AP Computer Science Programming Project - NumberGuessingGame

ASSIGNMENT OVERVIEW

In this assignment you'll be creating a main program called `NumberGuessingGame.java`, which will allow the user to play a number guessing game on the computer.

This assignment is worth 30 points and is due on the *crashwhite.polytechnic.org* server at 23:59:59 on the date given in class.

BACKGROUND

The rules for playing the Number Guessing Game are easy:

1. One person identifies a random secret integer between 1 and 10, inclusive.
2. The second person has three guesses to try to identify the number. After each guess, the guesser gets a hint as to whether they're guess was "too high" or "too low."
3. If the guesser identifies the secret number by the third guess, they win. Otherwise, they lose, and the secret number is revealed.

PROGRAM SPECIFICATION

Create a single Java main program that:

a. creates a random number between 1 and 10 inclusive.
b. asks the user to guess the number
c. if the user gets it right, congratulate them, and the game is over
d. if the user doesn't guess the number, tell them if they guessed too high or too low
e. give the user three tries to guess the number
f. if they get don't get it right in three tries, tell them what the number was, and the game is over
g. ask them if they want to play again, and keep playing the game until they indicate they want to quit.

DELIVERABLES

`NumberGuessingGame.java`

You may work on this file as a BlueJ project, but you will only upload the `NumberGuessingGame.java` file.

To submit your assignment for grading, copy your file to your directory in `/home/studentID/forInstructor/` at *crashwhite.polytechnic.org* before the deadline.

ASSIGNMENT NOTES

As you start to work on this assignment you'll want to consider a number of questions:

- How many loops does this program require?
- Are they going to be counting loops (with a known number of iterations) or conditional loops (that require checking a value)?
- If conditional, what are the conditions going to be on the loops?

You'll also be able to consider the different tasks that the program needs to manage, and make sure that your program includes these items:

- Create a random number for the user to guess.

- Get their guess
- Compare guess to random number and give appropriate feedback
- Keep track of how many guesses they've made (what kind of loop? counting? or conditional?)
- Ask them if they want to play again (what kind of loop? counting? or conditional?)

You might be tempted to write this program all at once, and then start testing it. Because there are a number of pieces to this program, and perhaps multiple loops and conditions being managed, it makes very good sense to break it down into pieces that can be written and tested individually.

Testing programs that work with random number generators can be challenging. One of the ways you can make testing your program a little easier is by *not* using a random number generator at first. Instead of having the computer select a random number:

```
    int randNum = (int) (Math.random() * 10 + 1);
```

set the random number to a fixed value:

```
    int randNum = 7;
```

Now you can test the logic of your program knowing exactly what the number is that your trying to guess.

Another way to to test your program is by including a `println` statement that displays the random number for you during the debugging of your program. You can see an example of this strategy in the *Sample Interactions* at the end of this document. Of course, you should remove this `println` statement before turning in your assignment or showing the game to someone else.

GETTING STARTED

1. With paper and pencil, and perhaps in collaboration with a partner, identify what the main components are that you'll need to include in your program.

2. Sketch out the basic flow of your program using a flowchart, and write some pseudocode that you can use to begin implementing those main components.

3. Create a new project in BlueJ that will allow you to manage this assignment.

4. Create a main program that will run the GuessingGame.

5. Enter pseudocode as comments in the editor, and then fill in more details for various parts of the code.

6. Test each bit of code as you go, making sure that one piece works before you proceed on to the next section. You'll repeatedly run through this edit-compile-test, edit-compile-test process to progressively find bugs and fix them *while* you're writing your program, not afterwards.

7. Save your program from time to time.

8. Once a day or so, archive/zip your BlueJ Craps folder and save a backup copy of it on another device or machine: a flash drive, your home folder on the *crashwhite.polytechnic.org* server, etc.

9. When your program is completed (but before the deadline), copy a final archived package to the server as indicated above.

QUESTIONS FOR YOU TO CONSIDER (NOT HAND IN)

1. Did you use `while`-loops with conditions to manage this game, or `for`-loops with a `break` statement? Why did you choose that style of loop for your program?
2. In the Number Guessing Game, is there an optimal guessing strategy that will always allow you to win the game? Is there an optimal guessing strategy that will improve your odds?
3. With only three guesses, the Number Guessing Game never lasts very long. What if you were to increase the numbers to choose from to 20, or 50, or 100? That's easily done, but you'd need to change the number of guesses allowed, too. (Having only three guess for a number between 1 and 100 wouldn't be very fun!) How would we calculate the correct number of guesses to allow so that the gameplay is the same, ie. you can optimize your winning strategy, but not guarantee and win?

SAMPLE INTERACTIONS

```
Let’s play the Number Guessing Game!
DEBUG: 9
Guess a number between 1 and 10.
Guess # 1: 5
You guessed too low.
Guess again!
Guess a number between 1 and 10.
Guess # 2: 10
You guessed too high.
Guess again!
Guess a number between 1 and 10.
Guess # 3: 8
You guessed too low.
Game over!
The number was 9
Wanna play again (Y/N)? y
DEBUG: 6
Guess a number between 1 and 10.
Guess # 1: 5
You guessed too low.
Guess again!
Guess a number between 1 and 10.
Guess # 2: 6
You got it!
Game over!
The number was 6
Wanna play again (Y/N)? n
Thanks for playing the Number Guessing Game!
```