// Chris Wibe, 10/3/22, Programming Projects Final Project

// Note: This has been developed and tested in Online GDB. I do not know how the format will work in other platforms when run.

import java.util.\*; // useful package

import java.time.LocalDateTime; // less useful package

import java.time.format.DateTimeFormatter; // less useful package

import java.lang.\*; // useful package

public class Main { // the class

static String[] profile = new String[2]; // array holding two slots: one for each mistake possible

public static void Destiny() { // method 1 of 2: this one shapes the character's destiny

Scanner scanner1 = new Scanner(System.in); // scanner

System.out.println((char)27 + "[32m" + "Quick, write me your name."); // first question: output name

System.out.println((char)27 + "[34m" + "Not so fast! Make sure when writing your name not to write the letter sometimes found in circles in email addresses." + (char)27 + "[31m"); // output name with no "a"s in it

String active = scanner1.nextLine(); // asks for string input for question 1

String[] activeChar = active.split(""); // splits string into array of chars

for (int i = 0; i < activeChar.length; i++) { // loops on length of array

String link = activeChar[i]; // converts each char into a string

if (link.equals("a") || link.equals("A")) { // looks for any "a"s in the input

System.out.println((char)27 + "[32m" + "Stop right there! You cannot use the letter 'a'. It is a sacred letter."); // character is scolded for inputting an "a"

profile[0] = "a"; // "a" added to profile of mistakes

break; // progresses to next question

} else { // character does not get scolded since user did not input an "a"

System.out.println((char)27 + "[31m" + "..."); // the story continues

}

}

System.out.println((char)27 + "[32m" + "...Right. Where do you come from?"); // second question: output where character is from

System.out.println((char)27 + "[34m" + "Listen closely: do not write the letter which is most common in the English language." + (char)27 + "[31m"); // output where character is from with no "e"s in it

String confident = scanner1.nextLine(); // asks for string input for question 2

System.out.println((char)27 + "[31m" + "Suddenly various echoes of forgotten voices are screaming my name!"); // echoes

String[] confidentChar = confident.split(""); // splits string into array of chars

int j = 0; // initializes j

while (j < confidentChar.length) { // loops on length of array

String confidant = confidentChar[j]; // converts each char into a string

if (confidant.equals("e") || confidant.equals("E")) { // looks for any "e"s in the input

System.out.println((char)27 + "[32m" + "Not that letter! ScrEEch! I cannot hear you further. Be on your way."); // character is scolded for inputting an "e"

profile[1] = "e"; // "e" added to profile of mistakes

}

j++;

String k = "[3" + j + "m"; // color formula

System.out.println((char)27 + k + active.toUpperCase()); // output color formula

}

System.out.println((char)27 + "[32m" + "It is hopeless! Your despair is becoming tangible."); // segway from Destiny to Fate

}

public static void Fate() { // method 2 of 2: this one takes one's destiny and lays out the final riddle

Scanner scanner2 = new Scanner(System.in); // scanner

System.out.println((char)27 + "[32m" + "A rockslide is plummeting toward you. How are you feeling?"); // enters the Fate method

String decision = scanner2.nextLine(); // asks for (meaningless) string input (for effect)

String directhere = Arrays.toString(profile); // converts profile array to string containing user mistakes

if (directhere.equals("[a, e]") || directhere.equals("[A, e]") || directhere.equals("[A, E]") || directhere.equals("[a, E]")) { // looks in profile string for both "a" and "e"

System.out.println("Pick any direction. Your only hint is '\*\*ST'."); // riddle hint for users with 2 mistakes: 2 missing letters (from "EAST")

} else if (directhere.equals("[a, null]") || directhere.equals("[A]")) { // looks in profile string for "a"

System.out.println("Choose a cardinal direction. Your hint is 'E\*ST'."); // riddle hint for users with 1 mistake: 1 missing letter (from "EAST")

} else if (directhere.equals("[null, e]") || directhere.equals("[E]")) { // looks in profile string for "e"

System.out.println("Choose a cardinal direction. Your hint is '\*AST'."); // riddle hint for users with 1 mistake: 1 missing letter (from "EAST")

} else { // looks in profile string for the null value

System.out.println("Head east (type 'east')."); // riddle hint for users without any mistakes: "EAST"

}

}

public static void main(String[] args) { // main-method: calls each method

Scanner scanner3 = new Scanner(System.in); // scanner

System.out.println((char)27 + "[36m" + "Narrator (Gray) You are trapped in a deep cave in a remote part of Tennessee and hear this screeching silkworm."); // initial setting of drama

System.out.println((char)27 + "[32m" + "Joe (Green): I am a TINY silkworm, Joe, in my nest of straw. My eyes are slitted."); // Joe, the antagonist

System.out.println((char)27 + "[31m" + "Myself (Red): Where am I?"); // exposition

String Pause = scanner3.nextLine(); // asks for string input

System.out.println("It is yet " + java.time.LocalTime.now() + " now and the darkness effervesces with evil sounds."); // more setting of drama

System.out.println((char)27 + "[34m" + "Augustus (Blue): I am your only hope. I am a bee of a man, AKA Augustus."); // Augustus, the helper

System.out.println((char)27 + "[32m" + "Listen to me. I am a silkworm, AKA Joe:"); // Joe is an evil silkworm

Destiny(); // calls method 1

Fate(); // calls method 2

String Outcome = scanner3.nextLine(); // scanner

String OutcomeFormatted = Outcome.toLowerCase(); // standardizes input to makes sure that it is all lowercase (to match with riddle answer)

if (OutcomeFormatted.equals("east")) { // compares input with "east": if they are the same, do this

System.out.println((char)27 + "[35m" + "You are saved. You and Augustus climb the silkworm's pale flesh to the safety of the SMC."); // good ending

} else { // otherwise, do this

System.out.println((char)27 + "[36m" + "You are dead. " + (char)27 + "[32m" + "A pearl-shaped boulder slaughtered you silently. \*Chuckles\*"); // bad ending

}

}

}