

## Project #3

### Mastermind Game

**Due Date: March 16, 2009 @ 11:55 p.m.**

In this project, you are going to implement a simple number guessing game that is often known as the game of Mastermind. Your application has to generate a secret number randomly and the user has to guess the number. Your program prompts the “game player” to guess the number and based on the guess the program will provide some feedback. Based on that feedback, the player makes another guess. Guessing continues until the “secret” number is guessed or until the maximum number of tries is reached. The game has three levels of difficulty: beginners, intermediate and advance corresponding to 3 digits secret number (30 guesses), 4 digits secret number (20 guesses) and 5 digits secret number (10 guesses) respectively. You should note that leading zero does not count as a digit. (Example: 021 is not a 3 digits number)

Write a Java program that allows a user to play Mastermind as many times as (s)he wishes. The program first asks if the user wants to play the game. If the answer is no, the program will display a message (see below) and terminate. Otherwise, the program will prompt the user about the amount of money (s)he has available to play. If this amount is less than the minimum amount allowed to play, \$2.00, then the program will indicate to the user what the minimum amount of money required to play the game is and terminate. If the user has enough money to play the game, the game will start. At this point, the program will request for the level of difficulty and the amount of money (s)he wants to want to play for this round and asks the player for his/her guess. The user’s guess should be k digits long (depending on the level of difficulty, for example 3 digits long for beginners). If the user enters an ill-formed “guess,” the program should print an error message and prompt again for a correctly formed guess. Given a k digit guess by the user, your program must display two results.

- First the number of digits in the guess which are in the correct positions.
- Second, the sum of those digits which are in the correct positions.

For example, assume that the secret number is 53840 and that the player guesses 83241.

The digits 3 and 4 are in the correct position. Thus the program should respond with 2 and 7.

If the user guesses the secret number correctly, (s)he will win an amount of money depending on the number of digits and the number of guesses it took him to guess the secret code. The formula is shown below:

$$\text{MoneyEarned} = \frac{\text{MoneyToPlay} * \text{SecretCodeDigits} * (\text{MaxGuesses} - \text{incorrectGuesses})}{\text{MaxGuesses}}$$

*MoneyToPlay*: The money the user wants to play in the current round of the game

*SecretCodeDigits*: The number of digits in the secret code according to the level of difficulty of the game

*MaxGuesses*: Maximum number of times the user can guess the secret code according to the level of difficulty of the game.

*incorrectGuesses*: Number of times the user did not guess the secret code.

If the user cannot guess the secret number, (s)he will lose the money (s)he played for that round. After each game, the program will display the money earned or lost by the user in the current game as well as how much money the user should have at that time of the game, let's call that amount the user's balance. Afterwards, the program will prompt the user whether (s)he wants to play again as long as the user's balance is greater than or equals to the minimum play money. Otherwise, if the user runs out of money or (s)he does not want to keep playing, the program will terminate the game. Your program should handle the user's answers correctly regardless of case. The program should display the summery of the game after each round. The summery will include: the status of the game (win/lost), the number of wrong guesses, the amount of money the user won/lost, and the current balance of the user. (Please see the examples below for more reference).

## Sample Output

In the following examples, the text in black is the program output, the text in green corresponds to the answers entered by the user.

### Example 1:

```
Would you like to play Mastermind (yes/no)? yes
Enter the amount of money you have to play: 1.0
Sorry, you should have at least 2.0 dollars to play the game. Bye!
```

### Example 2:

```
Would you like to play Mastermind (yes/no)? yEs
Enter the amount of money you have to play: 12.50
Please enter the level of difficulty (beginner, intermediate, advance): easy
Sorry , this is not a correct level.
Please enter the level of difficulty (beginner, intermediate, advance):
begINNER
Enter the amount of money you want to bet: 100
Sorry, you cannot bet more money than what you have.
Enter the amount of money you want to bet: 1.50
Sorry, the minimum amount of money is 2.00 dollars
Enter the amount of money you want to bet: 10
Please guess the 3 digits number: 10324
The guess you have entered is ill-formed.
Please guess the 3 digits number: 23
The guess you have entered is ill-formed.
Please guess the 3 digits number: 148
Number of correct digits: 1
Sum: 8
Please guess the 3 digits number: 238
Number of correct digits: 1
Sum: 8
Please guess the 3 digits number: 658
Number of correct digits: 2
Sum: 14
Please guess the 3 digits number: 608
```

Number of correct digits: 2

Sum: 14

Please guess the 3 digits number: 618

You won!!!

You have had 4 wrong guesses.

You have earned 26.0 dollars.

Your balance is now 38.5 dollars

Would you like to play again (yes/no)? Yes

Please enter the level of difficulty (beginner, intermediate, advance):

Intermediate

Enter the amount of money you want to bet: 38

Please guess the 4 digits number: 1234

Number of correct digits: 1

Sum: 3

Please guess the 4 digits number: 6932

Number of correct digits: 3

Sum: 11

Please guess the 4 digits number: 6332

Number of correct digits: 3

Sum: 11

Please guess the 4 digits number: 6032

Number of correct digits: 3

Sum: 11

Please guess the 4 digits number: 6132

You won!!!

You have had 4 wrong guesses.

You have earned 121.6 dollars.

Your balance is now 160.1 dollars

Would you like to play again (yes/no)? nO

The game has terminated.

Bye. Come to play again!!

### Example 3:

Would you like to play Mastermind (yes/no)? YES  
Enter the amount of money you have to play: 20.50  
Please enter the level of difficulty (beginner, intermediate, advance):  
adVaNce  
Enter the amount of money you want to play: 19  
Please guess the 5 digits number: 12345  
Number of correct digits: 0  
Sum: 0  
Please guess the 5 digits number: 23432  
Number of correct digits: 0  
Sum: 0  
Please guess the 5 digits number: 54678  
Number of correct digits: 0  
Sum: 0  
Please guess the 5 digits number: 35876  
Number of correct digits: 1  
Sum: 3  
Please guess the 5 digits number: 38410  
Number of correct digits: 2  
Sum: 11  
Please guess the 5 digits number: 38432  
Number of correct digits: 2  
Sum: 11  
Please guess the 5 digits number: 38912  
Number of correct digits: 2  
Sum: 11  
Please guess the 5 digits number: 38045  
Number of correct digits: 2  
Sum: 11  
Please guess the 5 digits number: 38563  
Number of correct digits: 2  
Sum: 11  
Please guess the 5 digits number: 38212  
Number of correct digits: 3  
Sum: 13

```
Sorry, you lost!!!  
You have guessed 10 times  
The number was 38289  
You have lost 19.0 dollars.  
Your balance is now 1.5 dollars
```

The game has terminated.  
Bye. Come to play again!!

#### Example 4:

```
Would you like to play Mastermind (yes/no)? No  
Bye, see you next time.
```

#### Hints:

- Use the method `random` of the class `Math` to compute random numbers. For example, to generate a 3-digit random number and stored it in an `int` variable `secretCode`, you should proceed in the following manner:

```
secretCode =(int) (Math.random()*900) + 100;
```

The method `random` of the class `Math` returns a double random number greater than or equal to zero and less than 1. Multiplying the random number returned by the `random` method by 900 and casting its result as an integer will generate an integer between greater than or equal to zero and 899. After adding 100 to this result, we will get a random integer between 100 and 999.

- The solution of the problem requires loops, nested loops and decision statements.
- Declare a double constant called `MIN_BET` that stores the minimum amount of money required to play the game.i.e.2
- You may use `System.exit(0)` to terminate the game whenever the user does not have enough money to participate in the game or (s)he does not want to start playing the game.
- You may use Boolean variables to keep track of the win/lost status of a game.

- You will need an outer loop to handle several games.
- An inner loop to request and validate the level of difficulty of the game.
- An inner loop to request and validate the money the user wants to play in a round of the game.
- An inner loop to validate that the guess of the user has the appropriate number of digits.
- An inner loop to determine the digits in the user's guess that matches the corresponding digit in the secret code.
- Several decision statements to complete the logic of the game.

### Requirements:

You should try to make your program output look just like the examples above.

The name of the class in your java program must be Mastermind. Therefore, the java source code file must be called Mastermind.java.

### Other requirements:

Your code must include a comment header like the following in every project you submit.

```

/*
* Mastermind.java
* Author: Amy Smith
* Last edited: 20/02/2009
*
* Purpose: A brief one or two paragraph description of the
* program. What does it do? How does it do it?)
*
* Statement of Academic Honesty:
*
* The following code represents my own work. I have neither
* received nor given inappropriate assistance. I have not copied
* or modified code from any source other than the course webpage
* or the course textbook. I recognize that any unauthorized
* assistance or plagiarism will be handled in accordance with
* the University of Georgia's Academic Honesty Policy and the
* policies of this course.
*/

```

### **Project Submission:**

Submit the file Mastermind.java via WebCT.

### **Project Grading:**

All projects are graded out of possible 100 points. In this project, 80 points will be for program correctness and 20 points for programming style. The following specific criteria will be applied to grade this project:

If your code does not compile, you will get a 0. Make absolutely certain your code compiles before you submit it via WebCT.

- Style (20 points )
  - Program header
  - Helpful comments
  - Mnemonic well-named variables
  - Correct Indentation and readability
  - Appropriate decision statements
  - Appropriate repetition statements
  - Appropriate assignment, input and output statements.
- Correctness (80 points)
  - Your program will be tested thoroughly to verify if it plays a series of Mastermind game correctly and computes the required statistics as required in the above description.