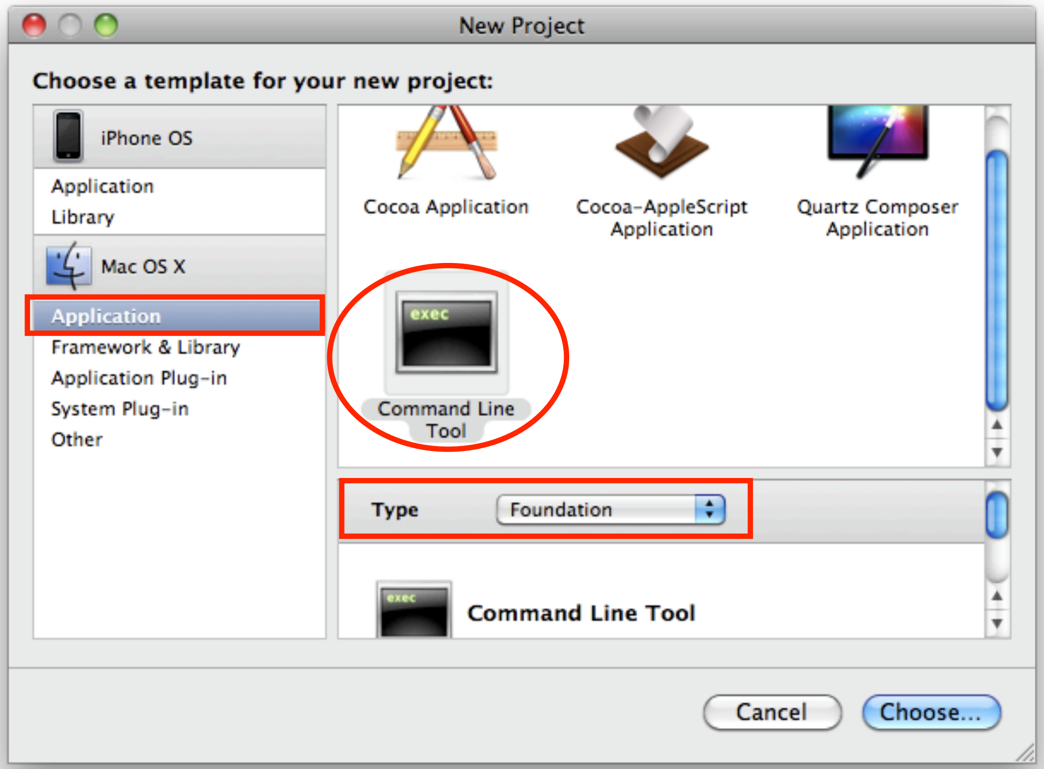
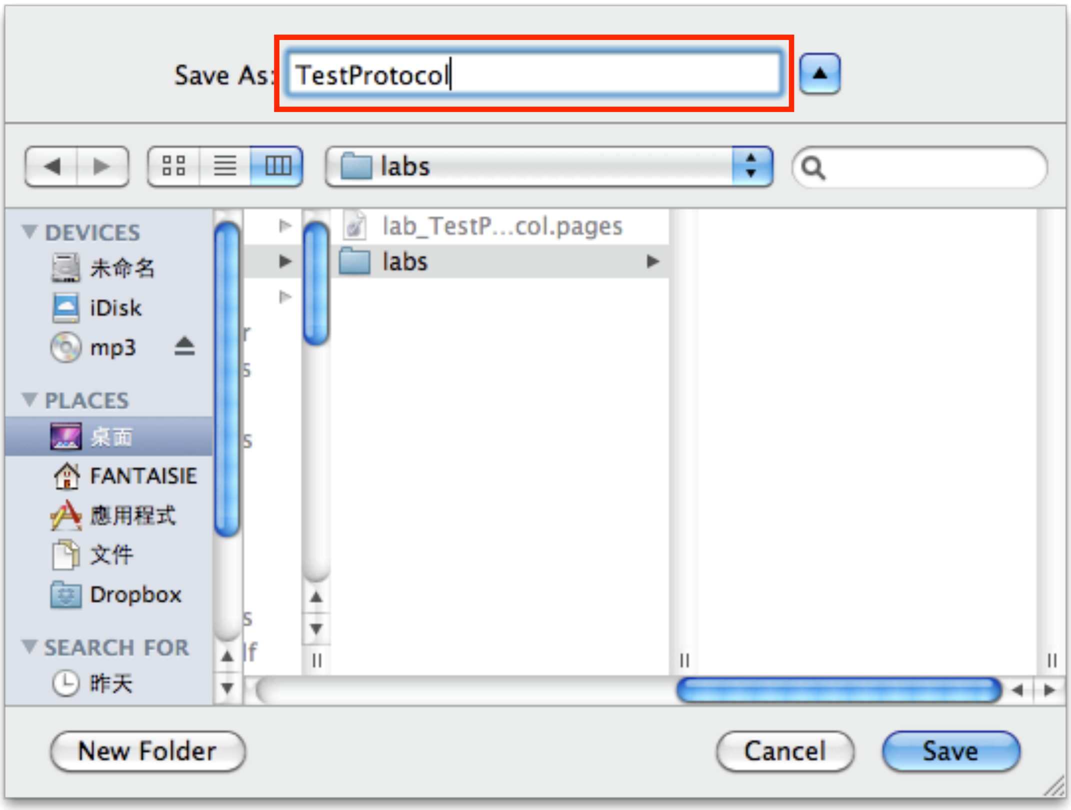
Lab TestProtocol

Step1. 在File開啓一個新的project, 選擇 MAC OS X的Command line Tool, Type選擇 Foundation, 將project命名為 TestProtocol

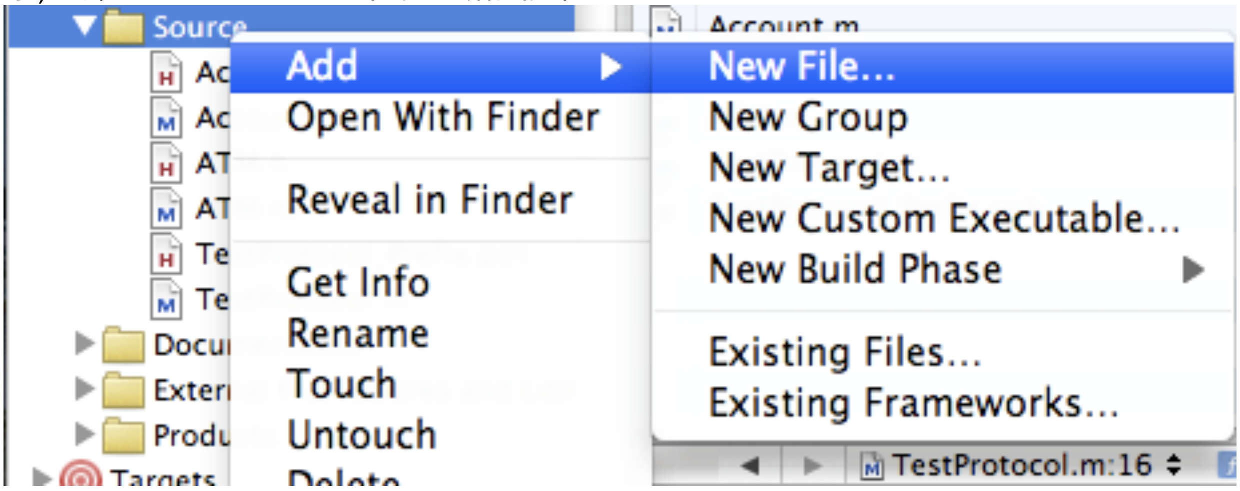


Type Foundation

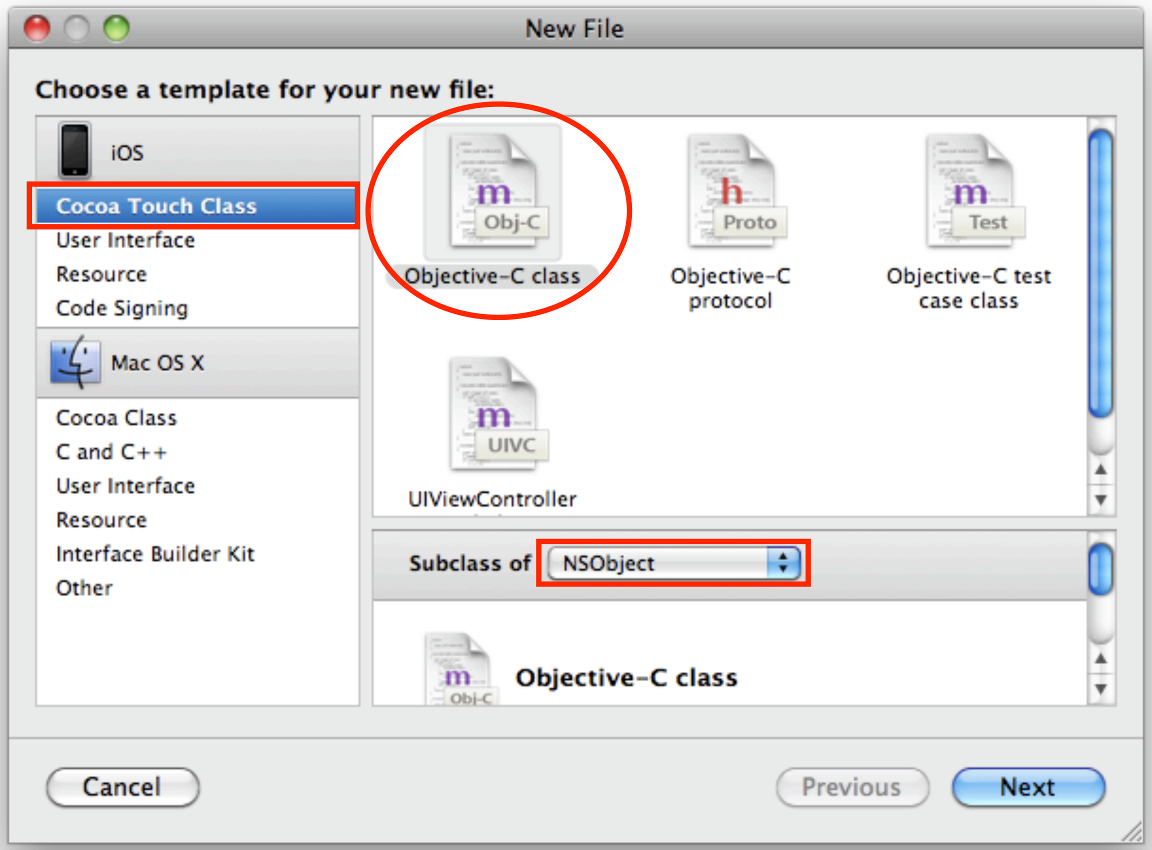


Save As: TestProtocol

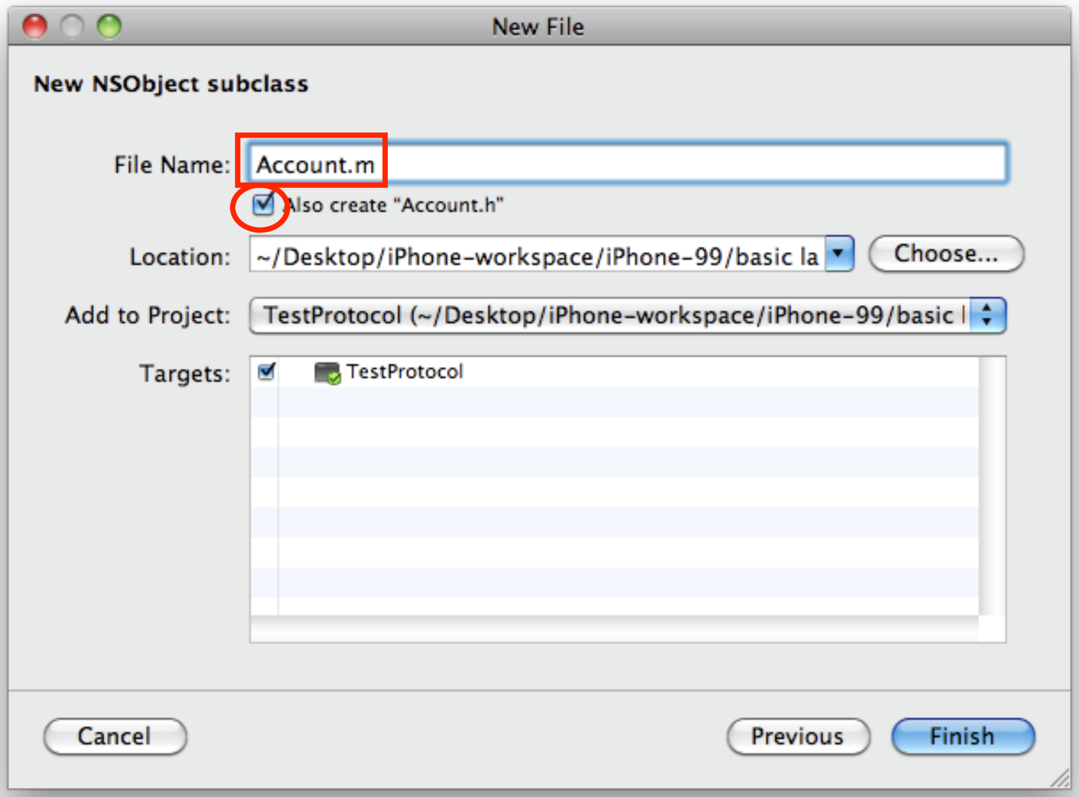
Step2. 在Xcode左邊Groups & Files 視窗中, 在Source這個路徑下點右鍵(若無滑鼠ctrl+點擊)選擇Add > New File...來增加新的檔案



Step3. 選擇iOS裡的Coca Touch Class裡的, 並選擇Subclass of **NSObject**下方有敘述這個Objective-C有includes <Foundation/Foundation.h> 這個標頭檔



Step4. 將新增的subclass的File命名為Account.m, 記得勾選Also create “Account.h”


New NSObject subclass

File Name: Account.m

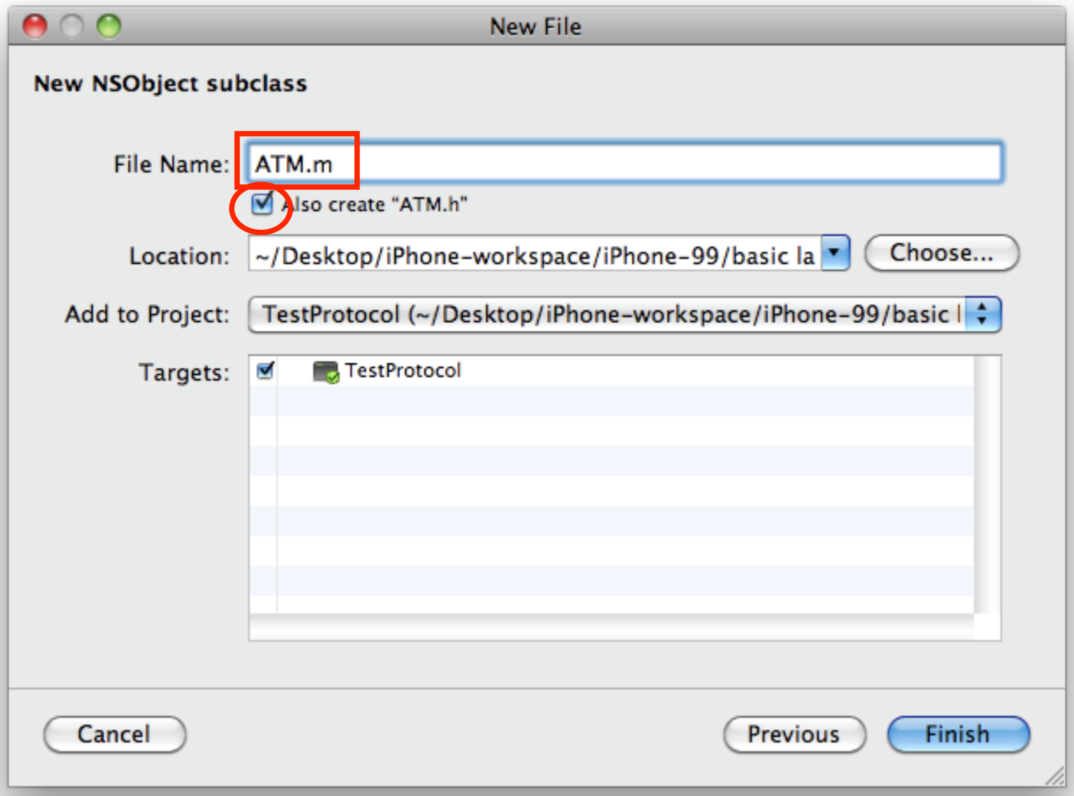
☑ Also create "Account.h"

Location: ~/Desktop/iPhone-workspace/iPhone-99/basic la

Add to Project: TestProtocol (~/Desktop/iPhone-workspace/iPhone-99/basic l

Targets: ☑ TestProtocol

Step5. 如上面新增Account.m和Account.h一樣的步驟來新增ATM.m和ATM.h


New NSObject subclass

File Name: ATM.m

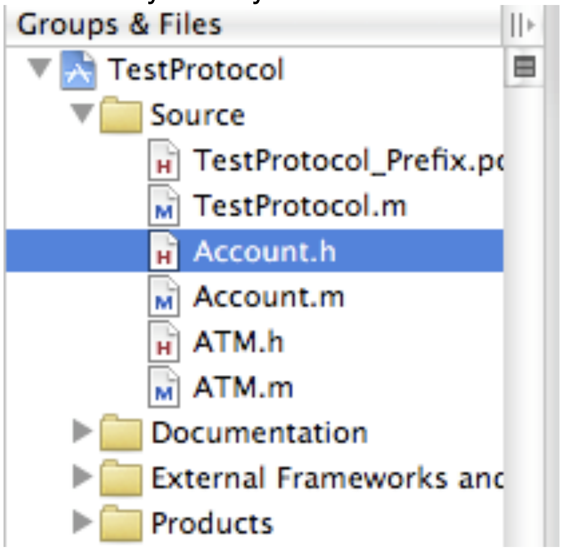
☑ Also create "ATM.h"

Location: ~/Desktop/iPhone-workspace/iPhone-99/basic la

Add to Project: TestProtocol (~/Desktop/iPhone-workspace/iPhone-99/basic l

Targets: ☑ TestProtocol

Step6. 在Xcode左邊Groups & Files 視窗中,開啓Source > Account.h
在裡面宣告withdraw這個protocol和他的method,之後會交由ATM這個NSObject作為Account的delegate去實作. 並將幾個變數的property都設定好, 並加入一個method叫做takeMoney:money:



```
#import <Foundation/Foundation.h>

@protocol withdraw;

@interface Account : NSObject {
    NSString *name;
    int deposit;
    id<withdraw> delegate;
}
-(void) takeMoney:(int) money;
@property (retain) NSString * name;
@property (assign) int deposit;
@property (assign) id<withdraw> delegate;
@end

@protocol withdraw<NSObject>

-(void) withdrawMoney:(Account *) account withName:(NSString *) name
withMoney:(int) money;

@end
```

Step7. 開啓Account.m,將剛剛已設定property的三個變數做synthesize, 並且實作takeMoney:money這個method, 主要就是如果Account的delegate有實作(respondsToSelector) `withdrawMoney:withName:withMoney:` 這個method的話,就把withdrawMoney這件工作交由delegate去完成.

```
#import "Account.h"

@implementation Account
@synthesize name, deposit, delegate;

-(void) takeMoney:(int)money{
    if ([delegate respondsToSelector:@selector
(withdrawMoney:withName:withMoney:)])
           [delegate withdrawMoney:self withName:name withMoney:money];
    else
          NSLog(@"Please implement withdrawMoney:withName:withMoney:");
}

@end
```

Step8. 開啓ATM.h並在裡面 #import "Account.h" 讓 ATM 知道 <withdraw> 和 Account 的存在,在 Class 最後寫 `NSObject<withdraw>` 來繼承 NSObject 而且遵循 <withdraw> 的規範

```
#import <Foundation/Foundation.h>
#import "Account.h"


@interface ATM : NSObject<withdraw> {

}

@end
```

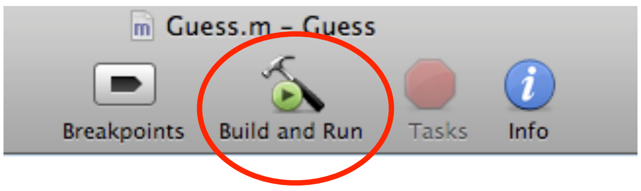
Step9. 開啓ATM.m,並在裡面實作 `withdrawMoney:withName:withMoney:` 這個method, 這個method的主要工作就是將領出的money從deposit扣掉,並將相關的資訊列印出來.

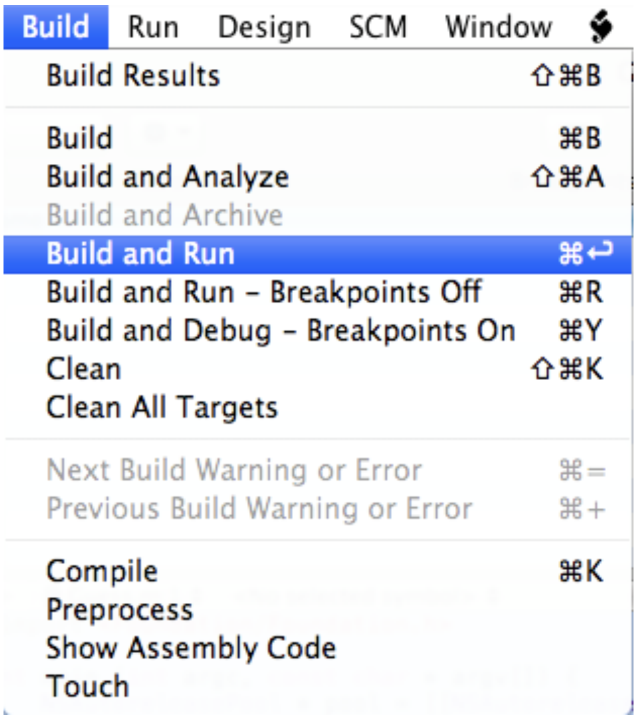
```
#import "ATM.h"

@implementation ATM

-(void) withdrawMoney:(Account *) account withName:(NSString *) name
withMoney:(int) money{
    if(account.deposit >= money){
          account.deposit -= money;
          NSLog(@"***Name: %@, Withdraw: %d, Deposit: %d***",
account.name, money, account.deposit);
    }
    else {
          NSLog(@"***Not enough money!! - Name: %@, Deposit:
%d***",account.name, account.deposit);
    }

}

@end
```

Step10. 開啓TestProtocol.m, 先 `#import "Account.h"` 和 `#import "ATM.h"` ,並將印出Hello, World!這行Mark掉, 並加入以下的程式, 來建立一個account並設定name和deposit, 之後再透過atm作為delegat去扣除領出的錢並將資訊列印出來

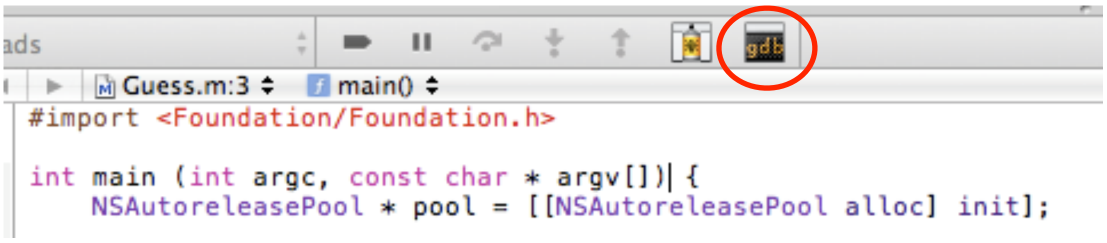
```
#import <Foundation/Foundation.h>
#import "Account.h"
#import "ATM.h"

int main (int argc, const char * argv[]) {
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];

    // insert code here...
    //NSLog(@"Hello, World!");
    Account *account = [Account new];
    account.name = @"Michael";
    account.deposit = 1000;
    ATM *atm = [ATM new];
    account.delegate = atm;
    [account takeMoney:700];
    [account takeMoney:500];
    [pool drain];
    return 0;
}
```

Step11. Build and Run (Command + enter)
在Xcode主頁上按下Build and Run, 或是在Build > Build and Run, 即開始Build code並執行.

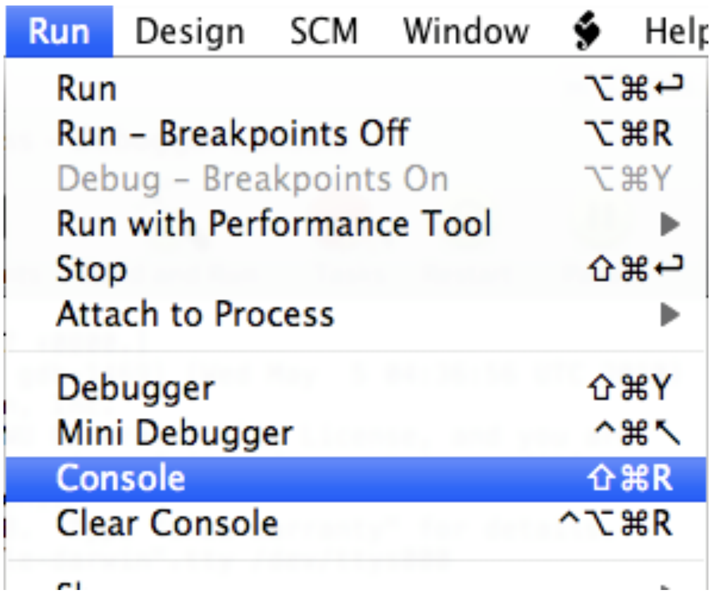




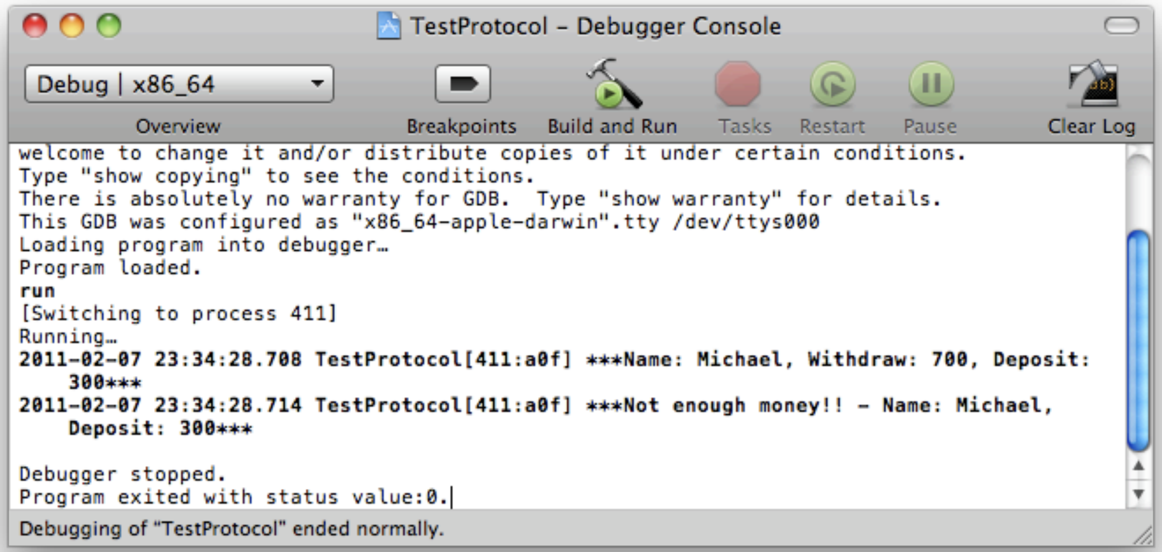
可在瀏覽程式視窗上方的 gdb 按下來開啓console, 或是在Run > Console來開啓.



```
#import <Foundation/Foundation.h>

int main (int argc, const char * argv[]) {
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
```



在Console中顯示第一次withdraw了700, 還剩下300, 而第二次要再領500時, money已經餘額不足了.



```
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "x86_64-apple-darwin".tty /dev/ttys000
Loading program into debugger…
Program loaded.
run
[Switching to process 411]
Running…
2011-02-07 23:34:28.708 TestProtocol[411:a0f] ***Name: Michael, Withdraw: 700, Deposit:
    300***
2011-02-07 23:34:28.714 TestProtocol[411:a0f] ***Not enough money!! - Name: Michael,
    Deposit: 300***

Debugger stopped.
Program exited with status value:0.
```

Debugging of “TestProtocol” ended normally.