Project Title: Clock Abstract Data Type (ADT)

Objective

The goal of this project is to design and implement a Clock Abstract Data Type (ADT) that represents a 24-hour digital clock. The Clock ADT should support various operations such as time initialization, updating time, and comparison between different clock instances.

Project Requirements

1. Features of the Clock ADT

- **Clock Creation:**
  - Initialize a clock using hours and minutes.
  - Initialize a clock using a string in the format "HH:MM".
- **Time Manipulation:**
  - `tic()` : Increments the clock by one minute.
  - `toc(int minutes)` : Increments the clock by a given number of minutes.
- **Comparison:**
  - `isEarlierThan(Clock other)` : Checks if the current clock time is earlier than another clock instance.
- **String Representation:**
  - `toString()` : Returns the clock time as a formatted string in `HH:MM` format.

2. Input and Output

- **Input Format:**
  - The program should accept user input to create and manipulate the clock.
  - Expected inputs:
    - `constructor(int, int)` : Initializes the clock with integer hour and minute values.
    - `constructor(String)` : Initializes the clock with a string-formatted time.
    - `tic()` : Advances time by one minute.
    - `toc(int)` : Advances time by a given number of minutes.
    - `isEarlierThan(Clock)` : Compares two clock instances.
    - `toString()` : Displays the clock time.
- **Output Format:**
  - The clock should return the updated time after operations.

- Boolean output for comparison.

Implementation Guidelines

1. **Class Design:**
   - Create a `Clock` class that encapsulates hour and minute attributes.
   - Use helper methods for validation and time updates.
   - Implement `toString()` for easy debugging and output formatting.
2. **Constraints:**
   - The clock should follow a 24-hour format (0 ≤ hours < 24, 0 ≤ minutes < 60).
   - Any invalid input should default to `00:00` (if exceptions are not used).
3. **Testing:**
   - Provide a set of test cases that validate the clock functionalities.
   - Ensure boundary cases like `23:59 -> 00:00` are handled correctly.

Expected Outcome

A functional Clock ADT that allows time manipulation and comparison efficiently, following best programming practices.

Extensions (Optional)

- Implement a stopwatch mode to count elapsed time.
- Add an alarm functionality that triggers at a specific time.
- Support 12-hour format with AM/PM.