CSCI 1302
Spring 2009

Homework 3

A Domino Class

The classic domino set is played with 28 dominoes sometimes referred to as bones. Each domino bone is divided into halves, each half has either a number of pips (black spots) ranging from 1 to 6 or is blank. The half with the greatest number of pips is called the major suit, and the other half is the minor suit. A domino bone whose minor suit has no pips is referred to as a blank. A domino that has identical halves is called a double. Additionally, the weight of a domino bone is the total number of pips in both halves. Moreover, a domino bone is ranked according to its major and minor suits. A domino bone is ranked higher than another domino when its major suit is greater than the other bone's major suit or when it has the same major suit as the other bone but its minor suit is greater than the minor suit of the other bone. Finally, a domino is usually named by specifying the major suit first and then the minor suit, for example, a domino whose major suit is 5 and minor suit is 3 is named as 5 - 3. The following table illustrates the major, minor and weight of several domino bones as wells as whether they are a double and/or a blank.

Major	Minor	Blank	Double	Weight	Name
2	0	Yes	Yes	2	2 - 0
5	3	No	No	8	5 - 3
4	4	No	Yes	8	4 - 4
5	2	No	No	7	5 - 2
0	0	Yes	Yes	0	0 - 0

The domino bones in the previous example can be sorted by their rank (from highest to lowest rank) as follows: 5 - 3, 5 - 2, 4 - 4, 2 - 0 and 0 - 0. Note that the 5 - 2 domino has a higher rank than the domino 4 - 4, even though the latter has a higher weight.

You have been given a **skeleton** for a class called Domino (**Domino.java**) which represents a domino bone in a standard domino set.

For the first part of the assignment, you will fill out the code so that the Domino class works as required. Each Domino object has two integer attributes - minor and major. Both major and the minor are integers between 0 and 6, but the major is always greater than or equal to the minor.

The **Domino** class should allow any domino in a standard domino set to be created, but it should **NOT** allow any other numbers for the major or the minor. If someone tries to create a domino with a major suit of 7 or with a minor suit of 10, for example, the program should default to the blank domino (minor 0 and major 0). You will need to fill out each method as described in the comments at the top of the method. The code currently has **return** statements for any method that returns something so that it will compile without errors - but you will need to change all of these lines so that the methods return the proper values.

As you are writing the code, use the **DominoTester** program to test your class to make sure it is working correctly. When every method of your Domino class is properly implemented with no errors, the output will look something like this:

```
Major Suit of Domino 1 is correct!
Minor Suit of Domino 1 is correct!
Major Suit of Domino 2 is correct!
Minor Suit of Domino 2 is correct!
Domino 1: 6 - 0
toString is working correctly for domino 1
Domino 2: 3 - 3
toString is working correctly for domino 2
isBlank is working correctly for domino 1
isDouble is working correctly for domino 2
getWeight is working correctly for domino 1
getWeight is working correctly for domino 2
Randomly Generated domino: 5 - 2
Invalid domino defaulted to: 0 - 0
```

Additionally, write a client class called **RandomDomino.java** that randomly generates 20 Domino objects (you can just use the default constructor) and prints them out, one by one. At the end of printing out the 20 domino bones, your program should print the heaviest domino bone with the highest rank, the lightest domino bone with the lowest rank, the number of blanks and the number of doubles among the dominos . The output of your program may look similar to the following:

```
20 randomly generated domino bones:
4 - 1
5 - 0
3 - 2
5 - 1
6 - 1
4 - 2
2 - 2
4 - 0
5 - 4
4 - 3
5 - 3
3 - 3
4 - 2
5 - 5
3 - 0
6 - 1
4 - 0
6 - 2
3 - 1
6 - 0

Lower rank: 2 - 2
Higher rank: 6 - 2
Heaviest domino: 5 - 5
Lightest domino: 3 - 0
Number of doubles:3
Number of blanks:5
```

Deliverables:

- Domino.java
- RandomDomino.java

Hints:

RandomDomino.java

- Use a **for** loop and one Domino object reference variable to create the 20 random domino objects.
  In each iteration of the loop:
  - use an assignment statement to create a new Domino object and reassign it to the Domino object reference variable.
  - keep track of the number of times a double or blank appears among the domino bones.
  - determine the heaviest bone
  - determine the lightest bone
  - determine the heavier bone.
  - determine the lightest bone.
- To determine the heaviest bone and the lightest bone.
  - Create an object lightDomino that has the highest weight of the domino set using the appropriate constructor of the class Domino.
  - Create an object heavyDomino that has the lowest weight of the domino set using the appropriate constructor of the class Domino.
  - Use the method getWeight to check whether a new generated domino is heavier than the current heavyDomino or lighter than the bone in lightDomino.
- To determine the bones with higher and smaller rank respectively.
  - Create an object minDomino that has the highest rank of the domino set using the appropriate constructor of the class Domino.
  - Create an object maxDomino that has the lowest weight of the domino set using the appropriate constructor of the class Domino.
  - Within the for loop:
    - Use the method isHigher to determine whether the rank of the new random domino generated within the loop is higher than the current maxDomino or lower than the current minDomino.
- Use the method toString of the Integer class to obtain a String object that stores the sequence of characters that represent an integer value passed as parameter to the method. For example, let `aNumber` be an integer variable with initial value 9863, Integer.toString(aNumber) will create String object that stores the sequence of characters "9863".