**Password Factory Classes, Variables, and Methods**

**App (Main Application Class)**

Central hub that manages the flow between screens (e.g., Main Menu, Game, Lab).

* start(Stage) - Initializes the application and starts the main stage (window).
* showGame() - Displays the PasswordGame screen.
* showLab() - Displays the PasswordLab screen.

**AboutScreen**

Displays the "About" window, providing project details and system information.

* aboutStage - The pop-up window for the About screen
* show() - Creates and displays the About screen with project information and system details
* Reason to use SystemInfo.java that Netbeans always includes
* Doubled as a sandbox window to test JavaFX things

**GameSetupScreen**

Explain gameplay and handles the difficulty selection before starting the game.

* private Stage mainMenuStage; – Stores a reference to the main menu window so we can go back to it
* public GameSetupScreen(Stage mainMenuStage) – Constructor that accepts the main menu stage
* public void show() – Displays the setup window, lets the player pick a difficulty, and starts the game
* "Start Game" → Opens PasswordGame.java with the selected difficulty.
* "Back to Main Menu" → Closes setup and reopens App.java

**PasswordGame**

Manages the core gameplay logic, including rounds, scoring, and displaying feedback.

* score : int - Tracks the player's current score.
* round : int - Tracks the current round of the game.
* difficulty : DifficultyLevel - Tracks the difficulty level (e.g., Easy, Medium, Hard).
* timer : int - Tracks the remaining time for a round (in seconds).
* feedbackList : List<String> - Stores feedback messages generated during gameplay.
* startGame() - Initializes and starts the game.
* updateScore() - Updates the player's score after each round.
* showFeedback() - Displays feedback to the player based on their password.
* showPasswordPrompt() - Shows the password entry prompt to the player.
* endGame() - Ends the game and displays the final results.

**PasswordLab**

Educational screen that teaches users about password security through tutorials and password testing.

* testPasswordResult : bool - Tracks whether the tested password passed or failed.
* genericHints : List<Hint> - Stores generic tutorial hints for display in the lab.
* testPassword() - Validates the password entered by the user.
* showTestResults() - Displays the results of the password test.
* loadHints() - Loads generic tutorial hints from HintManager.
* displayHints() - Displays the hints in the lab UI.

**PasswordComponent (Abstract Class)**

Ssuperclass for password-related logic. Implements PasswordChecker

* checkStrength(password: String) : int - Abstract method to evaluate password strength.
* giveFeedback(password: String) : String - Abstract method to provide feedback.
* containsCommonWord(password: String, commonWords: String[]) : boolean - **Helper method** to check for common weak words.

**PasswordChecker (Interface)**

Defines the structure for validating passwords and generating feedback.

* checkStrength() - Validates the strength of a given password.
* giveFeedback() - Provides feedback or hints about the weaknesses of the password.

**PasswordValidator (Extends PasswordComponent, Implements PasswordChecker)**

Handles password validation and generates detailed feedback.

* checkStrength(password: String) : int - Implements logic to validate a password's strength based on predefined criteria.
* giveFeedback(password String) : String - Implements logic to return feedback as hints about password weaknesses.

**Hint**

Holds individual hint data (message and category) for use in tutorials or feedback.

* message : String - The actual hint text (e.g., "Use at least 8 characters.").
* category : String - The category of the hint (e.g., "Length", "Complexity").
* getHint() - Returns the hint message.
* getCategory() - Returns the category of the hint.

**HintManager**

Manages both dynamic and static hints, providing them to PasswordGame and PasswordLab as needed.

* dynamicHints : List<Hint> - Stores temporary, dynamically generated hints during gameplay.
* genericHints : List<Hint> - Stores static, preloaded hints for tutorials.
* addDynamicHint(Hint) - Adds a dynamically generated hint during gameplay.
* getGenericHints() - Retrieves the list of generic hints.
* getDynamicHints() - Retrieves the list of dynamic hints for the current round.
* clearDynamicHints() - Clears the dynamic hints after a round ends.

**SystemInfo**

Retrieves system details such as the Java and JavaFX version. NetBeans automatically includes this

* javaVersion() - Returns the Java version as a string.
* javafxVersion() - Returns the JavaFX version as a string.