Akademia Nauk Stosowanych Teoretyczne i technologiczne podstawy multimediów – IS rok 3	
Imię i Nazwisko: Emilian Kochanek	**Grupa:** L2
Data: 18.10.2022	**Symbol:** TiTPM_03

Program po przekazaniu wiadomości do zaszyfrowania przez użytkownika, zlicza wystąpienia poszczególnych znaków i sortuje je w kolejności malejącej.

```
public static LinkedHashMap<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);
}

public static LinkedHashMap<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 dzieli na dwie równe połówki i ztych połówek wylicza kolejne podtablice przypisując odpowiednim znakom ich znak szyfrujący.

```
public static LinkedHashMap<Character, String> podzielNaRowne(LinkedHashMap<Character, Long> mapa) {
    long sumaPrzod = 0;
    long sumaTyl = 0;
    LinkedHashMap<Long, String> resultMap = new LinkedHashMap<>();
    List<Long> mapValues = mapa.values().stream().collect(Collectors.toList());
    int listSize = mapValues.size()-1;
    for (int i = 0; i < mapValues.size(); i++){
        if(sumaPrzod <= sumaTyl) {
            sumaPrzod += mapValues.get(i);
        }
        else {
            sumaTyl += mapValues.get(listSize);
            listSize--;
            i--;
        }
        if(listSize == i){
            break;
        }
    }
    resultMap.put(sumaPrzod, "0");
    resultMap.put(sumaTyl, "1");
    return podzielNaPodTablice(resultMap, mapa);
}
```

```java
public static LinkedHashMap<Character, String> podzielNaPodTablice(LinkedHashMap<Long, String> mapBin, LinkedHashMap<Character, Long> mapa){
    LinkedHashMap<Character, String> resultMap = new LinkedHashMap<>();
    boolean isFirst = true;
    int sum = 0;
    int inter = 0;
    for(Map.Entry<Character, Long> iterator : mapa.entrySet()){
        Map.Entry<Long, String> bin = mapBin.entrySet().stream().filter(value -> value.getKey() > 0).findFirst().get();
        if(bin.getKey() - sum <= 0){
            mapBin.remove(bin.getKey());
            sum = 0;
            isFirst = true;
            bin = mapBin.entrySet().stream().filter(value -> value.getKey() > 0).findFirst().get();
            inter = 0;
        }else if(inter == 2){
            resultMap.put(iterator.getKey(), bin.getValue()+1);
            mapBin.put(bin.getKey()-sum, bin.getValue()+1);
            mapBin.remove(bin.getKey());
            isFirst = true;
            sum = 0;
            inter = 0;
        }
        if (isFirst && bin.getKey() - sum > 0){
            sum += sum + iterator.getValue();
            resultMap.put(iterator.getKey(), bin.getValue() + 0);
            isFirst = false;
        } else{
            resultMap.put(iterator.getKey(), bin.getValue()+1);
            sum += iterator.getValue();
            if(bin.getKey() - sum > 0){
                resultMap.replace(iterator.getKey(), resultMap.get(iterator.getKey()) + 0);
                mapBin.replace(bin.getKey(), bin.getValue()+1);
            }
            mapBin.replace(bin.getKey(), bin.getValue() + 1);
            isFirst = true;
        }
        inter++;
    }
    return resultMap;
}
```

Na koniec tak utworzoną mapę szyfrującą wykorzystuje do zaszyfrowania wiadomości, a następnie do jej odszyfrowania.

```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;
}

private static String decryptMessage(Map<Character, String> wartoscBinarna, String encryptedMessage){
    LinkedHashMap<Character, String> reverseOrder = new LinkedHashMap<>();
            wartoscBinarna.entrySet().stream()
                    .sorted(Map.Entry.comparingByValue(Comparator.reverseOrder()))
                    .forEachOrdered(values -> reverseOrder.put(values.getKey(), values.getValue()));
    for (Map.Entry<Character, String> szyfr : reverseOrder.entrySet()){
        encryptedMessage = encryptedMessage.replace(szyfr.getValue(), szyfr.getKey().toString());
    }
    return encryptedMessage;
}
```

```
Podaj ciag znakow: aaaaaaaaddddddddddddddddddddddddddddddssssrrrrrrrree
{d=28, a=8, r=8, s=5, e=2}
{d=00, a=10, r=110, s=1110, e=11111}
Zaszyfrowana wiadomosc: 10101010101010100000000000000000000000000000000000000000000000000000000000000011101110111011101101101101101101101101101111111111
Odszyfrowana wiadomosc: aaaaaaaaddddddddddddddddddddddddddddddssssrrrrrrrree
```