Laboratory 6: Master Mind

Master Mind is a code breaking game. A secret code is selected by one player and another player tries to guess the code. The secret code has four positions and each position is assigned a color out of seven possible colors. It is legal to have multiple positions assigned the same color. In this version of Master Mind, the computer selects the secret code randomly.

To win the game, the player must determine the correct position and color of the randomly generated secret code. A guess is composed of selecting a color for each of four positions. After each guess, feedback is obtained which is used to improve the guess until an exact match of each color and position of the secret code is obtained. The secret code must be obtained in eight or fewer guesses.

Feedback is determined as follows:

- A black "button" is awarded for each of the four guess elements that matches both color and position of the secret code.
- A white "button" is awarded for each of the four guess elements that matches a color but is not in the correct position.
- It is likely that a given guess will have fewer than four (and maybe zero) buttons awarded.

Lab:

- Interfaces
- Dynamic Changeability
- Primitive Arrays and ArrayList
- Command Line Arguments

Part I: Evaluating a Guess

Complete **Guess.java**. You will be comparing a guess to the secret guess to determine the number of black buttons and white buttons to provide as feedback for that guess. The first element of the integer array is the number of black buttons, and the second element is the number of white buttons. The trick is to make sure that a given guess element is not awarded more than one button. You will probably want to assign the black buttons first.

Complete the **`“reportResult”`** method in this class. Count the black buttons to be awarded and award a black button and remove those elements and do the same for white.

Part II: Master Mind AI

Complete the following:

- **MasterMindAI.java (interface for MasterMindAIRandom class):**
  Add an interface method **`nextGuess(int guess_id)`**
- **MasterMindAIRandom.java**
  - **MasterMindAIRandom constructor**
  - **nextGuess(int guess_id) method:** Call **`makeRandomGuess`** method with the **`guess_id`** as parameter.
  - **makeRandomGuess(int guess_id) method:** Use **`getRandomNumberGenerator`** method of **`Random`** class.
- **MasterMindAIConsistent.java**
  - **nextGuess method:** make a random guess until you get one that works with all previous results.
  - **analyzeGuess method:** Compare the previous guess with the secret guess. Also, compare the next guess with previous guesses, except the secret guess.
  - **makeRandomGuess method:** Use the **`Random`** class to make a randomly generated guess.

Complete the **MasterMindAI.java** interface and then implement that interface to write **MasterMindAIRandom.java**. This simple AI makes a random guess, hoping for a miracle.

Implement the MasterMindAI interface again to complete **MasterMindAIConsistent.java**. Make a random guess where a guess is four integers (each integer represents a color) between 1 and 7. Make sure that the randomly selected guess is consistent with the feedback provided for all previous guesses. If it is, make that your official guess. If it is not, make another randomly generated guess, repeating the process until you identify a guess that is consistent with previous guesses.

Note that your AI has a reference to the MasterMind game so that it can obtain required information. This is better than having both the AI and MasterMind remember information about MasterMind (namely, all previous guesses and associated feedback). Your AI can simply request information when needed. Of particular importance are the two getResult methods in the MasterMind class. One of these can be used to obtain previous feedback (guesses are

tested against the secret code) and the other allows guesses to be compared to one another ("pretend" that one of the guesses, the second parameter, is the secret code).

Part III: Master Mind Game (MasterMind.java - Driver class)

Complete the following methods:

- **MasterMind constructor**
- **setAI method:** Change the AI used based on the value passed to this method.
- **main method:** Parse the command line argument as an integer and save it in **`ai_int`**. Implement exception handling for ArrayIndexOutOfBoundsException and NumberFormatException. In case of an exception, default your ai to be a human player (set ai_int = 0).

Complete **MasterMind.java** so that the user can specify the type of game desired at the command prompt. A 0 indicates a human player, a 1 indicates that the game should use MasterMindAIRandom, a 2 indicates MasterMindAIConsistent, and a 3 indicates MasterMindAIMiniMax. Your game should also allow the player to change between AIs and from AI to human by typing 0, 1, 2, or 3. It is not possible to change from human to AI as the change could occur in the middle of a guess!

Additional Info:

The relevant methods within the given Random Class (Singleton Pattern) are the following:

- **public static Random getRandomNumberGenerator()** //creates a new Random object
- **public int randomInt (int low, int high)** //Returns a randomly generated integer between low and high (inclusive of both).