Project 2: Simon Says

Your task for this project is to write a Simon Says game. The program should generate an ever-increasing sequence of random values, and the user needs to type in the sequence. The game should have three difficulty levels: easy (colors), medium (digits) and hard (letters). At the beginning of each round, the program should ask the user which difficulty level they want to play at. At the end of each round, it should tell them the length of the longest sequence they entered correctly before messing up. An example will hopefully make this more clear:

```
Which difficulty level do you want: easy, medium, or hard? potato

That is not a valid difficulty level!

Which difficulty level do you want: easy, medium, or hard? easy

Simon says: red

(The program should then pause for 2 seconds and then write out
200 blank lines so that the screen is empty.)

Input: red

Right! Your score is 1.

Simon says: red green

(The program should then pause for 2 seconds and then write out
200 blank lines so that the screen is empty.)

Input: red blue

Incorrect! Your score for this round was 1.

Do you want play again (yes or no)? potato

That is not a valid answer!

Do you want to play again? no

Ok, goodbye!
```

On the easy mode, your program should randomly choose between four colors: red, green, yellow and blue. In medium mode, it should randomly choose digits between 1 and 9, inclusive (e.g. a medium sequence might be `1 9 8 3 9 9`). In hard mode, it should randomly choose lowercase letters (e.g. `t e v b w t a`). When checking a user's answer,

spacing and capitalization should be ignored (e.g. if the sequence is `red green red red blue` and the user types `Redgreenredredblue`, that should be counted as correct.) As shown in the example above, be sure to handle invalid user responses to questions about the difficulty level and playing again.

Hints

One way to approach this would be to store the sequence Simon is saying in a string and then use the String class's equals method to compare what the user enters to the correct sequence. Recall that the String class has many useful methods, such as equalsIgnoreCase (useful for ignoring differences in capitalization when comparing two strings) and replace (useful for, among other things, removing spaces from within a string).

The statement `Thread.sleep(2000);` will cause a Java program to pause for two seconds before continuing. To use this method from within main, you need to make a slight change to the way your main method is declared:

```
public static void main(String[] args) throws Exception
```

Recall that to generate a random number between 1 and 10 inclusive, you can use:

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

You can do math with characters, which might be helpful when generating a random sequence of letters. For example:

```
char letter = 'a';
letter = (char) (letter + 4); // letter is now 'e'
```

Rubric

Projects that don't compile will receive a zero.

[10 pts] The user can choose the difficulty level, and invalid choices are rejected.

[15 pts] Easy mode works as described.

[15 pts] Medium mode works as described.

[15 pts] Hard mode works as described.

[10 pts] Spacing and capitalization are ignored.

[5 pts] The user's score is displayed at the end of each round.

[10 pts] The use can choose if they want to play again, and invalid choices are rejected.

[20 pts] The program is clearly organized, commented, and follows standard coding practices, including variable and class names.