

## 1. Daily Mood Journal



The Daily Mood Journal is a simple, interactive, and console-based mental health tool designed to help users track their daily emotional state. It allows users to record their mood, write a short note about their day, and receive personalized feedback based on the mood they enter. Each entry is organized by date, making it easy for users to review past logs at any time. By providing a structured way to document emotions and daily reflections, the program promotes self-awareness, emotional well-being, and the identification of personal mood patterns. Through consistent journaling, users can better understand their mental state, manage stress, and develop healthier coping habits over time.

## 3. OOP Concepts Applied

This project applies the four core principles of Object-Oriented Programming: Abstraction, Encapsulation, Inheritance, and Polymorphism.

### 3.1 Abstraction

Abstraction is used by creating an abstract class JournalEntry, which represents the general idea of any journal record. This class contains shared properties (date, note) and an abstract method displayEntry() that forces all subclasses to define their own display behavior. This hides unnecessary details and provides a clean, simplified view of journal entries.

### 3.2 Encapsulation

Encapsulation is applied through the use of private fields and public getters/setters in both JournalEntry and MoodEntry. For example, note, date, and mood cannot be accessed directly

from outside the class. Instead, they can only be accessed or modified through methods like `getNote()`, `setNote()`, and `getMood()`. This protects the data and enforces controlled access.

### 3.3 Inheritance

Inheritance is shown when the class `MoodEntry` extends `JournalEntry`. `MoodEntry` inherits all fields and behaviors from the abstract parent class, such as date, note, and the abstract method `displayEntry()`. It adds additional behavior by including the mood field and specific methods related to mood tracking.

### 3.4 Polymorphism

Polymorphism is evident when the program uses the same method name but executes different behavior. The `displayEntry()` method is overridden in `MoodEntry`, giving it a customized output. When viewing all entries in the `JournalManager` class, each object is handled through a `JournalEntry` reference, allowing the program to call the correct version of `displayEntry()` depending on the actual object type. This demonstrates runtime (dynamic) polymorphism.

## 4. Program Structure

### 4.1 Main Class Overview

#### Main

- Entry point of the program.
- Handles menu display, user input, and directs actions to `JournalManager`.

### 4.2 Class Descriptions

#### `JournalEntry` (Abstract Class)

- Represents a general journal entry.
- Fields: `LocalDate date`, `String note`
- Methods:
  - `getDate()`, `getNote()`, `setNote()`
  - `displayEntry()` → abstract method to be implemented by subclasses.
- Purpose: Base structure for all possible journal entry types.

#### `MoodEntry` (Subclass of `JournalEntry`)

- Specialized journal entry containing mood data.
- Additional field: String mood
- Overrides displayEntry() → adds mood + feedback from MoodAnalyzer.
- Purpose: Stores the user's mood and journal note for the day.

### MoodAnalyzer

- A utility class that provides feedback depending on user's mood.
- Contains:
  - Static method getFeedback(String mood)
- Purpose: Centralized logic for interpreting mood.

### JournalManager

- Stores and manages all journal entries.
- Fields:
  - ArrayList<JournalEntry> entries
  - Scanner scanner
- Methods:
  - addEntry() → creates a new MoodEntry and saves it
  - viewEntries() → displays all entries (uses polymorphism)
- Purpose: Controls data management, processing, and user interactions.

## 5. How to run the program

Follow the steps below to compile and run the Daily Mood Journal program.

5.1 Go to Programiz Java Online Compiler. Open your browser and go to:  
<https://www.programiz.com/java-programming/online-compiler/>

5.2 Copy and paste your entire Java code. Paste your full Main.java code, including all classes in the same file.

5.3 Click the green run button at the top-right.

5.4 Use the input box. Your program requires user input, so Programiz will show an input field under the console when needed.

```
===== Daily Mood Journal =====
1. Add New Entry
2. View Past Entries
3. Exit
Enter choice: 1
Enter your mood (Happy, Sad, Anxious, Calm): Happy
Write a short note about your day: Had a great day!
Entry saved successfully!
Feedback: Keep it up! Share your positivity with others
```

5.5 Continue interacting with your program. You can add entries, View entries, and Exit All through the same console area.

## 6. Sample Output

### 6.1. Add New Entry

Output Clear

```
===== Daily Mood Journal =====
1. Add New Entry
2. View Past Entries
3. Exit
Enter choice: 1
Enter your mood (Happy, Sad, Anxious, Calm): sad
Write a short note about your day: sad
Entry saved successfully!
Feedback: It's okay to feel sad. Take time to rest and care for yourself
```

## 6.2. View Past Entries

```
===== Daily Mood Journal =====
1. Add New Entry
2. View Past Entries
3. Exit
Enter choice: 2

===== PAST ENTRIES =====
-----
Date: 2025-11-23
Mood: sad
Note: sad
Feedback: It's okay to feel sad. Take time to rest and care for yourself
-----
```

## 6.3. Invalid Choice

```
=====
1. Add New Entry
2. View Past Entries
3. Exit
Enter choice: 4
Invalid choice. Please try again.
```

## 6.4. Exit

```
===== Daily Mood Journal =====
1. Add New Entry
2. View Past Entries
3. Exit
Enter choice: 3
Thank you for using your Mental Health Journal.

==== Code Execution Successful ===
```

## **7. Author and Acknowledgement**

*Authors:*

Alim, Arlene D.

Loterte, Samuel B.

Pepillo, Angel A.

We would like to thank everyone who helped us while working on this project. First, our instructor guided us and gave useful feedback that made the project easier to understand. A big thank you to our group members for supporting and helping each other throughout the project. We also want to thank online resources, especially Programiz, for providing examples and tools that helped us understand Java and apply Object-Oriented Programming concepts in our project. Without all the help we received, completing the Daily Mood Journal would have been much harder.

## **8. Other sections you may include**

### **8.1. Future Enhancements**

Although the current version of the Daily Mood Journal is functional, several improvements can be added in future versions to enhance usability, flexibility, and overall user experience.

1. **Load Entries from a File** - When the program starts, it can automatically read previously saved entries and display them. This will turn the program into a true long-term journal.
2. **Add More Mood Categories** - Expand the list of mood options (e.g., Stressed, Excited, Tired, Motivated), allowing for more detailed tracking.
3. **Add Mood Statistics** - Implement analytics such as the most common mood, mood trends per week, and average mood rating. These can help users better understand their emotional patterns.
4. **Edit or Delete Entries** - Users may want to modify or remove older entries.
5. **Password Protection** - Add a login system or password lock to keep the journal private and secure.

### **8.2. References**

<https://www.programiz.com/java-programming/online-compiler/>