Wizualizacja Grafów
Autor: Łukasz Maruszak
Język: Java + Swing

Program służy do graficznej wizualizacji grafu na podstawie pliku tekstowego. Program umożliwia użytkownikowi wybranie reprezentacji grafu na ekranie. Ma do wyboru cztery opcje:

- Graf Skierowany
- Graf Nieskierowany
- Etykiety Wierzchołków
- Etykiety Krawędzi

Program korzysta z mojej własnej implementacji Grafu, jest to obiektowa reprezentacja grafu. Wierzchołki są klasą, która w konstruktorze przyjmuje “id” wierzchołka czyli jego etykietę oraz Punkt w którym wierzchołek się znajduje. Krawędź to kolejna klasa przechowująca informacje o dwóch wierzchołkach, które łączy oraz etykietę krawędzi. Informacje o krawędziach i wierzchołkach w grafie przechowywane są w dwóch ArrayList.

Informacje o grafie odczytywane są z pliku tekstowego Graf.txt. Zaprezentuje przykładowy wygląd takiego pliku.

```
9
4
Kraków
Warszawa
Puławy
Gdańsk
1
2
3
4
5

Warszawa,Kraków,300km
Kraków,Gdańsk,550km
Warszawa,Puławy,120km
Puławy,Kraków,330km
1,3,9
3,2,10
5,4,blisko
```

Pierwsza liczba oznacza ilość wierzchołków. Kolejna to ilość krawędzi. Następnie użytkownik podaje kolejne nazwy wierzchołków, mogą to być liczby lub napisy. Musi ich być tyle ile wierzchołków, w przeciwnym razie wierzchołek będzie posiadał pustą etykietę. Po podaniu etykiet wierzchołków jest linia przerwy a następnie podaję się krawędzie miedzy wierzchołkami, użytkownik podaje nazwę wierzchołka początkowego, nazwę wierzchołka

końcowego oraz etykietę. Kolejność wierzchołków w grafie nieskierowanym nie ma znaczenia. Etykieta może być liczbą lub napisem.

- Interface użytkownika.

Interface składa się z dwóch pól. Pierwsze służy do wybierania opcji oraz przycisku “GO”, po kliknięciu którego w drugim polu zostaje narysowany graf odczytany z pliku tekstowego. Opcje można wybierać także po narysowaniu grafu, a zmiany zostają automatycznie naniesione na wizualizacje grafu. Program kończy działanie po zamknięciu okna.

- Wizualizacja grafu - analiza sposobu umiejscowienia wierzchołków.

Po podaniu ilości wierzchołków program oblicza odpowiednią ilość punktów punktów na planszy. Punkty te tworzą wielokąt foremny, dzięki temu udało mi się wyeliminować sytuacje w której krawędź przechodzi przez wierzchołek lub wierzchołki z którym nie jest połączona. Etykiety wierzchołków znajdują się po ich zewnętrznej stronie. Etykiety krawędzi znajdują się w ⅓ długości krawędzi. Kiedy użytkownik wybierzy Graf Skierowany groty pokazujące kierunek krawędzi zostają narysowane w ⅔ długości krawędzi.

- Analiza Kodu

  - Program zawiera funkcję obliczającą punkty rozmieszczenie w rogach wielokąta foremnego a następnie zostają one przeskalowane aby znajdowały się na środku okienka.

```
public static ArrayList<Point> obliczPunkty(int ilosc) {
        ArrayList<Point> punkty = new ArrayList<Point>();
        double n = (Math.PI * 2) / ilosc;
        for (int i = 0; i < ilosc; ++i) {
                int x = (int) (Math.cos(n * i) * 300) + 370;
                int y = (int) (Math.sin(n * i) * 300) + 350;
                Point p = new Point(x, y);
                punkty.add(p);
        }
        return punkty;
}

public static ArrayList<Point> obliczPunktyNapisu(int ilosc) {
        ArrayList<Point> punkty = new ArrayList<Point>();
        double n = (Math.PI * 2) / ilosc;
        for (int i = 0; i < ilosc; ++i) {
                int x = (int) (Math.cos(n * i) * 340) + 370;
                int y = (int) (Math.sin(n * i) * 340) + 360;
                Point p = new Point(x, y);
                punkty.add(p);
        }
        return punkty;
}
```

- Funkcja obliczająca punkty gdzie mają znajdować się etykiety działa w ten sam sposób, tylko obliczone punkty tworzą większy wielokąt aby etykiety nie zasłaniały wierzchołków oraz nie wchodziły na nie.

- Rysowanie wierzchołków polega na odczytaniu kolejnych współrzędnych oraz narysowaniu niebieskiego koła w zadanym punkcie. Korzystam z funkcji fillOval której podaje punkty gdzie ma się znajdować wierzchołek oraz wielkość koła.

```
// rysuje wierzcholki - punkty
public void draw(Graphics g) {

        for (int i = 0; i < node.size(); ++i) {
                g.setColor(new Color(9, 186, 249));
                g.fillOval(node.get(i).getPoint().x, node.get(i).getPoint().y, 25, 25);
                // System.out.println(node);
        }

}
```

- Wypisanie etykiet wierzchołków zaczynam od wywołania metody obliczającej współrzędne etykiet. Ustawiam odpowiednią czcionkę, rozmiar oraz kolor liter a następnie wywołują funkcje rysująca napis drawString. Przyjmuje ona napis, który ma zostać wypisany oraz współrzędne miejsca narysowania napisu.

```
// rysuje etykiety nad wierzcholkami
public void drawEtykietyWierzcholkow(Graphics g) {
        for (int j = 0; j < node.size(); ++j) {
                ArrayList<Point> punkty = new ArrayList<Point>();
                punkty = obliczPunktyNapisu(size);
                g.setFont(new Font("Arial", Font.BOLD, 13));
                g.setColor(Color.black);
                g.drawString(node.get(j).getNazwa(), punkty.get(j).x, punkty.get(j).y);
        }
}
```

- Połączenie wierzchołków czyli narysowanie linii, zaczynam od ustawienia koloru linii - czarny oraz grubości linii. Odcinek zostaje narysowany od za pomocą funkcji drawLine. Podaje współrzędne punktów które mają być połączone.

```
// rysuje krawedzie nieskierowane
public void drawEdge(Graphics e) {
        Graphics2D g = (Graphics2D) e;
        g.setColor(Color.black);
        g.setStroke(new BasicStroke(3.0f));

        for (int i = 0; i < edge.size(); ++i) {

                g.drawLine(edge.get(i).n1.getPoint().x + 12, edge.get(i).n1.getPoint().y + 12,
                                edge.get(i).n2.getPoint().x + 12, edge.get(i).n2.getPoint().y + 12);
        }
}
```

- Krawędzie skierowane zostają narysowane za pomocą tej samej funkcji drawLine, dodatkowo na rysunku pojawiają się groty strzałek pokazujące kierunek skierowania krawędzi.

```
// metoda do rysowanie strzalek na krawedziach
public void drawStrzalki(Graphics g, int x1, int y1, int x2, int y2) {
        Graphics2D g2 = (Graphics2D) g;
        g2.setColor(Color.black);
        g2.setStroke(new BasicStroke(3.0f));
        int ARR_SIZE = 9;

        double dx = x2 - x1, dy = y2 - y1;
        double angle = Math.atan2(dy, dx);
        int len = (int) Math.sqrt(dx * dx + dy * dy);
        // transform the draw
        AffineTransform at = AffineTransform.getTranslateInstance(x1, y1);
        at.concatenate(AffineTransform.getRotateInstance(angle));
        g2.transform(at);

        // Draw horizontal arrow starting in (0, 0)
        g2.drawLine(0, 0, len, 0);
        len = len + 75;
        g2.fillPolygon(new int[] { len / 2, len / 2 - ARR_SIZE, len / 2 - ARR_SIZE },
                       new int[] { 0, -ARR_SIZE, ARR_SIZE }, 3);

        try {
                g2.transform(at.createInverse());
        } catch (NoninvertibleTransformException ex) {
        }
}
```

- Podpisanie Krawędzi etykietą, polega na wyznaczeniu współrzędnych ⅔ krawędzi oraz wywołaniu funkcji drawString, w której podaje wyznaczone punkty oraz nazwę etykiety.

```
// rysowanie etykiet krawedzi
public void drawEtykietyKrawdzi(Graphics g) {
        g.setFont(new Font("Arial", Font.BOLD, 15));
        g.setColor(new Color(9, 96, 170));

        for (int i1 = 0; i1 < edge.size(); ++i1) {
                int x = (edge.get(i1).n2.p.x + 3 * edge.get(i1).n1.p.x) / 4;
                int y = (edge.get(i1).n2.p.y + 3 * edge.get(i1).n1.p.y) / 4;
                g.drawString(edge.get(i1).Etykietka(), x, y + 9);
        }
}
```

- Metoda szukaj(String text) służy do znalezienia wierzchołka w Liście wierzchołków i zwróceniu go. Dzięki niej określając połączenia, użytkownik może podać etykietę wierzchołka, która jest tekstem a nie tylko liczbą.

```
public static Node szukaj(String text) {

        for (int i = 0; i < nodeNazwy.size(); ++i) {
                if (nodeNazwy.get(i).id.equals(text)) {
                        return nodeNazwy.get(i);
                }
        }

        return null;

}
```

1. W klasie Zmienne, zainicjowane zostają zmienne globalne wykorzystywane do obsługi JCheckBox, które wykorzystywane są w innych klasach.

```
import javax.swing.JFrame;

public class Zmienne {
public static boolean skierowana = false;
public static boolean nieSkierowana = false;
public static boolean etykietyWierzcholkow = false;
public static boolean etykietyKrawedzi = false;
public static boolean go = false;

public static JFrame frame = new JFrame("Wizualizacja Grafów");;
}
```

2. Klasa Okno rozszerza klasę JFrame, inicjowane są tu elementy okna takie jak panel z opcjami do zaznaczania dzięki którym użytkownik może sterować wyglądem grafu, przycisk, pole na którym wyświetlana jest wizualizacja grafu. W tej klasie zainicjowana jest obsługa JCheckBox oraz JButton, po kliknięciu w guzik lub zaznaczeniu okna zostają zmienione na true odpowiednie zmienne.

3. Klasa Malowanko to klasa odpowiedzialna za namalowanie na ekranie grafu o zadanych parametrach przez użytkownika. Jeżeli któraś z opcji będzie miała wartość true to zostanie wywołana odpowiednia metoda rysująca krawędzie skierowane, nieskierowane lub etykiety nad wierzchołkami oraz krawędziami.

4. Klasa Main to główna klasa obsługująca program. W nieskończonej pętli while tworzony jest obiekt klasy Okno, następuje odczytanie grafu z pliku graf.txt, narysowane wierzchołki oraz odpowiednie krawędzie. Dzięki temu że znajduje się to w pętli nieskończonej, można dokonywać zmian w trakcie działania programu.

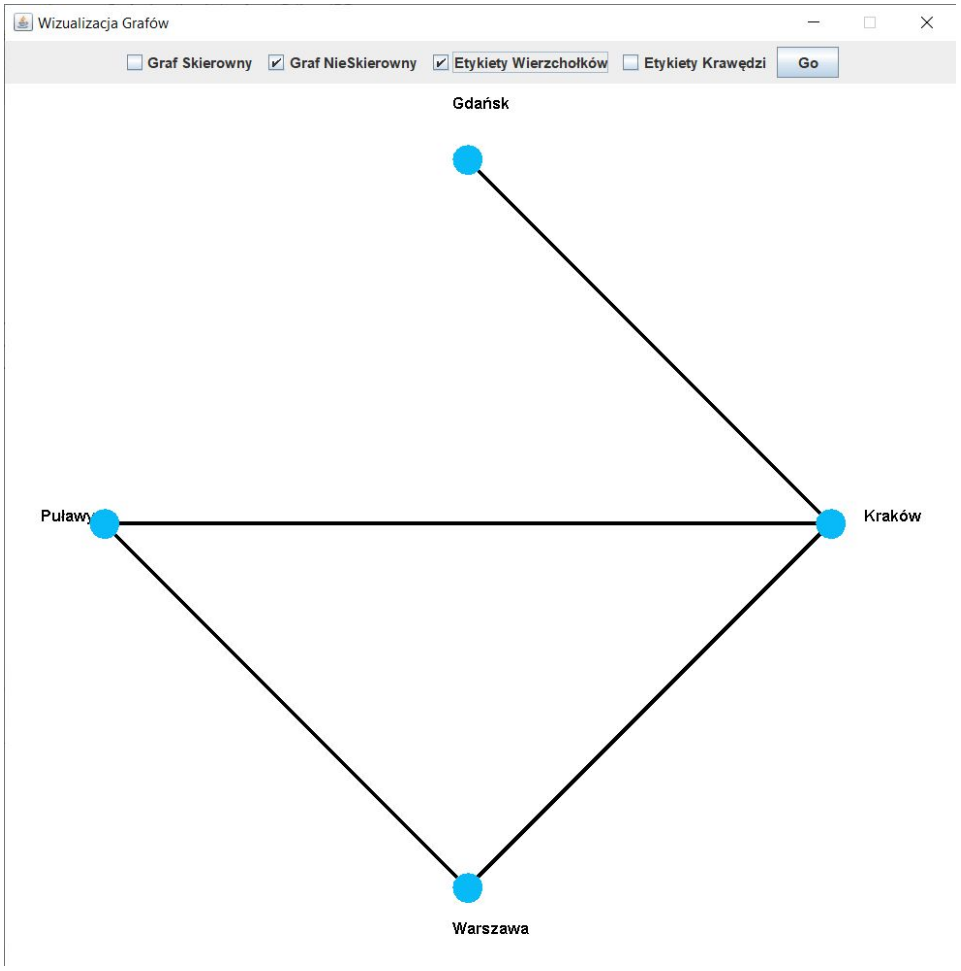
```
public class Main {

	public static void main(String[] args) throws FileNotFoundException {
		while (true) {

			Okno noweokno = new Okno();

		}
	}
}
```

- Interfejs użytkownika

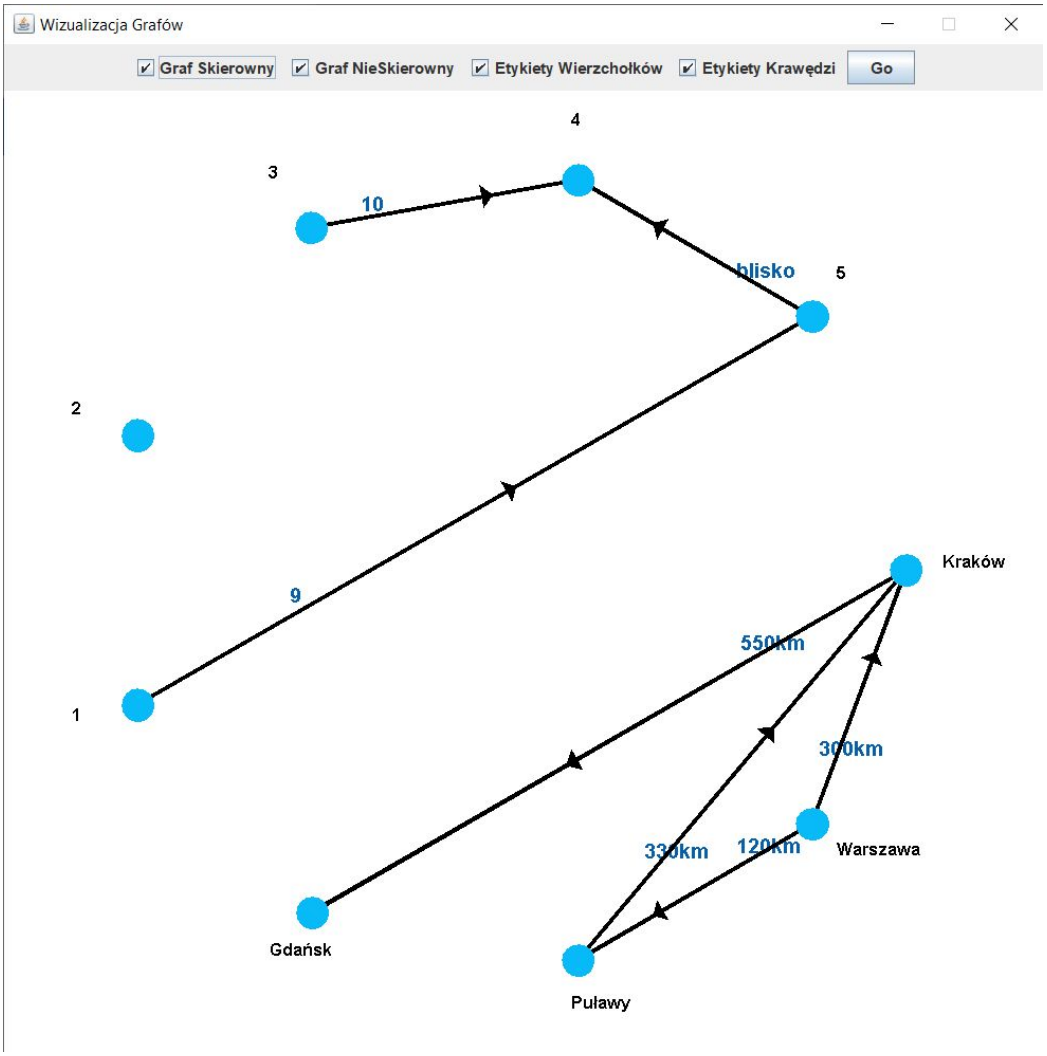
Po włączeniu programu użytkownikowi pokazuje się okno z pustym białym polem oraz panelem do wybierania ustawień. Po naciśnięciu przycisku “Go” zostaje narysowana wizualizacja Grafu.

Użytkownik może wybrać z czterech opcji klikając na kwadracik obok nazwy. Odznaczenie oznacza wyłączenie opcji.

Opcje, które użytkownik może wybierać.



W tym przykładzie mamy pokazane zastosowanie programu do wizualizacji Grafu przedstawiającego mapę połączeń między miastami. Krawędzie są nieskierwowane, ponieważ ruch może odbywać się w dwóch kierunkach. Etykiety wierzchołków zostały oznaczone nazwami miast.



Przykład numer dwa pokazuje trzy grafy niespójne grafy. Graf składający się z jednego wierzchołka numer dwa. Graf, w którym etykiety wierzchołków są liczbami a etykiety krawędzi są liczbami ale także mogą być napisem w tym przypadku “blisko”. Trzeci graf to ten sam graf z poprzedniego przykładu jednak teraz jest skierowany a etykiety krawędzi oznaczają odległości między wierzchołkami.