

Due: Monday, March 5, 2012 by 11:59 PM

Updated on 2/27/2012 (see red text below)

## Deliverables

The following project files must be submitted to the “graded” assignment in Web-CAT by the due date and time specified above (see the Lab Guidelines for information on submitting project files). You should plan to start on the project not later than Wednesday in lab, and you should make sure that your files are successfully submitted to the “ungraded” assignment in Web-CAT during lab on Wednesday. **Projects sent via e-mail past the deadline at 11:59 PM will not be accepted without a university-approved excuse.**

Files to submit to Web-CAT:

- Number.java
- NumberGuessingApp.java

Due to the nature of the project this week, you will need to demo your project in lab on Wed, Mar 7. Grading will be as follows:  
75 pts - Web-CAT  
25 pts - Demo in lab

## Specifications

**Overview:** Design an application that plays a guessing game with numbers. The program should pick four different random numbers between 0 and 9 (inclusive) in a certain combination, and then repeatedly prompt the user to guess the combination of these four numbers. The user’s guess input must follow the format “# # # #” (four different numbers separated by **one space**). For each guess, report to the user how many of the guessed numbers (out of the four) were totally right (i.e., both the number and position were correct) and how many of the guessed numbers were correct but in the wrong position. Continue accepting guesses until the user guesses correctly or until the user enters “key” or “end” to quit and see the answer. Count the number of guesses and report that value as part of the output that prefaces the guess as shown in the example below.

- **Number.java**

**Requirements:** The Number class will represent the four digits of the computer generated “number” which has four random digits between 0 and 9 inclusive that do not repeat.

**Design:** The Number class has fields, a constructor, and methods as outlined below.

- (1) **Fields:** Instance variables for the first number, the second number, the third number, and the fourth number, which represent the four different random numbers in certain combination.
- (2) **Constructor:** Accepts no parameters and generates four different random numbers between 0 and 9 (inclusive) and sets the four fields. Note that “Different” means no repeating numbers.
- (3) **Methods:** Usually a class provides methods to access (or read) and modify each of its instance variables (i.e., get and set methods) along with any other required methods. The methods for Number are described below.

- `setFirstNum`: accepts a `int` parameter representing the first number.
- `getFirstNum`: returns the first number as an `int`.
- `setSecondNum`: accepts a `int` parameter representing the second number.
- `getSecondNum`: returns the second number as an `int`.
- `setThirdNum`: accepts a `int` parameter representing the third number.
- `getThirdNum`: returns the third number as an `int`.
- `setFourthNum`: accepts a `int` parameter representing the fourth number.
- `getFourthNum`: returns the fourth number as an `int`.
- `toString`: returns a `String` representing of the correct answer which contains the first, second, third, and fourth numbers separated by one space as shown in the examples below.

**Code and Test:** See the Java API documentation (`Math` class or `Random` class) for details on how to generate a random number. Make sure that the integers generated are random numbers between 0 and 9 (inclusive) and that they do not repeat (i.e., they are different from each other).

**[Hint: When using interactions to test your class, you can use the Up-Arrow to avoid retyping previously entered interactions.]**

- **NumberGuessingApp.java**

**Requirements:** For each new game, the program creates an instance of the `Number` class, accepts the user's guess (entered as four numbers separated by spaces), compares the user's numbers to the numbers generated by the `Number` class, and then prints out useful information for user's next guess.

**Design:** `NumberGuessingApp.java` contains one main method and one supporting method called `compareNums`. The `compareNums` method accepts a `Scanner` object and returns no value. Note that the `Scanner` object passed in should be the one you declared in `main` for `System.in` to read input. In this method, a new number combination (an instance of the `Number` class) is created to start a new game. In a *while* statement, do the following: prompt user to input his/her four number guess, compare this combination to the number combination generated in `Number` class; print out the information about how many numbers of the four are correct and in the correct position, and print how many of the four number are correct but in the wrong position. Continue prompting the user to guess until the user gets the correct answer or until the user enters in "key" or "end" to end the game and see the correct answer. Note that non-numeric input from the user should not be case-sensitive (e.g., both `Y` or `y` should be acceptable).

**Code and Test:** After you read in the user's guess as a `String`, create a `Scanner` on the guess so that you can scan the individual numbers. Recall, this is the way you scanned the individual data for each used car in Project 5. In order to facilitate comparison of the two combinations of numbers, place the original numbers generated by `Number` class and user's guessing numbers into two `ArrayLists`. Then use nested *for* statements to compare these two combinations and count

how many numbers of the four are correct and in the correct position, and print how many of the four number are correct but in the wrong position.

Your program will need to accept all values as type string. Your program output should be **exactly** as follows:

```
Program output
Welcome to COMP 1210 Number Guessing Game.

Input four different numbers between 0 and 9 separated by one space,
or input "key" or "end" at any time to see key and end the game.

Start new game?(Y/N): Y
New Game Starting ...

Guess #1: 2 4 3 7
Correct number, Correct position: 1
Correct number, Wrong position: 0
Guess #2: 1 9 0 7
Correct number, Correct position: 1
Correct number, Wrong position: 1
Guess #3: 9 1 0 7
Correct number, Correct position: 1
Correct number, Wrong position: 1
Guess #4: key
The Key: 8 5 9 7

Try again?(Y/N): Y
New Game Starting ...

Guess #1: 2 1 3 6
Correct number, Correct position: 1
Correct number, Wrong position: 1
Guess #2: 1 9 0 7
Correct number, Correct position: 1
Correct number, Wrong position: 1
Guess #3: 4 1 0 2
Congratulations! Smart kid! Your answer is correct.

Try again?(Y/N): N
Thanks for playing.
```