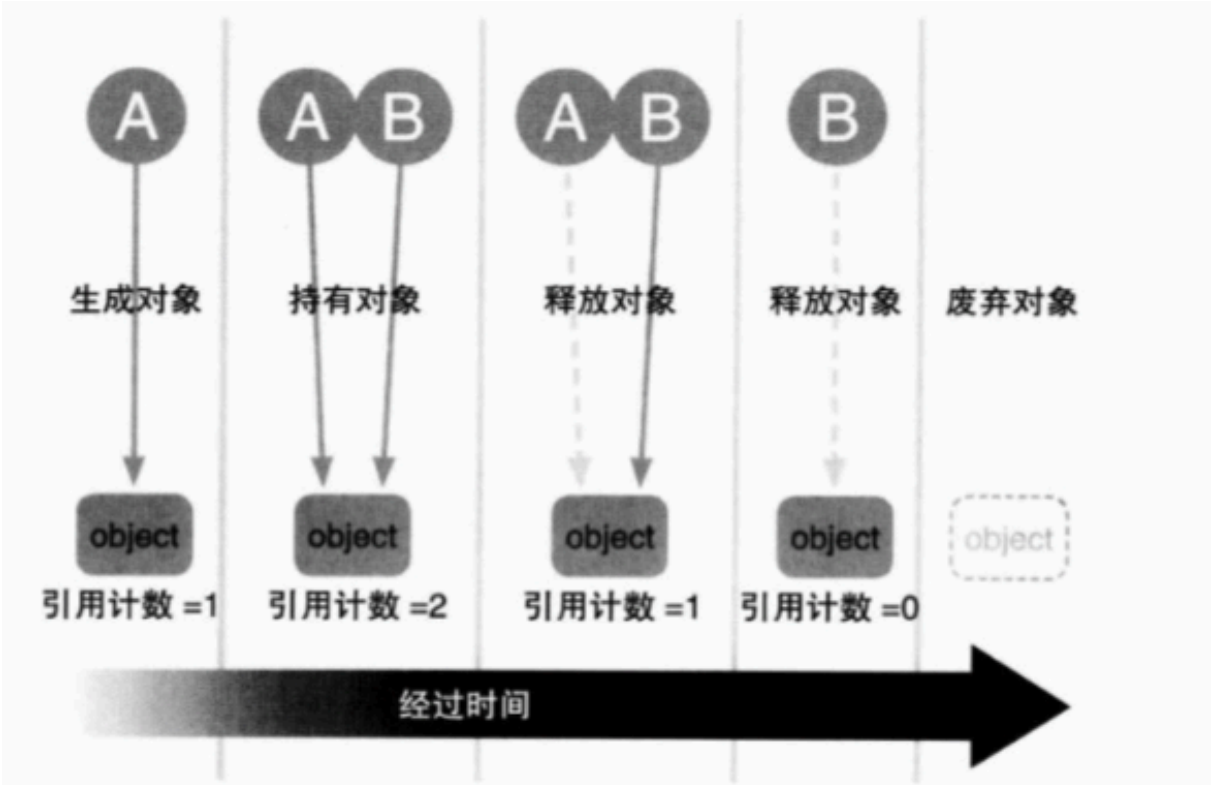
> *在LLVM中设置ARC为有效状态，就无需键入retain和release。*

- 需要满足的条件
  - Xcode4.2或以上版本。
  - LLVM编译器3.0或以上版本。
  - 编译器中设置ARC为有效状态。
- 引用计数图示



- 内存管理的思考方式：
  - 自已生成的对象，自己持有。
  - 非自己生成的对象，自己也能持有。
  - 不需要自己持有对象时释放。
  - 非自己持有的对象无法释放。
- 对象操作与Objective－C方法的对应

对象操作	**Objective－C方法**
生成并自己持有对象	alloc／new/copy/mutableCopy
持有对象	retain

释放对象	release
废弃对象	dealloc

`alloc`为类方法，其他为实例方法。

- 举例：
  - 自已生成的对象，自己持有。

```
//自已生成的对象，自己持有。
id obj = [[NSObject alloc] init];
```

  - 非自己生成的对象，自己也能持有。

```
//生成非自己持有对象
id array = [NSMutableArray array];
//持有对象
[array retain];
```

  - 不需要自己持有对象时释放。

```
//自已生成的对象，自己持有。
id obj = [[NSObject alloc] init];
//释放对象，指向对象的指针仍然保留在obj中，但是不可以访问!
[obj release];
```

  - 通过retain持有的对象也可以通过release释放。
- 通过方法生成对象，并将生成的对象返回调用方。
  - 调用方持有对象

```
- (id)allocObject
{
    id obj = [[NSObject alloc] init];
    return obj;
}
//取得非自己生成确持有的对象。
id obj1 = [obj1 allObjects];
```

  - 生成对象，不持有对象

```
- (id)obejct
{
    id obj = [[NSObject alloc] init];
    //不再持有对象
    [obj autorelease];
}
```

- copy/mutableCopy:copy方法基于NSCopying方法约定，由各类实现的copyWithZone：方法生成并持有对象的`副本`。mutableCopy方法基于NSMutableCopying方法约定，由各类实现的mutableCopyWithZone：方法生成并持有对象的`副本`。两者的区别为:
  `copy方法生成不可变更的对象，mutableCopy生成可变更对象`。