

CSA0961 – JAVA

JF PRACTISE 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

PROGRAM :

```
import java.util.Scanner;
```

```
import java.util.HashMap;
```

```
import java.util.Map;
```

```
public class Main {
```

```
    public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);
```

```
        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');
```

```
        StringBuilder decodedMessage = new StringBuilder();
```

```
        System.out.println("Enter 10 numbers, each representing a letter (1-7):");
```

```
        int count = 0;
```

```
        while (count < 10) {
```

```

        System.out.print("Enter number " + (count + 1) + ": ");

        int number = scanner.nextInt();

        if (codeMap.containsKey(number)) {

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

            count++;

        } else {

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

        }

    }

}

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

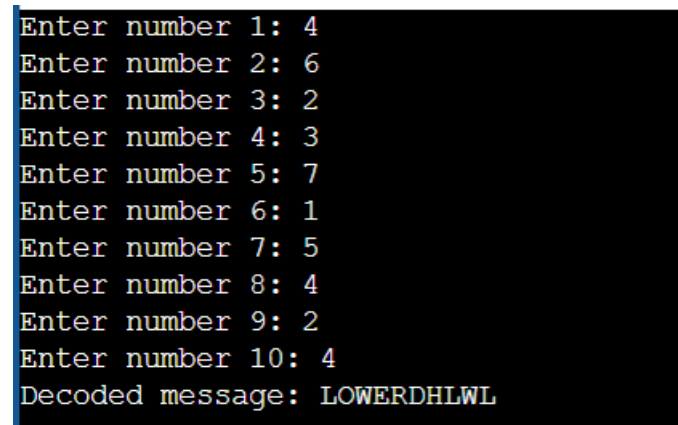
scanner.close();

}

}

```

OUTPUT :



```

Enter number 1: 4
Enter number 2: 6
Enter number 3: 2
Enter number 4: 3
Enter number 5: 7
Enter number 6: 1
Enter number 7: 5
Enter number 8: 4
Enter number 9: 2
Enter number 10: 4
Decoded message: LOWERDHLWL

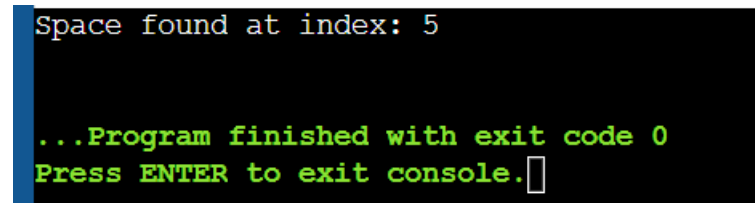
```

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?

PROGRAM : BY USING BREAK STATEMENT :

```
public class Main {  
    public static void main(String[] args) {  
        String str = "Hello World";  
        int index = 0;  
  
        while (index < str.length()) {  
            char ch = str.charAt(index);  
            if (ch == ' ') {  
                System.out.println("Space found at index: " + index);  
                break;  
            }  
            index++;  
        }  
    }  
}
```

OUTPUT :

A screenshot of a Java program's output in a console window. The text "Space found at index: 5" is displayed in a light blue font. Below it, the text "...Program finished with exit code 0" and "Press ENTER to exit console." are shown in a light green font. A small white cursor is visible at the end of the last line.

```
Space found at index: 5  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

WHY NOT USE 'CONTINUE' :

PROGRAM :

```
public class Main {  
    public static void main(String[] args) {  
        String str = "Hello World";  
        int index = 0;  
  
        while (index < str.length()) {  
            char ch = str.charAt(index);  
            if (ch == ' ') {
```

```

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

}

}

```

OUTPUT :

```

Space found at index: 5
Space found at index: 5
Space found at index: 5
Space found at index: 5
Space found at index: 5
Space found at index: 5
Space found at index: 5
Space found at index: 5
Space found at index: 5
Space found at index: 5
Space 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.

BY USING FOR LOOP :

PROGRAM :

```

public class Main {
    public static void main(String[] args) {

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

        int totalDays = 365;

        for (int i = 0; i < totalDays; i++) {
            String dayOfWeek = daysOfWeek[i % daysOfWeek.length];
            System.out.println("Day " + (i + 1) + ": " + dayOfWeek);
        }
    }
}

```

```

    }
}
}

```

OUTPUT :

.....



```

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

```

BY USING WHILE LOOP :

```

public class Main {
    public static void main(String[] args) {

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

        int totalDays = 365;
        int i = 0;

        while (i < totalDays) {

            String dayOfWeek = daysOfWeek[i % daysOfWeek.length];
            System.out.println("Day " + (i + 1) + ": " + dayOfWeek);
            i++;
        }
    }
}

```

OUTPUT :

```
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.

PROGRAM :

```
import java.util.Arrays;

public class Main {

    private static String normalizeString(String s) {
        return s.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();
    }

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

        String normalizedStr1 = normalizeString(str1);
        String normalizedStr2 = normalizeString(str2);

        if (normalizedStr1.length() != normalizedStr2.length()) {
            return false;
        }

        char[] array1 = normalizedStr1.toCharArray();
        char[] array2 = normalizedStr2.toCharArray();
```

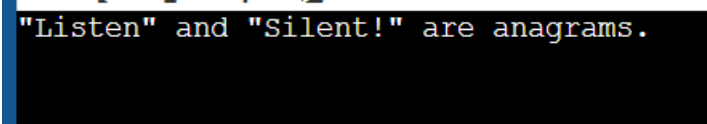
```
Arrays.sort(array1);
Arrays.sort(array2);

return Arrays.equals(array1, array2);
}

public static void main(String[] args) {
    String string1 = "Listen";
    String string2 = "Silent!";

    if (areAnagrams(string1, string2)) {
        System.out.println "\"" + string1 + "\" and \"" + string2 + "\" are anagrams.");
    } else {
        System.out.println "\"" + string1 + "\" and \"" + string2 + "\" are not anagrams.");
    }
}
}
```

OUTPUT :

A screenshot of a terminal window with a black background and a blue vertical bar on the left. The text displayed is "Listen" and "Silent!" are anagrams. in a monospaced font.

```
"Listen" and "Silent!" are anagrams.
```