

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.

```
main.java ×
package practise4;
import java.util.HashMap;
import java.util.Map;
public class Main{
    public static void main(String[] args)
    {
        Map<Integer,Character>code=new HashMap<>();
        code.put(1, 'D');
        code.put(2, 'W');
        code.put(3, 'E');
        code.put(4, 'L');
        code.put(5, 'H');
        code.put(6, 'O');
        code.put(7, 'R');

        int num=1234567;
        String sm="";
        while(num!=0)
        {
            int temp=num%10;
            num=num/10;
            if (temp<=7)
            {
                sm+=(code.get(temp));
            }
            else {
                System.out.println("invalid number please enter a valid number...");
            }
        }
        String reverse=new StringBuilder(sm).reverse().toString();

        System.out.println("decode message:"+reverse);
    }
}
```

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?

A. **break Keyword:** This keyword is used to exit the loop immediately. When you find the first space character, you want to stop searching the string and exit the loop. break allows you to do that instantly, terminating the loop and continuing with the next part of the program.

continue Keyword: This keyword skips the current iteration and proceeds to the next iteration of the loop. In your case, this would not be helpful because you want to stop the loop entirely when you find a space, not just skip processing the current character.

Using continue would cause the loop to proceed to the next character after the space, which is not what you want. You need the loop to stop once the space is found.

Using Break:

```
public class SearchSpace {  
    public static void main(String[] args) {  
        String text = "Hello World";  
        int index = 0;  
        while (index < text.length()) {  
            char currentChar = text.charAt(index);  
            if (currentChar == ' ') {  
                System.out.println("Space found at index: " + index);  
                break; // Exit the loop immediately  
            }    index++;  
        }  
        System.out.println("Search completed.");  
    }  
}
```

Using Continue:

```
public class SearchSpace {  
    public static void main(String[] args) {  
        String text = "Hello World";  
        int index = 0;  
        while (index < text.length()) {  
            char currentChar = text.charAt(index);  
            if (currentChar == ' ') {  
                System.out.println("Space found at index: " + index);  
                continue; // Skip to the next iteration (not desired in this case)  
            }  
            index++;  
        }  
    }  
}
```

```

    }

    System.out.println("Search completed.");
}
}

```

Conclusion

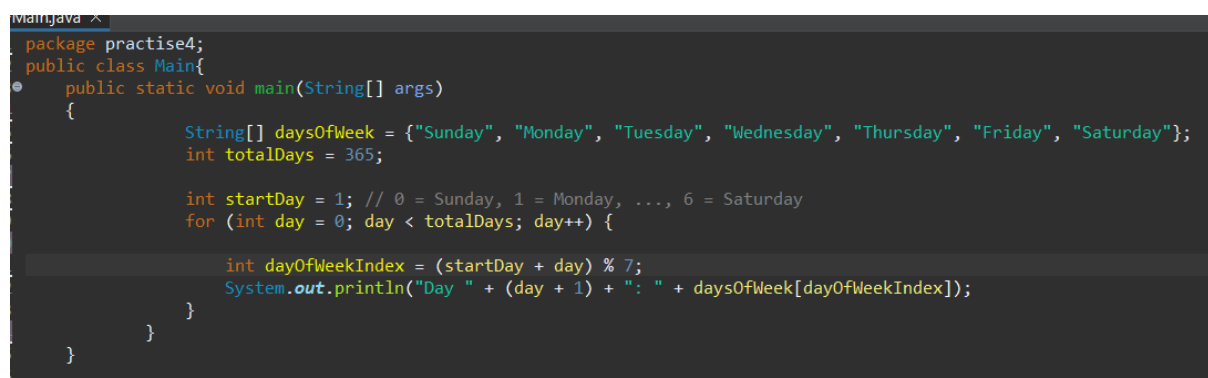
Use break to stop the loop immediately upon finding the first space character, as it directly meets the requirement of stopping the search once the space is found.

continue is not appropriate here because it would not stop the loop; it would just skip the rest of the current iteration and proceed to the next character, causing unnecessary continued processing of 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

A. The loop will execute 365 times, as there are 365 days in a non-leap year.

Given that we know the exact number of iterations (365), the most appropriate loop to use is a for loop. The for loop is ideal for scenarios where the number of iterations is known beforehand. It provides a clear, concise way to iterate a fixed number of times.



```

main.java x
package practise4;
public class Main{
    public static void main(String[] args)
    {
        String[] daysOfWeek = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};
        int totalDays = 365;

        int startDay = 1; // 0 = Sunday, 1 = Monday, ..., 6 = Saturday
        for (int day = 0; day < totalDays; day++) {

            int dayOfWeekIndex = (startDay + day) % 7;
            System.out.println("Day " + (day + 1) + ": " + daysOfWeek[dayOfWeekIndex]);

        }
    }
}

```

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

A.

```

Main.java ×
package practise4;
import java.util.Arrays;
public class Main {
    public static boolean isAnagram(String str1, String str2) {
        // Remove white spaces and punctuation
        str1 = str1.replaceAll("[^a-zA-Z]", "").toLowerCase();
        str2 = str2.replaceAll("[^a-zA-Z]", "").toLowerCase();
        // Check if the lengths are equal after removing spaces and punctuation
        if (str1.length() != str2.length()) {
            return false;
        }
        // Convert strings to char arrays and sort them
        char[] charArray1 = str1.toCharArray();
        char[] charArray2 = str2.toCharArray();
        Arrays.sort(charArray1);
        Arrays.sort(charArray2);

        // Compare sorted char arrays
        return Arrays.equals(charArray1, charArray2);
    }
    public static void main(String[] args)
    {
        String word1 = "parliament";
        String word2 = "partial men";
        if (isAnagram(word1, word2)) {
            System.out.println(word1 + " and " + word2 + " are anagrams.");
        } else {
            System.out.println(word1 + " and " + word2 + " are not anagrams.");
        }
    }
}

```

Problems @ Javadoc Declaration Console ×

minated> Main [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (6 Aug 2024, 9:33:31 am – 9:33:33 am) [pid: 219
 iament and partial men are anagrams.