**import** java.util.ArrayList; // import the ArrayList class

**import** java.util.Scanner; // for the interactive input

/\* TABLE OF CONTENTS

----------------------------

phrases = is the array of all the phrases

keywords = the inserted word to search for

wordFound = the numbers keywords in phrases

sentence = an item in phrases

----------------------------\*/

**public class** readingText {

**public static void** main(String[] args) {

//---------------------------------------------------------------------------------

ArrayList<String> phrases = **new** ArrayList<> ( );

// creating the ArrayList of the phrases

phrases.add ( "Programmable devices have existed at least as far back as 1206 AD, when the automata of Al-Jazari were programmable, via pegs and cams, to play various rhythms and drum patterns;[3] and the 1801 Jacquard loom could produce entirely different weaves by changing the \"program\" - a series of pasteboard cards with holes punched in them.\n" );

phrases.add ( "However, the first computer program is generally dated to 1843, when mathematician Ada Lovelace published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine.[4]\n" );

phrases.add ( "Data and instructions were once stored on external punched cards, which were kept in order and arranged in program decks.\n" );

phrases.add ( "In the 1880s Herman Hollerith invented the concept of storing data in machine-readable form.[5] Later a control panel (plugboard) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way; as were the first electronic computers. However, with the concept of the stored-program computers introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory.\n" );

phrases.add ( "Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. Assembly languages were soon developed that let the programmer specify instruction in a text format, (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses. However, because an assembly language is little more than a different notation for a machine language, any two machines with different instruction sets also have different assembly languages.\n" );

phrases.add ( "Wired control panel for an IBM 402 Accounting Machine.\n" );

phrases.add ( "High-level languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it targetable to varying machine instruction sets via compilation declarations and heuristics. They harness the power of computers to make programming easier[6] by allowing programmers to specify calculations by entering a formula using infix notation (e.g., Y = X\*2 + 5\*X + 9) for example. FORTRAN, the first widely used high-level language to have a functional implementation which permitted the abstraction of reusable blocks of code, came out in 1957[7] and many other languages were soon developed - in particular, COBOL aimed at commercial data processing, and Lisp for computer research.\n" );

phrases.add ( "Programs were mostly still entered using punched cards or paper tape. See computer programming in the punch card era. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Text editors were developed that allowed changes and corrections to be made much more easily than with punched cards.\n" );

//---------------------------------------------------------------------------------

Scanner scanner = **new** Scanner ( System.*in* ); // to use scanner!

System.*out*.print ( "Enter the keyword to search for: ");

String keyword = scanner.nextLine ( ); // the input string now becomes keyword

scanner.close ( ); // make sure it's closed

//---------------------------------------------------------------------------------

**int** wordFound = 0; // create the int for keyword counting

//---------------------------------------------------------------------------------

// the for loop that goes through the array to find the keywords

**for**(**int** i = 0; i< phrases.size ( ); i++) {

String sentence = phrases.get ( i ); // every item from the array becomes a sentence

String[] words = sentence.split(" "); // to look through each sentence (via words and seperated by a " " (space))

// this for loop inside a for loop will look through each word of each sentence to actually look for the keyword

**for**(**int** j = 0; j<words.length; j++) {

// when the keyword is found (ignore lowercase or capital)

**if** (words[j].equalsIgnoreCase ( keyword ) ) {

wordFound++; //add 1 to wordFound (which is the total word count)

// print out the found "keyword" EACH time in the loop that it is found --> in which sentence, and at which word in the sentence (including the line (sentence) and the entirety of it)

System.*out*.println ( "Keyword '" + keyword + "' found in sentence " + (i+1) + " at word " + (j+1) );

System.*out*.println ( "Found in line " + (i+1) + ": " + sentence );

}

}

}

// Then after looping through the entire array, print the total word count

System.*out*.println ( "Total word count: " + wordFound );

}

}