PROJECT / RELEASE #

Project Design Document

TEAM # + Name

John Doe <jdoe@rit.edu>

Jane Doa <jdoa@rit.edu>

Jill Doi <jdoi@rit.edu>

Jack Dou <jdou@rit.edu>

*Google Doc link to document with version history***:**

https://

# Project Summary

This section provides a brief overview of the project. Do not cut and paste the website description of the project exactly as is. You may use sections of the website description but summarize the project in your own words; a few paragraphs are sufficient.

# Design Overview

The design overview is the narrative that captures the thought process and evolution of the design from the preliminary design sketch through final implementation. The narrative should support how the project design addresses good design principles: separation of concerns, high cohesion, low coupling, dependency inversion via abstractions (interfaces), support for extendibility, etc.

It is equally important to document design decisions that did not go as anticipated, as it is decisions that worked out well. This is an extremely helpful background for future readers of the document to help them avoid solution paths that already had been attempted when extending the project with new or modified features. It is common for some design documents to have an entire section dedicated to “rejected alternatives”. Lastly, it is important to also document “assumptions”. Some of these should be documented here but others may be more pertinent in the corresponding section of the document.

Keep the narrative organized. This section ought not to be one huge paragraph.

# Subsystem Structure

This section provides a graphical model of the subsystems that comprise the application as a whole.

Subsystems are groups of closely related classes. For example, an email program might have subsystems for contact list management, message composition, and message delivery, among many others. In large applications, a subsystem might even have sub-subsystems, though this should not be necessary in this course.

A design goal is to have much lower coupling among classes of *different* subsystems than among the classes *within* a subsystem. Put another way, each subsystem should be highly cohesive in purpose, and the subsystems as a group should exhibit separation of concerns, just as the set of classes *within* a given subsystem should each be highly cohesive, while the set as a whole exhibits separation of concerns.

Draw the subsystems as simple boxes with lines (possibly terminated in arrows) showing relationships between them. Include in each subsystem box a brief narrative of the subsystem's purpose and/or responsibilities. Label important relationship lines with an indication of what the relationship represents.

***Replace all the blue text in this section with a diagram conforming to the description above and any complimentary textual description if appropriate. We suggest you ONLY remove blue sections instructions ONLY AFTER you have confirmed with all contributors that the instructions for a particular section have been fully covered prior to submission.***

***NOTE: the final Rationale section is meant to complement (not Duplicate) the information contained in other parts of the document.***

# Subsystems

This section contains one subsection for each of the subsystems in the high-level diagram above, describing the classes in the subsystem and their relationship. In what follows, ***class*** is a generic term for *concrete classes*, *abstract classes*, and *interfaces*).

Each subsystem has a CRC table naming each class in the subsystem, the class's responsibilities, and the class's collaborators (classes in this or other subsystems).

The table is followed by a UML class diagram showing inheritance (generalization) relationships between classes and interfaces and associations between objects in these classes.

***In the actual document, this entire blue section should simply be deleted.***

## **Subsystem name**

Fill in a CRC table like those below for each class or interface in the subsystem; feel free to copy, paste, and edit our examples.

The example tables, based on a hypothetical library media database system, illustrates all the possible collaborations between classes and objects that you must record for your design. ***Remove this blue paragraph from your document.***

| **Class** MediaCollection | |
| --- | --- |
| **Responsibilities** | Support access to the media in the library collection.  Add, find, delete an existing media in the collection.  Provide virtual collection for the entire library consortium collections. |
| **Collaborators**  **(uses)** | Media - the basic type for all different media in the collection.  network. Consortium - consortium communication (network subsystem). |

| **Class** Media (interface) | |
| --- | --- |
| **Responsibilities** | Provide a generic interface to all media types in the library.  Includes the media name, a unique media id.  Can retrieve a formatted media description. |

| **Class** Book | |
| --- | --- |
| **Responsibilities** | Represents a book in the collection.  Provides access to author, title, genre, and ISBN for the book.  Provides access to other attributes of a book (e.g., number of pages). |
| **Collaborators (inheritance)** | Media |
| **Class** DVD | |
| **Responsibilities** | Represents a DVD in the collection.  Provides access to title, genre for the DVD.  Provides access to other attributes of a DVD (time, cast, etc.). |
| **Collaborators (inheritance)** | Media |

Each subsystem will also include an ***in-line*** UML class diagram of the system. In the simplest case the interfaces and classes will simply have class boxes with the appropriate name. In the final document, and preferably in intermediate documents, classes will also include the public methods provided by the class.

If a class in this subsystem collaborates with a class in a different subsystem, simply include link to a box with the ***other subsystem's name***.

## **Subsystem name**

As described above for the second, third, etc. subsystems.

# Sequence Diagrams

## **Description of labeled Sequence diagram #1 and (what feature / operation / scenario the diagram shows).**

Include a sequence diagram that corresponds to the description above. Feel free to use a stick figure "pretend" object for whatever outside agent triggers the operation. In addition, it is acceptable to rotate your diagram to landscape orientation that makes the diagram easier to read.

Repeat as many times as necessary:

## **Description of labeled Sequence diagram #N (what feature / operation / scenario the diagram shows).**

Same as above. Provide diagrams for each of the basic operations in the problem description and/or those operations specified from discussion with your instructor/stakeholder.

# Pattern Usage

There will be a subsection for each pattern you use in your design (***including*** those that may be required in the project description or by your instructor).

## **Pattern #1 Name**

Create a table like the one below listing the generic "roles" from the pattern description along with the specific class(es) of objects in your design that are associated with each role. For instance, if we are using the Observer pattern in an Alarm system where one or more alarm reporting objects observe one or more sensor objects, we might see a table like the following:

| **Observer Pattern** | |
| --- | --- |
| **Observer(s)** | SoundAlarmReporter  ADPAlarmReporter  AlarmReporter911 |
| **Observable(s)** | TemperatureSensor  MotionSensor  DoorOpenSensor |

## **Pattern #2 Name**

Just repeat the information above for the second and following patterns.

# RATIONALE

Be sure to incorporate all major DESIGN and ARCHITECTURAL decisions and reasons for pursuing them. It helps to add entries with a time stamp (e.g., 9/27/2023 –We decided to..) >