Symbian OS Quick Overview of: Name Conventions and Types. *

For more Symbian programming resources visit:
http://kom.aau.dk/project/mobilephone/

Last updated: 17 October 2005

Contact: mvpe04@control.aau.dk

*Thanks to the guys at #symbian.dev on EFnet for their comments and suggestions regarding the content of this document. Go pay them a visit.

Static Classes

Classes consisting purely of static functions; these classes have no prefix letter and cannot be instantiated into an object. Examples: `Math, User` and `MyStaticClass`. Today static classes are not used anymore and our static functions should instead be wrapped in a namespace.

T Classes

A class prefixed with a T is a simple class that owns no heap memory, and therefore does not require any explicit destructor. One should be careful where to instantiate a T class, because even though it's simple it can consume a rather large amount of memory. All basic types have T prefix (see `e32def.h`and `e32std.h`):

Type	Symbian C++ replacement of	Comment
TAny	`void` (`TAny*` ≡ `void*`)	Shold only be used as `TAny*`
TInt	Integer: `int`	Minimum 32 bits
TInt8/16/32	Integer: `int`	8/16/32 bit versions
TUint	Unsigned integer: `unsigned int`	Minimum 32 bits
TUint8/16/32	Unsigned integer: `unsigned int`	8/16/32 bit versions
TReal	Floating point: `double`	64 bit
TReal32/64	Floating point: `float` or `double`	32/64 bit versions
TChar	Charater: `char`	32 bit
TText	Charater	16 bit
TText8/16	Charater	8/16 bit versions
TBool	Boolean: `bool`	32 bit. Use the emum `ETrue`/ `EFalse`

C Classes

A C-class derives from `CBase` or another class deriving form `CBase`.C-classes are always constructed on the heap. The prefix C means "Cleanup", and indicates that we should be careful to avoid memory leaks. You never derive from more than one C class. Example:

```
CMyExample : public CBase
```

```
{
    TInt iNumber;
    ...
public:
    ~CMyExample();
};
```

R Classes

The R-classes are special classes that are used as a client-side handle to a resource. Normally they are instantiated on the stack wrapped in a C-class, and then opened in some way (e.g. through a call to `connect()`,and closed with `close()`) Don't forget to close open R-class connections, otherwise memory leaks can occur. Example:

```
CFileAccess : public CBase
{
    // R-class, through which we connect to the file server
    RFs iFileServer;
    ...
public:
    ~CFileAccess()
    {
        iFileServer.Close();
    }
};
```

M Classes

A M class defines a set of abstract interfaces - consisting of purely virtual functions. M classes are the only classes in the Symbian OS that can be used in connection with multiple inheritance. Example:

```
class MNotify
{
    virtual void HandleEvent(TInt aEvent) = 0;
}
```

Structs

A ordinary C/C++ struct, without any member functions should be prefixed with a uppercase S. Not very often seen in Symbian OS, but here's a few examples from `eikdialg.h`:

```
struct SEikControlInfo;
struct SEikRange;
struct SEikDegreesMinutesDirection;
```

Normally replaced by a T-class.

Member variables

Member variables are prefixed with a lower-case i (for instance). Example:

```
CMyExample : public CBase
{
    TInt iNumber;
    TChar iLetter;
    ...
};
```

Automatic variables

The only rule applying to automatic variables are that they start with a lower-case letter.

Function names

- Should start with a upper-case letter e.g. `ShowPath(TFileName &aPath)`.
- Functions that might leave should be named with a trailing upper-case L e.g. `NewL()`.
- Functions that leaves pointers on the cleanup-stack should be named with a trailing upper-case C e.g. `NewLC()`.

- Functions with a trailing D indicates the deletion of an object e.g. ExecuteLD().

Function arguments

Function arguments are prefixed with a lower-case a (for argument). Example:

```
TInt CFileConnect::ShowPath(TFileName &aPath)
{
    return iFs.DefaultPath(aPath);
}
```

Constant data

Constant data are prefixed with a upper-case K. Example:

```
const TInt KNumber = 1982;
```

Namespaces

Namespaces are prefixed with a upper-case N. Namespaces should be used to group functions that don't belong anywhere else. Example:

```
namespace NMyFunctions
{
    ...
}
```

Emunerations

Enumerations are types therefore they are prefixed with a upper-case T. The emumeration members are prefixed with a upper-case E. Example:

```
enum TGames = {EMonopoly, ETetris, EChess = 0x05};
```