| Akademia Nauk Stosowanych<br>Teoretyczne i technologiczne podstawy multimediów<br>– IS rok 3 | |
|---|---|
| **Imię i Nazwisko:** Emilian Kochanek | **Grupa:** L2 |
| **Data:** 11.10.2022 | **Symbol:** TiTPM_02 |

Na początku program prosi nas o podanie ciągu znaków do zaszyfrowania. Następnie przechodzi do funkcji zliczającej wystąpienie. Funkcja ta wykorzystuje Stream api, która iteruje się po znakach w Stringu i grupuje je po znaku i zlicza ich wystąpienia. Wynik ten jest umieszczany w mapie.

```
public static Map<Character, Long> zliczWystapienia(String ciagZnakow){
    Map <Character, Long> mapStringToCount = ciagZnakow
            .chars().mapToObj(c -> (char) c)
            .collect(Collectors.groupingBy(c -> c, Collectors.counting()));
    return sort(mapStringToCount);
}
```

Mapa przekazywana jest do funkcji sortującej po wartościach w kolejności malejącej. Wartości te przekazywane są to nowej mapy, która utrzymuje wartości z kluczami w kolejności włożonej.

```
public static Map<Character, Long> sort(Map<Character, Long> mapa){
    LinkedHashMap<Character, Long> result = new LinkedHashMap<>();
    mapa.entrySet() Set<Map<K, V>.Entry<Character, Long>>
            .stream() Stream<Map<K, V>.Entry<Character, Long>>
            .sorted(Map.Entry.comparingByValue(Comparator.reverseOrder()))
            .forEachOrdered(value -> result.put(value.getKey(), value.getValue()));

    return result;
}
```

Następnie przypisujemy odpowiednio występującym wartościom ich odpowiedniki binarne.

```
public static Map<Character, String> convertToBinar(Map<Character, Long> mapa){
    Map<Character, String> wartoscBinarna = new LinkedHashMap<>();
    int i = mapa.size()-1;
    String bin = "1";
    for (Character wystapienie: mapa.keySet()){
        if(i == 0){
            wartoscBinarna.put(wystapienie, "0");
        }
        else {
            wartoscBinarna.put(wystapienie, bin.repeat(i) + "0");
        }
        i--;
    }
    return wartoscBinarna;
}
```

Teraz możemy zaszyfrować wiadomość poprzez porównanie znaku w wiadomości i zamianie na jej odpowiednik binarny przechowywany w mapie. Aby tego dokonać posłużyłem się funkcją z biblioteki String – replace.

```java
public static String encryptMessage(Map<Character, String> wartoscBinarna, String wiadomosc){
    for (Map.Entry<Character, String> szyfr : wartoscBinarna.entrySet()){
        wiadomosc = wiadomosc.replace(szyfr.getKey().toString(), szyfr.getValue());
    }
    return wiadomosc;
}
```

Tą samą mapę i funkcję wykorzystaliśmy do odszyfrowania. Funkcja replace działa poprawnie ponieważ porównuje ona zarówno wartość znaków jaki ich długość, aby móc dokonać zamiany.

```java
private static String decryptMessage(Map<Character, String> wartoscBinarna, String encryptedMessage){
    for (Map.Entry<Character, String> szyfr : wartoscBinarna.entrySet()){
        encryptedMessage = encryptedMessage.replace(szyfr.getValue(), szyfr.getKey().toString());
    }
    return encryptedMessage;
}
```

Efekt końcowy

```
Podaj ciag znakow: addwwwwwrrrttttttt
{t=7, w=5, r=3, d=2, a=1}
{t=11110, w=1110, r=110, d=10, a=0}
Zaszyfrowana wiadomosc: 0101011101110111011101110110110110111011110111011110111011101111011110111011110
Odszyfrowana wiadomosc: addwwwwwrrrttttttt
```