Guess-a-Word

1. Problem Description

Based on the phenomenal success on previous projects, your angel investor has come back to your firm for yet another application. You are to develop a simple 'Guess-a-Word' game in which the computer selects a random word, and the child repeatedly guesses letters until the word is uncovered. We will keep track of the number of guesses but there is no maximum number of guesses.

With each successive guess, show the word with the correct letters filled in, and also all the letters that have been guessed so far. If the child guesses the same letter more than once, do not count that guess against them. The guesses should be converted to upper case.

Provide a set of at least ten words to guess from. The instructor's example uses the words "one" through "ten", however you are free to provide as many words as you would like.

Keep track of the number of guesses the child takes and display this number at the end. Your application should only play the game once per execution (i.e. no "Would you like to play another game?" is required).

2. Notes

- Turn in only your source files. Make sure your class is not in a package (that is, it is in the *default* package).
- If you are using an eternal list of words in a file, you must turn that file in as well.

3. Required Main Class

GuessWord

4. Required Input

A series of uppercase or lowercase letters

5. Required Output

Your output should look something like the following example. Notice the multiple, repeated guesses ('H' and 'E'). It must include *your* name.

```
Guess-a-Word - E. Eckert

_ _ _ _ _    Used letters: {}

Enter a letter: a

_ _ _ _ _    Used letters: {A}

Enter a letter: b

_ _ _ _ _    Used letters: {AB}

Enter a letter: c

_ _ _ _ _    Used letters: {ABC}

Enter a letter: e

_ _ _ E E    Used letters: {ABCE}

Enter a letter: t

T _ _ E E    Used letters: {ABCET}

Enter a letter: h

T H _ E E    Used letters: {ABCETH}

Enter a letter: h

Enter a letter: e

Enter a letter: r

T H R E E    Used letters: {ABCETHR}

You guessed it in 7 tries.
```

6. Hint(s)

In Lecture 5c, I lay out the `main` method. Of course, you are free to do it a different way, as long as the program functions the same. In my approach, you must write four key methods: `pickWord`, `printWord`, `getUniqueGuess`, and `wordGuessed`. As usual, I suggest starting off by creating trivial versions of each of these just to get the program running. Perhaps `pickWord` can always return "TEST", `printWord` will always just print the secret word and all the guesses, `getUniqueGuess` can get a single letter from the user and not worry about it being unique and `wordGuessed` can always just return false. It won't be much fun, but your program will run. Next tackle each method to make your program more and more functional. Test as you go.

A useful method might be: `int String.contains(char c)`

Reading a single letter can be done with: `in.getLine().toUpperCase().charAt(0);`