

目次

第 1 章 Package bar	3
1.1 Class FileFactory	3
1.1.1 Method	3
1.2 Class Modifier	4
1.2.1 Method	4
1.3 Interface Openable	4
1.3.1 Method	4
1.4 Class Splint	6
1.4.1 Struct	6
1.4.2 Method	6
1.5 Class Thread	7
1.5.1 Struct	7
1.5.2 Method	7
1.5.3 Abstract Method	7
第 2 章 Package com_example	9
2.1 Class Double	9
2.1.1 Struct	9
2.1.2 Method	9
2.2 Class Integer	11
2.2.1 Macro	11
2.2.2 Typedef	11
2.2.3 Struct	11
2.2.4 Method	12
2.3 Class StringList	13
2.3.1 Struct	13
第 3 章 Package foo	15
3.1 Class Description	15
3.1.1 クラスの説明中のセクション	15
3.1.2 文字の装飾	15
3.1.3 ハイパーリンク	15
3.1.4 リスト	16

3.1.5	Struct	16
3.1.6	Method	17
3.2	Class File	17
3.2.1	Struct	18
3.2.2	Method	18
3.3	Class FunctionMember	18
3.3.1	Struct	18
3.4	Class NestedStruct	19
3.4.1	Struct	19
3.5	Class Opaque	20
3.5.1	Macro	20
3.5.2	Typedef	20
3.5.3	Method	21
3.6	Class OpaqueStruct	21
3.6.1	Struct	21
3.6.2	Method	21
3.7	Class Preference	22
3.7.1	Global Variable	22
3.8	Class Printer	22
3.8.1	Method	22
3.9	Class Time	23
3.9.1	Macro	23
3.10	Class Union	23
3.10.1	Union	23
3.11	Class Whence	24
3.11.1	Enum	24

第1章 Package bar

上級編のサンプルを提供します。初級編のサンプルと簡単なサンプルは[こちら](#)、中級編のサンプルは[こちら](#)です。

このパッケージのクラスはすべて名前空間展開を利用しています。

1.1 Class FileFactory

```
#define bar_FileFactory_IMPORT
#include <bar/FileFactory.h>
```

File クラスのインスタンスを生成する機能を提供します。

1.1.1 Method

1.1.1.1 FileFactory_open

```
foo_File * FileFactory_open(
    const void * cookie,
    int (*readfn)(void *, char *, int),
    int (*writefn)(void *, const char *, int),
    fpos_t (*seekfn)(void *, fpos_t, int),
    int (*closefn)(void *))
```

Openable インタフェースから File クラスのインスタンスを生成します。
生成できないときは NULL を返します。

Parameters

cookie [Openable インタフェース](#)のインスタンス

readfn see [Openable.readfn\(\)](#)

writefn see [Openable.writefn\(\)](#)

seekfn see [Openable.seekfn\(\)](#)

closefn see [Openable.closefn\(\)](#)

Returns

[File クラス](#)のインスタンス、または NULL

第1章 Package bar

1.2 Class Modifier

```
#define bar_Modifier_IMPORT  
#include <bar/Modifier.h>
```

修飾子の例を提供します。

1.2.1 Method

1.2.1.1 Modifier_abort

```
void Modifier_abort(void) __attribute__((noreturn))
```

コアをダンプして強制終了します。

1.2.1.2 Modifier_square

```
int Modifier_square(  
    int value) __attribute__((const))
```

自乗の値を返します。

Parameters

value 整数値

Returns

value の自乗の値

1.3 Interface Openable

オープン可能なインタフェースの記述例を提供します。

1.3.1 Method

1.3.1.1 Openable_readfn

```
int Openable_readfn(  
    void * cookie,  
    char * data,  
    int size)
```

ファイルからデータを読み込みます。

Parameters

cookie このインスタンスのポインタ
data 読み込むデータを格納する領域のポインタ
size data のサイズ

Returns

成功した場合は実際に読み込んだデータのサイズを返します。ファイルの最後の場合は 0 を返します。そうでなければ -1 を返します。

1.3.1.2 Openable_writfn

```
int Openable_writfn(  
    void * cookie,  
    const char * data,  
    int size)
```

ファイルにデータを書き込みます。

Parameters

cookie このインスタンスのポインタ
data 書き込むデータを格納した領域のポインタ
size data のサイズ

Returns

成功した場合は書き込んだデータのサイズを返します。そうでなければ -1 を返します。

1.3.1.3 Openable_seekfn

```
fpos_t Openable_seekfn(  
    void * cookie,  
    fpos_t offset,  
    int whence)
```

ファイルのオフセットを変更します。

whence の値には `foo_Whence_SET`、`foo_Whence_SET`、`foo_Whence_END` のいずれかを指定します。

Parameters

cookie このインスタンスのポインタ
offset ファイルのオフセット
whence offset の起点

第1章 Package bar

Returns

成功した場合はファイルの先頭からのオフセットを返します。そうでなければ `-1` を返します。

1.3.1.4 Openable_closefn

```
int Openable_closefn(  
    void * cookie)
```

ファイルをクローズします。

Parameters

`cookie` このインスタンスのポインタ

Returns

成功した場合は `0` を返します。そうでなければ `-1` を返します。

1.4 Class Splint

```
#define bar_Splint_IMPORT  
#include <bar/Splint.h>
```

Splint のアノテーションを埋め込んだ例を提供します。

1.4.1 Struct

1.4.1.1 struct Splint

Splint 構造体です。不透明な構造体です。

1.4.2 Method

1.4.2.1 Splint_initialize

```
void Splint_initialize(  
    /*@out@*/ struct Splint * s)
```

Splint 構造体を初期化します。

Parameters

`s` 初期化する Splint 構造体のポインタ

1.5 Class Thread

```
#define bar.Thread_IMPORT
#include <bar/Thread.h>
```

アブストラクトメソッドの例を提供します。

1.5.1 Struct

1.5.1.1 struct Thread

スレッドの実体となる構造体です。不透明な構造体です。

1.5.2 Method

1.5.2.1 Thread_new

```
void Thread_new(
    struct Thread * thread,
    void (*run)(struct Thread *))
```

スレッドのインスタンスを生成します。

Parameters

thread Thread クラスのインスタンス

run see Thread_run()

1.5.2.2 Thread_start

```
void Thread_start(
    struct Thread * thread)
```

新しいスレッドによる Thread_run() の実行を開始します。

Parameters

thread Thread クラスのインスタンス

1.5.3 Abstract Method

1.5.3.1 Thread_run

```
void Thread_run(
    struct Thread * thread)
```

スレッドが実行する処理を実装します。

第1章 Package bar

Parameters

thread Thread クラスのインスタンス

第2章 Package com_example

初級編のサンプルと簡単なサンプルを提供します。中級編のサンプルは[こちら](#)、上級編のサンプルは[こちら](#)です。

このパッケージのクラスはすべて名前空間展開を利用しています。

2.1 Class Double

```
#define com_example_Double_IMPORT  
#include <com/example/Double.h>
```

実数値を表す型と、その型への操作を提供します。

Integer クラスの struct Integer と異なり、構造体を型定義しています。

2.1.1 Struct

2.1.1.1 Double (struct)

Double クラスのインスタンスとなる構造体です。
実数を保持します。

value

```
double value
```

現在の実数値を保持します。

2.1.2 Method

2.1.2.1 Double_new

```
Double * Double_new(  
    double value)
```

Double クラスのインスタンスを生成します。
生成できないときは NULL を返します。

第2章 Package com.example

Parameters

value 初期値となる実数値

Returns

Double クラスのインスタンス、または NULL

2.1.2.2 Double.delete

```
void Double.delete(  
    Double * d)
```

Double クラスのインスタンスを破壊します。

Parameters

d Double クラスのインスタンス

2.1.2.3 Double.set

```
void Double.set(  
    Double * d,  
    double value)
```

実数値を設定します。

Parameters

d Double クラスのインスタンス

value 実数値

2.1.2.4 Double.get

```
double Double.get(  
    Double * d)
```

実数値を取得します。

Parameters

d Double クラスのインスタンス

Returns

実数値

2.2 Class Integer

```
#define com.example.Integer_IMPORT
#include <com/example/Integer.h>
```

整数値を表す型と、その型への操作を提供します。
チュートリアル以外に用途はありません。

2.2.1 Macro

2.2.1.1 Integer_MAX

Integer_MAX

整数の最大値です。

2.2.1.2 Integer_MIN

Integer_MIN

整数の最小値です。

2.2.2 Typedef

2.2.2.1 Integer_t

整数値を表す型です。

2.2.3 Struct

2.2.3.1 struct Integer

Integer クラスのインスタンスとなる構造体です。
整数を保持します。

value

Integer_t value

現在の整数値を保持します。

第2章 Package com_example

2.2.4 Method

2.2.4.1 Integer_new

```
struct Integer * Integer_new(  
    Integer_t i)
```

Integer クラスのインスタンスを生成します。
生成できないときは NULL を返します。

Parameters

i インスタンスの初期化

Returns

生成したインスタンス、または NULL

2.2.4.2 Integer_delete

```
void Integer_delete(  
    struct Integer * i)
```

Integer クラスのインスタンスを破壊します。
i が NULL のときは何もしません。

Parameters

i Integer クラスのインスタンス

2.2.4.3 Integer_set

```
void Integer_set(  
    struct Integer * i,  
    Integer_t value)
```

整数値を設定します。

Parameters

i Integer クラスのインスタンス

value 整数値

2.2.4.4 Integer.get

```
Integer_t Integer.get(
    struct Integer * i)
```

整数値を取得します。

Parameters

i Integer クラスのインスタンス

Returns

整数値

2.3 Class StringList

```
#define com.example.StringList_IMPORT
#include <com/example/StringList.h>
```

自己参照する構造体の例として文字列のリストを表すデータ型を提供します。
操作は省略されています。

2.3.1 Struct

2.3.1.1 StringList (struct StringList)

文字列のリストとなる構造体です。
構造体のリンクでリストを実現します。

next

```
struct StringList * next
```

リストの次の構造体を指します。
リストの最後の構造体の場合、NULL になります。

value

```
char * value
```

保持している文字列です。

第3章 Package foo

中級編のサンプルを提供します。初級編のサンプルと簡単なサンプルは[こちら](#)、上級編のサンプルは[こちら](#)です。

このパッケージのクラスはすべて名前空間展開を利用しています。

3.1 Class Description

```
#define foo.Description_IMPORT
#include <foo/Description.h>
```

Description クラスはドキュメントの書き方を提供します。
このクラスの説明を記述します。

3.1.1 クラスの説明中のセクション

セクションはこのように展開されます。

3.1.2 文字の装飾

強調したい語句はこのようになります。
タイプライタ体は `keyword` のようになります。
引数やパラメータは `argument` のようになります。
マイナス記号は `-1` のようになります。

3.1.3 ハイパーリンク

URL へのリンクは <http://www.example.com/> のようになります。
同一クラス内へのリンクは [このよう](#) になります。
他のクラスへのリンクは [com.example.Integer クラス](#) のように、パッケージの文書へのリンクは [com.example パッケージ](#) のようになります。

第3章 Package foo

3.1.4 リスト

リストは次のように展開されます。

- リストの項目 1
- リストの項目 2

See: 参照先の項目を記述します。

Note: 注意すべき項目を記述します。

3.1.5 Struct

3.1.5.1 struct Description

構造体のドキュメントの書き方を説明します。
この構造体の説明を記述します。

構造体の説明中のセクション セクションはこのように展開されます。

value

```
int value
```

構造体のメンバのドキュメントの書き方を説明します。
このメンバの説明を記述します。

メンバの説明中のセクション セクションはこのように展開されます。

method

```
int (*method)(  
    int in)
```

構造体の関数ポインタ型メンバのドキュメントの書き方を説明します。
この関数ポインタ型メンバの説明を記述します。

関数ポインタ型メンバの説明中のセクション セクションはこのように展開されます。

Parameters

in 関数の引数のドキュメントの書き方を説明します。
この引数の説明を記述します。

引数の説明中のセクション セクションはこのように展開されます。

Returns

関数の戻り値のドキュメントの書き方を説明します。
この戻り値の説明を記述します。

戻り値の説明中のセクション セクションはこのように展開されます。

3.1.6 Method

3.1.6.1 Description_method

```
int Description_method(  
    int in)
```

関数のドキュメントの書き方を説明します。
この関数の説明を記述します。

関数の説明中のセクション セクションはこのように展開されます。

Parameters

in 関数の引数のドキュメントの書き方を説明します。
この引数の説明を記述します。

引数の説明中のセクション セクションはこのように展開されます。

Returns

関数の戻り値のドキュメントの書き方を説明します。
この戻り値の説明を記述します。

戻り値の説明中のセクション セクションはこのように展開されます。

3.2 Class File

```
#define foo.File_IMPORT  
#include <foo/File.h>
```

include 要素の使用例を提供します。
include 要素はドキュメントに影響を与えません。

第3章 Package foo

3.2.1 Struct

3.2.1.1 File (struct)

File クラスのインスタンスとなる構造体です。

file

```
FILE * file
```

標準入出力ライブラリの FILE 型へのポインタです。

3.2.2 Method

3.2.2.1 File_new

```
File * File_new(  
    FILE * file)
```

File クラスのインスタンスを生成します。
生成できないときは NULL を返します。

Parameters

file 標準入出力ライブラリの FILE 型へのポインタ

Returns

File クラスのインスタンス、または NULL

3.3 Class FunctionMember

```
#define foo_FunctionMember_IMPORT  
#include <foo/FunctionMember.h>
```

関数ポインタ型のメンバをもつ構造体の例を提供します。

3.3.1 Struct

3.3.1.1 struct FunctionMember

関数ポインタ型のメンバをもつ構造体です。

func

```
int (*func)(
    int in)
```

入力に対応する出力を返します。

Parameters

in 入力値

Returns

出力値

3.4 Class NestedStruct

```
#define foo_NestedStruct_IMPORT
#include <foo/NestedStruct.h>
```

ネストする構造体の例を提供します。

3.4.1 Struct

3.4.1.1 struct @1

匿名の構造体です。

匿名の構造体には、便宜上、仮の名前が付けられます。仮の名前は@に続くユニークな ID (整数値) となります。

匿名の構造体の説明をここに記述します。

i

```
int i
```

何かの整数値です。

3.4.1.2 struct NestedStruct_Inner

構造体のメンバとして定義される構造体です。

この構造体はトップレベルで定義された場合と同様に扱えます。

構造体 *Inner* の説明をここに記述します。

第3章 Package foo

i

```
int i
```

何かの整数値です。

3.4.1.3 struct NestedStruct.Outer

メンバとして構造体を定義する構造体です。

s1

```
struct s1
```

型が「匿名の構造体」のメンバです。
メンバ *s1* の説明をここに記述します。

s2

```
struct NestedStruct.Inner s2
```

型が「foo_NestedStruct.Inner 構造体」のメンバです。
メンバ *s2* の説明をここに記述します。

3.5 Class Opaque

```
#define foo_Opaque_IMPORT  
#include <foo/Opaque.h>
```

import 要素の使用例を提供します。

3.5.1 Macro

3.5.1.1 Opaque_TYPE

Opaque_TYPE

コンストラクタの引数の型です。
型の詳細は隠蔽されています。

3.5.2 Typedef

3.5.2.1 Opaque

ある構造体のポインタ型です。
構造体の詳細は隠蔽されています。

3.5.3 Method

3.5.3.1 Opaque_new

```
Opaque Opaque_new(
    Opaque_TYPE s)
```

別のクラスで定義されている構造体のポインタ型から、不透明なデータ型のインスタンスを生成します。

Parameters

s 別のクラスで記述されている構造体のポインタ

Returns

インスタンスのポインタ

3.6 Class OpaqueStruct

```
#define foo.OpaqueStruct_IMPORT
#include <foo/OpaqueStruct.h>
```

export 要素の使用例を提供します。

3.6.1 Struct

3.6.1.1 struct OpaqueStruct

ある構造体です。
構造体の詳細は隠蔽されています。

3.6.2 Method

3.6.2.1 OpaqueStruct_new

```
struct OpaqueStruct * OpaqueStruct_new(
    struct another.Class * s)
```

別のクラスで定義されている構造体のポインタ型から、不透明なデータ型のインスタンスを生成します。

Parameters

s 別のクラスで記述されている構造体のポインタ

第3章 Package foo

Returns

インスタンスのポインタ

3.7 Class Preference

```
#define foo_Preference_IMPORT  
#include <foo/Preference.h>
```

globalvariable 要素の使用例を提供します。

グローバル変数を使用するべきではありません。設定、取得メソッドをもつシングルトンパターンのクラスを用意すべきです。

3.7.1 Global Variable

3.7.1.1 Preference_option

```
char * Preference_option
```

オプションを表す文字列です。

3.7.1.2 Preference_parameters

```
int Preference_parameters[3]
```

パラメータを表す整数の配列です。
配列の要素数は3です。

3.8 Class Printer

```
#define foo_Printer_IMPORT  
#include <foo/Printer.h>
```

variableparam 要素の使用例を提供します。

3.8.1 Method

3.8.1.1 Printer_print

```
int Printer_print(  
    const char * format,  
    ...)
```

指定したフォーマットに整形した文字列を出力します。
いわゆる printf() ライクな関数です。

Parameters

format フォーマット (printf(3) を参照してください)

Returns

出力したバイト数

3.9 Class Time

```
#define foo.Time_IMPORT
#include <foo/Time.h>
```

timeval 構造体を演算する操作を提供します。

3.9.1 Macro

3.9.1.1 Time_add

Time_add(tvp, uvp, vvp)

timeval 構造体の和を求めます。

tvp + uvp の値を計算し、vvp に格納します。

tvp, uvp, vvp は timeval 構造体のポインタでなければなりません。

Parameters

tvp timeval 構造体のポインタ

uvp timeval 構造体のポインタ

vvp 和を格納する timeval 構造体のポインタ

3.10 Class Union

```
#define foo.Union_IMPORT
#include <foo/Union.h>
```

int と double、unsigned int と unsigned char のビットマップを相互に変換する機能を提供します。

3.10.1 Union

3.10.1.1 union Union_IntDouble

int の配列 (要素数 2) と double の共用体です。

第3章 Package foo

i

```
int i[2]
```

int 型の配列としてアクセスします。

d

```
double d
```

double 型としてアクセスします。

3.10.1.2 Union.UCharUInt (union)

unsigned char の配列 (要素数 4) と unsigned int の共用体です。

c

```
unsigned char c[4]
```

unsigned char 型の配列としてアクセスします。

i

```
unsigned int i
```

unsigned int 型としてアクセスします。

3.11 Class Whence

```
#define foo.Whence_IMPORT  
#include <foo/Whence.h>
```

ファイルのシークに必要な指令を提供します。
lseek(2) の引数 whence を enum で表現してみました。

3.11.1 Enum

3.11.1.1 enum Whence

ファイルをシークする際にオフセットの基点を指定する列挙型です。

Whence_SET ファイル先頭からのオフセットを表します。

Whence_CUR 現在のファイル位置からのオフセットを表します。

Whence_END ファイル末尾からのオフセットを表します。