**Java Journal Template**

**Directions:** Follow the directions for each part of the journal template. Include in your response all the elements listed under the Requirements section. Prompts in the Inspiration section are not required; however, they may help you to fully think through your response.

Remember to review the Touchstone page for entry requirements, examples, and grading specifics.

**Name: Jesse Cooper**

**Date: 2/23/2025**

**Final IDE Program Join Link:** https://onlinegdb.com/bwCxKVMJd

Complete the following template. Fill out all entries using complete sentences.

## PART 1: Defining Your Problem

| **Task**  State the problem you are planning to solve.  **Requirements**   * Describe the problem you are trying to solve. * Describe any input data you expect to use. * Describe what the program will do to solve the problem. * Describe any outputs or results the program will provide.   **Inspiration**  When writing your entry below, ask yourself the following questions:   * Is your problem clearly defined? * Why do you want to solve this particular problem? * What source(s) of data do you believe you will need? Will the user need to supply that data, or will you get it from an external file or another source? * Will you need to interact with the user throughout the program? Will users continually need to enter data in and see something to continue? * What are your expected results or what will be the end product? What will you need to tell a user of your program when it is complete? |
| --- |
| I decided to copy over the project idea I had done from the Intro to Python course and test my ability to transfer to a new programming language using similar concepts. It will be interesting to see whether or not I’ll be able to complete this without too much trouble.  **The Problem**: I’m trying to the problem of translation of text-to-morse-code and morse-code-to-text. Morse code is an alphabet or code in which letters are represented by combinations of long and short signals of light or sound. The program will provide a tool to be utilized by users to convert text to morse and back.  **Input Data:** On text-to-Morse translation, the programs would expect the user to be inputting text containing alphanumeric characters, i.e. letters and numbers, and separating spaces. When doing the reverse, Morse-to-text translation, the program would expect the user to input Morse code with dots representing ‘.’ and dashes representing ‘-’ with spaces between, and ‘/’ to represent space between words.  **Programs Solve**: The program uses a dictionary named morse\_code\_dict to store mappings between characters—like letters, numbers, and spaces—and their Morse code equivalents. It offers two key functions:  text\_to\_morse(text): This function takes a piece of text and translates it into Morse code using the dictionary.  morse\_to\_text(morse\_code): This function does the reverse, converting Morse code back into readable text.  When you run the program, it shows a simple menu with two options:  Select "1" to turn text into Morse code.  Select "2" to translate Morse code back into text.  This setup makes it easy for users to switch between text and Morse code easily.  **Program’s Results:** When translating text to Morse code, the program takes your input text and converts it into its corresponding Morse code.  On the flip side, if you input Morse code, the program translates it back into regular text.  To make things easy, the program has a simple and intuitive interface that lets you pick which translation you want to use—whether it’s text to Morse or Morse to text—and then it quickly gives you the translated result. It’s designed to be straightforward and user-friendly! |
|  |
|  |

## PART 2: Working Through Specific Examples

| **Task**  Write down clear and specific steps to solve a simple version of your problem you identified in Part 1.  **Requirements**  Complete the three steps below **for at least two distinct examples/scenarios**.   * State any necessary input data for your simplified problem. * Write clear and specific steps in English (not Java) detailing what the program will do to solve the problem. * Describe the specific result of your example/scenario.   **Inspiration**  When writing your entry below, ask yourself the following questions:   * Are there any steps that you don’t fully understand? These are places to spend more time working out the details. Consider adding additional smaller steps in these spots. * Remember that a computer program is very literal. Are there any steps that are unclear? Try giving the steps of your example/scenario to a friend or family member to read through and ask you questions about parts they don’t understand. Rewrite these parts as clearly as you can. * Are there interesting edge cases for your program? Try to start one of your examples/scenarios with input that matches this edge case. How does it change how your program might work? |
| --- |
| **Morse-to-text:** Covert ".." "-." "-" ".-." "---" "-" "---" ".---" ".-" "...-" ".-" to the text “intro to java”  **Input for the problem:** User inputs the more code ".." "-." "-" ".-." "---" "-" "---" ".---" ".-" "...-" ".-" to be converted.  **Steps involved:**   1. Define a dictionary, or hashmap, to map characters to their Morse code representation.   morse\_code\_dict = { 'A': '.-', 'B': '-...', 'C': '-.-.', 'D': '-..', 'E': '.', 'F': '..-.', 'G': '--.', 'H': '....', 'I': '..', 'J': '.---', 'K': '-.-', 'L': '.-..', 'M': '--', 'N': '-.', 'O': '---', 'P': '.--.', 'Q': '--.-', 'R': '.-.', 'S': '...', 'T': '-', 'U': '..-', 'V': '...-', 'W': '.--', 'X': '-..-', 'Y': '-.--', 'Z': '--..', ' ': '/'  }   1. Take the input: ".. -. - .-. --- / - --- / .--- .- ...- .-" 2. Next I’ll split the morse code into individual words using the split() method to separate the morse into words (using / as the delimiter) and then separate letters (with spaces as the delimiter) 3. Translate Morse Code to Text: Loop through each morse code letter, looking it up in the HashMap, and appending the corresponding character to a result string. 4. Output the Result: Print the final translated text (System.out.println). Where the morse ".." "-." "-" ".-." "---" "-" "---" ".---" ".-" "...-" ".-" would output: INTRO TO JAVA.   **Text-to-morse:** Convert “intro to java” to Morse Code: .. -. - .-. --- / - --- / .--- .- ...- .-  **Input for the problem:** User inputs the words “intro to java”  **Steps involved:**   1. Make a hashmap, as before, to map each character to its corresponding morse code. 2. Input the text “intro to java”. 3. Text to Morse Conversion: The program starts to loop through each character in the input text, looking it up in the HashMap, and using cthe orresponding Morse code to the morseCode string. A space is then added between Morse code letters, and a / is added between words. 4. Output: Final morse code is printed with the output being Morse Code: .. -. - .-. --- / - --- / .--- .- ...- .- |

## PART 3: Generalizing Into Pseudocode

| **Task**  Write out the general sequence your program will use, including all specific examples/scenarios you provided in Part 2.  **Requirements**   * Write pseudocode for the program in English but refer to Java program elements where they are appropriate. The pseudocode should represent the full functionality of the program, not just a simplified version. Pseudocode is broken down enough that the details of the program are no longer in any paragraph form. One statement per line is ideal.   **Help With Writing Pseudocode**   * Here are a few links that can help you write pseudocode with examples. Remember to check out part 3 of the Example Journal Template Submission if you have not already. Note: everyone will write pseudocode differently. There is no right or wrong way to write it, other than to make sure you write it clearly and in as much detail as you can so that it should be easy to convert to code later.   + <https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/>   + <https://www.wikihow.com/Write-Pseudocode>   **Inspiration**  When writing your entry below, ask yourself the following questions:   * Do you see common program elements and patterns in your specific examples/scenarios in Part 2, like variables, conditionals, functions, loops, and classes? These should be part of your pseudocode for the general sequence as well. * Are there places where the steps for your examples/scenarios in Part 2 diverged? These may be places where errors may occur later in the project. Make note of them. * When you are finished with your pseudocode, does it make sense, even to a person that does not know Java? Aim for the clearest description of the steps, as this will make it easier to convert into program code later. |
| --- |
| PSEUDOCODE:  // Initialize Morse Code mapping dictionaries  MorseCodeMap = InitializeMorseCodeMap() // Mapping from English letters to Morse Code  MorseCodeReverseMap = InitializeMorseCodeReverseMap() // Mapping from Morse Code to English letters  // Define function to initialize Morse Code mapping from English letters to Morse Code  function InitializeMorseCodeMap():  map = {} // Empty dictionary  // Populate the dictionary with mappings for English letters and numbers  for each character from 'A' to 'Z' and '0' to '9':  map[character] = corresponding Morse Code for character  return map  // Define a function to initialize Morse Code mapping from Morse Code to English letters  function InitializeMorseCodeReverseMap():  reverseMap = {} // Empty dictionary  // Populate the dictionary by reversing the entries in MorseCodeMap  for each (key, value) pair in MorseCodeMap:  reverseMap[value] = key // Reverse mapping  return reverseMap  // Define the main function  function Main():  // Create a Scanner object for user input  scanner = CreateScanner(System.in)  Print "Morse Code Translator"  Print "Choose an option:"  Print "1. English to Morse Code"  Print "2. Morse Code to English"  // Read user's choice  choice = scanner.nextInt()  scanner.nextLine() // Consume the newline character  if choice == 1:  Print "Enter the text to translate to Morse Code: "  text = scanner.nextLine()  // Call the EnglishToMorseCode function and print the result  morseCode = EnglishToMorseCode(text)  Print "Morse Code: " + morseCode  else if choice == 2:  Print "Enter the Morse Code to translate to English: "  morseCode = scanner.nextLine()  // Call the MorseCodeToEnglish function and print the result  english = MorseCodeToEnglish(morseCode)  Print "English: " + english  else:  Print "Invalid choice. Please choose 1 or 2."  // Close the scanner  scanner.close()  // Define the function to translate English text to Morse Code  function EnglishToMorseCode(text):  morseCode = "" // Initialize an empty string for Morse Code  for each character in text:  if character is a space:  morseCode += " " // Preserve spaces  else if character is in MorseCodeMap:  morseCode += MorseCodeMap[character] + " " // Add Morse code for the character  else:  morseCode += character // Keep non-mapped characters as is  return morseCode  // Define the function to translate Morse Code to English  function MorseCodeToEnglish(morseCode):  english = "" // Initialize an empty string for English text  words = Split morseCode by " " // Split words by three spaces  for each word in words:  letters = Split word by " " // Split letters by single spaces  for each letter in letters:  if letter is in MorseCodeReverseMap:  english += MorseCodeReverseMap[letter] // Add English character for the Morse code  else:  english += letter // Keep non-mapped Morse Code as is  english += " " // Preserve spaces between words  return Trim(english) // Remove trailing spaces  // Call the main function to start the program  Main(): |

## PART 4: Testing Your Program

| **Task**  While writing and testing your program code, describe your tests, record any errors, and state your approach to fixing the errors.  **Requirements**   * For at least one of your test cases, describe how your choices for the test helped you understand whether the program was running correctly or not.   For each error that occurs while writing and testing your code:   * Record the details of the error from your IDE. A screenshot or copy-and-paste of the text into the journal entry is acceptable. * Describe what you attempted in order to fix the error. Clearly identify which approach was the one that worked.   **Inspiration**  When writing your entry below, ask yourself the following questions:   * Have you tested edge cases and special cases for the inputs of your program code? Often these unexpected values can cause errors in the operation of your program. * Have you tested opportunities for user error? If a user is asked to provide an input, what happens when they give the wrong type of input, like a letter instead of a number, or vice versa? * Did the outcome look the way you expected? Was it formatted correctly? * Does your output align with the solution to the problem you coded for? |
| --- |
| As I mentioned earlier, I’m not very experienced with Java, and I found it significantly more challenging compared to Python. While coding, I ran into several errors, and here are some of the most common ones I encountered:  Null Pointer Exception:  I forgot to properly initialize the Scanner object, which caused a NullPointerException.  Error: NullPointerException at line \*linenumber\*: scanner is null  Fix: I made sure to initialize the Scanner object correctly before using it.  Resource Leak:  I didn’t close the Scanner object properly, which led to a resource leak.  Error: Resource leak at line \*linenumber\*: Scanner not closed  Fix: I resolved this by calling the close() method on the Scanner object before the program ended.  String Manipulation Error:  The translated Morse code had missing spaces between characters, making it hard to read.  Fix: I adjusted the code to ensure proper spacing between Morse code characters.  Typos:  I made an endless number of typos while writing the code, which slowed me down quite a bit. |

## PART 5: Commenting Your Program

| **Task**  Submit your full program code, including thorough comments describing what each portion of the program should do when working correctly.  **Requirements**   * The purpose of the program and each of its parts should be clear to a reader who does not know the Java programming language.   **Inspiration**  When writing your entry, you are encouraged to consider the following:   * Is each section or sub-section of your code commented to describe what the code is doing? * Give your code with comments to a friend or family member to review. Add additional comments to spots that confuse them to make it clearer. |
| --- |
| Here is where I organized my program into two files and wrapped it all up into one to allow for a more seamless interaction.  **Main.java:**  import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  boolean running = true;  while (running) {  System.out.println("\nMorse Code Translator");  System.out.println("1. Text to Morse Code");  System.out.println("2. Morse Code to Text");  System.out.println("3. Exit");  System.out.print("Choose an option (1/2/3): ");  int choice = scanner.nextInt();  scanner.nextLine(); // Consume the newline character  switch (choice) {  case 1:  System.out.print("Enter text to translate to Morse code: ");  String text = scanner.nextLine();  String morseCode = MorseCodeTranslator.textToMorse(text);  System.out.println("Morse Code: " + morseCode);  break;  case 2:  System.out.print("Enter Morse code to translate to text: ");  String morseInput = scanner.nextLine();  String translatedText = MorseCodeTranslator.morseToText(morseInput);  System.out.println("Translated Text: " + translatedText);  break;  case 3:  running = false;  System.out.println("Exiting the program. Goodbye!");  break;  default:  System.out.println("Invalid option. Please try again.");  }  }  scanner.close(); // Close the scanner to prevent resource leak  }  }  **MorseCodeTranslator.java:**  import java.util.HashMap;  public class MorseCodeTranslator {  // Morse code mappings for letters, numbers, and space  private static final HashMap<Character, String> textToMorseMap = new HashMap<>();  private static final HashMap<String, Character> morseToTextMap = new HashMap<>();  static {  // Initialize the text-to-Morse mappings  textToMorseMap.put('A', ".-");  textToMorseMap.put('B', "-...");  textToMorseMap.put('C', "-.-.");  textToMorseMap.put('D', "-..");  textToMorseMap.put('E', ".");  textToMorseMap.put('F', "..-.");  textToMorseMap.put('G', "--.");  textToMorseMap.put('H', "....");  textToMorseMap.put('I', "..");  textToMorseMap.put('J', ".---");  textToMorseMap.put('K', "-.-");  textToMorseMap.put('L', ".-..");  textToMorseMap.put('M', "--");  textToMorseMap.put('N', "-.");  textToMorseMap.put('O', "---");  textToMorseMap.put('P', ".--.");  textToMorseMap.put('Q', "--.-");  textToMorseMap.put('R', ".-.");  textToMorseMap.put('S', "...");  textToMorseMap.put('T', "-");  textToMorseMap.put('U', "..-");  textToMorseMap.put('V', "...-");  textToMorseMap.put('W', ".--");  textToMorseMap.put('X', "-..-");  textToMorseMap.put('Y', "-.--");  textToMorseMap.put('Z', "--..");  textToMorseMap.put('0', "-----");  textToMorseMap.put('1', ".----");  textToMorseMap.put('2', "..---");  textToMorseMap.put('3', "...--");  textToMorseMap.put('4', "....-");  textToMorseMap.put('5', ".....");  textToMorseMap.put('6', "-....");  textToMorseMap.put('7', "--...");  textToMorseMap.put('8', "---..");  textToMorseMap.put('9', "----.");  textToMorseMap.put(' ', "/");  // Initialize the Morse-to-text mappings  for (Character key : textToMorseMap.keySet()) {  morseToTextMap.put(textToMorseMap.get(key), key);  }  }  // Translate text to Morse code  public static String textToMorse(String text) {  StringBuilder morseCode = new StringBuilder();  text = text.toUpperCase(); // Convert text to uppercase for consistency  for (char c : text.toCharArray()) {  if (textToMorseMap.containsKey(c)) {  morseCode.append(textToMorseMap.get(c)).append(" ");  } else {  morseCode.append(c); // Keep non-mapped characters as is  }  }  return morseCode.toString().trim();  }  // Translate Morse code to text  public static String morseToText(String morseCode) {  StringBuilder text = new StringBuilder();  String[] words = morseCode.split(" / "); // Split into words  for (String word : words) {  String[] letters = word.split(" "); // Split into letters  for (String letter : letters) {  if (morseToTextMap.containsKey(letter)) {  text.append(morseToTextMap.get(letter));  } else {  text.append(letter); // Keep non-mapped Morse code as is  }  }  text.append(" "); // Add space between words  }  return text.toString().trim();  }  } |

## PART 6: Your Completed Program

| **Task**  Provide the IDE link to your full program code.  **Requirements**   * The program must work correctly with all the comments included in the program.   **Inspiration**   * Check before submitting your Touchstone that your final version of the program is running successfully. |
| --- |
| It was an interesting project! I did need to spend a lot more time figuring out the syntax in Java than I did in Python, but it was interesting nonetheless.  Here’s my IDE code link:  https://onlinegdb.com/Zq2Nbl9py |