Two-Year Cycle Group Calendar

by

Praise Daramola, Praise24

Report submitted on 22 October 2019

EE 333 – Engineering Programming Using Objects

Department of Electrical and Computer Engineering

The University of Alabama at Birmingham

Abstract

This purpose of this project was to explore the object-oriented design process through development of a program which models a real-world system. This report discussed several object-oriented approaches and methods for solving problems. This report also explored several problems encountered and features discovered through the development process.

**Table of Contents**

[Abstract Error! Bookmark not defined.](#_Toc507684821)

[Introduction 1](#_Toc507684822)

[Project definition 1](#_Toc507684823)

[Constraints 1](#_Toc507684824)

[Goals 1](#_Toc507684825)

[Features (in MoSCoW list format) 1](#_Toc507684826)

[Applicable Standards 2](#_Toc507684827)

[Design 2](#_Toc507684828)

[Appropriate for Object Oriented Approach 4](#_Toc507684829)

[Design Decisions 4](#_Toc507684830)

[Object Oriented Design 6](#_Toc507684831)

[Discovery and USE OF ONLINE INformation 9](#_Toc507684832)

[Debug 9](#_Toc507684833)

[Results 9](#_Toc507684834)

[Discussion 10](#_Toc507684835)

[Conclusions 10](#_Toc507684836)

[References 10](#_Toc507684837)

[Appendix 11](#_Toc507684838)

Introduction

The goal of this project is to apply object-oriented design approach to development of a group calendar that works on a two-year cycle. Object-Oriented design is useful for modelling real-life systems on a computer. There are several Object-Oriented Programming languages including C++, Ruby, VB .NET, Python. For this project, I used the JAVA programming language. Also, I demonstrated knowledge of development tools within the NetBeans IDE by making appropriate use of these tools during my development process.

Project definition

The program to be designed in this project is a calendar which works on a two-year cycle. This calendar will be able to store processes which consist of multiple activities to be performed on specific dates. This calendar will also be able to assign different processes to different processes to different roles. The calendar should also allow the user to extract information related to several processes including the roles assigned (and any sub roles if needed) to that process, the date the process is to be executed, and the description of that process.

# Constraints

* CO-01: The system will be developed using the Java Programming language
* CO-02: The problem will be solved using the object-oriented programming design approach.
* CO-03: The modelling classes will not contain any I/O operations

# Goals

* GO-01: The program must be user friendly.
* GO-02: The program must be delivered by the due date.
* GO-03: The program must comply with standards associated with object-oriented design.
* GO-04: The program must adhere to rules described in the project description.

# Features (in MoSCoW list format)

* Must-01: Be able to search and retrieve information about processes
* Must-02: Keep track of Processes, roles related to processes, and dates to be executed.
* Must-03: Allow for the creation of new Processes
* Must-04: Allow for the assignment of processes to different roles.
* Must-05: Allow for the creation and deletion of processes.
* Should-01: Allow for the creation of new roles
* Should-02: Allow for manual addition of new sub roles to existing roles.
* Should-03: Allow the user to edit current existing processes.
* Should-04: Be presented in a meaningful sorted order.
* Could-01: Include a Graphical User Interface.
* Could-02: Inform roles about upcoming processes to be executed.
* Could-03: Inform roles about a new process addition to the calendar.
* Won’t-01: Automatically assign sub roles to roles.

# Applicable Standards

* STD-01: All files will comply with the acceptable documentation standards outlined on the Javadoc website.
* STD-02: All classes will comply with the format outlined in the EE 333 documentation page.
* STD-03: The programmed will be debugged and tested using the IDE debugger tools and other debugging methods to ensure the programs functionality.

Design

After receiving the assignment, I read through the entire assignment description. After my initial reading, starting from the beginning of the problem description, I began evaluating several possible design approaches based on my initial understanding of the problem. While reading through, I underlined and highlighted several keywords I believed would be important when working on a solution. I also took note of several potential objects that could be implemented if my solution could be solved using an object-oriented approach.

Next, I evaluated how the problem to be implemented would imitate a real-world calendar. To do this, I examined how I interact with my real-life calendar. I took note of each step I took throughout my interaction with the calendar. Some of the interactions include:

1. Viewing a task associated to a date in my calendar:

* Opening the calendar
* Searching for the desired year.
* Searching for the desired month.
* Searching for the desired day.
* Identifying the tasks/events assigned to that day.
* Exited the calendar.

1. Adding a task to my calendar:
   * Opened my calendar.
   * Searched for desired year.
   * Searched for the desired month
   * Searched for the desired day.
   * Selected the desired day.
   * Added the desired date.
   * Selected the duration of the task.
   * Selected when to be reminded about the task.
   * Exited the calendar.
2. Viewing a date associated with a task.
   * Opened my calendar.
   * Selected the search bar.
   * Typed the title of the task I was interested in.
   * Clicked search.
   * Identified the date [day, month, year] associated with the task.
   * Exited the calendar.

There were other recorded interactions associated with my calendar, however, I found the interactions listed above to be most relevant to this report.

Next, I began to identify features and behaviors that each object would exhibit. This was to identify my potential classes and determine the characteristics of these classes. I also evaluated possible inputs that would be required to create each object and the data that each class would store. I also evaluated the interaction between different classes. This helped me identify other possible data points that will be contained within an object and possible dependencies of each object.

After brainstorming possible classes, I eventually settled on the most relevant classes which include: A Calendar class to keep track of the date, processes associated with each date, role and activity. A Role class to keep track of the existing roles and their sub roles and the activities assigned to each role. An Activity class to keep track of each created activity, the roles associated with that activity, and the date associated with that activity. A Date class to keep track of the dates assigned to processes (this decision was made to account for date pairings such as “April/May”, “January/February”, etc.). I also evaluated possible commands and queries associated with each class. For example, an ‘addSubRole’ method for adding a sub role to a role, an ‘addActivity’ method for adding an activity to the calendar.

Next, I evaluated possible options for storing Processes: firstly, I determined that I could create a processes class to store a set of activities and assign a process object to each date in the calendar. Secondly, I considered creating an ArrayList named process which stores activities in either my Calendar class or Date class. Also, I decided I was going to create a file which stores the calendar history.

Although several potential approaches were evaluated, these approaches were used only as a blueprint for the final coding process. As the development process progressed, several changes were made to account for roadblocks with respect to code capabilities. Other alternative designs were also evaluated during the development process. These alternatives/changes were reflected on the final deliverable.

# Appropriate for Object Oriented Approach

For a program to be appropriate for an object-oriented approach, the model to be represented by the program should be able to be physically represented by objects in the real world. In other words, the model should have certain features, states, and behaviors. Upon evaluation of the problem description, the assignment contained several features that can be represented by a real-world system. For example, a feature of a calendar is that it has several dates-[feature] and can be used to monitor-[behavior] progress-[state], a process-[state] can be added to a physical calendar-[object] manually, an activity-[feature] can be assigned to a role-[object], a role-[feature] can be assigned to a person. As a result of this evaluation, it was determined that an object-oriented approach is appropriate for this project.

# Design Decisions

AL-01 How does the program store activities with respect to the calendar?

AL-01A Create a processes object for each date object, activities are stored in processes.

AL-01B Create a process ArrayList in each date object which stores activities

AL-01C Create a process ArrayList in the calendar object which relates to each date. This stores activities.

ALD-01C: My reason for choosing this solution is because it gave the calendar a way to track the activities involved with the calendar. This method simplified the activity search with respect to year and date. This is because it allowed me to filter the results based on the user input without having to create a new object.

AL-02 How does the user interact with the program?

AL-02A GUI for search/creation of activities, roles, and dates.

AL-02B Commands through the command line.

ALD-02 A I decided to create a GUI for the user to interact with the system. Initially, I assumed this would simplify my input process, however, it made development time significantly longer because I had to learn how to use JavaFX. Although the development time was significantly longer, creation of a GUI for user interaction would make it easier to use.

AL-03 Does the program assign ID’s to each activity and role?

AL-03A Yes, and gives the user the ability to search by ID.

AL-03B Yes, but only uses the ID internally.

AL-03C No, keep track of and roles using other relationships.

ALD-03 B Although I implemented this feature in the program, there is no real use for the activity ID feature feature currently, however, I decided to implement it incase any modifications are made in the future which requires an ability to track an activity. The role ID feature, however, is used to monitor roles created. Because if one role has the same name as another role, it is the same role, so it should have the same index in an arrayList of existing roles. This is to ensure that when the role is modified, it’s the role at that index of existing roles is modified.

AL-04 How to search for an activity, role or set of processes in the calendar?

AL-04A Overloaded getActivities(…) methods which take in parameters like activity, role, date, year.

AL-04B Single getActivity(…) method which takes in an activity parameter.

ALD-04 TBD

AL-04 How to search for an activity, role or set of processes in the calendar?

AL-04A Overloaded getActivities(…) methods which take in parameters like activity, role, date, year.

AL-04B Single getActivity(…) method which takes in an activity parameter.

ALD-04 TBD

AL-05 How will the history be stored?

AL-05A Use JSON format and use a JSON parser to read it.

AL-05B Use CSV format and create a .csv parser.

ALD-05 TBD

# Object Oriented Design

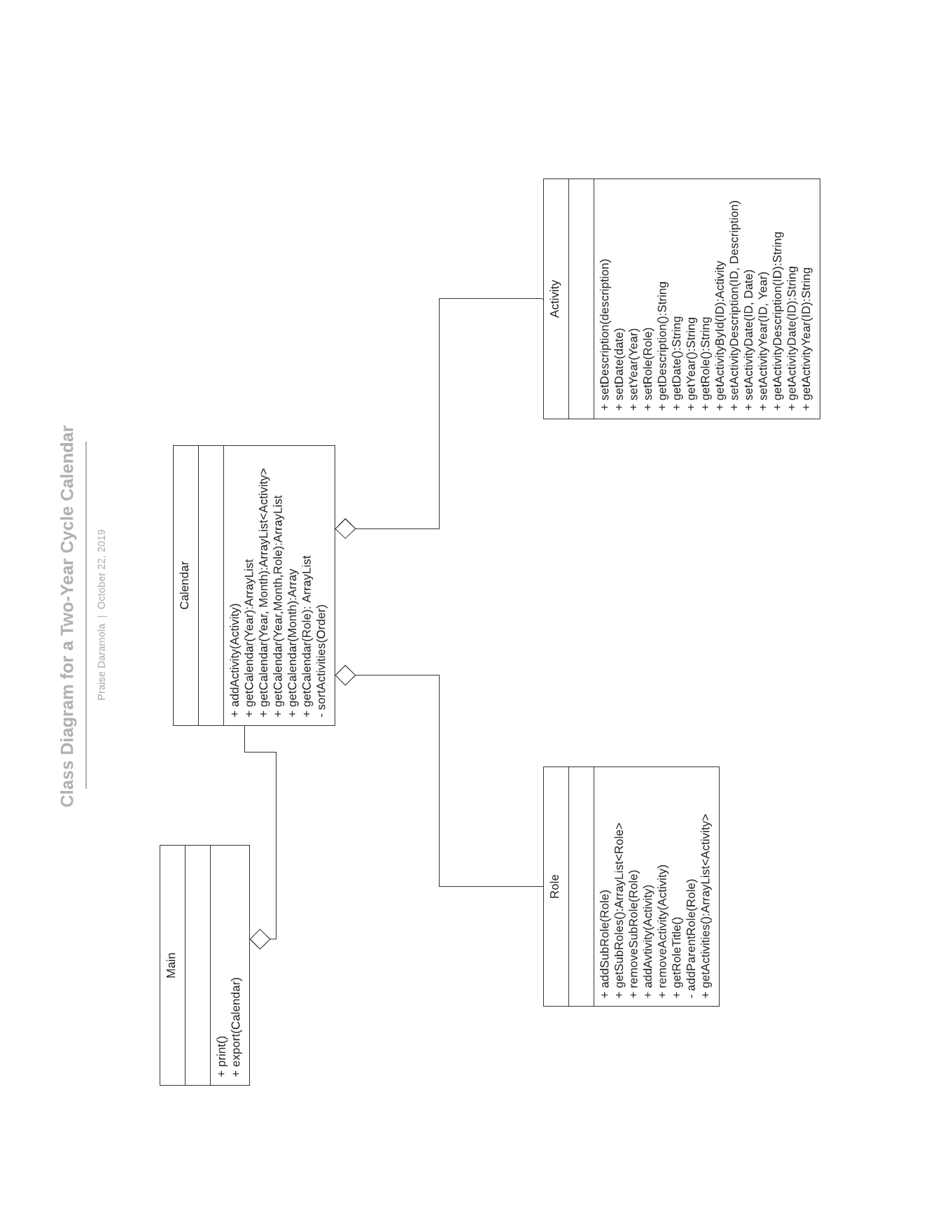


Fig 1. Class Diagram for the two-year cycle Diagram.

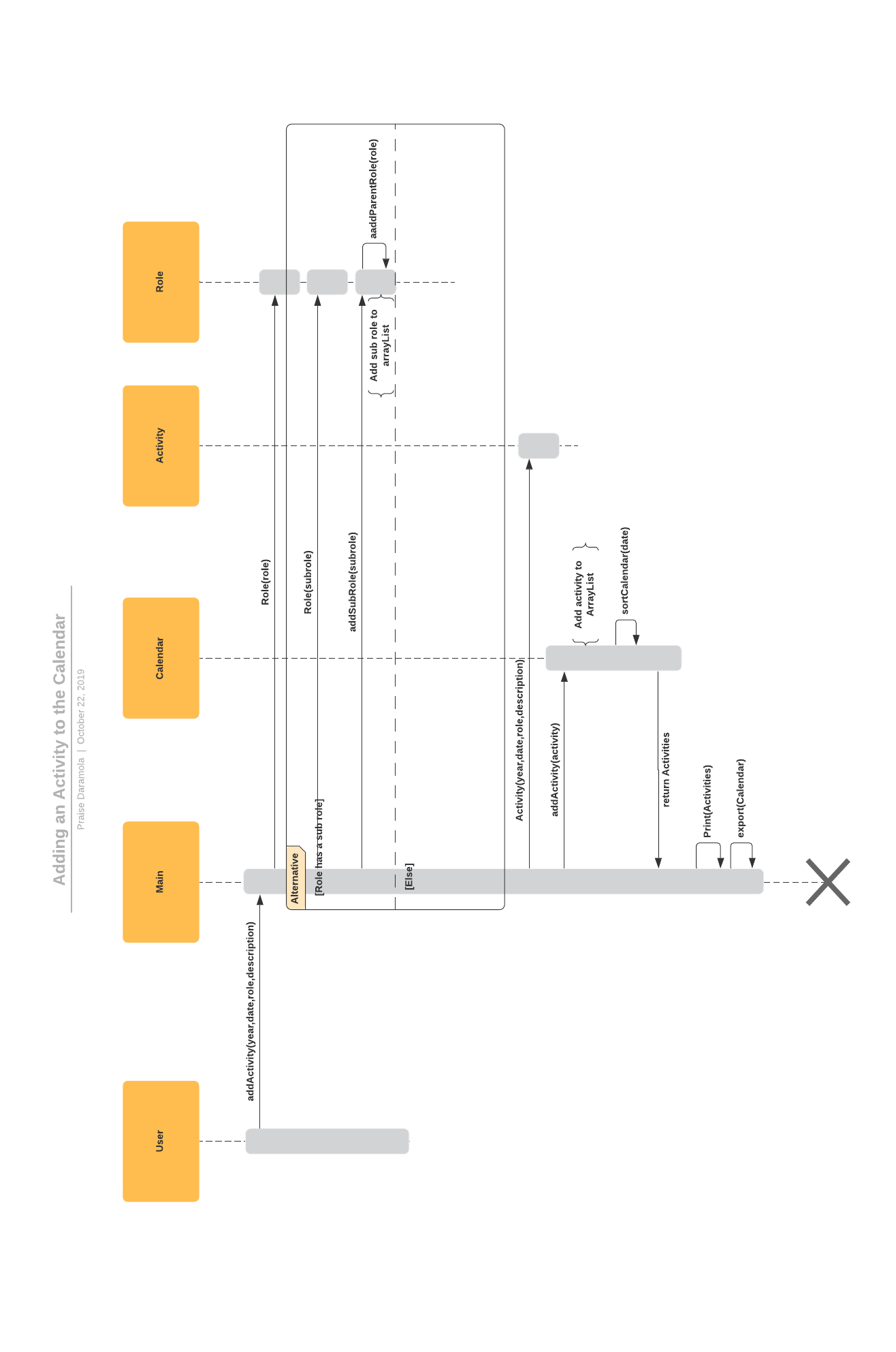


Fig 2. Interaction diagram for adding an activity to the calendar.

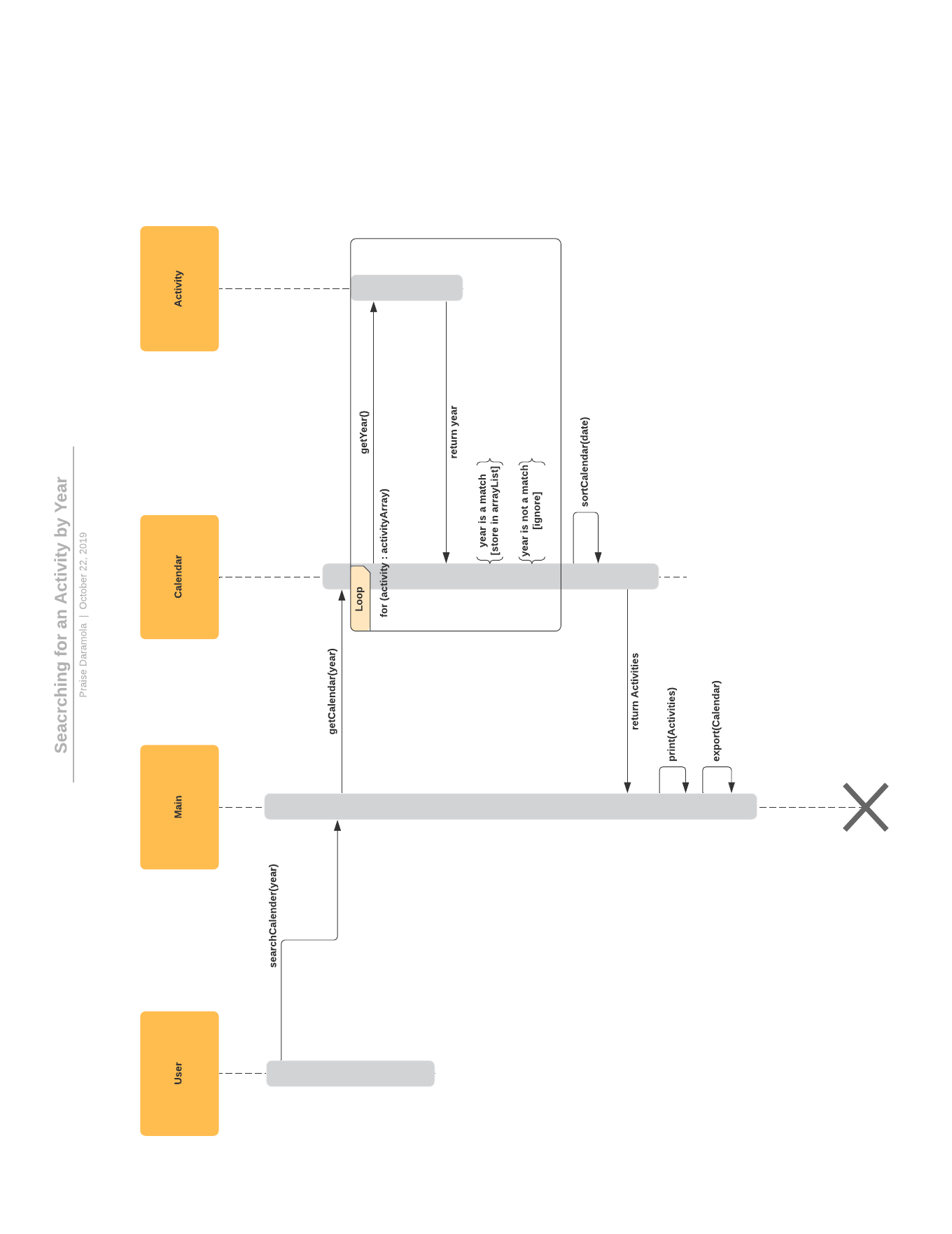


Fig 3. Interaction diagram for searching for an activity by year.

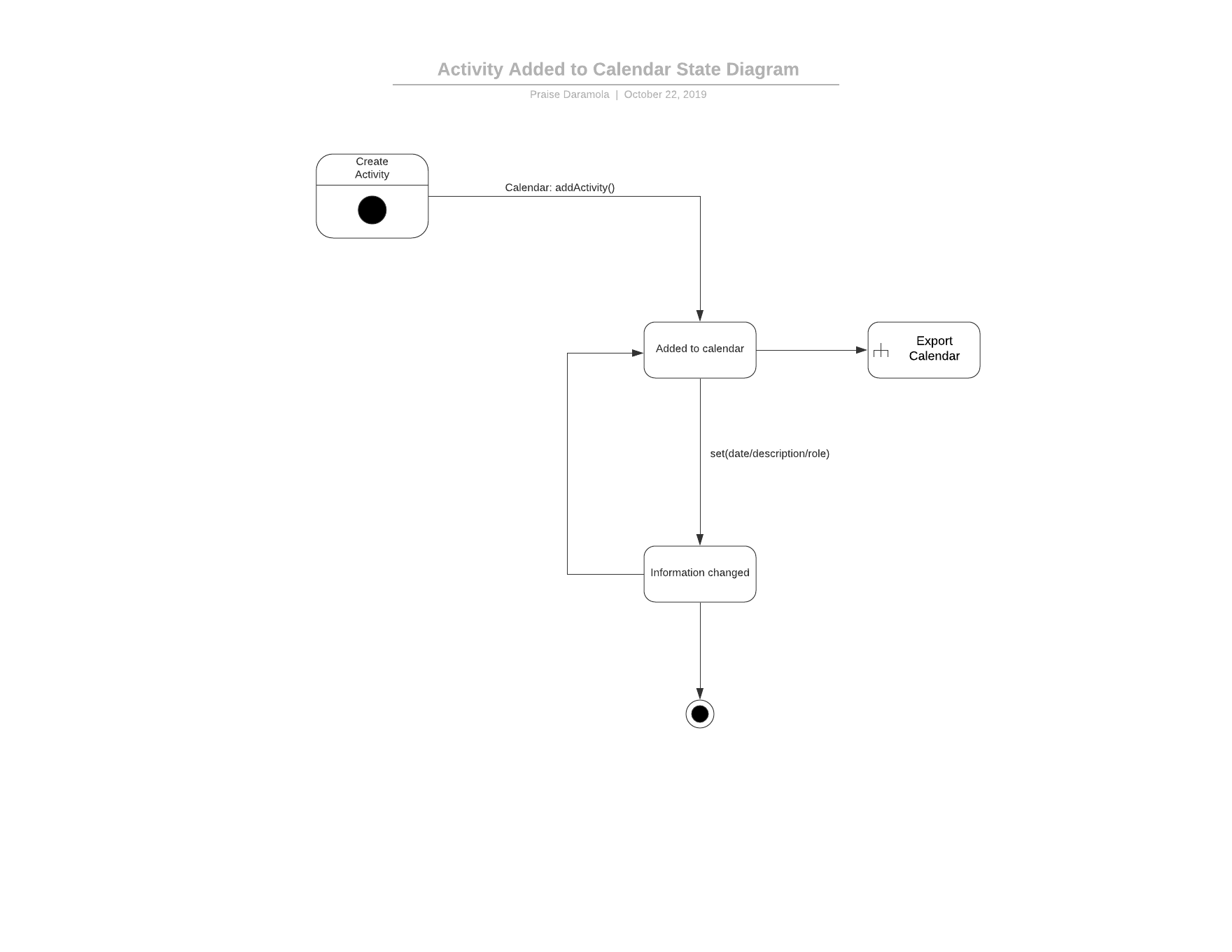


Fig 4. State diagram for a created activity.

Discovery and USE OF ONLINE INformation

TBD

<<Discuss at least one activity where you had to find additional information online to complete your design or prototype.>>

Debug

TBD

<<Discuss one difficult problem you had in implementation and the steps you took to debug the program. This section should demonstrate the developer’s ability to debug software.

Problem description.

Steps to debug.

Resolution.>>

Results

TBD

<<How was the program tested? Given the results of running the program annotating things so that one can understand what is being demonstrated.>>

Discussion

TBD

<<The discussion considers the results and describes how well the program achieved its purpose addressing all constraints, goals and standards. It is written in the past tense. Avoid trying to spin things.

Work that is not that of the report author and material such as handouts, assignments, papers, books, websites, and interviews that have been used in carrying out the activity should be referenced at the end of the report. If there is an assignment document, then it should be referenced. Works are cited in the report should be indicated by numbers placed in brackets before punctuation. For example:

* (a paper) … for determining loss coefficients [1].
* (a book) … for determining loss coefficients [2].
* (an interview) … for determining loss coefficients [3].
* (a URL with date viewed) … for determining loss coefficients [4].

In any document there should be only one method of citation used and it should be used consistently. The IEEE reference format [5] is to be used in this course. Citations should be listed using IEEE format [5] in the references section. Citations should be numbered in order of their first appearance in the text as follows. >>

Conclusions

TBD

<<Conclusions should be broad, closing statements that evaluate the design and development process, results and their significance including suggested changes for the next design project.>>

References

<<

1. K. L. Segwoth, C. Hang, and J. T. Barton, “Porosity malformation in modified and unmodified Fe-Si alloy castings,” *AFSP Transactions*, vol. 102, pp. 276-361, July, 1974.
2. H. W. Dann and W. E. Steelye, *Experimentation and Uncertainty Analysis for Engineers*. 1st edition. New York: John Wiley & Sons, pp. 67-80, 1989.
3. R. I. Peace, Informal communication, Professor, Department of Engineering, Great Midwest University, November 14, 1997.
4. (Basic Book/Monograph Online Sources) J. K. Author. (year, month, day). *Title* (edition) [Type of medium]. Volume(issue). Available: <http://www.(URL>)
5. (2006, May, 22) *IEEE Transactions LaTeX and Microsoft Word Style Files* .<http://www.ieee.org/portal/pages/pubs/transactions/stylesheets.html> >>

Appendix

<< Extensive reference material, tables, figures, handouts, source code or calculations should be presented in the appendix, and **NOT** in the text of the report. An **APPENDIX** may be presented as separate files if it is not feasible to integrate the listing into the report document, which may be the case for source code. >>