Ex2- Desktop README

Bonus Features Implemented:

1. **Skin Change Feature**
   Implemented a skin change feature with two additional color schemes: **Dark Theme** and **Twinkle Theme**.
2. **Animations for Sheet Loading:**
   I implemented three animations for sheet loading: **Fade-In** (sheet fades from 0% to 100% opacity), **Zoom-In** (sheet scales up from a smaller size), and **Slide-In** (sheet slides in from the left). These animations run **simultaneously** over **2 seconds** and can be **disabled** to improve performance if needed.
3. **Multi-Column Filtering**
   A dedicated **button** for this feature is available in the **Table Commands** section.
4. **Graph Generation**
   A dedicated **button** for this feature is available in the **Table Commands** section.

Overview

The **Sheet-Cell Desktop UI** project builds on the console-based spreadsheet system from Exercise 1, adding a graphical user interface (GUI) using JavaFX. Users can now interact with the system visually, manage a grid of cells that hold various data types, perform calculations, and reference other cells. The focus of Exercise 2 is on enhancing user experience with a desktop-based interface while maintaining the core engine functionality.

Assumptions and Decisions

1. **Text Clipping Approach:**
   In cases where the content of a cell exceeds the available width, the system uses a **clip approach** rather than wrapping the text. This ensures that the table remains visually clean and prevents the need for row height adjustments.
2. **Help Text on Hover:**
   To enhance user experience, **help text** is displayed when hovering the mouse over various UI elements. This provides users with guidance or descriptions of the element's functionality without needing additional clicks or navigating to separate help sections.
3. **Modulo Operation on Negative Numbers:**
   In Java, the modulo operation does not behave as expected with negative numbers, unlike calculators and Excel. For example, `10 % -3` in Java returns `1`, whereas, in Excel and according to the mathematical definition, it would return `-2`. To address this, I implemented custom logic to ensure the modulo operation in the project aligns with the expected mathematical behavior.

Design Choices

Modular Architecture

The project is divided into two main modules:

- **Engine Module:** Handles all the core logic, including cell management, expression evaluation, and overall sheet processing. Although unchanged from Exercise 1, two important classes in this module are(that added for ex2):
    - **ColumnRowPropertyManager:** Manages properties such as row height, column width, and cell alignment.
    - **RangeFactory:** Manages the handling and definition of cell ranges, ensuring operations on multiple cells are handled efficiently.
- **UI Module:** The newly added graphical user interface built with JavaFX. This module allows for a more user-friendly way to interact with the spreadsheet, providing features like theming, animations, and graph generation.

Key Classes in the UI Module

1. **BodyController & SheetController:**
   These two classes are responsible for managing the main graphical interface. **BodyController** oversees the core structure and layout, while **SheetController** handles user interactions with the spreadsheet grid, such as selecting cells.
2. **Skin and Theme Management:**
    - **dark-theme.css, light-theme.css, twilight-theme.css:**
      Three CSS files are provided for skin customization. These define the color schemes for the interface, and users can switch between themes using the GUI.
    - **ColorUtils:**
      A utility class that handles switching between color themes and ensuring consistency across the application's elements.
3. **FilterDialog & SortDialog:**
   These classes are responsible for managing user input related to filtering and sorting table data. They present a simple interface for selecting columns and defining filtering or sorting criteria.
4. **GraphGenerator:**
   This class is responsible for generating visual graphs from the spreadsheet's numeric data. It supports both bar and line charts, and users can select ranges of cells to use as the X and Y axes.
5. **AnimationManager:**
   This class handles the loading animations, such as fade-in and slide-in effects, during spreadsheet operations. These animations can be enabled or disabled by the user to control the visual experience without impacting performance.

6. **MultiColFilterParameters:**
   Manages the filtering parameters for multiple columns, allowing users to select values from multiple columns.

Other Utility Classes

- **SheetDisplayHelper:**
  Manages the rendering of the spreadsheet on the screen, ensuring that cell data is properly displayed and that scrolling works smoothly.
- **ValidationUtils:**
  Handles data validation for input fields, ensuring that user input meets the necessary criteria before being processed by the engine.

System Overview

The system integrates the **engine** and **UI** modules smoothly, with the engine performing all core functionality while the UI provides an intuitive interface for users to interact with the data. The primary advancements in this exercise lie in improving user experience through graphical interactions, animations, and customization options like skins and filtering.

Submitted by

- **Name:** Ahmad Danaf
- **ID:** 211787833
- **Email:** ahmadda@mta.ac.il