Team MIVC

|  |  |
| --- | --- |
| Project: | MIVC |
| Due: | 9-30-2013 |
| Members: | Piper Chester |
|  | Geoff Berl |
|  | Collin Ferris |
|  | Ty McGinnis-Kennedy |
|  | Allen Thomas |

*Insert table of contents here after document is complete…*

Narrative …how it all works

Describe High level architecture

Give rationale for major design choices made

Show how the design covers specific requirements

Outline how the design reflects a balance among low coupling, high cohesion, separation of concerns, information hiding, the Law of Demeter, extensibility, reusability, etc. (This can include UML, CRC, Sequence charts etc.

UML Class diagrams

Show main classes and interfaces in your design, along with appropriate relationships

Include cardinality as necessary

DO NOT include state or method information, this is to be captured in CRC cards

Indicate pattern participation using stereotypes (i.e. in guillemets << and >>).

Use multiple class diagrams showing from a top level using subsystems down to a detailed view of subsystems.

Class Responsibilities Collaborators cards

Example template

|  |  |
| --- | --- |
| **Class:** Motor |  |
| **Responsibilities:** The overall responsibility of this class is control of the motor  and prevention of unsafe operation. It provides start/stop control as well as the  ability to move the motor to a fixed angular location. The class can drive the  motor through a predefined time sequence of angular locations. Because this  class often needs to execute in a separate thread, many of its methods are  synchronized. The class is also responsible for providing status on the motor in  terms of current speed and angular position. | |
| **Collaborators** | |
| **Uses:**  SpeedEncoder, MotorPowerSwitch,  EmergencyAlarm, SystemTimer | **Used by:**  MotorThread, InletWaterSupply,  SolarPanelPositioner |
| **Author:** J. Smith | |

Descriptions of pattern usages

Example Template

|  |  |  |  |
| --- | --- | --- | --- |
| **Name:** Video Connection | | | **GoF pattern:** Template Method |
| **Participants** | | | |
| **Class** | **Role in pattern** | **Participant's contribution in the context of the application** | |
| VideoConnection | AbstractClass | This class' Setup method is the Template  Method defining the sequence of steps  needed to initialize a video connection. The  method uses abstract and non-abstract  methods defined in the class. The  AllocateHandles method is common to all  initializations and is implemented in this class.  Steps that must be implemented by the  ConcreteClass'es include: gaining access to  the video stream, setting the alert callback,  and initializing the connection's thread. | |
| FirewireConnection | ConcreteClass | This class defines the steps specific to  initializing a firewire video connection. | |
| IPStream | ConcreteClass | This class defines the steps specific to  initializing an IPStream video connection. | |
| **Deviations from the standard pattern:** None | | | |
| **Requirements being covered:**   * The application must setup several different types of video connections. * All network connections follow the same sequence of steps. The implementation * of some steps are common to all connections (allocate a handle), and some specific to the individual connection (access video, set alert, initialize thread). | | | |

Sequence Diagrams

for **at least** two non-trivial features in the implementation. See more details on the [second a under design documentations](http://www.se.rit.edu/~swen-262/projects/DesignProjectGuidelines.html)

The current state of the program

and any known issues

OTHER NOTES: This document should be professional with figure and table numbers as well as section numbers and page numbers.