

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());
```

```
}
```

```
}
```

The screenshot shows an IDE with a Java file named `*DecodeMessa...`. The code defines a `DecodeMessage` class with a `main` method. It uses a `HashMap` to map numbers 1-7 to characters 'D', 'W', 'E', 'L', 'H', 'O', 'R'. A `Scanner` is used to prompt the user for 10 numbers. For each number, it checks if it exists in the map and appends the corresponding character to a `StringBuilder`. The console output shows the user entering the numbers 5, 4, 3, 2, 6, 5, 2, 1, 7, 4, resulting in the decoded message "HLEWOHWDRL".

```
1 package helloworld;
2 import java.util.HashMap;
3 public class DecodeMessage {
4     public static void main(String[] args) {
5         Map<Integer, Character> codeMap = new HashMap<>();
6         codeMap.put(1, 'D');
7         codeMap.put(2, 'W');
8         codeMap.put(3, 'E');
9         codeMap.put(4, 'L');
10        codeMap.put(5, 'H');
11        codeMap.put(6, 'O');
12        codeMap.put(7, 'R');
13        Scanner scanner = new Scanner(System.in);
14        StringBuilder decodedMessage = new StringBuilder();
15        System.out.println("Enter 10 numbers (1-7) to decode the message:");
16        for (int i = 0; i < 10; i++) {
17            while (true) {
18                System.out.print("Enter a number: ");
19                int num = scanner.nextInt();
20                if (codeMap.containsKey(num)) {
21                    decodedMessage.append(codeMap.get(num));
22                    break;
23                } else {
24                    System.out.println("Invalid number. Please enter a number between 1 and 7.");
25                }
26            }
27        }
28    }
29 }
```

Problems Javadoc Declaration Console X

<terminated> DecodeMessage [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (06-Aug-2024, 8:27:46 am – 8:28:12 am) [pid: 1374]

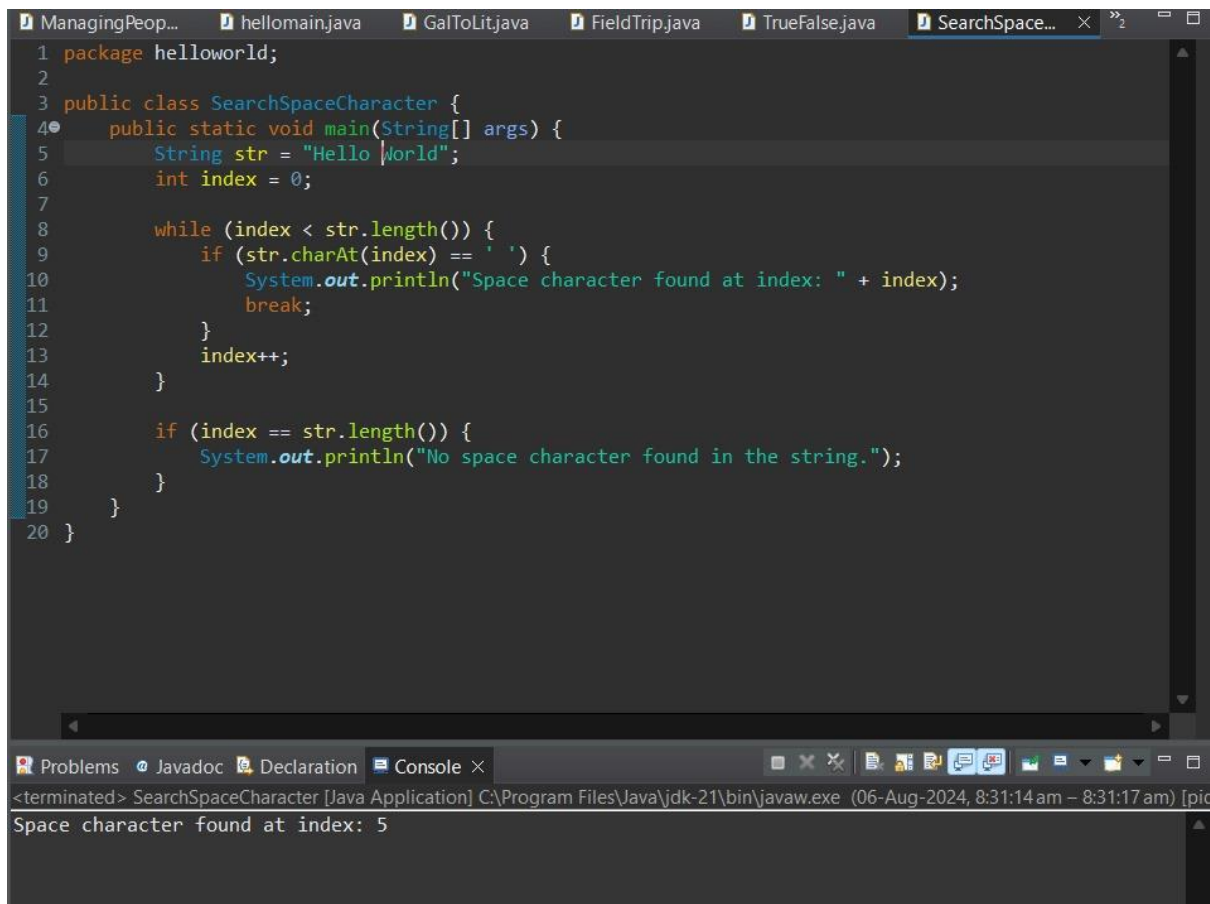
Enter 10 numbers (1-7) to decode the message:
Enter a number: 5
Enter a number: 4
Enter a number: 3
Enter a number: 2
Enter a number: 6
Enter a number: 5
Enter a number: 2
Enter a number: 1
Enter a number: 7
Enter a number: 4
Decoded message: HLEWOHWDRL

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.");  
}  
}  
}
```

A screenshot of an IDE window with several tabs open: 'ManagingPeop...', 'hellomain.java', 'GalToLit.java', 'FieldTrip.java', 'TrueFalse.java', and 'SearchSpace...'. The 'SearchSpace...' tab is active, showing a Java program. The code defines a class 'SearchSpaceCharacter' with a 'main' method. Inside 'main', a string 'str' is initialized to 'Hello World', and an 'index' variable is set to 0. A 'while' loop iterates as long as 'index' is less than 'str.length()'. Inside the loop, an 'if' statement checks if 'str.charAt(index)' is equal to a space character. If true, it prints 'Space character found at index: ' followed by the index value and then breaks the loop. If the loop completes without finding a space, it prints 'No space character found in the string.'. The console at the bottom shows the output: 'Space character found at index: 5'.

```
1 package helloworld;
2
3 public class SearchSpaceCharacter {
4     public static void main(String[] args) {
5         String str = "Hello World";
6         int index = 0;
7
8         while (index < str.length()) {
9             if (str.charAt(index) == ' ') {
10                 System.out.println("Space character found at index: " + index);
11                 break;
12             }
13             index++;
14         }
15
16         if (index == str.length()) {
17             System.out.println("No space character found in the string.");
18         }
19     }
20 }
```

Problems Javadoc Declaration Console ×

<terminated> SearchSpaceCharacter [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (06-Aug-2024, 8:31:14 am – 8:31:17 am) [pic
Space character found at index: 5

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;
```

```
1 package helloworld;
2
3 public class PrintDaysOfWeek {
4     public static void main(String[] args) {
5         String[] days = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};
6         int dayIndex = 0;
7
8         for (int day = 1; day <= 365; day++) {
9             System.out.println("Day " + day + ": " + days[dayIndex]);
10            dayIndex = (dayIndex + 1) % 7;
11        }
12    }
13 }
```

Problems Javadoc Declaration Console X

<terminated> PrintDaysOfWeek [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (06-Aug-2024, 8:32:21 am – 8:32:23 am) [pid: 11

Day 1: Sunday
Day 2: Monday
Day 3: Tuesday
Day 4: Wednesday
Day 5: Thursday
Day 6: Friday
Day 7: Saturday
Day 8: Sunday
Day 9: Monday
Day 10: Tuesday
Day 11: Wednesday
Day 12: Thursday
Day 13: Friday
Day 14: Saturday
Day 15: Sunday
Day 16: Monday
Day 17: Tuesday
Day 18: Wednesday
Day 19: Thursday
Day 20: Friday
Day 21: Saturday
Day 22: Sunday
Day 23: Monday
Day 24: Tuesday
Day 25: Wednesday
Day 26: Thursday

Problems Javadoc Declaration Console X

<terminated> PrintDaysOfWeek [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (06-Aug-2024, 8:32:21 am – 8:32:23 am) [pid: 11

Day 341: Thursday
Day 342: Friday
Day 343: Saturday
Day 344: Sunday
Day 345: Monday
Day 346: Tuesday
Day 347: Wednesday
Day 348: Thursday
Day 349: Friday
Day 350: Saturday
Day 351: Sunday
Day 352: Monday
Day 353: Tuesday
Day 354: Wednesday
Day 355: Thursday
Day 356: Friday
Day 357: Saturday
Day 358: Sunday
Day 359: Monday
Day 360: Tuesday
Day 361: Wednesday
Day 362: Thursday
Day 363: Friday
Day 364: Saturday
Day 365: Sunday

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);
    }
}
```

```
1 package helloworld;
2 import java.util.Arrays;
3
4 public class AnagramChecker {
5     public static void main(String[] args) {
6         System.out.println(areAnagrams("parliament", "partial men")); // Output: true
7         System.out.println(areAnagrams("software", "swear oft")); // Output: true
8         System.out.println(areAnagrams("hello", "world")); // Output: false
9     }
10
11     public static boolean areAnagrams(String str1, String str2) {
12         // Remove whitespace and punctuation, convert to lowercase
13         str1 = str1.replaceAll("\\W+", "").toLowerCase();
14         str2 = str2.replaceAll("\\W+", "").toLowerCase();
15
16         // Convert strings to char arrays and sort
17         char[] charArray1 = str1.toCharArray();
18         char[] charArray2 = str2.toCharArray();
19         Arrays.sort(charArray1);
20         Arrays.sort(charArray2);
21
22         // Compare sorted char arrays
23         return Arrays.equals(charArray1, charArray2);
24     }
25 }
```

Problems Javadoc Declaration Console ×

<terminated> AnagramChecker [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (06-Aug-2024, 8:35:53 am – 8:35:56 am)

true
true
false