Project Title: DigiPlanner

Authors: Diego Abanto Ibarguen, Brian Dion, Sara Alam

Course: COSC480E Intro to UI

Instructor: E. Fourquet

Statement

The DigiPlanner app is a digital form of a store-bought planner, with different sections for different purposes like task-planning, habit-tracking, short notes etc. Benefits beyond a paper planner include:

i. the aesthetic value added by digital images and gifs and

ii. a compact, paperless way to store the information recorded.

The user interface has been built using javafx components and the backend has been written in java.

Technical Outline

The app consists of multiple java classes, 4 of which are **Spreads**. A Spread is a layout used to plan and organize certain aspects of one's life in a journal. Each of these 4 objects correspond to a display in the app. They create and maintain instances of their own javafx components. These 4 classes have additional helper classes. One class integrates each of their own java classes to help them maintain their data. The main `DigiPlanner.java` class contains 12 instances of `SpreadBundle.java` which packages 4 distinct instances of these spreads. File i/o code is written separately for all classes. The following describes each java class in more detail.

1. `SpreadBundle.java`: This class creates an object that bundles the aforementioned objects together for a specific month. This does not hold any significant event-generating component. The following class files `ToDoMonth.java`, `Tracker.java`, `Journal.java`, and `MonthlyHome.java` contain the code for displaying the relevant spread.

2. `ToDoMonth.java`: This class creates and maintains the TableView of tasks added to the user. It also contains the Textfield for adding new tasks.

    - `ToDoList.java`: This class contains an `ObservableList` that is used to display the tasks for a certain day.

    - `ToDoTask.java`: This class contains a `BooleanProperty` and a `String` that holds whether the task is completed and its details.

3. `Tracker.java`: This class creates and maintains the colorable labels for each day of the month on a flowpane resembling a calendar.

    - `TrackerList.java`: This holds the drop down menu that is the main way to navigate between trackers of the month. It also holds the color slider. This slider allows the user to pick a color

based on how they performed a certain task, and color in a particular day's label on a tracker calendar.

4. `Journal.java`: This class will create and maintain a GridPane with: A smaller GridPane for journal entries and page flipping, a text area for adding new entries to the journal book in the previous GridPane.
    - `JournalEntry.java`: A supporting class that contains the content of each journal entry from Journal.java. It creates a container (TextArea) with the text, and a property with the date and time an entry was created (using the Date data type).
    - `JournalList.java`: This class will create and maintain the journal objects for each day of the month and store them in a HashMap to link to specific dates. The dates will be selected from the calendar navigator.
5. `MonthlyHome.java`: This will be an introduction to the month, displaying graphs that summarize the tracker information and possibly the task-completion rate for the month. The calendar view would look as follows:
    - The "<<" and ">>" buttons navigate to the previous and following year's tabpanes respectively. The "<" and ">" buttons navigate to the previous and next month's tabpanes respectively.
6. `DigiPlanner.java`: This class creates a SpreadBundle object for each month and the calendar view that allows the user to navigate between days of the year. Part of the remaining tasks for the project include adding a save button to the GUI through this class.

Aesthetic Elements

1. **CSS** The program uses one main css file that affects every different spread within the planner. Specific widgets or sections are given a unique style to fulfill the aesthetic and practical needs (e.g. transparent text-area for journal pages, gradient for tracker slider, trash can image for to-do list buttons).

- There are 4 additional css files that set the background color and gif based on the season (see Gifs below):

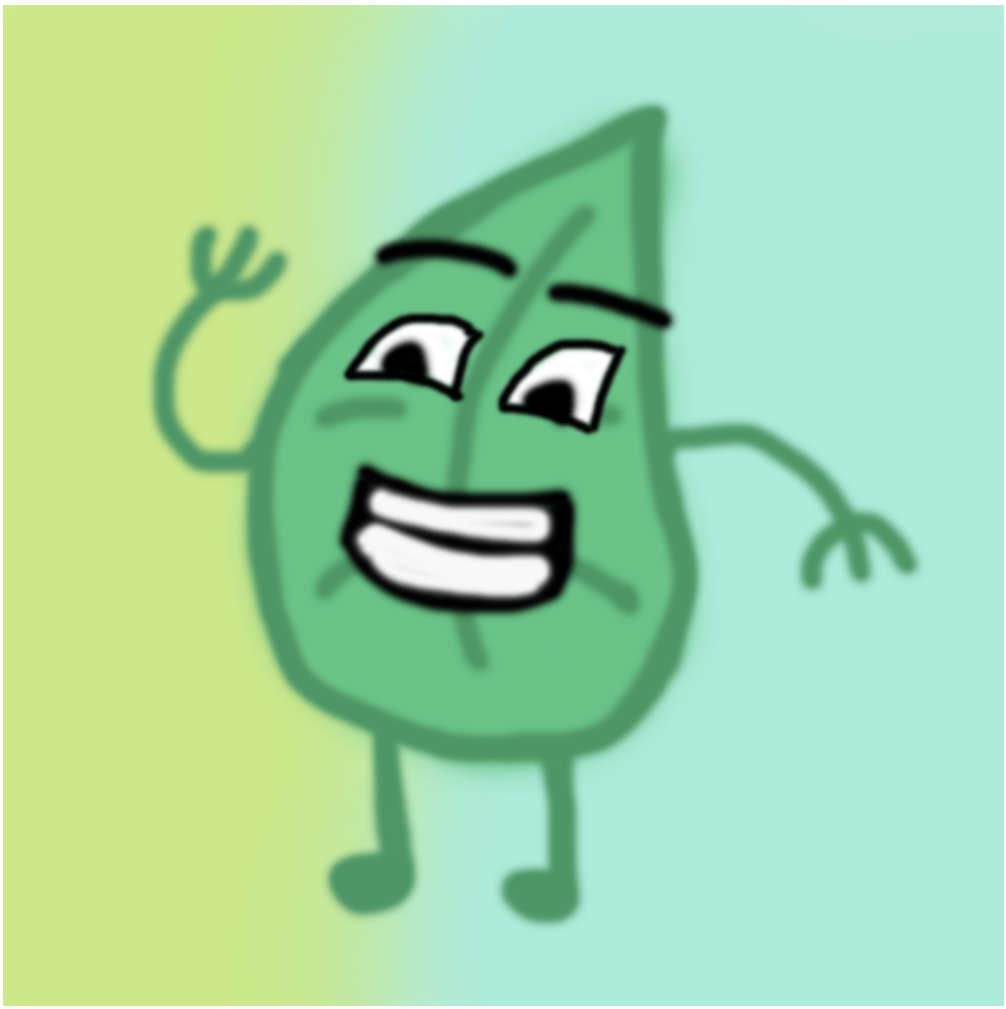
```
snow.css - Winter (December-February)
pink.css - Spring (March-May)
green.css - Summer (June-August)
burnt.css - Fall (September-November)
```

2. **Colors chosen based on color theory** Two color theory methods were employed to choose aestheticaly pleasing colors: Analagous and split complementary. Analagous chooses colors in a similar color region, while split complementary chooses colors complementary (on the opposite side of the color wheel). Based off of the mascot colors (Proposal Objectives Pending sections 6), two colors were chosen for each mascot iteration.
3. **Gifs** Gifs of floating bubbles populates the region behind the tabpane on the right side. This is intended to create a relaxing view. There are four gifs, one for each season (3 months).

4. **Monthly headings** Monthly headings were decorated with leaves or clouds based on the season.
5. **Task completion graphs** Line charts for the task completion rate obtained from the todolists of the month are displayed on each month's home tab.

Proposal Objectives Completed/Ommited/Pending

1. **Create a global calendar view** to allow the user to navigate based on year, month, and day with a menu on the left hand side.
    - The colors of the current calendar do not match the colors of the backgrounds. We are currently evaluating whether to make the color scheme for the calendar change in the same fashion, or to use a constant color palette regardless of the background.
    - Objectives 2-4 refer to tabs in the tabpane of every SpreadBundle object (there will be one per month).
2. Have a **tab called journal** that displays entries on the image of a book.
    - The goal for making it look like a page is flipped was ommitted due to difficulty. Instead, a task we have yet to complete is to play an audio of a page flipping when the user moves between journal entries for a day. The entries will also be removable and editable for the final version. There has been challenges with loading a handwritten font for the journal pages, but it will be solved through the use of built-in javafx methods instead of css.
3. Have a **tab called tracker**, with the calendar view of labels that the user can recolor to rate their day based on 5 different metrics: water, workout, sleep, stress, and study.
4. Have a **tab called toDo** which allows the user to have a to do list running for each day. This list allows the user to check off certain tasks as well as be deleted with a trash can shaped button.
5. **Make the color picker a gradient** formed using different hues of the same color. Allow the user to pick from many different hues. This was done using a slider.
6. **Have a tab called home**, which allows the user to view a summary of a certain month’s status in the form of a graph. The goal to add buttons for the other tabs was omitted since we use a tabpane. The tabpane sufficiently achieves the goal for navigating between views of the journal, tracker and todolist sections of the month.
7. **Make images of a mascot corresponding to each of the five trackers** water, workout, stress, study and sleep.
    - This task is still incomplete but we have made considerable progress on it. Four distinct leaves were created for the mascot, each representing a season. The background colors for these mascots were picked using color theory (see Aesthetic Elements section 2). The following are the four base mascots for Winter, Spring, Summer, and Fall respectively as well as their hand picked background colors.



8. **Make images of a mascot to display for each end of the tracker color slider.**
    - Also create images of the mascot speaking, for display when the program displays text to the user like errors, response to checking off tasks on the todo-list etc. NB: This task is still incomplete but we have made considerable progress on it. See previous section
9. Make 12 **background gifs** that are displayed based on which month it is. The TabPane for each month will sit on top of this gif background. We reduced the number of background gifs we used from 12 to 4, 1 for each season.
10. **Create a file IO hierarchy** of all text files to read and write to.

- The largest folder will be that for each year (due to the scope of this project, we plan to only create this for the current year for the current version of the application). It will contain a folder for each month. Each month's folder will contain text files to store journal entries, tracker information and daily todo-lists for the month.Each of these files will be created using specific naming conventions and formats, to aid the process of retrieving the information when the application is started in the future. This task is yet to be completed.

11. **Implement delete and save button handlers to different keypress events like Ctrl+S and Delete**, so that the user can perform those actions in multiple ways, regardless of the tab of the tabpane that is being displayed currently. This task is yet to be completed.

Bibliography

1. https://stackoverflow.com/questions/42350145/date-picker-selected-cell-css (for changing date at top of screen)
2. https://stackoverflow.com/questions/15189851/javafx-vertical-slider (can be used for objective, in trackers)
3. 18 Slider (Release 8) (oracle.com) (can be used for implementing the slider)
4. 11 Scroll Pane (Release 8) (oracle.com) (can be used to minimize space by placing widgets and data in a scrollable area)
5. 25 Color Picker (Release 8) (oracle.com)
6. 22 Tooltip (Release 8) (oracle.com) (adding tooltips to help user navigate)
7. java - Creating a using javafx - Stack Overflow (date picker/ calendar)
8. Part II: Using JavaFX UI Controls (Release 8) (oracle.com) (general widgets that may be useful)
9. Resizing images to fit the parent node - Stack Overflow (To bind the gif to the window and assist with resizing)
10. Resizing Font with main window in Java - Stack Overflow (For the journal entries as the book is resized)
11. How to add an image to a button (and position it) in JavaFX (For delete button column in To-do)
12. How to add a button to a TableView in JavaFX
13. Trash can image inspiration
14. Color Theory
15. Color Wheel- Color Palette Picker