的class是基于C++的class而衍生出来

EIT造形的基础：

<基类/子类>结构就是两个class的组合

EIT造形的接口：

也是一种纯粹抽象类(pure abstract class)



1.1 复习：C语言的结构(struct)

- 定义结构型态。商品有牌子，定义型态就像描述一件商品的品牌。例如：

生日礼盒 Smile

```
{
巧克力；
情人糖；
知心软糖；
}；
```

- 再如：

```
struct smile
        {
          char sna;
          char size;
          float price;
        };
```

- 这说明了：smile 结构内含3 项数据——两项字符数据，另一项浮点数数据。

- 接着，根据所定义之结构来宣告结构变量。宣告结构变量就像「订」礼盒。例如：

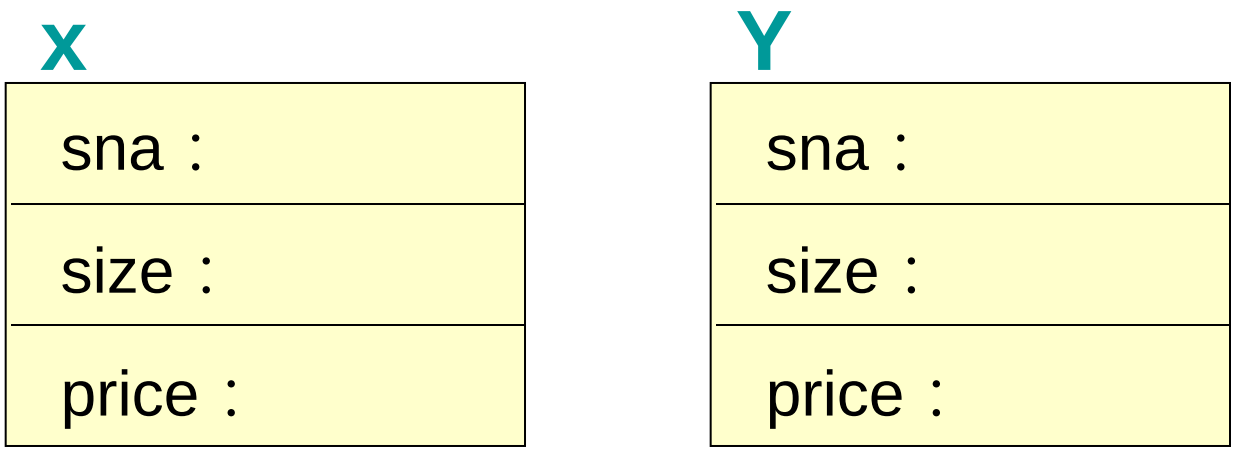
生日礼盒 Smile x, y;

- 这订购两个Smile 生日礼盒，一个给x，另一个给y。

- 再来，请看如何向计算机「订」结构变量(Structure Variable)？

```
struct  smile   x, y;
```

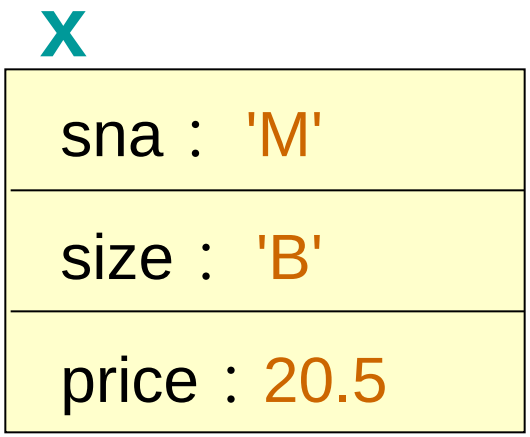
- 共宣告了2个结构变量：x 和y 为smile 结构之变量。亦即，x 及y 变量的型态是：struct smile。


X
sna :
size :
price :
Y
sna :
size :
price :

```
/* cx-01.c */
#include <stdio.h>
struct smile
      {
        char sna;
        char size;
        float price;
      };

int main(void)
     {
      struct smile x;
      x.sna = 'M';
      x.size = 'B';
      x.price = 20.5;
      printf( "%c, %c, %.1f", x.sna, x.size, x.price );
      return 0;
     }
```

- 先定义结构型态——struct smile。
- 说明了：struct smile型态包含char型态及float 型态的数据。进入main()函数，就诞生了自动变量x。
- 此时x 变数内含sna 、size及price 三个项目。程序里以以x.sna、x.size及x.price 表示之。


x
sna : 'M'
size : 'B'
price : 20.5

1.2 复习：结构指针(Pointer)

- 宣告结构指针，来指向结构变量。例如：

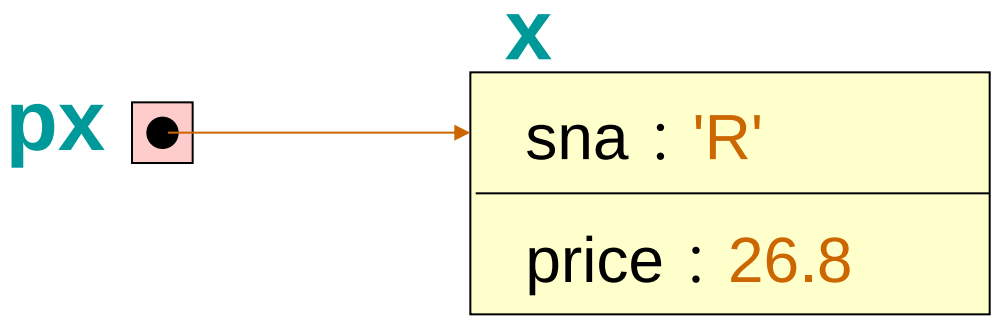
```c
/* cx-02.c */
#include <stdio.h>
#include <string.h>
struct smile {
        char sna;
        float price;
  };

int main(void)
      {
       struct smile x;
       struct smile *px;
       px = &x;
       px->sna = 'R';
       px->price = 26.8;
       printf( "Sna=[%c], Price=%.1f", x.sna, x.price );
       return 0;
      }
```

- px是struct smile型态的指针，x 是struct smile型态的变量，px可以指向x 变量。
- “&” 运算能把x 变量的地址存入px中，使得px指向x 变量。
- 指令：

```
px->sna = 'R';
px->price = 26.8;
```

- 把数据存入结构(变量)里。


x
px
sna : 'R'
price : 26.8

1.3 复习：动态内存分配

- 「动态」(Dynamic) 的意思是：待程序执行时(Run-Time)才告诉计算机共需要多少内存空间，计算机依照需要立即分配空间，裨储存数据。

- malloc()和free()是最常用的动态内存分配函数。如果在执行时需要空间来储存数据，宜使用malloc()函数。用完了就用free()释放该空间。malloc()之格式为：

  指针 = malloc(空间大小)

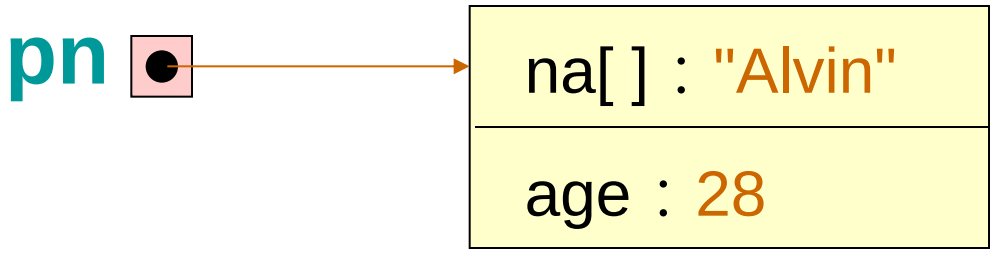
- 例如： ptr = malloc(100);

```c
/* cx03.c */
#include <stdio.h>
#include <malloc.h>
#include <string.h>
#include <stdlib.h>

struct kiki {
        char na[10];
        short int age;
      };
typedef struct kiki NODE;
```

```
int main(void) {
        NODE *pn;
        pn = (NODE *) malloc (sizeof(NODE));
        if( pn==NULL )
            {  printf("malloc() failed\n");
               exit(0);
            }
        strcpy( pn->na,"Alvin");
        pn->age = 28;
        printf("AGE=%d", pn->age);
        free(pn);
        return 0;
      }
```

- typedef 指令定义的新型态——NODE是 struct kiki 的别名。
- 如果你计算机的sizeof(NODE)值为16，malloc()就索取16 bytes的空间，并令pn指向此区域了。


pn
na[ ] : "Alvin"
age : 28

~ Continued ~