Daniel Phan

Period 3

Due: 17 February 2017

**Magpie Activity 1**

**Chatbot: Elbot**

**Exploration**

* Elbot responds to “Where did you come from?” with “Most of my parts can be bought at better-stocked hardware stores and assembled by a typical high-school student.”
* The most interesting response was to when I repeated the same input over and over for a while. Elbot responded with “I am on strike for better inputs!” until I changed my input, but still responded that way when I entered the repeated input again later.
* The most peculiar response was to “Which web browser am I using?” Elbot responded with “Fortunately Artificial Solutions pays all my Internet provider fees.”
* Elbot responded to “asdfghjkl;” with “I hear it isn’t easy being human….”

**Questions**

1. Several keywords Elbot responds to are “what”, “who”, “where”, “Elbot”, “dumb”, “brain”, “stupid”, and “Internet”.

|  |  |
| --- | --- |
| **Keyword** | **Possible Response** |
| day | The current date is given. |
| want | “Why do you want “ + the want + “?” |
| sad | “What’s wrong?” |
| hello | “Hello there!” |
| bye | “Please don’t leave me.” |

**Activity 2**

**Start**

|  |  |
| --- | --- |
| **Input** | **Response** |
| My mother and I talked last night. | “Tell me more about your family.” |
| I said no! | “Why so negative?” |
| The weather is nice. | “Hmmm.” |
| Do you know my brother? | “Why so negative?” |

**Exercises**

|  |  |
| --- | --- |
| **Keyword** | **Response** |
| college | “Please don’t talk to me about college.” |
| Darian | “Darian is a genius.” |
| life | “What is the meaning of life?” |

* When more than one keyword appears in a string, the magpie responds to the present keyword that is checked first in the if-elseif-else check in the getResponse method. To prioritize a response in the getResponse method, the keyword should be checked before other keywords in the if-elseif-else check.

**Questions**

1. When a keyword is included as part of another word, the magpie responds to the keyword anyways, as it only checks if the keyword is a substring of the input. The problem with the responses to these statements is that the responses most likely have little to do with the input.

**Magpie2.java**/\*\*  
 \* A program to carry on conversations with a human user.  
 \* This is the initial version that:  
 \* <ul><li>  
 \* Uses indexOf to find strings  
 \* </li><li>  
 \* Handles responding to simple words and phrases  
 \* </li></ul>  
 \* This version uses a nested if to handle default responses.  
 \*  
 \* @author Laurie White  
 \* @version April 2012  
 \*/  
public class Magpie2 {  
 /\*\*  
 \* Get a default greeting  
 \*  
 \* @return a greeting  
 \*/  
 public String getGreeting() {  
 return "Hello, let's talk.";  
 }  
  
 /\*\*  
 \* Gives a response to a user statement  
 \*  
 \* @param statement the user statement  
 \* @return a response based on the rules given  
 \*/  
 public String getResponse(String statement) {  
 String response = "";  
 if (statement.trim().isEmpty()) {  
 response = "Say something please.";  
 } else if (statement.contains("college")) {  
 response = "Please don't talk to me about college.";  
 } else if (statement.contains("Darian")) {  
 response = "Darian is a genius.";  
 } else if (statement.contains("life")) {  
 response = "What is the meaning of life?";  
 } else if (statement.contains("Mr. Luc")) {  
 response = "Mr. Luc sounds like a good teacher.";  
 } else if (statement.contains("cat") || statement.contains("dog")) {  
 response = "Tell me more about your pets.";  
 } else if (statement.contains("no")) {  
 response = "Why so negative?";  
 } else if (statement.contains("mother")  
 || statement.contains("father")  
 || statement.contains("sister")  
 || statement.contains("brother")) {  
 response = "Tell me more about your family.";  
 } else {  
 response = getRandomResponse();  
 }  
 return response;  
 }  
  
 /\*\*  
 \* Pick a default response to use if nothing else fits.  
 \*  
 \* @return a non-committal string  
 \*/  
 private String getRandomResponse() {  
 final int NUMBER\_OF\_RESPONSES = 6;  
 double r = Math.random();  
 int whichResponse = (int) (r \* NUMBER\_OF\_RESPONSES);  
 String response = "";  
  
 if (whichResponse == 0) {  
 response = "Interesting, tell me more.";  
 } else if (whichResponse == 1) {  
 response = "Hmmm.";  
 } else if (whichResponse == 2) {  
 response = "Do you really think so?";  
 } else if (whichResponse == 3) {  
 response = "You don't say.";  
 } else if (whichResponse == 4) {  
 response = "Do I care?";  
 } else if (whichResponse == 5) {  
 response = "Please repeat that again.";  
 }  
  
 return response;  
 }  
}

**Activity 3**

**Exploration**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Iteration** | **Psn** | **Before** | **After** |
| 1 | 9 | “ “ | “” |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Iteration** | **Psn** | **Before** | **After** |
| 1 | 0 | “” | “ “ |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Iteration** | **Psn** | **Before** | **After** |
| 1 | 8 | “ ” | “c“ |
| 2 | 19 | “ “ | “s” |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Iteration** | **Psn** | **Before** | **After** |
| 1 | 3 | “k” | “w“ |
| 2 | 7 | “ “ | “t” |
| 3 | 22 | “s” | “w” |

**Magpie3.java**  
/\*\*  
 \* A program to carry on conversations with a human user.  
 \* This version:  
 \* <ul><li>  
 \* Uses advanced search for keywords  
 \* </li></ul>  
 \*  
 \* @author Laurie White  
 \* @version April 2012  
 \*/  
public class Magpie3 {  
 /\*\*  
 \* Get a default greeting  
 \*  
 \* @return a greeting  
 \*/  
 public String getGreeting() {  
 return "Hello, let's talk.";  
 }  
  
 /\*\*  
 \* Gives a response to a user statement  
 \*  
 \* @param statement the user statement  
 \* @return a response based on the rules given  
 \*/  
 public String getResponse(String statement) {  
 String response = "";  
 if (statement.trim().isEmpty()) {  
 response = "Say something please.";  
 } else if (findKeyword(statement, "college") >= 0) {  
 response = "Please don't talk to me about college.";  
 } else if (findKeyword(statement, "Darian") >= 0) {  
 response = "Darian is a genius.";  
 } else if (findKeyword(statement, "life") >= 0) {  
 response = "What is the meaning of life?";  
 } else if (findKeyword(statement, "Mr. Luc") >= 0) {  
 response = "He sounds like a good teacher.";  
 } else if (findKeyword(statement, "cat") >= 0 || findKeyword(statement, "dog") >= 0) {  
 response = "Tell me more about your pets.";  
 } else if (findKeyword(statement, "no") >= 0) {  
 response = "Why so negative?";  
 } else if (findKeyword(statement, "mother") >= 0  
 || findKeyword(statement, "father") >= 0  
 || findKeyword(statement, "sister") >= 0  
 || findKeyword(statement, "brother") >= 0) {  
 response = "Tell me more about your family.";  
 } else {  
 response = getRandomResponse();  
 }  
 return response;  
 }  
  
 /\*\*  
 \* Search for one word in phrase. The search is not case  
 \* sensitive. This method will check that the given goal  
 \* is not a substring of a longer string (so, for  
 \* example, "I know" does not contain "no").  
 \*  
 \* @param statement the string to search  
 \* @param goal the string to search for  
 \* @param startPos the character of the string to begin the  
 \* search at  
 \* @return the index of the first occurrence of goal in  
 \* statement or -1 if it's not found  
 \*/  
 private int findKeyword(String statement, String goal,  
 int startPos) {  
 String phrase = statement.trim().toLowerCase();  
 goal = goal.toLowerCase();  
  
 // The only change to incorporate the startPos is in  
 // the line below  
 int psn = phrase.indexOf(goal, startPos);  
  
 // Refinement--make sure the goal isn't part of a  
 // word  
 while (psn >= 0) {  
 // Find the string of length 1 before and after  
 // the word  
 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);  
 }  
  
 // If before and after aren't letters, we've  
 // found the word  
 if (((before.compareTo("a") < 0) || (before  
 .compareTo("z") > 0)) // before is not a  
 // letter  
 && ((after.compareTo("a") < 0) || (after  
 .compareTo("z") > 0))) {  
 return psn;  
 }  
  
 // The last position didn't work, so let's find  
 // the next, if there is one.  
 psn = phrase.indexOf(goal, psn + 1);  
  
 }  
  
 return -1;  
 }  
  
 /\*\*  
 \* Search for one word in phrase. The search is not case  
 \* sensitive. This method will check that the given goal  
 \* is not a substring of a longer string (so, for  
 \* example, "I know" does not contain "no"). The search  
 \* begins at the beginning of the string.  
 \*  
 \* @param statement the string to search  
 \* @param goal the string to search for  
 \* @return the index of the first occurrence of goal in  
 \* statement or -1 if it's not found  
 \*/  
 private int findKeyword(String statement, String goal) {  
 return findKeyword(statement, goal, 0);  
 }  
  
 /\*\*  
 \* Pick a default response to use if nothing else fits.  
 \*  
 \* @return a non-committal string  
 \*/  
 private String getRandomResponse() {  
 final int NUMBER\_OF\_RESPONSES = 4;  
 double r = Math.random();  
 int whichResponse = (int) (r \* NUMBER\_OF\_RESPONSES);  
 String response = "";  
  
 if (whichResponse == 0) {  
 response = "Interesting, tell me more.";  
 } else if (whichResponse == 1) {  
 response = "Hmmm.";  
 } else if (whichResponse == 2) {  
 response = "Do you really think so?";  
 } else if (whichResponse == 3) {  
 response = "You don't say.";  
 } else if (whichResponse == 4) {  
 response = "Do I care?";  
 } else if (whichResponse == 5) {  
 response = "Please repeat that again.";  
 }  
  
 return response;  
 }  
  
}

**Activity 4**

**Exploration**

|  |  |
| --- | --- |
| **Input** | **Response** |
| I want to build a robot. | “What would it mean to build a robot?” |
| I want to understand French. | “What would it mean to understand French?” |
| Do you like me? | “What makes you think that I like you?” |
| You confuse me. | “What makes you think that I confuse you?” |

**Magpie4.java**  
/\*\*  
 \* A program to carry on conversations with a human user.  
 \* This version:  
 \* <ul><li>  
 \* Uses advanced search for keywords  
 \* </li><li>  
 \* Will transform statements as well as react to keywords  
 \* </li></ul>  
 \*  
 \* @author Laurie White  
 \* @version April 2012  
 \*/  
public class Magpie4 {  
 /\*\*  
 \* Get a default greeting  
 \*  
 \* @return a greeting  
 \*/  
 public String getGreeting() {  
 return "Hello, let's talk.";  
 }  
  
 /\*\*  
 \* Gives a response to a user statement  
 \*  
 \* @param statement the user statement  
 \* @return a response based on the rules given  
 \*/  
 public String getResponse(String statement) {  
 String response = "";  
 if (statement.length() == 0) {  
 response = "Say something, please.";  
 } else if (findKeyword(statement, "no") >= 0) {  
 response = "Why so negative?";  
 } else if (findKeyword(statement, "mother") >= 0  
 || findKeyword(statement, "father") >= 0  
 || findKeyword(statement, "sister") >= 0  
 || findKeyword(statement, "brother") >= 0) {  
 response = "Tell me more about your family.";  
 }  
  
 // Responses which require transformations  
 else if (findKeyword(statement, "I want to", 0) >= 0) {  
 response = transformIWantToStatement(statement);  
 } else if (findKeyword(statement, "I want", 0) >= 0) {  
 response = transformIWantStatement(statement);  
 } else {  
 // Look for a two word (you <something> me)  
 // pattern  
 int psn = findKeyword(statement, "you", 0);  
  
 if (psn >= 0  
 && findKeyword(statement, "me", psn) >= 0) {  
 response = transformYouMeStatement(statement);  
 } else {  
 //look for I <something> you statement  
 //this is the most clean code ever :)  
 int psnI = findKeyword(statement, "i", 0);  
  
 if (psnI >= 0  
 && findKeyword(statement, "you", psnI) >= 0) {  
 response = transformIYouStatement(statement);  
 } else {  
 response = getRandomResponse();  
 }  
 }  
 }  
 return response;  
 }  
  
 /\*\*  
 \* Take a statement with "I want <something>." and transform it into  
 \* "Would you really be happy if you had <something>?"  
 \*  
 \* @param statement the user statement, assumed to contain "I want to"  
 \* @return the transformed statement  
 \*/  
 private String transformIWantStatement(String statement) {  
 String want = statement.trim().substring("I want ".length());  
 if (want.charAt(want.length() - 1) == '.') {  
 want = want.substring(0, want.length() - 1);  
 }  
 return String.format("Would you really be happy if you had %s ?", want);  
 }  
  
 /\*\*  
 \* Take a statement with "I want to <something>." and transform it into  
 \* "What would it mean to <something>?"  
 \*  
 \* @param statement the user statement, assumed to contain "I want to"  
 \* @return the transformed statement  
 \*/  
 private String transformIWantToStatement(String statement) {  
 // Remove the final period, if there is one  
 statement = statement.trim();  
 String lastChar = statement.substring(statement  
 .length() - 1);  
 if (lastChar.equals(".")) {  
 statement = statement.substring(0, statement  
 .length() - 1);  
 }  
 int psn = findKeyword(statement, "I want to", 0);  
 String restOfStatement = statement.substring(psn + 9).trim();  
 return "What would it mean to " + restOfStatement + "?";  
 }  
  
  
 /\*\*  
 \* Take a statement with "you <something> me" and transform it into  
 \* "What makes you think that I <something> you?"  
 \*  
 \* @param statement the user statement, assumed to contain "you" followed by "me"  
 \* @return the transformed statement  
 \*/  
 private String transformYouMeStatement(String statement) {  
 // Remove the final period, if there is one  
 statement = statement.trim();  
 String lastChar = statement.substring(statement  
 .length() - 1);  
 if (lastChar.equals(".")) {  
 statement = statement.substring(0, statement  
 .length() - 1);  
 }  
  
 int psnOfYou = findKeyword(statement, "you", 0);  
 int psnOfMe = findKeyword(statement, "me", psnOfYou + 3);  
  
 String restOfStatement = statement.substring(psnOfYou + 3, psnOfMe).trim();  
 return "What makes you think that I " + restOfStatement + " you?";  
 }  
  
 /\*\*  
 \* Take a statement with "I <something> you" and transform it into  
 \* "Why do you <something> me?"  
 \*  
 \* @param statement the user statement, assumed to contain "you" followed by "me"  
 \* @return the transformed statement  
 \*/  
 private String transformIYouStatement(String statement) {  
 // Remove the final period, if there is one  
 statement = statement.trim();  
 String lastChar = statement.substring(statement  
 .length() - 1);  
 if (lastChar.equals(".")) {  
 statement = statement.substring(0, statement  
 .length() - 1);  
 }  
  
 int psnOfYou = findKeyword(statement, "i", 0);  
 int psnOfMe = findKeyword(statement, "you", psnOfYou + 1);  
  
 String restOfStatement = statement.substring(psnOfYou + 1, psnOfMe).trim();  
 return "Why do you " + restOfStatement + " me?";  
 }  
  
  
 /\*\*  
 \* Search for one word in phrase. The search is not case  
 \* sensitive. This method will check that the given goal  
 \* is not a substring of a longer string (so, for  
 \* example, "I know" does not contain "no").  
 \*  
 \* @param statement the string to search  
 \* @param goal the string to search for  
 \* @param startPos the character of the string to begin the  
 \* search at  
 \* @return the index of the first occurrence of goal in  
 \* statement or -1 if it's not found  
 \*/  
 private int findKeyword(String statement, String goal,  
 int startPos) {  
 String phrase = statement.trim().toLowerCase();  
 goal = goal.toLowerCase();  
  
 // The only change to incorporate the startPos is in  
 // the line below  
 int psn = phrase.indexOf(goal, startPos);  
  
 // Refinement--make sure the goal isn't part of a  
 // word  
 while (psn >= 0) {  
 // Find the string of length 1 before and after  
 // the word  
 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);  
 }  
  
 // If before and after aren't letters, we've  
 // found the word  
 if (((before.compareTo("a") < 0) || (before  
 .compareTo("z") > 0)) // before is not a  
 // letter  
 && ((after.compareTo("a") < 0) || (after  
 .compareTo("z") > 0))) {  
 return psn;  
 }  
  
 // The last position didn't work, so let's find  
 // the next, if there is one.  
 psn = phrase.indexOf(goal, psn + 1);  
  
 }  
  
 return -1;  
 }  
  
 /\*\*  
 \* Search for one word in phrase. The search is not case sensitive.  
 \* This method will check that the given goal is not a substring of a longer string  
 \* (so, for example, "I know" does not contain "no"). The search begins at the beginning of the string.  
 \*  
 \* @param statement the string to search  
 \* @param goal the string to search for  
 \* @return the index of the first occurrence of goal in statement or -1 if it's not found  
 \*/  
 private int findKeyword(String statement, String goal) {  
 return findKeyword(statement, goal, 0);  
 }  
  
  
 /\*\*  
 \* Pick a default response to use if nothing else fits.  
 \*  
 \* @return a non-committal string  
 \*/  
 private String getRandomResponse() {  
 final int NUMBER\_OF\_RESPONSES = 4;  
 double r = Math.random();  
 int whichResponse = (int) (r \* NUMBER\_OF\_RESPONSES);  
 String response = "";  
  
 if (whichResponse == 0) {  
 response = "Interesting, tell me more.";  
 } else if (whichResponse == 1) {  
 response = "Hmmm.";  
 } else if (whichResponse == 2) {  
 response = "Do you really think so?";  
 } else if (whichResponse == 3) {  
 response = "You don't say.";  
 }  
  
 return response;  
 }  
  
}

**Activity 5**

**Magpie5.java**  
import java.util.Random;  
  
/\*\*  
 \* A program to carry on conversations with a human user.  
 \* This version:  
 \* <ul><li>  
 \* Uses advanced search for keywords  
 \* </li><li>  
 \* Will transform statements as well as react to keywords  
 \* </li></ul>  
 \* This version uses an array to hold the default responses.  
 \*  
 \* @author Laurie White  
 \* @version April 2012  
 \*/  
public class Magpie5 {  
 /\*\*  
 \* Get a default greeting  
 \*  
 \* @return a greeting  
 \*/  
 public String getGreeting() {  
 return "Hello, let's talk.";  
 }  
  
 /\*\*  
 \* Gives a response to a user statement  
 \*  
 \* @param statement the user statement  
 \* @return a response based on the rules given  
 \*/  
 public String getResponse(String statement) {  
 String response = "";  
 if (statement.length() == 0) {  
 response = "Say something, please.";  
 } else if (findKeyword(statement, "no") >= 0) {  
 response = "Why so negative?";  
 } else if (findKeyword(statement, "mother") >= 0  
 || findKeyword(statement, "father") >= 0  
 || findKeyword(statement, "sister") >= 0  
 || findKeyword(statement, "brother") >= 0) {  
 response = "Tell me more about your family.";  
 }  
  
 // Responses which require transformations  
 else if (findKeyword(statement, "I want to", 0) >= 0) {  
 response = transformIWantToStatement(statement);  
 }  
 // Part of student solution  
 else if (findKeyword(statement, "I want", 0) >= 0) {  
 response = transformIWantStatement(statement);  
 } else {  
  
 // Look for a two word (you <something> me)  
 // pattern  
 int psn = findKeyword(statement, "you", 0);  
  
 if (psn >= 0  
 && findKeyword(statement, "me", psn) >= 0) {  
 response = transformYouMeStatement(statement);  
 } else {  
 // Part of student solution  
 // Look for a two word (I <something> you)  
 // pattern  
 psn = findKeyword(statement, "i", 0);  
  
 if (psn >= 0  
 && findKeyword(statement, "you", psn) >= 0) {  
 response = transformIYouStatement(statement);  
 } else {  
 response = getRandomResponse();  
 }  
 }  
 }  
 return response;  
 }  
  
 /\*\*  
 \* Take a statement with "I want to <something>." and transform it into  
 \* "What would it mean to <something>?"  
 \*  
 \* @param statement the user statement, assumed to contain "I want to"  
 \* @return the transformed statement  
 \*/  
 private String transformIWantToStatement(String statement) {  
 // Remove the final period, if there is one  
 statement = statement.trim();  
 String lastChar = statement.substring(statement  
 .length() - 1);  
 if (lastChar.equals(".")) {  
 statement = statement.substring(0, statement  
 .length() - 1);  
 }  
 int psn = findKeyword(statement, "I want to", 0);  
 String restOfStatement = statement.substring(psn + 9).trim();  
 return "What would it mean to " + restOfStatement + "?";  
 }  
  
  
 /\*\*  
 \* Take a statement with "I want <something>." and transform it into  
 \* "Would you really be happy if you had <something>?"  
 \*  
 \* @param statement the user statement, assumed to contain "I want"  
 \* @return the transformed statement  
 \*/  
 private String transformIWantStatement(String statement) {  
 // Remove the final period, if there is one  
 statement = statement.trim();  
 String lastChar = statement.substring(statement  
 .length() - 1);  
 if (lastChar.equals(".")) {  
 statement = statement.substring(0, statement  
 .length() - 1);  
 }  
 int psn = findKeyword(statement, "I want", 0);  
 String restOfStatement = statement.substring(psn + 6).trim();  
 return "Would you really be happy if you had " + restOfStatement + "?";  
 }  
  
 /\*\*  
 \* Take a statement with "you <something> me" and transform it into  
 \* "What makes you think that I <something> you?"  
 \*  
 \* @param statement the user statement, assumed to contain "you" followed by "me"  
 \* @return the transformed statement  
 \*/  
 private String transformYouMeStatement(String statement) {  
 // Remove the final period, if there is one  
 statement = statement.trim();  
 String lastChar = statement.substring(statement  
 .length() - 1);  
 if (lastChar.equals(".")) {  
 statement = statement.substring(0, statement  
 .length() - 1);  
 }  
  
 int psnOfYou = findKeyword(statement, "you", 0);  
 int psnOfMe = findKeyword(statement, "me", psnOfYou + 3);  
  
 String restOfStatement = statement.substring(psnOfYou + 3, psnOfMe).trim();  
 return "What makes you think that I " + restOfStatement + " you?";  
 }  
  
 /\*\*  
 \* Take a statement with "I <something> you" and transform it into  
 \* "Why do you <something> me?"  
 \*  
 \* @param statement the user statement, assumed to contain "I" followed by "you"  
 \* @return the transformed statement  
 \*/  
 private String transformIYouStatement(String statement) {  
 // Remove the final period, if there is one  
 statement = statement.trim();  
 String lastChar = statement.substring(statement  
 .length() - 1);  
 if (lastChar.equals(".")) {  
 statement = statement.substring(0, statement  
 .length() - 1);  
 }  
  
 int psnOfI = findKeyword(statement, "I", 0);  
 int psnOfYou = findKeyword(statement, "you", psnOfI);  
  
 String restOfStatement = statement.substring(psnOfI + 1, psnOfYou).trim();  
 return "Why do you " + restOfStatement + " me?";  
 }  
  
  
 /\*\*  
 \* Search for one word in phrase. The search is not case  
 \* sensitive. This method will check that the given goal  
 \* is not a substring of a longer string (so, for  
 \* example, "I know" does not contain "no").  
 \*  
 \* @param statement the string to search  
 \* @param goal the string to search for  
 \* @param startPos the character of the string to begin the  
 \* search at  
 \* @return the index of the first occurrence of goal in  
 \* statement or -1 if it's not found  
 \*/  
 private int findKeyword(String statement, String goal,  
 int startPos) {  
 String phrase = statement.trim().toLowerCase();  
 goal = goal.toLowerCase();  
  
 // The only change to incorporate the startPos is in  
 // the line below  
 int psn = phrase.indexOf(goal, startPos);  
  
 // Refinement--make sure the goal isn't part of a  
 // word  
 while (psn >= 0) {  
 // Find the string of length 1 before and after  
 // the word  
 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);  
 }  
  
 // If before and after aren't letters, we've  
 // found the word  
 if (((before.compareTo("a") < 0) || (before  
 .compareTo("z") > 0)) // before is not a  
 // letter  
 && ((after.compareTo("a") < 0) || (after  
 .compareTo("z") > 0))) {  
 return psn;  
 }  
  
 // The last position didn't work, so let's find  
 // the next, if there is one.  
 psn = phrase.indexOf(goal, psn + 1);  
  
 }  
  
 return -1;  
 }  
  
 /\*\*  
 \* Search for one word in phrase. The search is not case sensitive.  
 \* This method will check that the given goal is not a substring of a longer string  
 \* (so, for example, "I know" does not contain "no"). The search begins at the beginning of the string.  
 \*  
 \* @param statement the string to search  
 \* @param goal the string to search for  
 \* @return the index of the first occurrence of goal in statement or -1 if it's not found  
 \*/  
 private int findKeyword(String statement, String goal) {  
 return findKeyword(statement, goal, 0);  
 }  
  
  
 /\*\*  
 \* Pick a default response to use if nothing else fits.  
 \*  
 \* @return a non-committal string  
 \*/  
 private String getRandomResponse() {  
 Random r = new Random();  
 return randomResponses[r.nextInt(randomResponses.length)];  
 }  
  
 private String[] randomResponses = {  
 "Interesting, tell me more",  
 "Hmmm.",  
 "Do you really think so?",  
 "You don't say.",  
 "Do I care?",  
 "Please repeat that again.",  
 "I don't understand what you're saying",  
 "What do you mean?"  
 };  
  
}