MasterMind Game Implementation
Object-Oriented Programming Project

Erkin Tunc Boya

April 2025

1 Introduction

This project implements a Java-based MasterMind game featuring multiple game modes, configurable difficulty settings, and an intelligent computer opponent. The implementation demonstrates object-oriented programming principles while providing an engaging gaming experience through its modular and extensible design.

2 Game Overview

The core gameplay revolves around players attempting to guess a secret combination of colored pegs within a limited number of attempts. After each attempt, players receive feedback indicating:

- The number of pegs with correct color and position
- The number of pegs with correct color but wrong position

The game offers three distinct modes:

- Single Player: Players compete against computer-generated combinations
- Multiplayer: Supports player vs. player competition with tournament capabilities
- AI Opponent: Players challenge an adaptive AI that employs sophisticated guessing strategies

3 Technical Implementation

The implementation is structured around seven key classes that handle different aspects of the game:

3.1 Core Components

- `MasterMind`: Serves as the main controller, managing game initialization and mode selection
- `Game`: Handles game session management, including turn progression and scoring
- `Board`: Maintains game state, tracks attempts, and validates moves
- `Player`: Provides base functionality for human players
- `Combination`: Manages color combinations and validation logic
- `Pion`: Represents individual pegs with color and position properties

3.2 AI Implementation

The `AutomaticPlayer` class implements an intelligent computer opponent using three progressive strategies:

1. **Initial Phase:** Employs random generation for early attempts to explore the solution space
2. **Pattern Recognition:** Analyzes previous attempts to identify:

- Successful color patterns
- Position-specific color frequencies
- Color combination effectiveness

3. **Intelligent Analysis:** Utilizes weighted scoring systems to:

- Track position-specific color success rates
- Adapt strategy based on feedback
- Optimize color selection for each position

4 Configuration Options

The game supports various customization options to adjust difficulty and gameplay experience:

- Number of pegs (3-5)
- Available colors (4-8)
- Maximum attempts (5-20)
- Detailed feedback mode

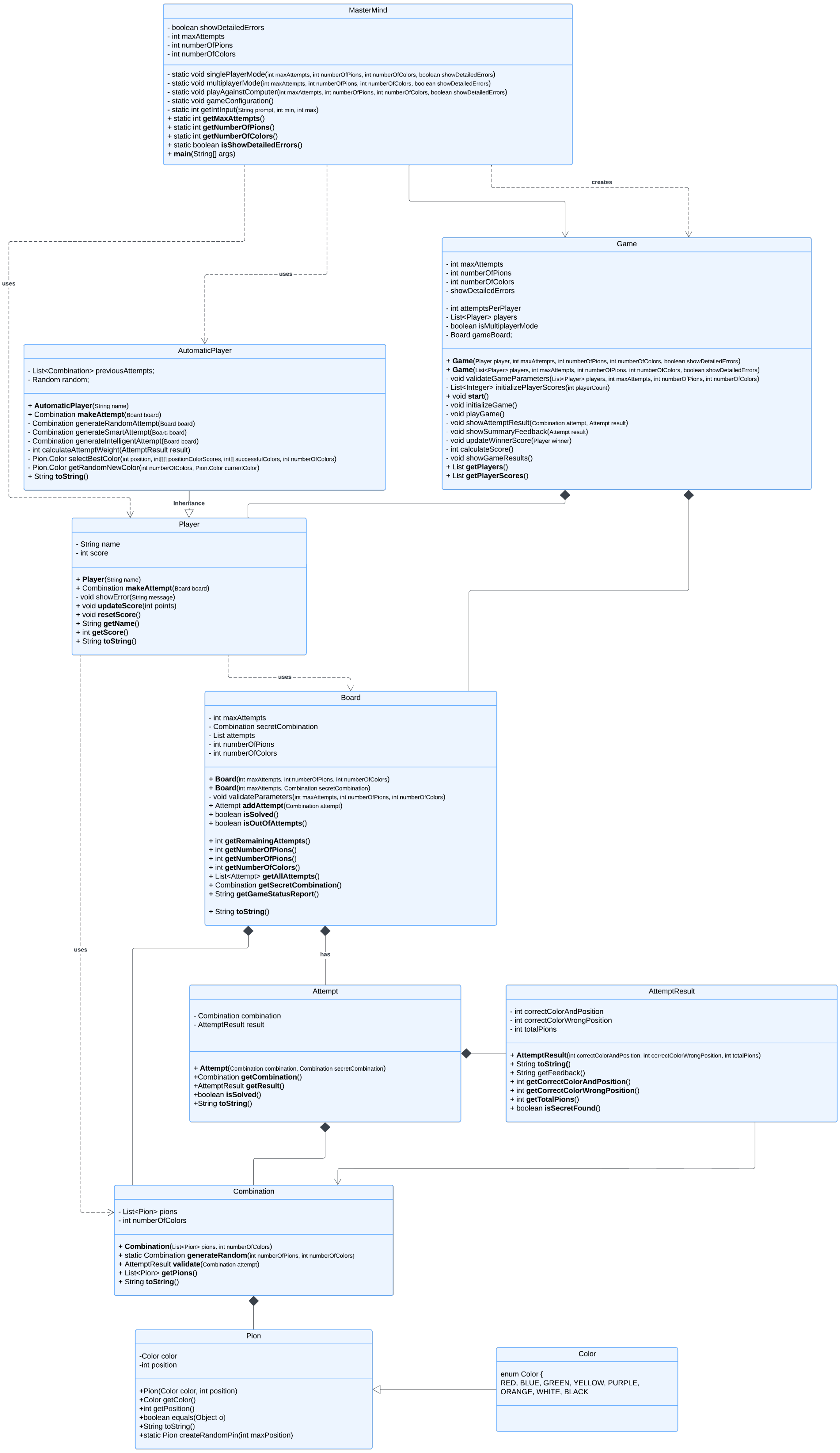
5 Future Enhancements

Potential improvements for future versions include:

- Enhanced AI strategies using machine learning
- Graphical user interface implementation
- Online multiplayer capabilities
- Achievement and progression systems

6 Class Diagram

Figure 1: Master-Mind UML diagram


MasterMind
- boolean showDetailedErrors
- int maxAttempts
- int numberOfPions
- int numberOfColors
- static void singlePlayerMode(int maxAttempts, int numberOfPions, int numberOfColors, boolean showDetailedErrors)
- static void multiplayerMode(int maxAttempts, int numberOfPions, int numberOfColors, boolean showDetailedErrors)
- static void playAgainstComputer(int maxAttempts, int numberOfPions, int numberOfColors, boolean showDetailedErrors)
- static void gameConfiguration()
- static int getIntInput(String prompt, int min, int max)
+ static int getMaxAttempts()
+ static int getNumberOfPions()
+ static int getNumberOfColors()
+ static boolean isShowDetailedErrors()
+ main(String[] args)
creates
uses
uses
Game
- int maxAttempts
- int numberOfPions
- int numberOfColors
- showDetailedErrors
- int attemptsPerPlayer
- List<Player> players
- boolean isMultiplayerMode
- Board gameBoard;
+ Game(Player player, int maxAttempts, int numberOfPions, int numberOfColors, boolean showDetailedErrors)
+ Game(List<Player> players, int maxAttempts, int numberOfPions, int numberOfColors, boolean showDetailedErrors)
- void validateGameParameters(List<Player> players, int maxAttempts, int numberOfPions, int numberOfColors)
- List<Integer> initializePlayerScores(int playerCount)
+ void start()
- void initializeGame()
- void playGame()
- void showAttemptResult(Combination attempt, Attempt result)
- void showSummaryFeedback(Attempt result)
- void updateWinnerScore(Player winner)
- int calculateScore()
- void showGameResults()
+ List getPlayers()
+ List getPlayerScores()
AutomaticPlayer
- List<Combination> previousAttempts;
- Random random;
+ AutomaticPlayer(String name)
+ Combination makeAttempt(Board board)
- Combination generateRandomAttempt(Board board)
- Combination generateSmartAttempt(Board board)
- Combination generateIntelligentAttempt(Board board)
- int calculateAttemptWeight(AttemptResult result)
- Pion.Color selectBestColor(int position, int[][] positionColorScores, int[] successfulColors, int numberOfColors)
- Pion.Color getRandomNewColor(int numberOfColors, Pion.Color currentColor)
+ String toString()
Inheritance
Player
- String name
- int score
+ Player(String name)
+ Combination makeAttempt(Board board)
- void showError(String message)
+ void updateScore(int points)
+ void resetScore()
+ String getName()
+ int getScore()
+ String toString()
uses
Board
- int maxAttempts
- Combination secretCombination
- List attempts
- int numberOfPions
- int numberOfColors
+ Board(int maxAttempts, int numberOfPions, int numberOfColors)
+ Board(int maxAttempts, Combination secretCombination)
- void validateParameters(int maxAttempts, int numberOfPions, int numberOfColors)
+ Attempt addAttempt(Combination attempt)
+ boolean isSolved()
+ boolean isOutOfAttempts()
+ int getRemainingAttempts()
+ int getNumberOfPions()
+ int getNumberOfPions()
+ int getNumberOfColors()
+ List<Attempt> getAllAttempts()
+ Combination getSecretCombination()
+ String getGameStatusReport()
+ String toString()
uses
has
Attempt
- Combination combination
- AttemptResult result
+ Attempt(Combination combination, Combination secretCombination)
+Combination getCombination()
+AttemptResult getResult()
+boolean isSolved()
+String toString()
AttemptResult
- int correctColorAndPosition
- int correctColorWrongPosition
- int totalPions
+ AttemptResult(int correctColorAndPosition, int correctColorWrongPosition, int totalPions)
+ String toString()
+ String getFeedback()
+ int getCorrectColorAndPosition()
+ int getCorrectColorWrongPosition()
+ int getTotalPions()
+ boolean isSecretFound()
Combination
- List<Pion> pions
- int numberOfColors
+ Combination(List<Pion> pions, int numberOfColors)
+ static Combination generateRandom(int numberOfPions, int numberOfColors)
+ AttemptResult validate(Combination attempt)
+ List<Pion> getPions()
+ String toString()
Pion
-Color color
-int position
+Pion(Color color, int position)
+Color getColor()
+int getPosition()
+boolean equals(Object o)
+String toString()
+static Pion createRandomPin(int maxPosition)
Color
enum Color {
RED, BLUE, GREEN, YELLOW, PURPLE,
ORANGE, WHITE, BLACK