[GUI] Ćwiczenia V:

```
// Ważne znaczenie w zastosowaniu kolekcji mają metody
public boolean equals(Object ...)

// oraz
public int hashCode()

// z klasy Object
// Te metody należy przedefiniować (ang. override) jeśli ich pierwotna wersja (w klasie Object) nie działa w konkretnych sytuacjach
```

```
class K {

    // ...
    @Override
    public boolean equals(Object obj){

        if (this == obj) return true;
        if (obj == null || this.getClass() != obj.getClass())
            return false;

        K k = (K)obj;
        // porównywanie składowych obiektów: this, k
        // ...
        return ...;
    }
}
```

```
import java.util.*;

public class Maps {

    public static void main(String[] args) {

        // mapa = zbiór par (klucz, wartość)
        Map<String, Integer> telephoneBook = new HashMap<>();

        // para (nazwisko, numer tel)
        telephoneBook.put("Kowalski", 123);
        telephoneBook.put("Nowak", 456);
        telephoneBook.put("Nowak", 789);
        telephoneBook.put("Kowalska", 111);
        //telephoneBook.putIfAbsent("Kowalska", 111);

        // pobieranie wartości dla danego klucza
        System.out.println(telephoneBook.get("Kowalska"));          // 111
```

```
        // usuwanie pary dla danego klucza
        telephoneBook.remove("Kowalska");
        System.out.println(telephoneBook.get("Kowalska"));          // null

        int number = telephoneBook.get("Nowak");
        System.out.println(number);                                 // 789

        // czy mapa zawiera podany klucz?
        System.out.println(telephoneBook.containsKey("Kowalski"));  // true
        System.out.println(telephoneBook.containsKey("Kowal"));     // false

        // czy mapa zawiera podaną wartość?
        System.out.println(telephoneBook.containsValue(456));       // false
        System.out.println(telephoneBook.containsValue(789));       // true
        System.out.println(telephoneBook.containsValue(111));       // false

        // zbiór kluczy mapy
        System.out.println(telephoneBook.keySet());                 // [Kowalsk
i, Nowak]

        // zbiór wartości mapy
        System.out.println(telephoneBook.values());                 // [123, 78
9]

        // drukowanie wartości mapy
        for (String k : telephoneBook.keySet())
            System.out.print(telephoneBook.get(k) + " ");           // 123 789

        System.out.println();

        // drukowanie par (klucz, wartość) mapy: Kowalski - 123 Nowak - 789
        telephoneBook.forEach((key, value) -> System.out.print(key + " - " + val
ue + " "));
    }
}
```

```
/*
1. Statyczne metody klasy java.nio.file.Files zwracają strumień Stream<String>

lines(Path path) - Wszystkie wiersze pliku (z kodowaniem UTF-8)
lines(Path path, Charset cs) - Wszystkie wiersze pliku (z podanym kodowaniem)

2. String[] split(...) - rozbiór napisu na podnapisy

3. Przydatne metody z java.util.stream.*;

Stream sorted()
Collector Collectors.groupingBy(...) - dostarcza mapę (java.util.Map) grupując o
biekty
```

```
*/

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Arrays;
import java.util.function.Predicate;
import java.util.regex.Pattern;
import java.util.stream.Stream;

/*  Przykładowy plik:
Maria 12c 78
Janina 12c 90
Bartek 13c 68
Wiktor 12c 45
Kasia 12c 66
Janek 13c 66
*/

public class Program {

    public static void main(String[] args) throws IOException {

        String fname = "students.txt";

            // użycie strumieni I/O  plikowych
        BufferedReader br = new BufferedReader(new FileReader(fname));
        String wiersz;
        while ((wiersz = br.readLine()) != null) {
            System.out.println(wiersz);
        }

        System.out.println();

        // użycie Stream API
        Stream<String> wiersze;
        wiersze = Files.lines(Paths.get(fname), StandardCharsets.UTF_8);
        wiersze.forEach(System.out::println);

            // wiersze zaczynające się od znaku 'M'
//      wiersze.filter(s -> s.startsWith("M")).forEach(System.out::println);

            // jak wyżej, z klasą anonimową
//      wiersze.filter(new Predicate<String>() {
//          @Override
//          public boolean test(String s) {
//              return s.startsWith("M");
//          }
```

```
//      }).forEach(System.out::println);

        System.out.println();
        // rozbiór napisu
        wiersz = "Maria 12c 78";
        String[] slowa = wiersz.trim().split(" +");    // separator = 1 lub więc
ej spacji

        for (String s : slowa)
            System.out.println(s);                          // Maria 12c 78

        // lub
        //Stream.of(slowa).forEach(System.out::println);        // Maria 12c 78
        // lub
        //Arrays.stream(slowa).forEach(System.out::println);    // Maria 12c 78

        // jeszcze inaczej
        //Pattern.compile(" +").splitAsStream(wiersz).forEach(System.out::printl
n);
    }
}
```