Państwowa Wyższa Szkoła Zawodowa w Nowym Sączu Teoretyczne i technologiczne podstawy multimediów	
Imię i Nazwisko: Norbert Lachner	**Grupa: L2**
Data: 18.10.2022	**Ocena:**

Kodowanie Shannona-Fano – metoda kompresji bezstratnej autorstwa Roberta Fano. Kodowanie to dla dyskretnego źródła danych znajduje kod prefiksowy, który charakteryzuje się dość dobrą efektywnością – lepszą od kodowania Shannona (słowa kodowe krótsze o 1 bit), nie tworzy jednak optymalnych kodów.

```java
package com.company;

import java.util.*;
import java.util.stream.Collectors;

public class Main {

    public static void main(String[] args) {
        Scanner cin = new Scanner(System.in);
        String ciagZnakow;
        System.out.print("Podaj ciag znakow: ");
        ciagZnakow = cin.nextLine();
        LinkedHashMap<Character, Long> characterLongMap = zliczWystapienia(ciagZnakow);
        System.out.println(characterLongMap);
        Map<Character, String> characterStringMap = convertToBinar(characterLongMap);
        System.out.println(characterStringMap);
        String encryptedMessage = encryptMessage(characterStringMap, ciagZnakow);
        System.out.println("Zaszyfrowana wiadomosc: " + encryptedMessage);
        System.out.println("Odszyfrowana wiadomosc: " + decryptMessage(characterStringMap, encryptedMessage));
        System.out.println(podzielNR(characterLongMap)) ;
    }

    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<>();
```

```java
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--;S
    }
    return
}
    public static LinkedHashMap<Long, Long> podzielNR(LinkedHashMap<Character, Long> mapa) {
        long sumaP = 0;
        long sumaT = 0;
      LinkedHashMap<Long, Long> 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(sumaP <= sumaT){
                sumaP += mapValues.get(i);
            }else if(listSize == i){
                break;
            }else {
                sumaT += mapValues.get(listSize);
                listSize--;
            }
        }
        resultMap.put(sumaP, 0L);
```

```java
            }
            resultMap.put(sumaP, 0L);
            resultMap.put(sumaT, 1L);
            return resultMap;
        }



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

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

Wynik działania kodu :

```
"C:\Program Files\Java\jdk-17.0.4.1\bin\java.exe" "-javaagent:
Podaj ciag znakow: qweqweqwewqedd
{q=4, e=4, w=4, d=2}
{q=1110, e=110, w=10, d=0}
Zaszyfrowana wiadomosc: 11101011011101011011101011010111011000
Odszyfrowana wiadomosc: qweqweqwewqedd
{4=0, 2=1}

Process finished with exit code 0
```