MagPie Lab

**getResponse()**

My code:

if (statement.length() == 0)

{

return "Say something, please";

}

The getResponse() method works by finding words in the user’s responses and then saying something that fits their response depending on the words the user uses. In this code in particular, the rule was that if the user doesn’t say anything and thus, the length(or # of words in the response) is 0, the response is “Say something, please.” because no words were entered.

Similar?

My code is similar, yet not the same because the answer uses “response = “Say something, please.” ” instead of the return method, which I put as “return “Say something, please.””

More or Less Efficient?

I think both codes have the same level of efficiency because they both work the same, returning“Say something, please” when the user input is nothing.

What could you have done differently?

I should have done the “response =” way to be consistent with the rest of the code, but for the purposes of efficiency, both work out the same.

What did you learn from comparing your code to the provided answer?

I learned that there are other options other than the return method within the if statement. I could do multiple if statements with the print statement equalling a certain word and then at the end just do “return word”.

**findKeyword()**

MY CODE:

String phrase = statement.trim();

int psn = phrase.toLowerCase().indexOf(goal.toLowerCase(), startPos);

while(psn >= 0)

{

String before = " ";

String after = " ";

if(psn > 0)

{

before = phrase.substring(psn - 1, psn).toLowerCase();

}

if (psn + goal.length() < phrase.length())

{

after = phrase.substring(psn + goal.length(),

psn + goal.length() + 1).toLowerCase();

}

if(((before.compareTo("a") < 0 ) || (before.compareTo("z") > 0)) && ((after.compareTo("a") < 0 ) || (after.compareTo("z") > 0)))

{

return psn;

}

psn = phrase.indexOf(goal.toLowerCase(), psn + 1);

}

SIMILAR?: The code is pretty similar to the one above since they both work out the same, efficiently and effectively. However, in the answer, phrase and goal are already initiated to equal the .LowerCase() method so I didn’t have to repeat the same method over and over again. I could have just initiated phrase and psn to equal it in the first place. Also, in my code, I didn’t initiate “goal” to equal goal.toLowerCase(), so my code was a bit longer in that regard. Instead of having to repeat the .toLowerCase() method, I should just initiate my variables to equal the method in the beginning.

MORE or LESS EFFICIENT? The efficiency is the same, but if /I wanted to be less repetitive, I could have stated that phrase, psn and goal were all the .LowerCase() methods so I didn’t have to keep typing that over and over again throughout my code.

WHAT COULD YOU HAVE DONE DIFFERENTLY? I could have initiated the variables as .LowerCase() to avoid making meaningless mistakes and making my code more readable.

WHAT DID YOU LEARN FROM COMPARING YOUR CODE TO THE PROVIDED ANSWER? I learned that to make my code more readable and accessible, I should do as much as I can to make it less complicated by initiating my variables with the methods they continuously use.

**Exercise 3:**

else if (statement.indexOf("dog") >= 0

|| statement.indexOf("cat") >= 0

|| statement.indexOf("turtle") >= 0

|| statement.indexOf("fish") >= 0)

{

response = "Tell me more about your pets.";

}

else if (statement.indexOf("Robinette") >= 0)

{

response = "He sounds like a pretty dank teacher.";

}

HOW IT WORKS:

This code creates specific responses to certain words that the user inputs. If the statement.indexOf(“word”)>= 0, then the word is in the user input and the Magpie response is “Tell me more about the “word.” The “||” symbol means “or” so if any of those words are found, then the same response is executed. For example, “dog,” “cat,” “turtle,” and “fish” are all pets so they’re all in the same if statement and the same response is outputted. For the second one about Robinette, the computer has a different specified response if the user types in Robinette and the computer finds it with the “statement.indexOf(“Robinette”).

WHAT I DID WRONG:

However, instead of using the indexOf() method, I should have used the findKeyword() method so that the computer knows that “I know” isn’t “No” but “know.” I changed it to the findKeyword() method and it ended up working perfectly:

else if (findKeyword(statement, "dog",0) >= 0)

|| (findKeyword(statement, "cat",0)>= 0)

|| (findKeyword(statement, "turtle",0) >= 0)

|| (findKeyword(statement, "frog",0)>= 0)

{

response = "Tell me more about your pets.";

}