**Magpie Chatbot Lab: Student Guide**

**Activity 3: Better Keyword Detection**

In the previous activity, you discovered that simply searching for collections of letters in a string does not always work as intended. For example, the word “cat” is in the string “Let’s play catch!,” but the string has nothing to do with the animal. In this activity, you will trace a method that searches for a full word in the string. It will check the substring before and after the string to ensure that the keyword is actually found.

You will use some more complex String methods in this activity. The String class has many useful methods, not all of which are included in the AP Computer Science Java Subset. But they can be helpful in certain cases, so you will learn how to use the API to explore all of the methods that are built into Java.

**Prepare**

Have available:

* the code for the Magpie3.java
* the code for the MagpieRunner3.java
* the code for the StringExplorer.java
* the API for the String class
* a computer with your Java development tools

Create a project and folder for your files named LastnameMagpieThree

Name your copy of this word document LastnameMagpieChatbotThree.dox.

**Exploration: Using the API**

One of the major benefits of using Java as a programming language is that so many library classes have already been created for it.

Open the program StringExplorer. It currently has code to illustrate the use of the indexOf and toLowerCase methods.

 Open the API for String. Scroll down to the Method Summary section and find the indexOf(String str) method. Follow the link and read the description of the indexOf method. What value is returned by indexOf if the substring does not occur in the string?  \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Add the following lines to StringExplorer to see for yourself that indexOf behaves as specified:

int notFoundPsn = sample.indexOf("slow");

System.out.println("sample.indexOf(\"slow\") = " + notFoundPsn);

Read the description of indexOf(String str, int fromIndex). Add lines to StringExplorer that illustrate how this version of indexOf differs from the one with one parameter.

This lab activity will use a variety of different String methods. Consult the API whenever you see one with which you are unfamiliar.

**Exploration: Understand the new method**

This version of the Magpie class has a method named findKeyword to detect keywords. This method will only find exact matches of the keyword, instead of cases where the keyword is embedded in a longer word. Run it.

private int findKeyword(String statement, String goal, int startPos)

{ String phrase = statement.trim().toLowerCase();

goal = goal.toLowerCase();

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

while (psn >= 0)

{ String before = " ", after = " ";

if (psn > 0)

{

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

}

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

{

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

psn + goal.length() + 1);

}

**/\* determine the values of psn, before, and after at this point in the method. \*/**

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

&&

{ ((after.compareTo ("a") < 0 ) || (after.compareTo("z") > 0)))

} return psn;

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

} return -1;

Read through the findKeyword method. To ensure that you understand it, trace the following method calls.

findKeyword("She's my sister", "sister", 0); findKeyword("Brother Tom is helpful", "brother", 0); findKeyword("I can't catch wild cats.", "cat", 0); findKeyword("I know nothing about snow plows.", "no", 0);

Write the value of each of the variable psn, before, and after each time the program control reaches the point in the method indicated by the comment.

Example: findKeyword("yesterday is today's day before.", "day", 0);

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration | psn | before | after |
| 1 | 6 | “r” | “ “ |
| 2 | 15 | “o” | “’” |
| 3 | 21 | “ “ | “ “ |

Use the tables below to trace the calls.

findKeyword("She's my sister", "sister", 0);

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration | psn | before | after |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

findKeyword("Brother Tom is helpful", "brother", 0);

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration | psn | before | after |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

findKeyword("I can't catch wild cats.", "cat", 0);

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration | psn | before | after |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

findKeyword("I know nothing about snow plows.", "no", 0);

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration | psn | before | after |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Exercise: Use the new method**

Repeat the changes you made to the program in Activity 2, using this new method to detect keywords.

**Questions: Prepare for the next activity**

Single keywords are interesting, but better chatbots look for groups of words. Consider statements like “I like cats,” “I like math class,” and “I like Spain.” All of these have the form “I like *something*.” The response could be “What do you like about *something*?” The next activity will expand on these groups. You will get to add one of your own, so it’s a good idea to start paying close attention to common phrases in your own conversations.

Zip your project and submit in Edmodo along with your Magpie Chatbot word document in the Magpie Chatbot Three assignment. Be sure to include your StringExplorer.java file, too.