Design Document

**Product Name**

**List Group Member Names**

**Revision Number**

**Revision Date**

Table of Contents

[Introduction 3](#_Toc219813623)

[Introduction to this outline 3](#_Toc219813624)

[Visual of User Interface 4](#_Toc219813625)

[List of User Commands 4](#_Toc219813626)

[Design of Features 5](#_Toc219813627)

[Manifest Constants and Macros 6](#_Toc219813628)

[Global Data Map 6](#_Toc219813629)

[Class N Design 7](#_Toc219813630)

[Class N Properties 7](#_Toc219813631)

[Class N Methods 7](#_Toc219813632)

[Name 7](#_Toc219813633)

[Pre Conditions 7](#_Toc219813634)

[Post Conditions 7](#_Toc219813635)

[Inputs 7](#_Toc219813636)

[Outputs 7](#_Toc219813637)

[Local Variables 7](#_Toc219813638)

[Algorithm 8](#_Toc219813639)

[Class N Event Handlers 8](#_Toc219813640)

[Application Design 9](#_Toc219813641)

[Organization of Code 10](#_Toc219813642)

[Other Design Considerations 10](#_Toc219813643)

# Introduction

This is often just copied from the Project Plan.

## Introduction to this outline

The outline and topics for this document will greatly based on the nature of the product being designed, but some general items should be assumed. This document assumes it is being used for a product that is following an Object Oriented Design process. Adjust as necessary.

# Visual of User Interface

This section should describe the interface. It is usually created from a drawing tools and shows what the user can expect to see. If there are multiple views, each one should be described. Every view should show what options the user will see and what action they will perform. It is not intended to show every possible view of jut one screen, but should describe each base screen.

IT IS NOT TO BE FILLED IN FROM THE IMPLEMENTED PRODUCT. That is it better not be reversed engineered from the product.

In product design, prototypes are sometimes created to demonstrate a non functional product along with the user interfaces. Screen captures can be used for this section from product prototypes.

# List of User Commands

This may be in addition to the above section or a replacement. It is usually used for command based or menu driven applications.

# Design of Features

For EVERY feature listed in the Requirements document provide a list of items needed to implement the feature. This can be done in many ways using many methodologies and notations. This outline assumes a product using Object Oriented Design methodology. Again note, this is a list!

For each feature list the following

* Data Structures used and whether they are global or local

This includes items like stacks, queues, lists, trees etc:. If possible, list them by name and how many are used.

* Classes used and whether they are global or local

List all classes from which objects are instantiated that the feature needs. If possible, list them by name and how many are used.

* Variables used and whether they are global or external

List all the non-local variables used by this feature. List them by name. These are not data members or properties of a class.

* Functions used and whether they are global or external

List all the non-local support functions used by this feature. List them by name. These are not member functions or methods of a class.

Please note that not every item on the list applies to every feature and some implementations do not support all items.

# Manifest Constants and Macros

These are symbolic constants defined for the application. As an example C++ allows the use of “#define” (a preprocessing directive) statements to create manifest constants that are replaced with real values as part of the compilation process. This also applies to any defined macros.

# Global Data Map

This section lists all global and external items. This includes

* Data Structures
* Objects Instantiated from Classes
* Variables

Data Structures should be identified by name (or pointer name) and their initial state. Are they fixed sized or dynamic? If fixed size, what is the capacity of the structure? Initial state should be defined as empty, not initialized or special.

Objects should be identified by name (or pointer name) and their initial state. Initial date should be defined as default, not initialized or special.

Variables should be identified by name (or pointer name) and their type and initial state. The variable type should be things like integer, real number, string etc: DO NOT use language keywords. The initial state should be the actual value starting value or not initialized.

# Class N Design

This section is repeated for each unique object. Replace “N” with the name of the class.

Note any relation this class has to any other class or classes. As applicable, note the following relationships.

* Uses a – Does this class use any other classes?
* Knows a – Does this class have a “friend” relationship with any other classes?
* Has a – Does this class have any other objects as data members or properties?
* Is a – Does this class inherit attributes from any other classes?

Diagrams are often used to show thee relationships.

## Class N Properties

This is a description of all properties or data members for this class. For each property, list the type and initial state as done for global variables. This includes the accessibility of the property. Is it private or public?

## Class N Methods

Describe every method or member function of the class. Use the following format

### Name

List the method name and the accessibility of the method. Is it private or public?

### Pre Conditions

List any conditions or assumptions that must be met before this method is activated.

### Post Conditions

List any changes that occur to the global data map made by this method.

### Inputs

List all inputs needed by this method. Describe them exactly as variables are described including the non-use of key words.

### Outputs

List all outputs returned by this method. Describe them exactly as variables are described including the non-use of key words.

### Local Variables

Describe all local variables created by this method exactly as variables are described including the non-use of key words.

### Algorithm

This is a description of the algorithm used by the method to accomplish its task. The algorithm is a step by step sequence of events the code must perform.

Code algorithms are usually described in one of two ways. The method used is up to the designer.

The first is flow charts. These are pictorial representations of the algorithm. To create them a tool such as Visio is often used.

The second is the use of pseudo code. This is a semi formal (sometimes formal) prose description of the algorithm. Pseudo code is a text description of the algorithm.

Note that neither flowcharts nor pseudo code is real code.

THE ALGORITHM IS NOT TO BE FILLED IN FROM THE IMPLEMENTED PRODUCT. That is it better not be reversed engineered from the product.

## Class N Event Handlers

Document this section the exact same way the Methods were described. Note that some implementations do not differentiate between Methods and Event Handlers.

# Application Design

If the application is implemented as a class (JAVA does this) the this section does not apply since it would already have been described as a class. If using a language like C++, then the application source file will have at least a “main” function and probably some support functions.

For every function (that is not a method in a class) describe the function exactly as you described a class method. List all sections for these functions that are listed a class method.

# Organization of Code

This section describes what files will be used to assemble the application. Typically each C++ class has a separate header and body file. Each Java class has a separate body file. Non-class code is usually in files separate from all the class files.

Regardless of how the files are organized, each file listed should include the following items. Note that not every item on the list applies to all files.

* What manifest constants are defined
* What macros are defined
* What global data structures are created
* What global objects are created
* What external variables are created
* What global variables are created
* What functions (including external) are defined
* What other files are needed (included) by this file

# Other Design Considerations

This is the catchall to address other design issues no covered in this outline.