Create a Base Class Clock, this class initials the starting time for any clock object

1. Properties:

    a) public -> hours, minutes, seconds
    b) private -> IntervalId

2. Constructor:

    a) Accept an initial time (e.g., HH:MM:SS) as a string.
    b) Parse the time and store it as hours, minutes, and seconds Properties.

3. Static Member:

    a) Add a static method formatTime(hours, minutes, seconds) to format the time as HH:MM:SS from the class properties

4. Private Methods:

    a) Create a private method #tick() to increment the clock's time by one second.

5. Public Methods:

    a) start(): Starts the clock using setInterval to call the #tick() method every second.
    b) stop(): Stops the clock.
    c) getTime(): Returns the current time formatted as HH:MM:SS (now it's time to use formatTime method to convert properties to string)

Create a Subclass AlarmClock:

- Inherit from the Clock class.
- private alarmTime property to store the alarm time as , HH:MM:SS
- Constructor:
    - Accept an additional parameter alarmTime (e.g., HH:MM:SS) for the alarm.
- Private Methods:
    - Create a private method #checkAlarm() to compare the current time with the alarm time.
- Public Methods:
    - Override the start() method to call #checkAlarm() every second.
    - Add setAlarm(newAlarmTime) to update the alarm time.

Testing the Implementation:

- Instantiate an AlarmClock object with an initial time and alarm time.

- Start the clock and ensure it prints the time to the console every second.
- If the alarm time matches the current time, log **"Alarm! Wake up!"** and stop the clock.

Example for using the classes:

```
const sleepAlarmClock = new AlarmClock('14:59:55', '15:00:00');
sleepAlarmClock.start();

setTimeout(() => sleepAlarmClock.setAlarm('15:01:00'), 10000); // Update alarm after 10 seconds
```

Try to control the message instead of log **"Alarm! Wake up!"** every time