CSE 142, Autumn 2006
Programming Assignment #5: Guess 2D (20 points)

Part A Due: Monday, October 30, 2006, 9:00 PM (optional; +1 late day)
Part B Due: Wednesday, November 1, 2006, 9:00 PM (20 points)

Problem Description:

This assignment will give you practice with `while` loops and random numbers. Your program allows the user to play a guessing game. The program thinks of a random 2D point, and the user guesses points until the right answer is found.

This assignment will be due in two parts: a partial version and a second complete version. **The initial turnin, Part A, is optional and is worth no points, but will earn you an extra late day. Part B is worth 20 points**. For Part A, you will turn in a Java class named `Guess2Da` in a file named `Guess2Da.java`. For Part B, you will turn in a Java class named `Guess2Db` in a file named `Guess2Db.java`. Let's examine Part B first and then discuss what's different about Part A.

Program Behavior (Part B):

```
This program is a 2-D guessing game.
I will think of a point
between (1, 1) and (10, 10)
and give hints until you guess it.

Guess x and y: 6 6
Incorrect.
Guess x and y: 3 6
Warmer.
Guess x and y: 3 3
Warmer.
Guess x and y: 2 2
Colder.
Guess x and y: 1 2
Warmer.
Guess x and y: 1 5
Hot!
Guess x and y: 1 4
You got it right in 7 guesses!
Do you want to play again? y

Guess x and y: 5 5
Incorrect.
Guess x and y: 8 5
Hot!
Guess x and y: 9 5
You got it right in 3 guesses!
Do you want to play again? y

Guess x and y: 5 2
Incorrect.
Guess x and y: 2 2
Warmer.
Guess x and y: 1 2
Warmer.
Guess x and y: 1 8
Warmer.
Guess x and y: 1 7
Hot!
Guess x and y: 1 6
You got it right in 6 guesses!
Do you want to play again? n

Overall results:
Games played  = 3
Total guesses = 16
Guesses/game  = 5.3
Best game     = 3
```

In the final second version of the game, the computer chooses a point between (1, 1) and (10, 10) inclusive. The user is repeatedly asked to guess the point. For each incorrect guess, the program will tell the user whether it is warmer (closer than the previous guess), colder (equidistant or further away than the previous guess), or hot (within a distance of 1 from the right answer). When the game ends, the program reports how many guesses were needed.

After each game, the program asks the user to play again. You may assume that the user will give a one-word answer. The program should continue playing if the user's response begins with a lower- or upper-case Y. That is, answers such as `"y"`, `"Y"`, `"YES"`, `"yes"`, `"Yes"`, or `"yeehaw"` would all indicate that the user wants to play again. If the answer is not a definitive Yes answer, assume that the user does not want to play again. For example, responses such as `"no"`, `"No"`, `"0"`, and `"Pass"` are all assumed to mean No.

Once the user ends a game and chooses not to play again, the program prints overall statistics about the games played. The total number of games, total guesses for all games, average number of guesses per game (as a real number rounded to the nearest tenth), and best game (fewest guesses) are displayed. Your statistics should present correct information regardless of how many guesses were needed or games were played, within the limits of Java's maximum values. (In other words, the statistics should be correct even if the user has a game where they need a large number such as 100 guesses to get the right answer, and even if the user plays a large number of games, and even if only 1 game is played, and so on.)

The log of execution on this page demonstrates your program's behavior. Your program may generate different random points, but your output structure should match this one exactly.

You may assume that the user will need fewer than 1000 guesses to solve any given game.

Program Behavior (Part A):

```
CORRECT ANSWER: 9 5
Guess x and y: 1 2
Incorrect.
Guess x and y: 8 6
Warmer.
Guess x and y: 5 6
Colder.
Guess x and y: 10 6
Warmer.
Guess x and y: 8 5
Hot!
Guess x and y: 9 5
You got it right in 6 guesses!
```

In Part A, only a single guessing game is played. Part A does not need to print any initial welcome message or prompt the user to play additional games. To help your debugging, Part A should also print the correct random answer at the start of the program.

You will receive an extra late day if you submit Part A on time. No bonus will be awarded if Part A is submitted late or is not submitted.

To get the late day, your program must roughly match the correct expected behavior such as the example log of execution at left. Your program may generate different random points, but your output structure should match this one. It will be examined for external correctness (output) only. You will not receive feedback on your Part A turnin before you submit Part B.

Implementation Guidelines:

The hints about guesses being warmer or colder are based on distances between points. The formula to compute the distance between two points is to take the square root of the squares of the differences in x and y between the two points. For example, in the log of execution for Part A, the distance between (9, 5) and (1, 2) is $\sqrt{((9-1)^2 + (5-2)^2)}$ or roughly 8.544. The distance between (9, 5) and (8, 6) is $\sqrt{((9-8)^2 + (5-6)^2)}$ or roughly 1.414. Therefore, (8, 6) is closer than (1, 2) and the hint given is "Warmer." If the current guess is equidistant or further from the right answer than the previous guess, the hint should be "Colder." Regardless of the previous guess, if the current guess is within a distance of 1.0 from the right point, the hint should be "Hot!". The first wrong guess has no point of comparison and therefore just prints "Incorrect.".

You may wish to implement this behavior by representing the correct answer and user's guess as `Point` objects (as shown in Chapter 3) and using their `distance` method. Remember to `import java.awt.*;` if you use `Point` in your program.

If the user guesses the number correctly in one try, you may still print the text `"You got it right in 1 guesses"` though the word `"guess"` would be more appropriate. We will not test this case when grading.

Assume valid user input. When the user is prompted for numbers, the user will type valid integers in the proper range. When the user is prompted to play again, the user will type a one-word string as their answer. To deal with the yes/no response from the user, you may want to use some of the `String` class methods described in Chapters 3 and 4 of the book.

Stylistic Guidelines for Part B:

Structure your solution using static methods that accept parameters and return values where appropriate. For full credit on Part B, you must have at least 2 methods other than main in your program: a method to play a single game with the user (not multiple games), and a method to report the overall statistics to the user.

You should define other methods if they are useful to eliminate redundancy. If you like, you can place the loop that plays multiple games and prompts the user to play another game in your `main` method.

For full credit, you must define a class constant for the maximum x/y number used in the guessing game. The sample log shows the user making guesses from 1 to 10, but by introducing a constant for 10, you should be able to make the program play the game over any other range starting with (1, 1) just by changing the constant's value. For example, if it is changed to 5, your program picks (x, y) points between (1, 1) and (5, 5). See the course web site for expected output.

For this assignment you are limited to the language features in Chapters 1 through 5 of the textbook. Indent your code properly and use white space to make your program more readable. Give meaningful names to methods and variables, and follow Java's naming standards as specified in Chapter 1 of the textbook. Localize variables whenever possible. Include a comment at the beginning of your program with basic information and a description of the program and include a comment at the start of each method. Since this program has longer methods than past programs, also put brief comments inside the methods explaining the more complex sections of your code.