University of Toronto Mississauga

CSC207 Fall 2022

Design Document

AUTHORS

Jacob Youssef
Krish Patel
Khubaib Ahmed
Shivank Goel

GITHUB TEAM: **The Hamburglars**
GITHUB URL: **https://github.com/KrishPatel13/The-Hamburglars**

December 06 2022

Contents

2 Project Identification

"Chronos" was developed with the intention of creating a flexible time management tool that encourages goal accomplishment and enables for simple adjustment. Chronos is designed for students who need assistance in striking and maintaining a healthy work-life balance while remaining adaptable to changing situations.

Chronos' calendar system will meet this demand by enabling users to set up events that can be modified later. Chronos will allow users to design prizes for themselves that will only be "awarded" if specific events are fulfilled, further encouraging work-life balance. This will help users remember to take time for themselves and prevent burnout from relentlessly grinding.

Users will find Chronos useful since the project blends life-gamification and motivational aspects with the schedule-making functionality of other applications like Notion. Chronos will assist users in maintaining a healthy work-life balance and developing themselves along the way by merging the two aspects and generalising their application to any issue like school or exercise.

3 User Stories

Name	**ID**	**Owner**	**Description**	**Implementation Details**	**Priority**	**Effort**
Create Events	1.1	Krish	As a user, I want to create events that cover a certain time frame and cover a certain goal.	Create an Event class that stores essential details such as the time and a description of what the event represents.	1	2
Calendar View	1.2	Khubaib	As a user, I want to see my schedule in the format of a calendar where I can place my events.	Make a GUI that allows the user to pick and view any date.	1	2
Goals and Rewards	1.3	Jacob	As a user, I want to give incentives to achieve my goals by being able to set my own rewards for completing certain tasks.	Goals will be observer objects that are notified when an event is completed.	1	2
Edit Color, Theme of the Display	1.4	Shivank	As a user, I want to be able to more clearly see my tasks by editing the colors of my schedule to suit my needs.	Add a Color Picker from JavaFX that can be accessed by the user to edit the colour of the background and font.	1	3

Name	**ID**	**Owner**	**Description**	**Implementation Details**	**Priority**	**Effort**
Display and complete events	1.5	Khubaib	As a user, I want to be able to see my events in a concise list, sorted by date, where I can select an event to complete the event.	Use a JavaFX ListView to display events of the date that is selected on the calendar. Also, add a button that marks events as complete.	2	1
Editing all Details of Events	1.6	Krish	As a user, I want to freely edit the names, descriptions, times and point values of all my events.	Make a GUI that will allow the user to edit the attributes the event that they selected on the calendar.	2	1
Saving and Loading	1.7	Jacob	As a user, I want to have my progress saved when I make changes, so I can later continue where I left off.	The calendar model, events, goals and time behaviours will be serializable. The memento pattern will be used to save and load progress automatically.	2	1

4 Software Design

4.1 Design Pattern 1: The Observer Pattern

Overview: This pattern will be used to implement User Story 1.3. The user will be able to set goals to earn badges. These goals will be observers that track when events are completed.
UML Diagram: Refer to Figure 1, below.

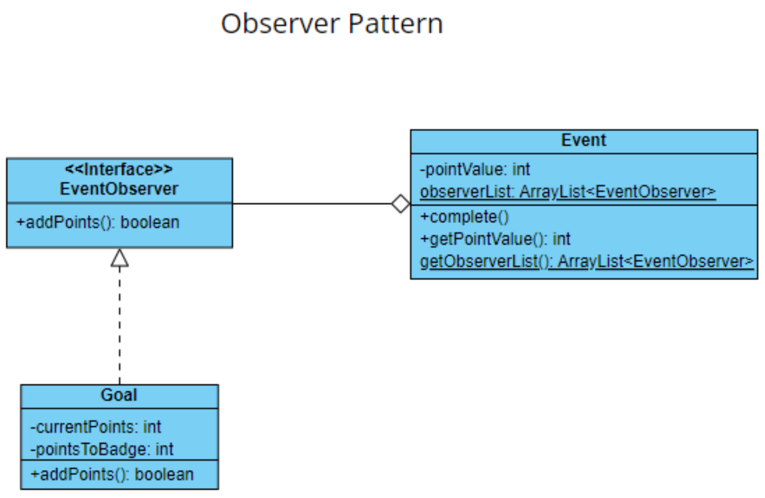

Figure 1: The Observer Pattern

Implementation Details: The UML diagram outlines these main components:

1. The EventObserver interface, which has the addPoints method.
2. The Goal class, which extends the Observer interface. This acts as a concrete observer.
3. The Event class, which notifies observers using the complete method.

When the user creates a new goal, this goal will be added to the Event class' static observerList. After the user indicates they have completed an event, that event will notify all observers with the complete method. Each event has a point value which gets accumulated in the goals' currentPoints attribute upon completion. When the value of currentPoints reaches the value of pointsToBadge for a certain goal, the user is awarded a badge and the goal is considered completed.

4.2 Design Pattern 2: The Strategy Pattern

Overview: This pattern will be used to implement User Story 1.1. Specifically, the strategy pattern will assign different behaviours to different types of events.
UML Diagram: Refer to Figure 2, below.

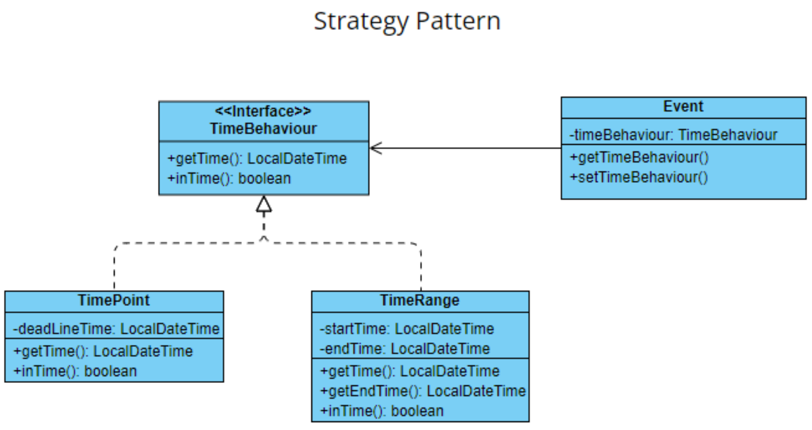
Strategy Pattern
<<Interface>>
TimeBehaviour
+getTime(): LocalDateTime
+inTime(): boolean
Event
-timeBehaviour: TimeBehaviour
+getTimeBehaviour()
+setTimeBehaviour()
TimePoint
-deadLineTime: LocalDateTime
+getTime(): LocalDateTime
+inTime(): boolean
TimeRange
-startTime: LocalDateTime
-endTime: LocalDateTime
+getTime(): LocalDateTime
+getEndTime(): LocalDateTime
+inTime(): boolean

Figure 2: The Strategy Pattern

Implementation Details: The UML diagram outlines these main components:

1. The TimeBehaviour interface, which defines the abstract methods getTime and setTime.
2. An Event class which has two subclasses Block and Deadline, and contain a TimeBehaviour object.
3. TimeRange and TimePoint, which implement the TimeBehavior interface.

All events will contain objects that implement the TimeBehaviour interface, either a TimePoint or a TimeRange. TimePoints define an event to have one point in time, while TimeRanges have a separate start time and end time. This allows for different events to have different behaviours, while they are still instances of the same class.

4.3 Design Pattern 3: The Memento Pattern

Overview: This pattern will be used to implement User Story 1.7. The application will automatically save and load progress, without revealing the underlying implementation to the user.
UML Diagram: Refer to Figure 3, below.

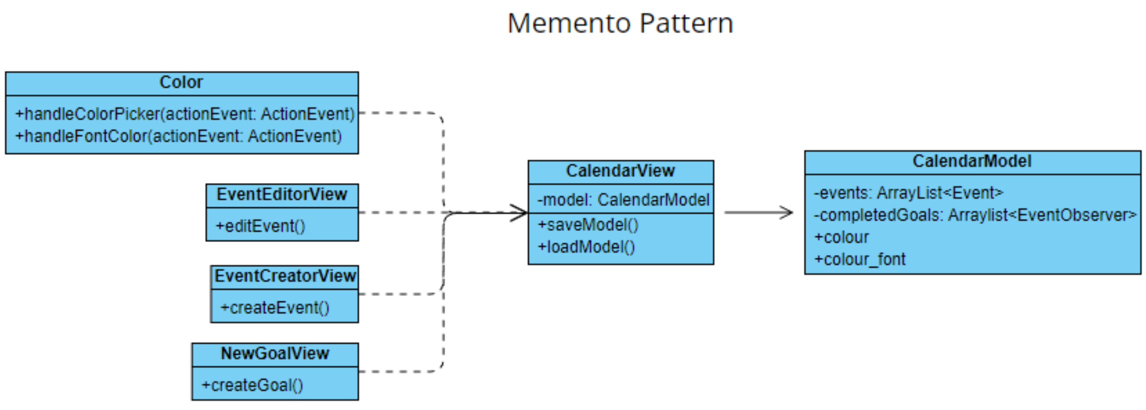

Figure 3: The Memento Pattern

Implementation Details: The UML diagram outlines these main components:

1. The CalendarView, which is the main GUI that the user interacts with.
2. The CalendarModel, which stores key data about the application such as the list of all events.
3. Color, EventEditorView, EventCreatorView and NewGoalView, which are GUIs used to add or edit the CalendarModel's data.

Every time the user makes a change to their data, such as creating an event or changing the theme, the CalendarModel is saved to the file save/model.ser. Upon starting the application, the previous CalendarModel is loaded automatically, so the state of the CalendarModel is rolled back to the one that the user was previously interacting with.

4.4 Design Pattern 4: The Singleton Pattern

Overview: This pattern will be used to implement User Story 1.2. The CalendarView will be a singleton, ensuring that there is only one CalendarView and making easy for the entire application to access CalendarView's instance.
UML Diagram: Refer to Figure 4, below.

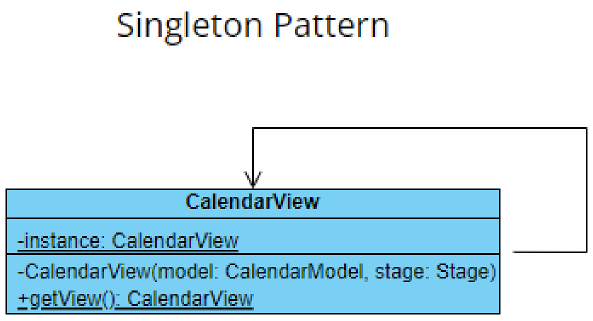

Figure 4: The Singleton Pattern

Implementation Details: The UML diagram outlines these main components:

1. The CalendarView class, which is a singleton containing a private constructor and one static instance.

The CalendarView has a private constructor, which ensures that other classes cannot instantiate additional instances of the class. CalendarView also has the static method getView, which returns the static instance of CalendarView. This is useful as the method allows other classes to access and interact with the CalendarView freely.

5 Expected Timeline

The major milestone the team have attained so far is finishing up the Phase 01 of project, as well as developing an overview of our implementation.

Project Timeline			
Task	Start Date	Number of Days Required	Completed?
Project Identification	October 18, 2022	2	Yes
Develop Broad Plan	October 20, 2022	4	Yes
Mentor TA Approval	October 30, 2022	1	Yes
Identifying Design Patterns	October 31, 2022	2	Yes
Develop detailed plan	November 02, 2022	2	Yes
Designing UML Diagrams	November 08, 2022	3	Yes
Mentor TA Review	November 12, 2022	1	Yes
Develop an overview of Implementation	November 13, 2022	6	Yes
Implementing Sprint 1 features	November 15, 2022	7	Yes
Testing Sprint 1 features	November 22, 2022	3	Yes
Merging and integrating Sprint 1 features	November 25, 2022	2	Yes
Implementing Sprint 2 features	November 27, 2022	6	Yes
Testing Sprint 2 features	December 03, 2022	2	Yes
Merging and integrating Sprint 2 features	December 04, 2022	2	Yes
Final testing, debugging and fixes	December 05, 2022	2	Yes