آشنایی با برنامه‌نویسی
سیمبین سری ۶۰

S60 Getting Started

Mousa Moradi

نویسنده:

موسی مرادی

مقدمه:

سلام به همه خوانندگان عزیز.

دوباره فرصتی به وجود آمد تا بتوانم مقاله‌ای در مورد آموزش برنامه‌نویسی موبایل بنویسم. وقتی مقاله قبلی را که ترجمه کرده بودم، منتشر کردم، خوشبختانه با استقبال خوبی مواجه شد. دوستان زیادی در این مورد به من ایمیل فرستادند و اظهار لطف کردند. امید است این مقاله نیز بتواند کمکی به شما کرده باشد.

ضمناً اگر شما هنوز مقاله قبلی را ندیده‌اید، می‌توانید آن را از وبلاگ بنده دانلود کنید.

در این مقاله می‌خواهم نحوه نوشتن برنامه Hello World را برایتان آموزش دهم. در هر زبان برنامه‌نویسی که می‌خواهید این برنامه را بنویسید، معمولاً بیش از سه چهار خط طول نمی‌کشد ولی در اینجا بیش از ده فایل ایجاد می‌کنیم تا بتوانیم این برنامه را بنویسیم. البته ساده‌تر از این هم می‌شد گفت، ولی خواستم برنامه را با استانداردهای کامل بنویسم تا شما بهتر یاد بگیرید، به همین دلیل طولانی شد.

برای نوشتن و اجرای این برنامه شما به IDE نیازی ندارید. مثلاً لازم نیست که حتماً C++BuilderX و یا ابزارهای مشابه را نصب کنید، ولی باید نرم‌افزارهای زیر را داشته باشید:

۱) ActivePerl (www.activastate.com)

۲) یک SDK متناسب با موبایلی که می‌خواهید برایش برنامه بنویسید. (www.forum.nokia.com)

۳) Microsoft Debugging Tools (www.microsoft.com/whdc/devtools/debugging)

۴) ویژوال استودیو ۶ یا بالاتر و یا Visual C++ 2005

توجه) نرم‌افزارها را باید به ترتیب گفته شده نصب کنید.

توجه) نرم‌افزارهای بالا را علاوه بر اینکه می‌توانید از سایت‌های یاد شده دانلود کنید، نسخه‌ای از آنها را در وبلاگم قرار داده‌ام؛ می‌توانید از وبلاگ من هم استفاده کنید.

سعی کنید، کدهایی را که در اینجا نوشته‌ام و از شما می‌خواهم آنها را بنویسید، خودتان بنویسید و کپی نکنید، تا دستتان با آنها آشناتر شود. چون همیشه با کلاس‌هایی که در اینجا ایجاد می‌کنیم، سروکار خواهید داشت.

و نکته‌ای که باید متذکر شوم، این است که برای استفاده کردن از این مقاله باید زبان C++ را به خوبی بلد باشید وگرنه مطالب گفته شده را خوب یاد نمی‌گیرید.

ضمناً به همراه این فایل، پروژه کاملی که در اینجا می‌خواهیم بنویسیم را نیز قرار داده‌ام تا اگر با مشکلی برخوردید، بتوانید از آن استفاده کنید. (اگر شما از جایی فقط خود فایل pdf را دانلود کرده‌اید و فایل zip پروژه همراه آن نیست، می‌توانید از وبلاگ من دوباره دانلود کنید.)

سعی کرده‌ام مقدماتی را که برای برنامه‌نویسی سیمبین لازمند به طور اختصار و به قدر نیازمان در نوشتن این پروژه، برایتان توضیح دهم. برای اطلاعات کاملتر می‌توانید به مستندات SDK مراجعه کنید. اطلاعات کاملی در آن وجود دارد. سایت http://developer.symbian.com هم حاوی اطلاعات به مراتب کاملتری است.

با تشکر

سی‌ام دیماه ۱۳۸۵

موسی مرادی کندلجی

mousamk@gmail.com

http://series60.blogfa.com

http://symbiandeveloper.blogfa.com

فهرست محتویات

مقدماتی در مورد برنامه‌نویسی سیمبین سری ۶۰:

برای ساخت یک برنامه سیمبین باید فایل‌های مشخصی را داشته باشید که در مقاله «فایل‌های سیمبین» در مورد آنها توضیحاتی را داده‌ام. در این مقاله می‌خواهیم بدون استفاده از یک IDE خودمان یک پروژه ایجاد کنیم. تنها ابزار ما برای نوشتن پروژه، فقط یک ویرایشگر متن ساده مانند Notepad است. می‌توانید از هر ویرایشگر دیگری که متن ساده (Plain Text) تولید می‌کند نیز استفاده کنید. مثلاً من خودم از برنامه Notepad++ استفاده می‌کنم. قبل از این که شروع به کار کنیم، باید با مقدماتی آشنا شویم.

انواع داده‌ها در سیمبین:

در سیمبین نوع‌های int و char و double و ... را نداریم ولی در عوض از اینها استفاده می‌کنیم:

TInt: معادل همان int است که یک متغیر چهار بایتی است.

TBool: معادل bool در C++ است و می‌تواند مقادیر ETrue و EFalse داشته باشد.

TReal: معادل double در C++ است و یک متغیر ۸ بایتی است.

TUid: نوع داده‌ای است که می‌تواند یک عدد Uid ذخیره کند. Uid یک عدد ۸ رقمی در مبنای ۱۶ است و ۴ بایت جا می‌گیرد.

TRect: نوع داده‌ای است که یک مستطیل تعریف می‌کند و کاربردهای زیادی هم دارد. شبیه کلاس Rect در ویژوال سی است.

کلاس‌های اصلی برنامه:

هر برنامه سیمبین دارای چهار کلاس اصلی است که باید در همه برنامه‌های GUI سیمبین وجود داشته باشند. این کلاس‌ها را به اختصار در زیر توضیح می‌دهیم:

کلاس Application: کلاس اصلی برنامه است و هنگام اجرا شدن برنامه، آبجکتی از این کلاس ایجاد می‌شود و آدرسش به سیستم عامل برگردانده می‌شود. این کلاس ویژگی‌های اصلی برنامه را مشخص می‌کند. ساخت یک شیء از کلاس Document هم برعهده این کلاس است. کلاس پایه این کلاس، CAknApplication است که در فایل aknapp.h تعریف شده است.

کلاس Document: هر برنامه باید آبجکتی از این کلاس داشته باشد. این کلاس برای ساخت آبجکتی از کلاس AppUi به کار می‌رود. کلاس پایه‌اش CAknDocument است. این کلاس هم در فایل akndoc.h تعریف شده است.

کلاس AppUi: این کلاس برای مدیریت رویدادها (مثل رویداد زده شدن کلیدها توسط کاربر)، بستن فایل‌ها و ... استفاده می‌شود. خودش هیچ خروجی در صفحه ندارد. در عوض رابط صفحه با Viewهای خودش است. (می‌تواند تعداد زیادی View داشته باشد) کلاس پایه‌اش CAknAppUi و یا CAknViewAppUi است.

کلاس View: که یک کنترل است که اطلاعاتی را در صفحه نشان می‌دهد و کاربر می‌تواند با آن تعامل داشته باشد. معمولاً در SDK سری ۶۰ واژه Container به جای View به کار می‌رود؛ گرچه با هم معادلند و هیچ فرقی ندارند. از CCoeControl یا CAknDialog مشتق می‌شوند. و یا اگر برنامه با معماری Application/View طراحی شده باشد از CAknView مشتق می‌شوند.

سازنده دو فازی:

وقتی برای موبایل برنامه می‌نویسیم، همیشه امکان کمبود حافظه وجود دارد. بنابراین باید همیشه حافظه را به خوبی مدیریت کنیم. فرض کنیم تابعی داریم که یک آبجکت از یک کلاس ایجاد می‌کند و اشاره‌گر به آن را در یک متغیر اتوماتیک داخل همان تابع ذخیره می‌کند. اگر اجرای برخی از دستورات متد در حالی که کار متد تمام نشده است، ناتمام رها شود، متد شما هم متوقف خواهد شد و متغیرهای اتوماتیک متد، از بین خواهند رفت. در این حالت، آبجکتی که ایجاد کرده‌ایم، هنوز باقی است ولی اشاره‌گر به آن را از دست داده‌ایم و نمی‌توانیم آن را delete کنیم در نتیجه فضای محدود حافظه، بی‌دلیل اشغال می‌شود. برای جلوگیری از این کار ما از یک سیستم سازنده دو فازی یا دو مرحله‌ای استفاده می‌کنیم. به این صورت که از Constructor خود کلاس استفاده نمی‌کنیم و آن را خالی باقی می‌گذاریم. در عوض سه تابع دیگر به صورت زیر می‌سازیم:

۱) متد NewLC: این تابع آرگومان‌های مورد نیاز را می‌گیرد و یک آبجکت جدید new می‌کند. سپس این آبجکت را در Stackی به نام CleanupStack، Push می‌کند و تابع ConstructL را Call می‌کند تا متغیرهای آبجکت ساخته شوند. نتیجه را به تابع NewL که آن را Call کرده است، برمی‌گرداند.

۲) متد NewL: این تابع ابتدا NewLC را Call می‌کند و مقدار بازگشتی آن را از Cleanup Stack، Pop می‌کند و آن را برمی‌گرداند.

۳) تابع ConstructL: این تابع خروجی ندارد و کارش فقط ساختن متغیرهای عضو کلاس است.

هنگامی که از این حالت استفاده می‌کنیم، اگر اشاره‌گر به آبجکت را از دست دهیم، سیمبین، آبجکت ایجاد شده را که در Cleanup Stack هست، خودش پاک می‌کند.

مثال: فرض کنیم کلاسی به نام CMyClass داریم که مثلاً یک متغیر با نام iNum از نوع TInt* دارد و این سه تابع برایش تعریف شده‌اند. پیاده‌سازی این توابع را در زیر می‌بینیم.

```
CMyClass* CMyClass::NewL()
    {
    CMyClass* self = CMyClass::NewLC();
    CleanupStack::Pop(self);
    return self;
    }

CMyClass* CMyClass::NewLC()
    {
    CMyClass* self = new (ELeave) CMyClass;
    CleanupStack::PushL(self);
    self->ConstructL();
    return self;
    }

void CMyClass::ConstructL()
    {
    iNum = new (ELeave) TInt;
    }
```

برای ساخت آبجکتی از این کلاس، به صورت زیر عمل می‌کنیم:

```
CMyClass* newObject;
newObject = CMyClass->NewL();
```

Resourceها:

Resourceها ساختارهایی(Structure) هستند که به کمک آنها منابع برنامه (مثل منوها، متن‌ها و ...) را تعریف می‌کنیم. ساختار همه Resourceها به صورت زیر است:

```
RESOURCE struct-name [resource-name]
    {
    resource-initializer-list
    }
```

struct-name یک struct است که قبلاً در یک دستور STRUCT تعریف شده است. تعاریف structها در فایل‌های rh. انجام می‌شوند. مثلاً تعاریف Avkon در فایل epoc32\include\avkon.rh\ هستند.

avkon کتابخانه‌ای است که خیلی از ملزومات سری ۶۰ در آن تعریف شده است و می‌توانیم به راحتی از آن استفاده کنیم. ولی فقط مخصوص سری ۶۰ هستند. Uikon هم کتابخانه‌ای است که برای همه رابط‌های سیمبین استفاده می‌شود.

اجزای Avkon UI:

چند تا از کنترل‌های پرکاربرد Avkon:

- Dialog
- Query
- Form
- Popup Notes
- Listbox
- Grid
- منوی Options

(در این مقاله فقط منوی Options را توضیح می‌دهیم. انشاءالله لیست کامل آنها را در مقاله جداگانه‌ای شرح خواهم داد.)

منوی Options:

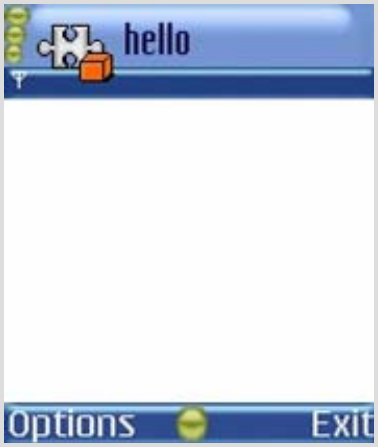
در برنامه در حال اجرا در موبایل، کاربر با زدن دکمه سمت چپ، منوی Options را اجرا می‌کند. منوهای Options در فایل‌های resource ذخیره می‌شوند.

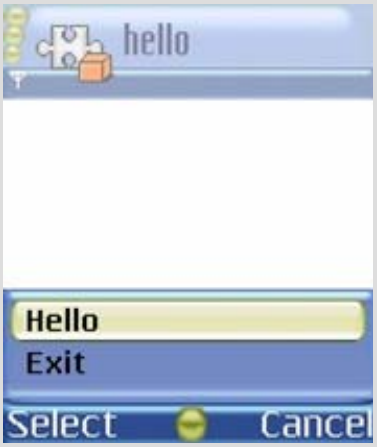
مثالی از نحوه ساخت یک منوی Options برای یک برنامه:

```
RESOURCE AIK_AIP_INFO
    {
    menubar = r_helloworld_menubar;
    cba = R_AVKON_SOFTKEYS_OPTIONS_BACK;
    }

RESOURCE MENU_BAR r_helloworld_menubar
    {
    titles =
        {
        MENU_TITLE
            {
            menu_pane = r_helloworld_menu;
            }
        };
    }

RESOURCE MENU_PANE r_helloworld_menu
    {
    items =
        {
        MENU_ITEM
            {
            command = EHelloWorldCommand1;
            txt = "Hello";
            },
        MENU_ITEM
            {
            command = EAknSoftkeyExit;
            txt = "Exit";
            }
        };
    }
```

سه تا ساختار در بالا تعریف کردیم. ساختار اولی به سیستم عامل می‌گوید که برای برنامه ما دو تا منوی Exit و Options در نظر بگیرد. حالت‌های دیگری هم موجود است که می‌توان از آنها استفاده کرد ولی برای پنجره اصلی برنامه معمولاً همین حالت Options-Exit را در نظر می‌گیرند. در بخش MENU_BAR به سیستم می‌گوییم که برای خود منوی Options، زیرمنوی r_helloworld_menu را در نظر بگیرد. در قسمت بعد هم، همین زیر منو را تعریف می‌کنیم. در اینجا ما دو گزینه Hello و Exit ایجاد کردیم و برای هر منو هم یک Command تعریف کردیم. سیمبین از Command برای شناسایی منو استفاده می‌کند. مثلاً اگر کاربر منوی Exit از منوی Options را انتخاب کرد، متد HandleCommandL از AppUi را صدا می‌زند (Call) و EAknSoftkeyExit را که همان Command منوی Exit است، به عنوان آرگومان به آن می‌فرستد. بعداً در کلاس AppUi این تابع را مدیریت خواهیم کرد.

منویی که ما ساختیم به صورت زیر درمی‌آید:





توجه:

a) «EAknSoftkeyExit» از کلمات تعریف شده است و می‌توانیم از آن استفاده کنیم. ولی EHelloWorldCommand1 تعریف نشده است و باید خودمان آن را در یک فایل hrh. تعریف کنیم. علاوه بر این اگر منوهای دیگری وجود داشت و Commandهای دیگری در نظر گرفتیم، باید آنها را هم تعریف کنیم.

b) این یک استاندارد است که زدن "Back" یا "Exit" باعث می‌شود تا از برنامه خارج شویم. همچنین این نیز یک استاندارد است که برنامه در پاسخ به EEikCmdExit یا EAknSoftkeyExit اجرای برنامه را تمام کند و از برنامه خارج شود.

Control Stack:

سیمبین Stackی به نام Control Stack دارد. هر کنترلی که می‌خواهد رویداد فشرده شدن کلید را بگیرد باید در این Stack ثبت شود. ولی معمولاً فقط Viewها خودشان را به Control Stack اضافه می‌کنند. بعداً سیمبین، رویداد را به View می‌فرستد و View هم توسط متد OfferKeyEventL عکس‌العمل مناسب را انجام می‌دهد.

مثالی از نحوه اضافه کردن یک dialog (dialog هم نوعی View است.) به Control Stack:

```
void CTestAppUi::ConstructL()
    {
    BaseConstructL();
    iAppDialog = new (ELeave) CTestADialog;
    iAppDialog->setMopParent(this);
    iAppDialog->ExecuteLD(R_TESTA_DIALOG);
    AddToStackL(iAppDialog);
    }
```

در اینجا تابع ConstructL() از AppUi، dialog را می‌سازد و آن را به Control Stack اضافه می‌کند. در نتیجه dialog می‌تواند رویدادهای کلید را بگیرد.

رویدادهای کلید به کنترل‌های موجود در Control Stack توسط تابع OfferKeyEventL() پیشنهاد می‌شوند. (اول به آخرین کنترل اضافه شده پیشنهاد می‌شود.) اگر کنترل مورد نظر رویداد را مدیریت (Handle) نکند و EKeyWasNotConsumed برگرداند، سیستم عامل کلید را به کنترل بعدی Control Stack می‌دهد و الی آخر.

نوشتن برنامه Hello Symbian:

ایجاد فایل bld.inf:

ابتدا به مسیری که SDK را نصب کرده‌اید مثلاً (C:\Symbian) بروید در آنجا یک پوشه با نام HelloSymbian ایجاد کنید. در داخل این پوشه، چهار پوشه با نام‌های group، inc، src و sis ایجاد کنید. group برای نگه داشتن فایل‌های اصلی پروژه است. inc برای فایل‌های h. و src برای فایل‌های cpp. است. در پوشه sis هم فایل sis را خواهیم ساخت. توجه داشته باشید که می‌توانید هر نام دلخواه دیگری هم به کار ببرید؛ ولی معمولاً در برنامه‌نویسی سیمبین از این اسامی استفاده می‌کنند. همانطور که در مقاله «فایل‌های سیمبین» هم گفتم، فایل اصلی پروژه، فایل **bld.inf** است. این فایل، mmpهای استفاده شده در برنامه را مشخص می‌کند که در برنامه‌های ساده که فقط از یک فایل mmp استفاده می‌کنند، این فایل هم به همان یک فایل mmp اشاره می‌کند.

ویرایشگر متن را باز کنید و دستورات زیر را در آن بنویسید:

```
PRJ_MMPFILES

hellosymbian.mmp
```

فایل را با نام bld.inf در پوشه group ذخیره کنید. دقت کنید که حتماً تمام فایل‌هایی که در اینجا می‌نویسیم را به صورت ANSI ذخیره کنید، فقط برای فایل‌های خاصی می‌توانید از یونیکد استفاده کنید، که آنها را بعداً خواهم گفت.

ایجاد فایل hellosymbian.mmp:

کارمان با فایل bld.inf تمام شد. یک سند جدید باز کنید و عبارت زیر را در آن بنویسید و سپس آن را با نام hellosymbian.mmp در پوشه group ذخیره کنید:

```
TARGET                  hellosymbian.app
TARGETTYPE              app
UID                     0x100039CE 0x10006F90
TARGETPATH              \system\apps\HelloSymbian

SOURCEPATH              ..\src
SOURCE                  hellosymbian.cpp
SOURCE                  hellosymbianApplication.cpp
SOURCE                  hellosymbianDocument.cpp
SOURCE                  hellosymbianAppUi.cpp
SOURCE                  hellosymbianAppView.cpp

SOURCEPATH              ..\group
RESOURCE                hellosymbian.rss

USERINCLUDE             ..\inc

SYSTEMINCLUDE           \epoc32\include

LIBRARY                 euser.lib
LIBRARY                 apparc.lib
LIBRARY                 cone.lib
LIBRARY                 eikcore.lib
LIBRARY                 avkon.lib
```

فایل mmp فایل تعریف پروژه است که تمام اجزای پروژه‌مان را مشخص می‌کند. سطرهای بالا را با هم بررسی می‌کنیم:

TARGET: هدف نهایی پروژه را مشخص می‌کند. در اینجا هدف ما ساخت فایل hellosymbian.app است. حتماً می‌دانید که فایل‌های اجرایی GUI سیمبین، فایل‌های app هستند.

TARGETTYPE: نوع برنامه را مشخص می‌کند. در اینجا ما می‌خواهیم یک برنامه GUI اجرایی بنویسیم؛ پس از app استفاده

می‌کنیم. اگر هدف برنامه‌مان ساخت یک dll بود، در اینجا dll می‌نوشتیم.

UID: uid برنامه را مشخص می‌کند. در اینجا باید دو تا عدد در مبنای ۱۶ وارد کنیم. عدد اول معمولاً در همه برنامه‌ها 0x100039CE می‌باشد ولی عدد دوم باید منحصر بفرد باشد و قبلاً نباید در هیچ برنامه‌ای استفاده شده باشد. می‌توانید این عددها را از سایت www.symbiansigned.com بگیرید. ولی اعداد بین 0x10000000 و 0x1FFFFFFF برای ساخت اولیه برنامه‌ها رزرو شده‌اند. می‌توانید یک عدد دلخواه از بین اینها انتخاب کنید و برنامه‌تان را برای خودتان بسازید ولی اگر قصد دارید که برنامه را برای عموم منتشر کنید، باید از عددهای بالای 0x20000000 استفاده کنید و آنها را باید از سایت یاد شده دریافت کنید. توجه کنید که هرگز نباید از خودتان عدد تصادفی وارد کنید، چون در این صورت هم برای برنامه خودتان و هم برای برنامه‌ای که شماره‌اش تصادفاً همان عدد است، مشکلاتی ایجاد می‌شود.

TARGETPATH: مسیر پروژه را در موبایل مشخص می‌کند. همه برنامه‌های GUI سیمبین در مسیر \system\apps\appname ذخیره می‌شوند.

SOURCEPATH: مسیر سورس‌فایل‌های برنامه را مشخص می‌کند.

SOURCE: خود سورس‌فایل‌ها را مشخص می‌کند.

RESOURCE: فایل‌های Resource استفاده شده در پروژه را نشان می‌دهد.

USERINCLUDE: مسیر فایل‌هایی را که خودتان include خواهید کرد را مشخص می‌کند.

SYSTEMINCLUDE: مسیر فایل‌هایی را که در SDK وجود دارند و شما include خواهید کرد را مشخص می‌کند.

توجه) اگر در کدهایتان فایلی را به صورت "file.h" ضمیمه کنید، کامپایلر در مسیر USERINCLUDE دنبال آن می‌گردد و اگر به صورت <file.h> ضمیمه کنید، در مسیر SYSTEMINCLUDE.

LIBRARY: کتابخانه‌هایی را که در برنامه‌تان از آنها استفاده کرده‌اید را مشخص می‌کند.

ایجاد فایل hellosymbian.hrh:

یک فایل جدید با نام hellosymbian.hrh ایجاد کنید و آن را در پوشه inc ذخیره کنید و داخلش کدهای زیر را بنویسید:

```
#ifndef _HELLO_HRH_
#define _HELLO_HRH_

enum THelloIds
	{
	EHelloSymbianCommand1 = 1
	};

#endif
```

همانطور که می‌بینید در اینجا یک enum تعریف کرده‌ایم که فقط حاوی یک عضو است. این همان Commandی است که به منوی Hello نسبت خواهیم داد. اگر منوهای بیشتری خواستیم درست کنیم، باید در همین‌جا برایشان Commandهای جدیدی تعریف کنیم.

ایجاد فایل Resource برنامه:

فایل‌های resource را بخش اول توضیح دادیم. این فایل‌ها قبل از کامپایل کردن به صورت متنی هستند و با پسوند rss. ذخیره می‌شوند. بعد از کامپایل کردن برنامه، به صورت باینری و با پسوند rsc. ذخیره می‌شوند.

یک سند جدید در ویرایشگر باز کنید و دستورات زیر را در آن بنویسید و با نام hellosymbian.rss در پوشه group ذخیره کنید:

```
NAME HELL

#include <eikon.rh>
#include <avkon.rh>
#include <avkon.rsg>

#include "hellosymbian.hrh"

RESOURCE RSS_SIGNATURE
	{
	}

RESOURCE TBUF r_default_document_name
	{
	buf="";
	}

RESOURCE EIK_APP_INFO
	{
	menubar = r_hellosymbian_menubar;
	cba = R_AVKON_SOFTKEYS_OPTIONS_EXIT;
	}

RESOURCE MENU_BAR r_hellosymbian_menubar
	{
	titles =
		{
		MENU_TITLE
			{
			menu_pane = r_hellosymbian_menu;
			}
		};
	}

RESOURCE MENU_PANE r_hellosymbian_menu
	{
	items =
		{
		MENU_ITEM
			{
			command = EHelloSymbianCommand1;
			txt = "Say Hello!";
			},
		MENU_ITEM
			{
			command = EAknSoftkeyExit;
			txt = "Exit";
			}
		};
	}
```

اکثر قسمت‌های این فایل را در بخش اول مقاله توضیح داده‌ام و فقط چند تا مورد کوچک باقی است:

NAME HELL: کلمه دوم مشخصه فایل است و باید یک کلمه چهارحرفی باشد. البته زیاد مهم نیست. در اینجا ما چهار حرف اول نام پروژه‌مان را وارد کردیم.

فایل hellosymbian.hrh را هم که قبلاً نوشتیم. در اینجا چون از EHelloSymbianCommand1 که در آن فایل تعریف شده است، استفاده کرده‌ایم، باید آن را ضمیمه کنیم. فایل‌های eikon.rh، avkon.rh و avkon.rsg معمولاً همیشه ضمیمه می‌شوند (تمام کلمه‌های کلیدی که در اینجا استفاده کرده‌ایم در این فایل‌های تعریف شده‌اند). یک RESOURCE هم با نام RSS_SIGNATURE باید تعریف شود

که لازم نیست داخلش چیزی بنویسید. سپس یک RESOURCE هم از نوع TBUF تعریف می‌کنیم و نام پیش‌فرض برنامه را در آن می‌نویسیم.

در اینجا معمولاً همیشه یک نام خالی ("") می‌نویسیم.

کدنویسی برنامه:

تمام فایل‌ها و resourceهای لازم را آماده کردیم و الان نوبت بخش اصلی برنامه یعنی کدنویسی است.

ابتدا فایل‌های header را ایجاد می‌کنیم و سپس توابع این فایل‌ها را در فایل‌های cpp. پیاده‌سازی می‌کنیم.

ایجاد فایل hellosymbianapplication.h:

این فایل کلاس Application برنامه را می‌سازد. یک فایل متنی در پوشه inc با نام hellosymbianapplication.h ایجاد کنید و آن را باز کنید. ابتدا دستورات زیر را در فایل بنویسید تا محتویات فایل، بیش از یک بار include نشوند:

```
#ifndef _HELLO_HRH_
#define _HELLO_HRH_

//Codes will be here...

#endif
```

حال مکان‌نما را به خط بعد از کامنت ببرید و دستور زیر را در آنجا بنویسید:

```
#include <aknapp.h>
```

فایل aknapp.h حاوی تعریف کلاس CAknApplication است که ما از آن کلاس به عنوان کلاس پایه Application استفاده خواهیم کرد. در ادامه دستورات زیر را بنویسید:

```
class CHelloSymbianApplication : public CAknApplication
	{
public:
	TUid AppDllUid() const;

protected:	//from CAknApplication
	CApaDocument* CreateDocumentL();
	};
```

کلاس Application را که کلاس اصلی برنامه است با نام CHelloSymbianApplication تعریف کردیم. این کلاس دارای یک تابع با نام AppDllUid است که Uid برنامه را برمی‌گرداند به همین دلیل خروجی‌اش از نوع TUid است. یک متد دیگر هم با نام CreateDocumentL وجود دارد که یک آبجکت از کلاس Document می‌سازد و یک اشاره‌گر به آن برمی‌گرداند.

ایجاد فایل hellosymbiandocument.h:

در پوشه inc یک فایل دیگر با نام hellosymbiandocument.h ایجاد کنید و دستورات زیر را در آن بنویسید:

```
#ifndef _HELLOSYMBIANDOCUMENT_H_
#define _HELLOSYMBIANDOCUMENT_H_

#include <akndoc.h>

class CHelloSymbianAppUi;

class CHelloSymbianDocument : public CAknDocument
	{
public:
	static CHelloSymbianDocument* NewL(CEikApplication& aApp);
	static CHelloSymbianDocument* NewLC(CEikApplication& aApp);
	~CHelloSymbianDocument();
public:
	CEikAppUi* CreateAppUiL();
private:
	void ConstructL();
	CHelloSymbianDocument(CEikApplication& aApp);
	};
#endif
```

توضیح کد: ابتدا فایل akndoc.h را ضمیمه کردیم. این فایل حاوی تعریف کلاس CAknDocument است. سپس یک forward declaration از کلاس CHelloSymbianAppUi انجام دادیم؛ به علت اینکه در اینجا از این کلاس استفاده کردیم ولی کلاس را بعداً تعریف خواهیم کرد. سپس تعریف کلاس CHelloSymbianDocument را انجام دادیم. این کلاس دارای متدهای NewL و NewLC است که اجزای سازنده دوفازی هستند. متد ConstructL که کار ساخت آبجکت را انجام می‌دهد؛ یک سازنده (Constructor) و یک مخرب (Destructor) هم تعریف کرده‌ایم و بالاخره یک متد با نام CreateAppUiL که یک آبجکت از کلاس Document برنامه می‌سازد و یک اشاره‌گر به آن برمی‌گرداند.

ایجاد فایل hellosymbianappui.h:

یک فایل جدید با این نام در پوشه inc ایجاد کنید. کدهای زیر را در آن بنویسید:

```
#ifndef _HELLOSYMBIANAPPUI_H_
#define _HELLOSYMBIANAPPUI_H_

#include <aknappui.h>

class CHelloSymbianAppView;             //Forward declaration

class CHelloSymbianAppUi : public CAknAppUi
	{
public:
	void ConstructL();
	CHelloSymbianAppUi();
	~CHelloSymbianAppUi();

public:
	void HandleCommandL(TInt aCommand);

private:
	CHelloSymbianAppView* iAppView;
	};


#endif
```

توضیح کد: فایل aknappui.h که ضمیمه کرده‌ایم حاوی تعریف کلاس CAknAppUi است. از این کلاس، کلاس AppUi خودمان را با نام CHelloSymbianAppUi مشتق می‌کنیم. این کلاس دارای یک متغیر عضو از نوع کلاس View برنامه‌مان است (که کلاس را بعداً تعریف

خواهیم کرد. در اینجا چون از آن استفاده کرده‌ایم، یک Forward declaration هم از کلاس CHelloSymbianAppView انجام داده‌ایم.

یکی از متدهای کلاسمان، ConstructL است که همانطور که می‌دانید برای ساخت متغیرهای عضو کلاس (در اینجا iAppView) از آن استفاده خواهیم کرد. یک سازنده هم داریم، که در پیاده‌سازی آن چیزی نخواهیم نوشت. مخرب کلاس، کار delete کردن متغیر عضو را انجام خواهد داد. و بالاخره یک متد با نام HandleCommandL که با این متد، ورودی کاربر را مدیریت خواهیم کرد.

ایجاد فایل hellosymbianappview.h:

این فایل را هم در پوشه inc ایجاد کنید و کدهای زیر را در بنویسید:

```
#ifndef _HELLOSYMBIANAPPVIEW_H_
#define _HELLOSYMBIANAPPVIEW_H_

#include <coecntrl.h>

class CHelloSymbianAppView : public CCoeControl
	{
public:
	static CHelloSymbianAppView* NewL(const TRect& aRect);
	static CHelloSymbianAppView* NewLC(const TRect& aRect);
	~CHelloSymbianAppView();

public:
	void Draw(const TRect& aRect) const;

private:
	void ConstructL(const TRect& aRect);
	CHelloSymbianAppView();
	};


#endif
```

توضیح کد: در فایل coecntrl.h کلاس CCoeControl تعریف‌شده‌است. سپس کلاس خودمان را با نام CHelloSymbianAppView از این کلاس مشتق می‌کنیم. برای این کلاس هم ملزومات سازنده دو فازی را نوشته‌ایم (متدهای NewL و NewLC). کار متد ConstructL را هم که می‌دانید. یک سازنده و یک مخرب هم داریم. و بالاخره یک تابع با نام Draw. این تابع از کلاس CCoeControl به ارث برده شده است و در اینجا آن را over load خواهیم کرد. کار این تابع، کشیدن شکل یا عکس یا متن یا ... در صفحه است.

فایل‌های header لازم را ایجاد کردیم. اکنون سورس‌فایل‌های cpp را ایجاد می‌کنیم.

ایجاد فایل hellosymbian.cpp:

اکنون یک فایل با این نام در پوشه src ایجاد کنید. این فایل نقطه شروع برنامه ما است. کدهای زیر را در آن بنویسید:

```
#include "hellosymbianapplication.h"

GLDEF_C TInt E32Dll(TDllReason /*aReason*/)
	{
	return KErrNone;
	}

EXPORT_C CApaApplication* NewApplication()
	{
	return (static_cast<CApaApplication*>(new CHelloSymbianApplication));
	}
```

توضیح کد: در این فایل دو تابع تعریف کرده‌ایم که بدین صورت هستند:

تابع E32Dll: این تابع نقطه ورودی dll است. در برنامه‌های که با dll سروکار نداریم، برای این تابع دستور نوشته شده در بالا را می‌نویسیم.

دقت کنید که الگوی این تابع همیشه به صورت بالا است. این تابع یک آرگومان هم از نوع TDllReason دارد که ما چون در اینجا آن را لازم نداریم، این پارامتر را نمی‌گیریم.

تابع NewApplication: کار این تابع اجرای برنامه است. این تابع تقریباً معادل تابع main در C++ است. این تابع یک آبجکت از کلاس Application برنامه می‌سازد و یک اشاره‌گر به آن برمی‌گرداند. پیاده‌سازی این تابع را هم می‌بینید. یک Application جدید new می‌کنیم و آدرس آن را به نوع CApaApplication، cast می‌کنیم و برمی‌گردانیم.

ایجاد فایل hellosymbianapplication.cpp:

این فایل را در src ایجاد کنید و کدهای زیر را در آن بنویسید:

```
#include "hellosymbiandocument.h"
#include "hellosymbianapplication.h"

const TUid KUidHelloApp = {0x10006F90};

CApaDocument* CHelloSymbianApplication::CreateDocumentL()
	{
	return (static_cast<CApaDocument*>(CHelloSymbianDocument::NewL(*this)));
	}


TUid CHelloSymbianApplication::AppDllUid() const
	{
	return KUidHelloApp;
	}
```

توضیح کد: ابتدا فایل‌های لازم را ضمیمه می‌کنیم. سپس یک ثابت از نوع TUid تعریف می‌کنیم و uid برنامه را در آن ذخیره می‌کنیم. و بالاخره پیاده‌سازی دو تابع تعریف شده در hellosymbianapplication.h را می‌نویسیم. تعریف این دو تابع واضح است و نیازی به توضیح ندارند.

ایجاد فایل hellosymbiandocument.cpp:

این فایل را هم در پوشه src ایجاد کنید و کدهای زیر را در آن بنویسید:

```
#include "hellosymbianappui.h"
#include "hellosymbiandocument.h"

CHelloSymbianDocument* CHelloSymbianDocument::NewL(CEikApplication& aApp)
	{
	CHelloSymbianDocument* self = NewLC(aApp);
	CleanupStack::Pop(self);
	return self;
	}

CHelloSymbianDocument* CHelloSymbianDocument::NewLC(CEikApplication& aApp)
	{
	CHelloSymbianDocument* self = new (ELeave) CHelloSymbianDocument(aApp);
	CleanupStack::PushL(self);
	self->ConstructL();
	return self;
	}

void CHelloSymbianDocument::ConstructL()
	{
	}
CHelloSymbianDocument::CHelloSymbianDocument(CEikApplication& aApp) :
CAknDocument(aApp)
	{
	}
```

```
CHelloSymbianDocument::~CHelloSymbianDocument()
	{
	}

CEikAppUi* CHelloSymbianDocument::CreateAppUiL()
	{
	return (static_cast <CEikAppUi*> (new (ELeave) CHelloSymbianAppUi));
	}
```

توضیح کد: توضیح توابع NewL و NewLC را قبلاً گفته‌ام. تابع ConstructL هم چون در اینجا متغیر عضوی نداریم، خالی است. سازنده هم خالی است ولی یک کار کوچک انجام می‌دهد و آن هم این که آرگومان‌های لازم را (در اینجا aApp) به کلاس پایه‌اش می‌فرستد. مخرب هم خالی است، چون چیزی برای نابود کردن نداریم. پیاده‌سازی تابع CreateAppUiL هم واضح است. کار این تابع ساخت یک آبجکت از کلاس document برنامه و برگرداندن یک اشاره‌گر به آن است.

ایجاد فایل hellosymbianappui.cpp:

این فایل را در پوشه src ایجاد کنید و کدهای صفحه بعد را در آن بنویسید:

توضیح کد: ابتدا فایل‌های لازم را include می‌کنیم. فایل avkon.hrh حاوی تعریف EAknSoftkeyExit است و فایل aknnotewrappers.h هم حاوی تعریف CAknInformationNote است. در اینجا از آنها استفاده خواهیم کرد.

تابع ConstructL ابتدا تابع BaseConstructL را call می‌کند. این تابع هم بعضی کارهای پایه‌ای را انجام می‌دهید. سپس متغیر iAppView را new می‌کند. تابع ClientRect() که در اینجا از آن استفاده کرده‌ایم، محوطه‌ای که در اختیار برنامه ما است را از نوع TRect برمی‌گرداند.

پیاده‌سازی سازنده و مخرب هم که واضح است.

و بالاخره به تابع HandleCommandL می‌رسیم. این تابع را سیمبین موقعی که یک کلید زده شد، call می‌کند. ورودی این تابع یک عدد از نوع TInt است. در این تابع با یک دستور switch بررسی می‌کنیم تا ببینیم که چه رویدادی رخ داده است. اگر کاربر منوی Exit را زده

```
#include <avkon.hrh>
#include <aknnotewrappers.h>

#include "hellosymbianappview.h"
#include "hellosymbianappui.h"
#include "hellosymbian.hrh"

void CHelloSymbianAppUi::ConstructL()
	{
	BaseConstructL();

	iAppView = CHelloSymbianAppView::NewL(ClientRect());
	AddToStackL(iAppView);
	}

CHelloSymbianAppUi::CHelloSymbianAppUi()
	{
	}

CHelloSymbianAppUi::~CHelloSymbianAppUi()
	{
	if(iAppView)
		{
		iEikonEnv->RemoveFromStack(iAppView);
		delete iAppView;
		iAppView = NULL;
		}
	}

void CHelloSymbianAppUi::HandleCommandL(TInt aCommand)
	{
	switch(aCommand)
		{
		case EEikCmdExit:
		case EAknSoftkeyExit:
			Exit();
			break;
		case EHelloSymbianCommand1:
			{
			_LIT(message, "Hello Symbian World!");
			CAknInformationNote* informationNote = new (ELeave)
CAknInformationNote();
			informationNote->ExecuteLD(message);
			}
			break;
		default:
			break;
		}
	}
```

باشد، با دستور Exit() از برنامه خارج می‌شویم. ولی اگر منوی Hello را زده باشد، پیغام Hello Symbian World را نمایش می‌دهیم. این بخش از کد را بیشتر توضیح می‌دهم: ابتدا یک رشته تعریف می‌کنیم. تعریف رشته‌ها در برنامه‌نویسی سیمبین متفاوت از C++ استاندارد است. در سیمبین به رشته‌ها **واصف** (descriptor) می‌گوییم. کار با این نوع از رشته‌ها مفصل است و خود نیاز به مقاله جداگانه‌ای دارد و در اینجا فقط کافی است این را بدانید که برای تعریف یک واصف باید از تابع LIT_ استفاده کنید. این تابع دارای دو آرگومان است که اولی نام واصف و دومی رشته‌ای است که باید در آن واصف ذخیره شود. در اینجا ما رشته "Hello Symbian World!" را ذخیره کردیم. واصف‌ها کمتر از رشته‌ها فضا می‌گیرند، به همین دلیل در برنامه‌نویسی موبایل از آنها استفاده می‌شود.

سپس یک آبجکت از نوع CAknInformationNote تعریف می‌کنیم. این آبجکت یک Message Box ایجاد می‌کند. در سطر بعدی هم این جعبه پیغام را نمایش می‌دهیم (با متن مشخص شده).

ایجاد فایل hellosymbianappview.cpp:

این فایل را هم در پوشه src ایجاد کنید و کدهای زیر را در آن بنویسید:

```
#include <coemain.h>
#include "hellosymbianappview.h"

CHelloSymbianAppView* CHelloSymbianAppView::NewL(const TRect& aRect)
	{
	CHelloSymbianAppView* self = CHelloSymbianAppView::NewLC(aRect);
	CleanupStack::Pop(self);
	return self;
	}

CHelloSymbianAppView* CHelloSymbianAppView::NewLC(const TRect& aRect)
	{
	CHelloSymbianAppView* self = new (ELeave) CHelloSymbianAppView;
	CleanupStack::PushL(self);
	self->ConstructL(aRect);
	return self;
	}

CHelloSymbianAppView::CHelloSymbianAppView()
	{
	}

CHelloSymbianAppView::~CHelloSymbianAppView()
	{
	}

void CHelloSymbianAppView::ConstructL(const TRect& aRect)
	{
	CreateWindowL();
	SetRect(aRect);
	ActivateL();
	}

void CHelloSymbianAppView::Draw(const TRect& /*aRect*/) const
	{
	CWindowGc& gc = SystemGc();
	TRect rect = Rect();
	gc.Clear(rect);
	}
```

توضیح کد: توابع مورد نیاز را در ابتدا ضمیمه کردیم. فایل coemain.h حاوی توابع کار کردن با gcها (Graphic Context) است. از اینها در تابع Draw استفاده خواهیم کرد.

توابع NewL و NewLC و سازنده و مخرب را خودتان می‌فهمید که چه کرده‌ایم. ولی تابع ConstructL: ابتدا یک پنجره برای View ایجاد می‌کنیم. سپس اندازه پنجره ایجاد شده را مشخص می‌کنیم. و بالاخره پنجره را فعال می‌کنیم. حال می‌توانیم روی این پنجره عملیات Draw را انجام دهیم.

تابع Draw: در اینجا ما قصد کشیدن چیزی را نداریم و فقط می‌خواهیم صفحه را پاک کنیم. ابتدا یک gc تعریف می‌کنیم و آن را set می‌کنیم. سپس اندازه کنترل را در rect ذخیره می‌کنیم. و در پایان محوطه مشخص شده را پاک می‌کنیم.

ساخت برنامه HelloSymbian:

تمام کدنویسی‌های لازم را انجام دادیم. الان می‌خواهیم پروژه را کامپایل کنیم. ابتدا یک پنجره Command Prompot باز کنید و توسط دستور cd به پوشه‌ای که پروژه را ایجاد کرده‌اید، بروید (مثلاً مسیر C:\Symbian\HelloSymbian). در اینجا وارد پوشه group شوید و دستور زیر را وارد کنید:

```
bldmake bldfiles
```

این دستور، فایل abld.bat را در پوشه جاری ایجاد می‌کند. سپس دستور زیر را وارد کنید:

```
abld build armi urel
```

این دستور پروژه را کامپایل می‌کند. اگر تمام کدها را درست نوشته باشید و اسامی فایل‌ها را هم درست تنظیم کرده باشید، بدون error پروژه شما کامپایل خواهد شد. ولی اگر error داشتید، errorها در صفحه نشان داده خواهند شد. آنها را درست کنید و دوباره دستور را وارد کنید. اگر نتوانستید errorها را برطرف کنید، پروژه کامل را من همراه این فایل PDF قرار داده‌ام. آن را باز کنید و کدهای خودتان را با آن مقایسه کنید.

نحوه ساخت فایل sis:

فایل اصلی برنامه یعنی فایل hellosymbian.app قبلاً ساخته شده است ولی برای این که بتوانیم برنامه را به راحتی در موبایل نصب کنیم، باید فایل sis را ایجاد کنیم. فایل sis فایل نهایی پروژه‌مان است و قابل نصب بر روی موبایل است. فایل sis شبیه فایل‌های zip است؛ زیرا کار این فایل بسته‌بندی فایل‌هایی است که باید در موبایل کپی شوند.

برای این که بتوانید فایل sis را بسازید اول باید یک فایل pkg. بسازید. این فایل به سازنده sis می‌گوید که فایل sis را چگونه بسازد. یک فایل متنی جدید با نام hellosymbian.pkg در پوشه sis بسازید و دستورات زیر را در آن بنویسید:

```
&EN

#{"Hello"},(0x10006F90),1,0,0

(0x101F7960),0,0,0,{"Series60ProductID"}

"C:\Symbian\7.0s\Series60_v21\epoc32\release\armi\urel\hellosymbian.APP"-"!:\system\apps\hellosymbian\hellosymbian.app"
"C:\Symbian\7.0s\Series60_v21\epoc32\data\z\system\apps\hellosymbian\hellosymbian.rsc"-"!:\system\apps\hellosymbian\hellosymbian.rsc"
```

توضیح دستورات:

توجه) شرح کامل فایل pkg را در مقاله «فایل‌های پروژه‌های برنامه‌نویسی سیمبین» نوشته‌ام؛ می‌توانید به آن مراجعه کنید. در اینجا فقط شرح مختصری می‌دهم.

در سطر اول می‌گوییم که برنامه ما فقط از یک زبان پشتیبانی می‌کند و آن هم انگلیسی است.

در سطر بعدی اسم، uid و ورژن برنامه را مشخص می‌کنیم.

سطر سوم مشخصات دستگاه مقصد است. برای موبایل‌های سیمبین سری ۶۰، همیشه این گونه است.

در دو سطر بعدی فایل‌هایی که باید به موبایل کپی شوند را مشخص می‌کنیم. بخش اول آدرس فایل در کامپیوتر است و بخش دوم مسیری است که آن فایل در موبایل باید به آن کپی شود. در اینجا ما فقط دو تا فایل داریم که اولی فایل اجرایی برنامه ما است و دومی هم ترجمه فایل hellosymbian.rss است که حاوی منابع برنامه است.

توجه داشته باشید که مسیر فایل‌ها در کامپیوتر را مطابق کامپیوتان تصحیح کنید. مثلاً اگر SDK را در D نصب کرده‌اید، آن را تصحیح

کنید. دستوراتی که در اینجا نوشته‌ام، مطابق کامپیوتر خود من است.

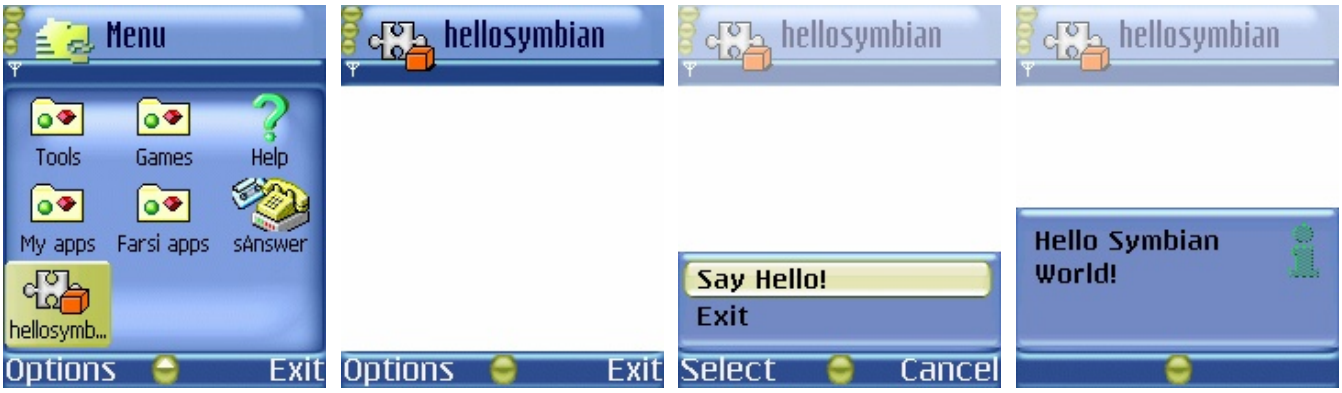
حالا دوباره سراغ Command Prompt می‌رویم. از پوشه group بالا بیایید و وارد sis شوید. در اینجا دستور زیر را وارد کنید:

```
makesis hellosymbian.pkg HelloSymbian(v1.00).sis
```

در اینجا برنامه makesis فایل pkg پاس شده به خود را می‌خواند و طبق دستورات آن، فایل sis با نام مشخص شده را می‌سازد. دقت داشته باشید که دادن نام فایل sis اختیاری است. اگر آن را ننویسید یک فایل sis با نام فایل pkg می‌سازد.

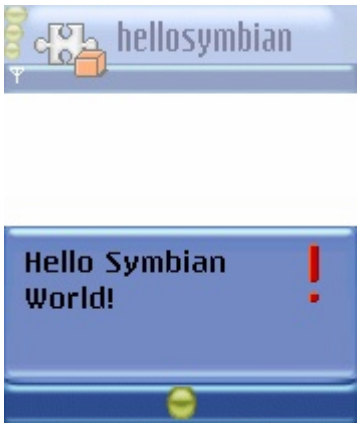
حال می‌توانید فایل sis را به موبایلتان انتقال دهید و آن را نصب کرده و از اولین برنامه خودتان که با دست‌های خودتان نوشته‌اید و حاصل دسترنج خودتان است و منت IDEها را نکشیده‌اید(!) لذت ببرید.

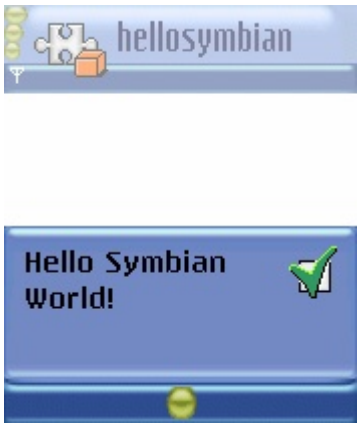
و تصاویری از این برنامه در موبایل من:



تمرین

این برنامه را طوری تغییر دهید که پیام‌های زیر را نشان دهد:





راهنمایی: به جای CAknInformationNote از CAknErrorNote و CAknConfirmationNote استفاده کنید. (البته راهنمایی که چه عرض کنم، همه جواب همین بود!)

سخن آخر

خوشحالم که توانستم نوشتن این مقاله را تمام کنم. برخلاف مقالات قبلی، این مقاله ترجمه نبود و خودم آن را نوشتم، به همین دلیل وقت زیادی برد.

امیدوارم این مقاله برایتان مفید بوده باشد. ضمناً تصمیم دارم ادامه این مقاله را در pdf جداگانه‌ای بنویسم و نحوه فارسی کردن و بعضی چیزهای دیگر را توضیح دهم. البته شما منتظر من نمانید و خودتان دست به کار شوید. بهترین منبع هم برای یادگیری همانطور که در مقدمه هم گفتم، مستندات SDK است.

اکنون که توانستید مقدمات برنامه‌نویسی سیمبین را یاد بگیرید، ادامه راه برایتان آسان شده است. برای افزایش اطلاعاتتان علاوه بر یادگیری از طریق منابع مختلف، می‌توانید از روش کد خواندن هم استفاده کنید. روشی بسیار مؤثر که من خودم از این طریق چیزهای زیادی یاد گرفته‌ام. برای این کار می‌توانید از پروژه‌های نمونه‌ای که همراه SDK آمده‌اند، استفاده کنید. اول برنامه‌های ساده را باز کنید و کدهایشان را بخوانید و ببینید که چگونه کارهای برنامه را پیاده‌سازی کرده‌اند.

سؤالات و مشکلاتتان را در مورد این مقاله به ایمیل mousamk@gmail.com بفرستید. انتقاد و پیشنهادی هم بود در خدمتیم.

بازدید از وبلاگ من هم یادتان نرود:

http://series60.blogfa.com

http://symbiandeveloper.blogfa.com

پایان