Literate Object Oriented Library.

Brandon Koepke

February 9, 2016

Contents

Abstract

Implemented of a literate object oriented library with deep integration with design by contracts.

1 Class Definition

Simple definition of a class contains the class name and nothing else. In the future we may want to add additional members here for introspection but at the moment this is sufficient.

1a ⟨*class.h* 1a⟩≡

```
#pragma once
#include <contract.h>
typedef struct {
  const char * const name;
} Class;

/**
 * Gets the class name.
 *
 * @param c the class to get the name of.
 * @return the name of the class.
 */
const char * const class_name(const Class * const c);
```

1b ⟨*class.c* 1b⟩≡

```
#include "class.h"

const char * const class_name(const Class * const c) { return c->name; }
```

2 Object Definition

This is a simple object model. By default objects can be freed and you can get a string representation of the object. There are default functions for both of these. The _object_free just calls the regular stdlib free, and _object_to_string prints an object similar to how objects are printed in java.

2 ⟨*object.h* 2⟩≡

```
#pragma once
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <contract.h>
#include "class.h"

typedef struct _object_vtable object_vtable;
typedef struct {
  object_vtable *vtable;
} Object;
struct _object_vtable {
  const Class class;
  const char * const (*to_string)(const Object * const);
  void (*free)(Object *);
};

/**
 * Frees the memory used by the specified object.
 *
 * @param o the object to free.
 */
void object_free(Object *o);

/**
 * Gets a string representation of the specified object.
 *
 * @param o the object to get the string representation of.
 * @return a string representation of the object.
 */
const char * const object_to_string(const Object * const o);

void _object_free(Object *o);
const char * const _object_to_string(const Object * const o);
```

3 ⟨*object.c* 3⟩≡

```
#include "object.h"

void _object_free(Object *o) { free(contract_requires_non_null(o)); }

void object_free(Object *o) { o->vtable->free(contract_requires_non_null(o)); }

const Class * const object_class(const Object * const o) {
  contract_requires_non_null(o);
  return &o->vtable->class;
}

const char * const object_to_string(const Object * const o) {
  return o->vtable->to_string(contract_requires_non_null(o));
}

const char * const _object_to_string(const Object * const o) {
  const char * const name = class_name(object_class(o));
  size_t size = sizeof(void *) + sizeof('@') + strlen(name) + sizeof('\n');
  char *buffer = malloc(size);
  snprintf(buffer, size, "%s@%x", name, (unsigned int)o);
  return buffer;
}
```