**Madlibs**CPSC 24500  
Due October 23, 2023 at 6pm  
38 points

Madlibs is a game you might have played on long trips in the car. A Madlib is a story that contains placeholders for words that you are supposed to fill in randomly. Your friend might ask you to offer an adjective or a verb or a noun. That word would then be inserted into the story at the location of the placeholder. The result would be a randomly generated tale that might be complete nonsense and but also might be funny.

For this assignment, you will create your own Madlibs game. Using readymade files of singular nouns, plural nouns, singular verbs, plural verbs, past-tense verbs, adjectives, and adverbs, along with story files named story1.txt, story2.txt, etc., you will implement a game in which the user can load a story and generate a Madlib automatically. Files you can use [are kept here](https://www.dropbox.com/scl/fo/siz00erdnfj4kzf3bkm7v/h?rlkey=l95cllysn5q1ogap1rhscpv3l&dl=0).

The wordlist files each contain words of a particular type, one word per line. Here is what the file adj.txt, which contains adjectives, looks like:

A black text on a white background

Description automatically generated

The story files contain the text of stories except placeholders appear throughout. The placeholders indicate the kind of word that should be placed at that point in the story. Placeholders look like <this>. Here’s what a story file looks like:

A screen shot of a computer code

Description automatically generated

You can infer that <adj> must be a placeholder for an adjective, <singnoun> for a singular noun, <adv> for an adverb, <plunoun> for a plural noun, <singverb> for a singular verb, <pluverb> for a plural verb, <pastverb> for a past verb. These placeholders match the names of the word files: there are files named adj.txt, adv.txt, singnoun.txt, plunoun.txt, singverb.txt, pluverb.txt, and pastverb.txt.

In reading the stories and deciding where to substitute, you will have to be careful to consider that some placeholders might end in punctuation but should still be treated as placeholders to be replaced by actual words.

Your game will begin by displaying a welcome header with a message and a brief instruction. As the player proceeds, they can type the number of the story they want to load. Your program will load the story and automatically insert words from the appropriate word list (singular nouns, plural nouns, singular verbs, etc.) to fill in for the placeholders embedded in the story. It will display the newly generated story, with the right kind of word replacing each placeholder.

Here’s an example of playing this game:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Welcome to Madlibs V1.0 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

This program generates random stories using wordlists you supply.

Enter the name of the folder where the stories and wordlists are.

Or just press Enter to accept the default location: c:\temp\ml

Enter a story number or q to quit: 1

Here is your Madlib:

There was once a inspired destiny who shouted in a cat.

Everyone in the pilot was gangly of the anguished missile,

but no one had the scabs to love. So we exuberantly schemed

past the peasant and exclaimed upon the cobwebbed locusts,

and everyone felt musty about their lot in life, until, of course,

the mountains descended upon us like lives and took pilot

from us. Rue that day! Rue it exuberantly.

Enter a story number or q to quit: 2

Here is your Madlib:

Almost everybody had shouted in the house for the

tiger to love over the villages in the distance.

One could dance mountains in the air and eat feelings

in the countryside. I smugly jumped over the missile to

simmer a inspired tiger overhead. I didn't breathe, but I

felt musty for having leaped a pilot.

Enter a story number or q to quit: 3

That story does not exist. Choose again.

Enter a story number or q to quit: three

That story does not exist. Choose again.

Enter a story number or q to quit: 1

Here is your Madlib:

There was once a dank pilot who schemed in a destiny.

Everyone in the pilot was inspired of the yellow pilot,

but no one had the mountains to dream. So we slowly leaped

past the pilot and leaped upon the musty lives,

and everyone felt moist about their lot in life, until, of course,

the locusts descended upon us like feelings and took destiny

from us. Rue that day! Rue it contentedly.

Enter a story number or q to quit: q

Thank you for using this program.

Here’s how your program will be graded:

|  |  |
| --- | --- |
| Points | Requirement |
| 2 | Print welcome message centered as a banner as shown in the sample output followed by a brief description. |
| 2 | You ask the user for the path of the folder where the wordlists and story files are stored. If they just hit enter in response to this question, the program will read the word lists and story files from whatever is the default location for your IDE on your computer. If they specify a value here, the program will read the word lists and story files from the path they typed. You must implement this feature, because I will be using it to read my wordlists and story files from a different folder when I grade your work. If I have to modify your code or my setup to accommodate a different path for the story files and word lists, the program will crash, and you will be penalized 9.5 points. |
| 5 | Load the 7 word lists into arrays or arraylists or hash maps (whichever collection you prefer). You must use the file names I’ve posted: adj.txt, adv.txt, pastverb.txt, plunoun.txt, pluverb.txt, singnoun.txt, singverb.txt |
| 3 | Show the one-line menu of options repeatedly until the user decides to quit using a do loop or while loop that will terminate if the user enters q or Q. |
| 2 | Indicate that a story doesn’t exist if the user enters a nonexistent story number. This includes entering text that might cause an exception to occur, such as entering text where there should have been an integer. You program must not crash if they do enter a non-integer, or else there will be a 9.5-point penalty. |
| 5 | Read the selected story file into memory, storing it as an ArrayList<String>, where each entry is a line of the story. |
| 10 | Replace each placeholder in the story with the correct kind of word, not letting punctuation get in the way of that, but still preserving punctuation (the proper handling of punctuation is worth 3 points of this; randomly selecting the right word is worth another 4 points. Printing the result is worth another 3 points. |
| 3 | Divide up your code into multiple, single-purposed, clearly named functions. |
| 3 | Include both Javadoc and regular comments. Javadoc comments must precede functions that you think require documentation because they do the important work of the program. Javadoc comments for functions that take in values and / or return values must have @param and @return tags. |
| 3 | Submit your solution by posting the link to your GitHub repository. Make sure you use git’s tools to commit and push. Do not upload your code there. Make sure you commit your code multiple times during the time you have to work on this assignment, including within 3 days of this assignment being assigned. If you do not get started on this code by 10/16, making a commitment to the repository by that date, you will be penalized 3 points because you need to start working on this assignment as soon as possible and work gradually and regularly on it. |

In total, then, this assignment is worth 38 points.

If you copy any part of your solution from another student or from the web or from an AI tool, you will earn -38 points out of 38.

If your program crashes, you will earn a 9.5-point penalty.

If your program doesn’t compile, you will earn a 19-point penalty.

Have fun with this! If you can do this problem, then you know how to read from files and deal with collections, random numbers, exceptions, loops, and strings effectively.

PrintWelcome

Define data that you will use in multiple places as class level variable

Public class madlibs{

Private static Random rnd;

Private static ArrayList<String> singNouns, singVerbs, pluNouns, etc;

}

Create the word lists

Public static void createWordList( ){

singNouns = new ArrayList<String>( );

singVerbs = new ArrayList<String>( );

}

Public static ArrayList<String> readStory(folder, storyNum){

ArrayList<String> result = new ArrayList<String>();

Try{

Scanner fsc = new Scanner(new File(folder, “story” +storyNum +”.text”));

While there are lines in the file

Results.add(the line we read)

Return result

Fsc.close()

}catch(Exception e){

Return null;

In main ask the user for the folder name 🡪 folder

Load word lists

Public static Boolean loadWordLists(String folder){

//load in the singNouns

Try{

Scanner fsc = new Scanner(new File(folder, “singnoun.txt”));

Use a while loop to read its contents

singNouns.add(each line);

fsc.close( );

do the same for verbs etc

return true;

}catch (Exception e){

Return false;

}

}

In main, do loop (assuming you’ve already called createWordList and loadWordList

Do{

Ask them for story number

If it is not q:

Story = readStory(folder, choice); 🡪 ArrayList<String>

If story is null,

show that there was a problem

else

tellStory(story); //shows the lines of the story with the placeholders replaced

while(choice was not q)

while (phrase.indexOf(“<noun>”) > 0 {

replace with randomly chosen noun from list

phrase = phrase.replaceFirst(“<noun>”, “pizza”);

Random rnd = new Random( );

Words.get(rnd.nextInt(words.size( ))); do this in here🡪 phrase = phrase.replaceFirst(“<noun>”, “pizza”);

Public static void tellStory(ArrayList<String> story){

For (String line : story){

One while loop for each part of speech that replaces < > with randomly chosen word

Print resulting line;