

# **MindWell Mental Health Application**

## **Project Documentation - CSS 217**

**Team Members:** Vakhitova Dilyara , Kyzaibai Makpal , Beibit Assylzat



### **1. Application Overview**

MindWell is a console-based mental health application that helps users track mood, practice exercises, and connect with community support.

#### **Core Features:**

- Mood tracking with daily logging and history
- Four guided exercises: Breathing, Meditation, Relaxation, Mindfulness
- Personalized recommendations based on mood
- Community forums with subscription notifications
- Progress statistics and goal setting

**Technical Stack:** Java, ArrayList data structure, console interface, three design patterns

### **2. Design Patterns**

#### **2.1 Singleton Pattern (Creational)**

**Classes:** MindWellApp, UserManager, ExerciseManager, CommunityManager

**Structure:** Private static instance variable, private constructor, public getInstance() method

**Purpose:** Ensures single instance of each manager throughout application. Prevents data inconsistency across different parts of the system.

**Why Used:** If multiple UserManager instances existed, user registration in one would be invisible to another. Singleton guarantees all components access same data.

## 2.2 Decorator Pattern (Structural)

**Classes:** Exercise (base), ExerciseDecorator (abstract decorator), TimerDecorator, ReminderDecorator

**Structure:** Decorator wraps Exercise object and adds functionality through enhanced execute() method

**Purpose:** Adds features to exercises dynamically without modifying exercise classes.

**Why Used:** Users want different feature combinations. Some need timers, others need reminders, some need both. Decorator allows flexible runtime composition without creating separate classes for each combination.

## 2.3 Strategy Pattern (Behavioral)

**Classes:** RecommendationStrategy (interface), LowMoodStrategy, MediumMoodStrategy, HighMoodStrategy

**Structure:** Each strategy implements getRecommendation() method with mood-appropriate advice

**Purpose:** Selects recommendation algorithm based on user mood level.

**Why Used:** Different moods require different approaches. Low mood needs gentle support, high mood can handle active challenges. Strategy pattern encapsulates each approach and selects at runtime.

## 3. SOLID Principles

### Single Responsibility

Each class has one job. User manages data, UserManager handles authentication, MoodEntry represents records, Exercise defines structure, managers handle collections.

### Open/Closed

New functionality added without modifying existing code. New exercises extend Exercise class, new strategies implement interface, new decorators extend decorator base.

## **Liskov Substitution**

Subclasses replace parent classes seamlessly. All Exercise types work through Exercise reference. All strategies work through strategy interface.

## **Interface Segregation**

Small focused interfaces. Observer has only update() method, RecommendationStrategy has only getRecommendation() method.

## **Dependency Inversion**

Code depends on abstractions not concrete classes. Methods use Exercise abstraction not specific exercise types, strategy interface not specific strategies.

## **4. System Architecture**

**Core Classes:** Main (entry point), MindWellApp (controller), User (data model), MoodEntry (mood record)

**Managers:** UserManager (authentication), ExerciseManager (exercise library), CommunityManager (community features)

**Exercise Hierarchy:** Exercise (abstract base), four concrete exercises, ExerciseDecorator (abstract), two concrete decorators

**Strategy Classes:** Interface and three implementations for different mood levels

**Observer Classes:** Observer interface, Subject interface, User implements Observer, CommunityManager implements Subject

### **Data Flow:**

- Authentication: User enters credentials, UserManager validates, returns User object
- Mood logging: Creates MoodEntry, adds to User history, triggers recommendations if low
- Exercise: Retrieves from manager, wraps with decorators, executes, records completion
- Community: Subscribes to topics, posts create notifications for all subscribers

## **5. User Guide**

**Installation:** Compile with javac Main.java, run with java Main

**First Use:** Register with username and password, then login

**Mood Diary:** Log daily mood (1-10) with notes, view history and average, get personalized recommendations

**Exercises:** Browse available exercises, start selected exercise with timer and reminder, view completion history

**Community:** View topics, post messages, subscribe for notifications, view your posts

**Progress:** View mood statistics and exercise counts

**Settings:** Set goals (reduce stress, improve sleep, increase focus)

## 6. Testing

**Functional Tests:** Registration accepts valid users and rejects duplicates. Login validates credentials. Mood logging saves entries and triggers warnings for low mood. Exercises execute with decorators and record completion. Community posts notify subscribers.

**Pattern Tests:** Singleton maintains single instance across multiple references. Decorator combines features correctly. Strategy selects based on mood level. Observer notifies all subscribers.

**Integration Tests:** Complete user journey from registration through all features works correctly with consistent data.

## 7. Technical Decisions

**ArrayList Usage:** Chosen over HashMap for simplicity in educational context. Performance adequate for expected data sizes.

**Console Interface:** Focus on design patterns rather than UI complexity. Enables rapid development and clear demonstration.

**In-Memory Storage:** Data persists during runtime only. Simplifies implementation for academic purposes.

## Conclusion

MindWell demonstrates practical implementation of design patterns and SOLID principles. Three patterns from different categories (Singleton, Decorator, Strategy) create maintainable architecture. All five SOLID principles applied throughout ensure extensible, understandable code structure.