Java Programming 1 - Week 8 Notes

Hia Al Saleh

October 24th, 2024

Contents

1 SpyTek Caesar Cipher **2**
1.1 Project Request 2
1.2 Plan of Action 2

2 Caesar Cipher **2**
2.1 Key Information 4

3 String Manipulation **4**

4 Typecasting **5**

5 ASCII Table **6**

1 SpyTek Caesar Cipher

In this week's lesson, we covered topics related to encryption and decryption using the Caesar Cipher. We focused on the following core concepts:

- String manipulation
- Typecasting
- The `&&` and `||` operators
- Introduction to Object-Oriented Programming (OOP)

1.1 Project Request

We were introduced to a client project called **SpyTek**, where the task was to build a program capable of encrypting and decrypting email messages using the Caesar Cipher.

The client provided the following requirements:

- The program should ask the user if they wish to encrypt or decrypt a message.
- It should then prompt the user to input the message and the key.
- The program will output the encrypted or decrypted message to the user.

1.2 Plan of Action

To solve this problem, we created a function that handles both encryption and decryption. The function takes the following parameters:

- `text` - the message to be processed.
- `key` - the number of positions to shift in the alphabet.
- `mode` - whether to encrypt or decrypt.

2 Caesar Cipher

The Caesar Cipher is a substitution cipher that shifts each letter in the alphabet by a certain number of positions based on the key. Here's how it works:

- Encryption shifts each letter forward by the key value.
- Decryption shifts each letter backward by the key value.

For example:

- Shifting the letter 'a' by 5 results in 'f'.

- The reverse operation decrypts the message back to 'a'.

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

public class CaesarCypher1 {
    public static void main(String[] args) {
        final boolean DEBUG = false;
        Scanner input = new Scanner(System.in);
        //encryption or decryption
        System.out.println("What would you like to do?\n"
            +
                "1) encrypt\n" +
                "2) decrypt");
        int mode = input.nextInt();
        //key
        System.out.println("Enter the key size");
        int key = input.nextInt();
        //message
        input.nextLine();
        System.out.println("Enter the message");
        String msg = input.nextLine();
        //output result
        if (mode == 1){
            System.out.println(encrypt(msg, key));
        }
        else if(mode ==2) {
            System.out.println(decrypt(msg, key));
        }
        else {
            System.out.println("Choose 1 or 2");
        }
        if (DEBUG) System.out.println(mode+ " "+ key +" \n
            "+msg);

        input.close();
    }

    public static String encrypt(String msg, int key){
        StringTokenizer tokenizer = new StringTokenizer(
            msg.toLowerCase(), " ");
        String cypher = "";
        while (tokenizer.hasMoreTokens()){
            String value = tokenizer.nextToken();
            for (int i = 0; i< value.length(); i++){
                char character = value.charAt(i);
                int asciiValue = (int)value.charAt(i)+key;
                if(asciiValue > 122){
                    asciiValue-=26;
                }
```

```
                character = (char)asciiValue;
                cypher+=character;
            }
            cypher+=" ";
        }
        return cypher; // Change to message
    }
    public static String decrypt(String cypher, int key){
        StringTokenizer tokenizer = new StringTokenizer(
            cypher.toLowerCase(), " ");
        String text = "";
        while (tokenizer.hasMoreTokens()){
            String value = tokenizer.nextToken();
            for (int i = 0; i< value.length(); i++){
                char character = value.charAt(i);
                int asciiValue = (int)value.charAt(i)-key;
                if(asciiValue < 97){
                    asciiValue+=26;
                }
                character = (char)asciiValue;
                text+=character;
            }
            text+=" ";
        }
        return text; // Change to message
    }
}
```

2.1 Key Information

Before diving deeper into the implementation, we reviewed the following key topics:

- **Encryption**: The process of encoding a message so that only the intended recipient can read it.
- **Decryption**: The process of converting the ciphertext back into readable text.

3 String Manipulation

String manipulation is key to the Caesar Cipher, and Java provides several tools to handle strings:

- `substring(begin, end)` - Extracts part of a string.
- `toUpperCase()` - Converts a string to uppercase.
- `toLowerCase()` - Converts a string to lowercase.

- `replace(old, new)` - Replaces characters in a string.
- `charAt(index)` - Returns the character at the specified index.
- `length()` - Returns the length of the string.

4 Typecasting

In Java, typecasting allows you to convert between data types. There are two types of casting:

- **Implicit casting**: Java automatically converts smaller data types to larger ones (e.g., from int to double).
- **Explicit casting**: Requires manual conversion using the target type in parentheses (e.g., `(int) value`).

```
// Example of typecasting
public class TypeCastingExample {
    public static void main(String[] args) {
        // Implicit casting (int to double)
        int intValue = 10;
        double doubleValue = intValue;
        System.out.println("Implicit cast: " + doubleValue
            );

        // Explicit casting (double to int)
        double anotherDouble = 9.78;
        int anotherInt = (int) anotherDouble;
        System.out.println("Explicit cast: " + anotherInt)
            ;
    }
}
```

5 ASCII Table

We also used the ASCII table, which represents characters as numeric values. This allowed us to shift characters in the cipher by manipulating their ASCII values directly.

```
// Example of using ASCII values in Caesar Cipher
public class ASCIICipher {
    public static void main(String[] args) {
        char c = 'a';
        int asciiValue = (int) c;
        System.out.println("ASCII value of 'a': " +
            asciiValue);

        // Shift character by 3 using ASCII
        char shiftedChar = (char) (c + 3);
        System.out.println("Shifted character: " +
            shiftedChar);
    }
}
```