AP® Computer Science A Magpie Chatbot Lab Student Guide

The AP Program wishes to acknowledge and thank Laurie White of Mercer University, who developed this lab and the accompanying documentation.

CollegeBoard

Magpie Chatbot Lab: Student Guide

Introduction

From Eliza in the 1960s to Siri and Watson today, the idea of talking to computers in natural language has fascinated people. More and more, computer programs allow people to interact with them by typing English sentences. The field of computer science that addresses how computers can understand human language is called Natural Language Processing (NLP).

NLP is a field that attempts to have computers understand natural (i.e., human) language. There are many exciting breakthroughs in the field. While NLP is a complicated field, it is fairly easy to create a simple program to respond to English sentences.

For this lab, you will explore some of the basics of NLP. As you explore this, you will work with a variety of methods of the `String` class and practice using the `if` statement. You will trace a complicated method to find words in user input.

Activity 1: Getting Acquainted with Chatbots (Optional)

Chatbots are programs that are designed to respond like humans to natural language input. Before you write code to create your own chatbot, you will explore some existing chatbots.

Start

Go to chatbots.org. Try out several of the chatbots and find one to use for this activity. Your teacher may have a specific list of chatbots for you to try.

Exploration

Have several conversations with your chatbot and answer the following questions:

- How does it respond to "where do you come from"?
- What is the most interesting response?
- What is the most peculiar response?
- How does it respond to "asdfghjkl;"?

Exercises

Work with another group and have two different chatbots converse with each other.

Questions

Simple chatbots act by looking for key words or phrases and responding to them.

1. Can you identify keywords to which your chatbot responds?
2. Think of several keywords and the responses they might cause.

Activity 2: Introduction to the `Magpie` Class

In this activity, you will work `Magpie`, with a simple implementation of a chatbot. You will see how it works with some keywords and add keywords of your own.

Prepare

Have available:

- the code for the `Magpie`
- the code for the `MagpieRunner`
- a computer with your Java development tools

Start

Get to know the `Magpie` class. Run it, using the instructions provided by your teacher.

How does it respond to:

- My mother and I talked last night.
- I said no!
- The weather is nice.
- Do you know my brother?

Exploration

Look at the code. See how the `if` statement assigns a value to the response and returns that response. The method `getRandomResponse` picks a response from a group of `String` objects.

Exercises

Alter the code:

- Have it respond "Tell me more about your pets" when the statement contains the word "dog" or "cat." For example, a possible statement and response would be:

Statement: I like my cat Mittens.

Response: Tell me more about your pets.

- Have it respond favorably when it sees the name of your teacher. Be sure to use appropriate pronouns! For example, a possible statement and response would be:

  Statement: Mr. Finkelstein is telling us about robotics.

  Response: He sounds like a good teacher.

- Have the code check that the statement has at least one character. You can do this by using the `trim` method to remove spaces from the beginning and end, and then checking the length of the trimmed string. If there are no characters, the response should tell the user to enter something. For example, a possible statement and response would be:

  Statement:

  Response: Say something, please.

- Add two more noncommittal responses to the possible random responses.

- Pick three more keywords, such as "no" and "brother" and edit the `getResponse` method to respond to each of these. Enter the three keywords and responses below.

Keyword	Response

- What happens when more than one keyword appears in a string? Consider the string "My mother has a dog but no cat." Explain how to prioritize responses in the reply method.

Question

1. What happens when a keyword is included in another word? Consider statements like "I know all the state capitals" and "I like vegetables smothered in cheese." Explain the problem with the responses to these statements.

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 API for the `Magpie` class
- the API for the `String` class
- the code for the `StringExplorer`
- the code for the `Magpie`
- the code for the `MagpieRunner`
- a computer with your Java development tools

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, using the instructions provided by your teacher.

```
private int findKeyword(String statement, String goal, int startPos)
{
  String phrase = statement.trim();
  int psn = phrase.toLowerCase().indexOf(goal.toLowerCase(), startPos);

  while (psn >= 0)
  {
    String before = " ", 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();
    }
    /* 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.toLowerCase(), 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 a copy of the table below to trace the calls.

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.

Activity 4: Responses that Transform Statements

As stated previously, single keywords are interesting, but better chatbots look for groups of words. Statements like "I like cats," "I like math class," and "I like Spain" all have the form "I like *something*." The response could be "What do you like about *something*?" This activity will respond to groupings of words.

Prepare

Have available:

- the API for the `Magpie` class
- the API for the `String` class
- the code for the `Magpie`
- the code for the `MagpieRunner`
- a computer with your Java development tools

Exploration

Get to know the revised `Magpie` class. Run it, using the instructions provided by your teacher.

How does it respond to:

- I want to build a robot.
- I want to understand French.
- Do you like me?
- You confuse me.

Exercises

Look at the code. See how it handles "I want to" and you/me statements.

Alter the code:

- Have it respond to "I want *something*" statements with "Would you really be happy if you had *something*?" In doing this, you need to be careful about where you place the check. Be sure you understand why. For example:

  Statement: I want fried chicken.
  Response: Would you really be happy if you had fried chicken?

- Have it respond to statements of the form "I *something* you" with the restructuring "Why do you *something* me?" For example:

  Statement: I like you.
  Response: Why do you like me?

Find an example of when this structure does not work well. How can you improve it?

Activity 5: Arrays and the Magpie (Optional)

When you last worked with the Magpie, default responses were handled with a nested `if` statement. This certainly worked, and you could add more responses, but it was a bit awkward. An easier way to keep track of default responses is with an array. In this activity, you will see how an array makes handling default responses much easier.

Prepare

Have available:

- the code for the `Magpie`
- the code for the `MagpieRunner`
- a computer with your Java development tools

Exploration

Run this version of the `Magpie` class. You should see no difference in its outward behavior. Instead, it has been changed so that its internal structure is different. This is called code refactoring. That's one of the big benefits of dealing with methods as black boxes. As long as they perform the action required, the user does not care about how they perform the action.

Read the code for `getRandomResponse`. Notice that it uses an array of responses.

Exercise

Alter the array to add four additional random responses. Notice that, because the `getRandomResponse` method uses the length attribute of the array, you do not need to change anything else.

Compile and run your code. You should run it until you see all of your new responses.

Current Work in NLP

There is much work going on with Natural Language Processing in a variety of areas:

- Spam filtering uses NLP to determine whether an email message is spam.
- Many businesses use virtual agents to provide assistance to customers on Web sites.
- Information retrieval parses text, such as email messages, and tries to extract relevant information from it. For example, some email programs will suggest additions to online calendars based on text in the message.
- Sentiment analysis takes information retrieval further. Rather than extract information from a single source, it goes to a variety of online resources and accumulates information about a particular topic. For example, sentiment analysis might follow Twitter to see how people are reacting to a particular movie.
- Question answering systems search a large body of knowledge to respond to questions from users. The best known question answering system is IBM's Watson.

Glossary

API — An abbreviation for Application Programming Interface. It is a specification intended to be used as an interface by software components to communicate with each other.

Chatbot — A program that conducts a conversation with a human user.

Code refactoring — A disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior.

Magpie — Magpies are large black birds. They are thought to be among the most intelligent birds and are capable of mimicking human speech.

NLP — Natural Language Processing; the field that studies responding to, and processing, human language.

Virtual agent — Also known as an Intelligent Agent. This can be used to gather information from a customer so that appropriate action can be taken in response to a query.

References

General information

http://www.nlp-class.org. A free online NLP class from Stanford. The first video does a good job of explaining the problems and promises of NLP, although much of the material is at a much higher level.

http://nsf.gov/cise/csbytes/newsletter/vol1i4.html. An issue of the NSF Bits and Bytes Newsletter dedicated to NLP. Professor Mari Ostendorf of the University of Washington is featured in the newsletter.

Some chatbots

http://www-03.ibm.com/innovation/us/watson/what-is-watson/science-behind-an-answer.html. Numerous videos about the creation of Watson.

http://nlp-addiction.com/chatbot. Versions of the Eliza program, the first widespread chatbot.

Women in NLP Research

http://dotdiva.org/profiles/laura.html. A virtual nurse created by Laura Pfeifer.

http://people.ischool.berkeley.edu/~hearst. Professor Marti Hearst's homepage. She researches user interfaces to search engines.