CSCI 136 Programming Exam #0
Fundamentals of Computer Science II
Spring 2012

This part of the exam is like a mini-programming assignment. You will create a program, compile it, and debug it as necessary. This part of the exam is open book and open web. You may use code from the course web site or from your past assignments. When you are done, submit all your Java source files to the Moodle exam #0 dropbox. Please ***double check you have submitted all the required files***.

You will have 100 minutes. No communication with any non-staff members is allowed. This includes all forms of real-world and electronic communication.

Grading. Your program will be graded on correctness and to a lesser degree on clarity (including comments) and efficiency. You will lose a substantial number of points if your code does not compile or crashes on typical inputs.

Overview. You will be developing classes for use in a dice-related game. You will be developing three classes, `Die.java`, `Dice.java`, and `LoadDice.java`. You should probably develop the classes in this order. Stub versions of these classes as well as two test files `fixed.txt` and `random.txt` can be downloaded from here:

http://katie.mtech.edu/classes/csci136/dice.zip

Die class. Your first task is to develop a class that represent a single die. The most familiar type of die is 6-sided and can take a value of 1, 2, 3, 4, 5, or 6. As players of Dungeons and Dragons know, there are other types of dice such as 4-sided, 8-sided, 12-sided and 20-sided.

You class will support dice with between 1 and 100 sides inclusive. A 1-sided die is a ball that always has a value of 1, a 100-sided die has 100 faces and can take any value between 1 and 100. In your implementation, any attempt to create a `Die` object with ***fewer than 1 side or more than 100 should result in a runtime exception.***

A die can be rolled to get a random value in its range. Alternatively a die can also be set to a specific value. The specific value must be within the allowed range of that die. Any attempt to ***set a die to an invalid value causes the die to be set to a random value*** (i.e. it is like someone rolled the die instead).

Here is the public API to the Die class:

```
public class Die
-----------------------------------------------------------------------------------------
        Die(int sides)             // Create die with given number of sides
        Die(int sides, int value)  // Create die with given # of sides and initial value
   void roll()                     // Roll the die setting a new random value
   void setValue(int value)        // Set the die to the given value, set to a random
                                   // value if the value is out of range
    int getValue()                 // Get the current value of the die
    int getSides()                 // Get how many sides the die has
 String toString()                 // Return string representation, format example:
                                   // "3(6d)" is a 6-sided die with current value 3
```

You may want to create a `main` method for testing purposes, but it is not required and will not be graded.

Dice class. The `Dice` class represents a collection of 0 or more `Die` objects. This class can do the following: add a new die to the collection, roll all the dice, return a string representation of the dice, and calculate various quantities about the dice. For example, assume our collection has three 6-sided dice with values 1, 3, 4 and two 10-sided dice with values 3 and 9. The string representation would be "1(6d) 3(6d) 4(6d) 3(10d) 9(10d)". The calculated quantities are:

- sum - The sum of the values of all dice in the collection, 1 + 3 + 4 + 3 + 9 = 20
- max - The largest value of any dice in the collection, max(1, 3, 4, 3, 9) = 9
- same - The maximum number of dice that have the same value. So for the above example with dice having values 1, **3**, 4, **3**, 9 the "same" value is 2 since there are 2 dice with the same value (a value of 3). If we had a collection of dice with values 1, **9**, 8, **9**, **9**, 8, 3, the "same" value would be 3 since there are 3 dice with the same value (a value of 9).

Here is the public API to the Die class:

```
public class Dice
-------------------------------------------------------------------------------------
   void add(int sides)            // Add die with the given sides and a random value
   void add(int sides, int value) // Add die with the given sides and given value
   void roll()                    // Roll all the dice
    int sum()                     // Compute the sum of all the dice values
    int max()                     // Compute max value over all the dice
    int same()                    // Compute max number of dice having the same value
 String toString()                // Return a string representation of all the dice
                                  // Format example: "3(4d) 5(6d) 12(20d)"
```

You may want to create a `main` method for testing purposes, but it is not required and will not be graded.

LoadDice program. This program has a `main` method that reads in data from a text file ***using Java file I/O*** (NOT standard input). The filename is given as the first command line argument. If no arguments are given, the error "No filename!" is printed and the program exits. If the file can't be read, "File read error!" is printed and the program exits. Here are two example files:

```
georgia   4    4
sammy     6    3
georgia   6    2
ROSS      20   15
pedro     4    2
georgia   10   8
georgia   10   8
sammy     1    1
Sammy     50   50
ross      8    7
georgia   12   8
sammy     100  50
```

fixed.txt

```
sammy    6   0
Georgia  4   0
ross     20  0
pedro    4   0
georgia  4   0
Georgia  4   0
sammy    1   0
sammy    50  0
Ross     8   0
georgia  4   0
georgia  4   0
georgia  4   0
```

random.txt

Each line in the file contains three columns separated by whitespace. The first column is the name of a player. Names of players are ***case insensitive*** (i.e. sammy and Sammy are the same player). The second column is the number of sides of the die. The third column is the current value of the die (an invalid value means that die should be given a random value). The program should read in all the data from the file and produce for each player: their name (in lowercase), a summary of the dice held, the player's *sum*, *max* and *same* values. Example runs:

```
% java LoadDice fixed.txt
pedro
2(4d)
sum  = 2
max  = 2
same = 1

ross
15(20d)  7(8d)
sum  = 22
max  = 15
same = 1

sammy
3(6d)  1(1d)  50(50d)  50(100d)
sum  = 104
max  = 50
same = 2

georgia
4(4d)  2(6d)  8(10d)  8(10d)  8(12d)
sum  = 30
max  = 8
same = 3
```

```
% java LoadDice random.txt
pedro
4(4d)
sum  = 4
max  = 4
same = 1

ross
20(20d)  4(8d)
sum  = 24
max  = 20
same = 1

sammy
6(6d)  1(1d)  12(50d)
sum  = 19
max  = 12
same = 1

georgia
3(4d)  2(4d)  1(4d)  1(4d)  4(4d)  1(4d)
sum  = 12
max  = 4
same = 3
```

The values in this example are random! You are unlikely to get the same output.

```
% java LoadDice blahblah.txt
File read error!
```

```
% java LoadDice
No filename!
```