Oracle academy

Practice session 5.2:

1.Consider you are asked to decode a secret message. The coded message is in numbers and each number stands for a specific letter. You discover enough of the secret code to decode the current message. So far, you know:

• 1 represents “D”

• 2 represents “W”

• 3 represents “E”

• 4 represents “L”

• 5 represents “H”

• 6 represents “O”

• 7 represents “R”

Write a program that prompts the user for 10 numbers, one at a time, and prints out the decoded message. If the user enters a number that is not one of those already deciphered, prompt him/her for a new number. Test your code with the following input: 5 3 4 4 6 2 6 7 4 1

Code:

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class DecodeMessage {

public static void main(String[] args) {

Map<Integer, Character> codeMap = new HashMap<>();

codeMap.put(1, 'D');

codeMap.put(2, 'W');

codeMap.put(3, 'E');

codeMap.put(4, 'L');

codeMap.put(5, 'H');

codeMap.put(6, 'O');

codeMap.put(7, 'R');

Scanner scanner = new Scanner(System.in);

StringBuilder decodedMessage = new StringBuilder();

System.out.println("Enter 10 numbers (1-7) to decode the message:");

for (int i = 0; i < 10; i++) {

while (true) {

System.out.print("Enter a number: ");

int num = scanner.nextInt();

if (codeMap.containsKey(num)) {

decodedMessage.append(codeMap.get(num));

break;

} else {

System.out.println("Invalid number. Please enter a number between 1 and 7.");

}

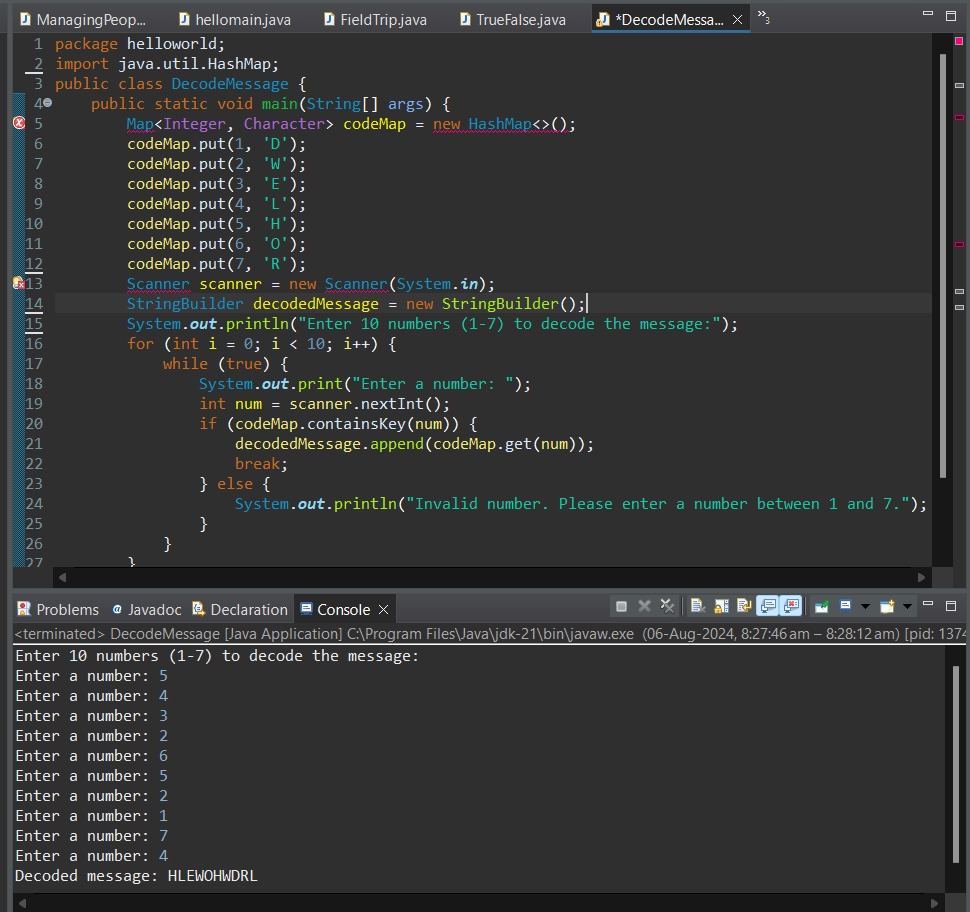
}

}

System.out.println("Decoded message: " + decodedMessage.toString());

}

}



2. Suppose you are implementing a search routine that searches through a String, character by character, until it finds a space character. As soon as you find the first space character, you decide that you do not want to continue searching the string. If you are using a WHILE loop and your loop will continue to execute until you have gone through the entire string, should you use the keyword break or continue when you find the first space character? Why? Why would you not use the other keyword?

Code:

public class SearchSpaceCharacter {

public static void main(String[] args) {

String str = "Hello World";

int index = 0;

while (index < str.length()) {

if (str.charAt(index) == ' ') {

System.out.println("Space character found at index: " + index);

break;

}

index++;

}

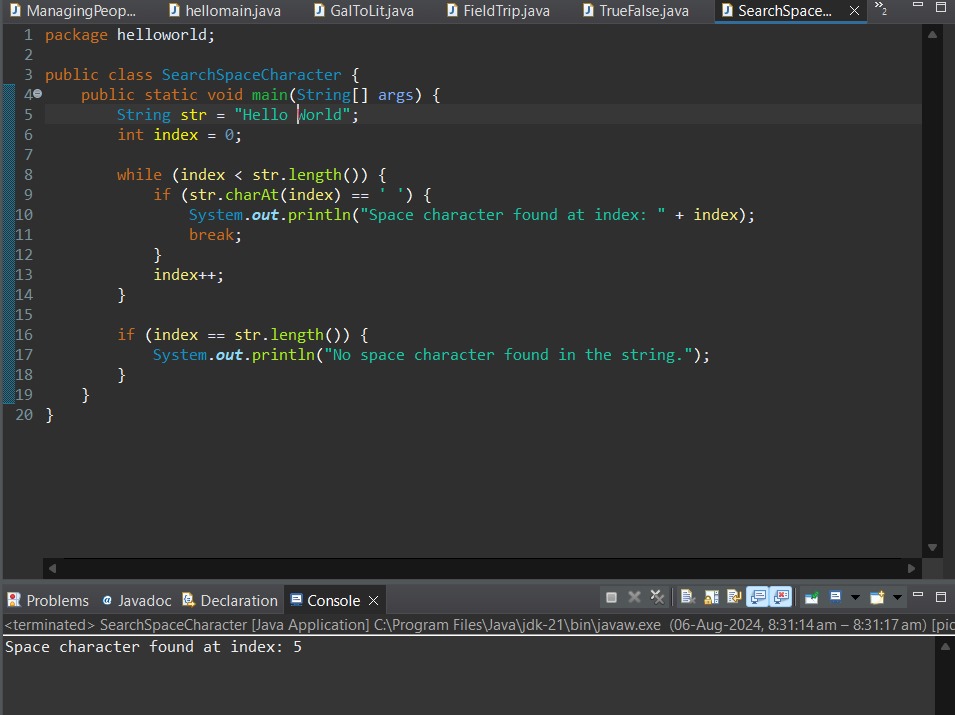
if (index == str.length()) {

System.out.println("No space character found in the string.");

}

}

}



3. Imagine you are writing a program that prints out the day of the week (Sunday, Monday, Tuesday, etc.) for each day of the year. Before the program executes, can you tell how many times the loop will execute? Assume the year is not a Leap year. Given your answer, which type of loop would you need to implement? Explain your reasoning

Code:

public class PrintDaysOfWeek {

public static void main(String[] args) {

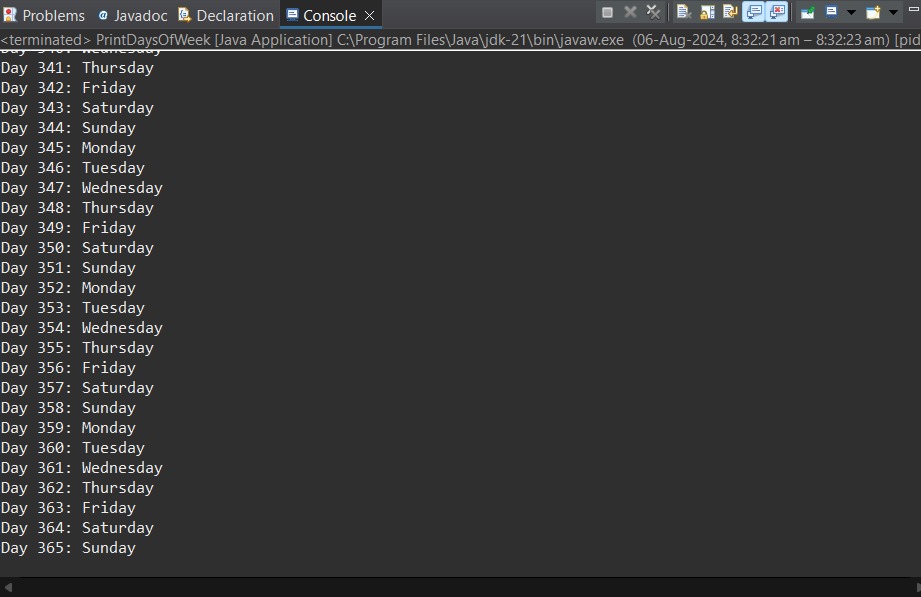
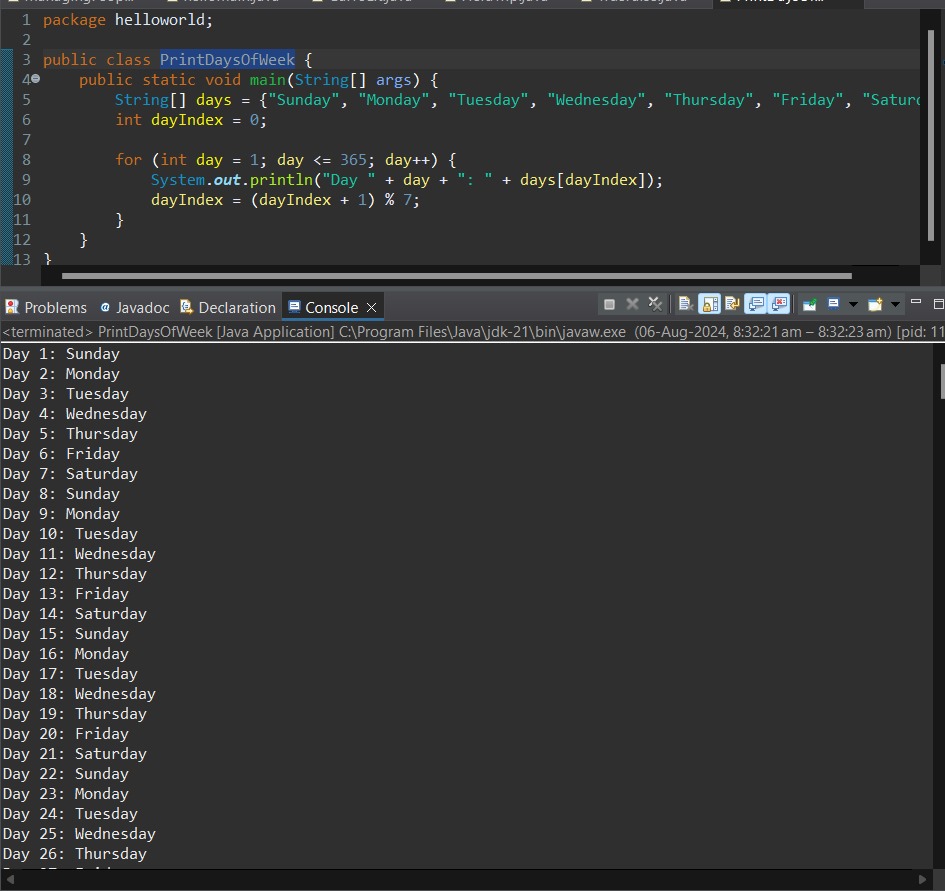
String[] days = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};

int dayIndex = 0;

for (int day = 1; day <= 365; day++) {

System.out.println("Day " + day + ": " + days[dayIndex]);

dayIndex = (dayIndex + 1) % 7;



4. An anagram is a word or a phrase made by transposing the letters of another word or phrase; for example, "parliament" is an anagram of "partial men," and "software" is an anagram of "swear oft." Write a program that figures out whether one string is an anagram of another string. The program should ignore white space and punctuation.

Code:

import java.util.Arrays;

public class AnagramChecker {

public static void main(String[] args) {

System.out.println(areAnagrams("parliament", "partial men")); // Output: true

System.out.println(areAnagrams("software", "swear oft")); // Output: true

System.out.println(areAnagrams("hello", "world")); // Output: false

}

public static boolean areAnagrams(String str1, String str2) {

// Remove whitespace and punctuation, convert to lowercase

str1 = str1.replaceAll("\\W+", "").toLowerCase();

str2 = str2.replaceAll("\\W+", "").toLowerCase();

// Convert strings to char arrays and sort

char[] charArray1 = str1.toCharArray();

char[] charArray2 = str2.toCharArray();

Arrays.sort(charArray1);

Arrays.sort(charArray2);

// Compare sorted char arrays

return Arrays.equals(charArray1, charArray2);

}

}

