

Programowanie III

Dokumentacja projektu

***"Sudoku" – program do sprawdzania poprawności
rozwiązania Sudoku w oparciu o zadanie 5 „Sudoku” z
konkursu Algorytmion z 2011r.***

Wojciech Grzywocz gr. lab. 1/2

1. Opis projektu.

Program działa w oparciu o zadanie 5 pt. „Sudoku” z konkursu Algorytmion z 2011r. W skrócie polega ono na tym, iż otrzymujemy rozwiązany sudoku w pliku txt, w którym mamy 81 liczb zapisanych w 9 wierszach po 9 liczb w każdym, przy czym liczby są od siebie oddzielone spacjami. Naszym zadaniem jest stworzenie programu, który sprawdzi poprawność liczb w pliku i poprawność rozwiązania sudoku, a następnie zapisze w pliku z Sudoku informacje czy błąd został znaleziony czy nie. Podejmując się tego zadania przyjąłem również dodatkowe założenie, aby program zapisywał do pliku z Sudoku dokładną informację gdzie znajdują się błędy, aby ułatwić ich wyszukanie użytkownikowi, ponieważ dla niektórych użytkowników sama informacja o zaistnieniu błędu była by niewystarczająca.

Dokładne polecenie zadania z konkursu Algorytmion brzmi:

Adam przesyła Basi plik "dane5.txt" z rozwiązaniem gry sudoku. Plik ten zawiera dziewięć liczb oddzielonych spacjami, w każdym z dziewięciu wierszy (razem 81 liczb). Basia nie chce sama sprawdzać czy rozwiązanie jest poprawne. Prosi Cię o pomoc w napisaniu programu, który wykona pracę za nią i jeśli jest to poprawne rozwiązanie to w pliku "wynik5.txt" znajdzie się słowo „TAK”, a jeśli nie to słowo „NIE”.

2. Budowa i funkcjonalności.

Projekt znajduje się w folderze „PROJEKT_Grzywocz_Wojciech_inf1_2”.

Wewnątrz folderu mamy: plik z instrukcją „INSTRUKCJA.txt”, plik z dokumentacją projektu „Dokumentacja projektu Java.pdf”, plik z zadaniami konkursu algorytmion z 2011r. w formacie pdf, dwa pliki z danymi w formacie txt: „dane.txt” oraz „blednySudoku.txt”, które odpowiednio zawierają poprawnie i błędnie rozwiązany sudoku i służą jako przykład do zademonstrowania działania programu; plik

PROJEKT_Grzywocz_Wojciech_inf1_2.iml oraz podfoldery: .idea, out oraz src, który to zawiera 6 plików z kodem programu napisanym w całości w

Javie(JDK14) za pomocą edytora IntelliJ IDEA 2024, te 6 plików z kodem to: Main.java, OdczytZpliku.java, SprawdzanieMalychKwadratow.java, SprawdzenieDuzegoKwadratu.java, ZapisDoPliku.java i GUI.java.

Podstawowym zadaniem programu, które wyznaczała treść polecenie zadania z Algorytmionu było to, aby program sprawdzał czy sudoku jest rozwiązany poprawnie a następnie zapisywał w tym pliku txt z Sudoku wynik tego sprawdzenia. Ja postanowiłem rozszerzyć trochę podejście do tego zadania i dodałem dodatkowe funkcjonalności, które nie były z góry określone przez treść zadania. Postanowiłem dodać dokładny opis błędu – program sprawdza wszystkie możliwości i zwraca użytkownikowi gdzie popełnił błąd oraz dodałem interfejs graficzny w formie formularza Javy, który wygodnie pozwala użytkownikowi programu wprowadzić nazwę pliku w którym użytkownik ma zapisany plik ze swoim rozwiązaniem Sudoku do sprawdzenia.

3. Przebieg realizacji.

Kod jest podzielony na 6 plików .java: Main, GUI, OdczytZpliku, SprawdzenieDuzegoKwadratu, SprawdzanieMalychKwadratow, ZapisDoPliku.

Wykorzystane biblioteki: java.util, java.swing, java.awt, java.awt.event, java.io, java.nio.charset.StandardCharsets

Plik Main:

```
Main.java x GUI.java OdczytZpliku.java SprawdzenieDuzegoKwadratu.java SprawdzanieMalychKwadratow.java
1 import java.util.*;
2
3 public class Main {
4     public static void main(String[] args) {
5         try {
6             GUI gui = new GUI();
7             while (gui.getPlik() == null) {
8                 try {
9                     Thread.sleep(1000); // Czekamy, aż użytkownik poda nazwę pliku
10                } catch (InterruptedException e) {
11                    e.printStackTrace();
12                }
13            }
14            String plik = gui.getPlik();
15
16            int[] liczby = new int[81];
17            OdczytZpliku odczyt = new OdczytZpliku();
18            liczby = odczyt.odczytajZpliku(plik);
19            if(liczby==null){
20                System.out.println("Zatrzymano działanie programu bo wystąpił błąd z danymi wejściowymi.");
21                throw new Exception();
22            }
23
24            ArrayList<ArrayList<String>> listaBledowSudoku = new ArrayList<>();
25            ArrayList<String> komunikatyObledzie = new ArrayList<>();
```

Na początku funkcja main tworzy Ramkę gui dzięki której można podać nazwę pliku. Ramka wyświetla się tak długo aż użytkownik nie poda nazwy pliku, co gwarantuje pętla while. Następnie przypisujemy pobraną nazwę pliku do zmiennej plik dzięki niej korzystamy z klasy do odczytu danych. Jeśli dane są w niepoprawnym formacie program rzuca wyjątek i kończy działanie programu z informacją o błędzie. (Dane są zapisane do tablicy 81-elementowej, indeksacja zgodnie z zapisem książkowym – od lewej do prawej)

```

public class Main {
    public static void main(String[] args) {
        throw new Exception();
    }

    ArrayList<ArrayList<String>> listaBledowSudoku = new ArrayList<>();
    ArrayList<String> komunikatyObledzie = new ArrayList<>();

    SprawdzenieDuzegoKwadratu sprawdzanieDuzego = new SprawdzenieDuzegoKwadratu();
    komunikatyObledzie=sprawdzanieDuzego.sprawdzWiersze(liczby);
    if(komunikatyObledzie != null) {
        listaBledowSudoku.add(komunikatyObledzie);
    }
    komunikatyObledzie=sprawdzanieDuzego.sprawdzKolumny(liczby);
    if(komunikatyObledzie !=null) {
        listaBledowSudoku.add(komunikatyObledzie);
    }

    SprawdzanieMalychKwadratow sprawdzanieMalych = new SprawdzanieMalychKwadratow();
    for (int i = 1; i <= 9; i++) {
        komunikatyObledzie=sprawdzanieMalych.sprawdzMaly(i, liczby);
        if(komunikatyObledzie != null) {
            listaBledowSudoku.add(komunikatyObledzie);
        }
    }
}

```

Następnie sprawdzamy poprawność danych w sudoku według kolejno: dużych wierszy, dużych kolumn i poprawność w małych dziewięciu kwadratach. Każda z metod sprawdzających poprawność zwraca `ArrayList<String>` natomiast w klasie main do zbierania wszystkich błędów razem mamy `ArrayList<ArrayList<String>>`.

```

        if (listaBledowSudoku.isEmpty()) {
            ArrayList<String> wewnetrzna = new ArrayList<>();
            wewnetrzna.add("Gratulacje rozwiązano bezbłędnie.");
            listaBledowSudoku.add(wewnetrzna);
        }
        ZapisDoPliku zapis = new ZapisDoPliku();
        zapis.zapisDoPliku(plik, listaBledowSudoku);

        System.out.println("Wykonano wszystkie operacje");
    }
    catch (Exception e) {
        System.out.println(e.getMessage());
        System.exit( status: 1);
    }
}
}

```

Jeśli błędy nie występują dodajemy komunikat że rozwiązano bezbłędnie i zapisujemy komunikaty do pliku wykorzystując klasę do zapisu. Poniżej tego znajdują się również obsługa wyjątków klasy Main.

Plik GUI:

```

Main.java  GUI.java x  OdczytZPliku.java  SprawdzenieDuzegoKwadratu.java  SprawdzanieMalychKwadratu
1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.*;
4
5  class GUI extends JFrame { 2 usages
6      private JTextField poleTekstowe; 4 usages
7      public String plik; 3 usages
8
9      // Tworzenie okna aplikacji
10     public GUI() { 1 usage
11         setTitle("Podawanie pliku");
12         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13         setSize( width: 500, height: 500);
14         setLocationRelativeTo(null);
15
16         JPanel panel = new JPanel();
17         panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
18         panel.setBorder(BorderFactory.createEmptyBorder( top: 150, left: 50, bottom: 150, right: 50));
19
20         JLabel tekst = new JLabel( text: "Podaj nazwe pliku z Sudoku (np. dane.txt): ");
21         tekst.setAlignmentX(Component.CENTER_ALIGNMENT);
22         poleTekstowe = new JTextField( columns: 15);
23         poleTekstowe.setMaximumSize(new Dimension( width: 200, height: 30));
24         JButton przycisk = new JButton( text: "Zatwierdz");
25         przycisk.setAlignmentX(Component.CENTER_ALIGNMENT);
26
27         panel.add(tekst);
28         panel.add(Box.createVerticalStrut( height: 10));
29         panel.add(poleTekstowe);

```

Utworzona została ramka a w niej panel z tekstem, polem do wpisywania tekstu oraz przyciskiem.

```

Main.java  GUI.java x  OdczytZPliku.java  SprawdzenieDuzegoKwadratu.java  SprawdzanieMalychKwadratu.java  ZapisDoPliku.java
class GUI extends JFrame { 1 usage
    public GUI() { 1 usage
        panel.add(tekst);
        panel.add(Box.createVerticalStrut( height: 10));
        panel.add(poleTekstowe);
        panel.add(Box.createVerticalStrut( height: 20));
        panel.add(przycisk);

        // Dajemy kliknięciu przycisku
        przycisk.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                plik = poleTekstowe.getText().trim();
                if (plik.isEmpty()) {
                    dispose(); // Zamknij okno po zatwierdzeniu
                } else {
                    JOptionPane.showMessageDialog( getWindow(), message: "Wprowadziłeś poprawną nazwę pliku!", title: "Błęd", JOptionPane.ERROR_MESSAGE);
                    try {
                        Thread.sleep( time: 3000);
                    } catch (InterruptedException blad) {
                        blad.printStackTrace();
                    }
                }
            }
        });

        add(panel);
        setVisible(true);
    }
}

```

Dodanie akcji do przycisku, czyli pobranie wpisanej nazwy pliku lub zgłoszenie błędu.

```
    }  
});  
  
add(panel);  
setVisible(true);  
}  
  
public String getPlik() { 2 usages  
    return plik;  
}  
}
```

Metoda w klasie GUI która pozwala na przekazanie nazwy pliku do Main.

Plik OdczytZpliku:

```
Main.java  GUI.java  OdczytZpliku.java  SprawdzanieDuzegoKwadratu.java  SprawdzanieMalychKwadratow.java  ZapisD  
1  import java.io.BufferedReader;  
2  import java.io.FileNotFoundException;  
3  import java.io.FileReader;  
4  import java.nio.charset.StandardCharsets;  
5  import java.util.Arrays;  
6  
7  class OdczytZpliku { 2 usages  
8  OdczytZpliku() {} 1 usage  
9  public int[] odczytajZpliku(String nazwa_pliku){ 1 usage  
10     try(BufferedReader input = new BufferedReader(new FileReader(nazwa_pliku, StandardCharsets.UTF_8))){  
11         int[] liczby=new int[81];  
12         int indeks=0;  
13         int i=0;  
14         String linia;  
15         while ((linia = input.readLine()) != null) {  
16             linia=linia.trim();  
17             if(linia.isEmpty()){  
18                 continue;  
19             }  
20             String[] liczbyWlini=linia.split(" ");  
21  
22             if(liczbyWlini.length != 9){  
23                 System.out.println("Bład: Nie ma 9 cyfr w kazdej linii");  
24                 return null;  
25             }  
26             indeks += liczbyWlini.length;  
27             if (indeks > 81){  
28                 System.out.println("Bład: w pliku jest za duzo (ponad 81) liczb");  
29                 return null;  
30             }  
31         }  
32     }  
33 }
```


Wczytujemy liczby z pliku, zapisujemy je początkowo jako tablicę Stringów, sprawdzamy czy jest 9 cyfr w każdej linii, następnie sprawdzamy czy nie przekroczyliśmy 81 liczb, stosujemy w tym celu dodatkową zmienną indeks aby wiedzieć w którym miejscu w liczbach z Sudoku się znajdujemy.

```
class OdczytZpliku { 2 usages
    public int[] odczytajZpliku(String nazwa_pliku){ 1 usage
    }
    i=indeks-liczbyWlini.length;
    for(String l: liczbyWlini) {
        liczby[i] = Integer.parseInt(l);
        if(liczby[i]<1 || liczby[i]>9){
            System.out.println("Błąd: Podano liczbę która nie mieści się w zakresie 1-9.");
            return null;
        }
        i++;
    }
    }
    input.close();
    if(i==81) {
        return liczby;
    }
    else{
        System.out.println("W pliku jest mniej niż 81 liczb.");
        return null;
    }
    }
    catch(FileNotFoundException e1){
        System.out.println("Plik nie został znaleziony.");
        return null;
    }
    catch(Exception e){
        System.out.println(e.getMessage());
        return null;
    }
}
```

Następnie parsujemy liczby z formatu String na int i jeśli mamy 81 liczb zwracamy je do Maina, a jeśli nie to zwracamy null, który następnie w Mainie wywoła wyjątek, który zakończy program z komunikatem o błędzie.

Plik SprawdzenieDuzegoKwadratu:

Klasa SprawdzenieDuzegoKwadratu zawiera dwie metody, jedną która szuka błędów w każdym dużym wierszu, a druga metoda szuka błędów w każdej dużej kolumnie. Obie metody zwracają listę z błędami w postaci ArrayList<String> na której każdy string to jeden komunikat o błędzie w danym wierszu lub kolumnie.

```

1  import java.util.ArrayList;
2  import java.util.Arrays;
3
4  public class SprawdzenieDuzegoKwadratu { 2 usages
5      SprawdzenieDuzegoKwadratu(){} 1 usage
6      public ArrayList<String> sprawdzWiersze(int[] liczby){ 1 usage
7          int[] sprawdzane=new int[9];
8          int[] checkList=new int[9];
9          ArrayList<String>listaBledow=new ArrayList<>();
10         int licznik=0;
11
12         for(int i=0;i<9;i++){
13             //Wczytanie jednego wiersza
14             int indeks = 0;
15             for (int j = licznik; j < licznik + 9; j++) {
16                 sprawdzane[indeks] = liczby[j];
17                 indeks++;
18             }
19             licznik += 9;
20
21             //Sprawdzanie danego wiersza:
22             Arrays.fill(checkList, val: 0);
23             for(int l: sprawdzane) {
24                 checkList[l-1]++;
25             }
26             int[] arr={1,1,1,1,1,1,1,1,1};
27             if (Arrays.equals(checkList,arr)==false) {
28                 listaBledow.add("Błąd w "+(i+1)+" duzym wierszu sudoku");
29             }

```

Sprawdzenie poprawności danych opiera się na porównaniu dwóch tablic 9 elementowych, w jednej początkowo mamy same zera, a w drugiej, kontrolnej, same jedynki. Jak nie trudno zauważyć, wystarczy liczby z danego wiersza/kolumny sudoku pomniejszyć o 1 aby ich wartości odpowiadały kolejnym indeksom tablicy wypełnionej zerami. Dlatego też możemy powiększać wartości na określonych przez liczby z Sudoku indeksach w tablicy z zerami. Jeśli w danym wierszu lub danej kolumnie jakaś wartość się powtarza lub brakuje jakiejś wartości to na którejś pozycji w tablicy do sprawdzania będziemy mieli wartość różną od jeden i wtedy tablice sprawdzające nie będą sobie równe. Jeśli zaś w danym wierszu i kolumnie każda z liczb występuje dokładnie raz to 9-elementowa tablica, z tablicy zer, zamieni się na tablicę samych jedynek i będzie równa drugiej tablicy sprawdzającej co będzie świadczyć o poprawności danych w tym wierszu lub kolumnie.

```

Main.java  GUI.java  OdczytZpliku.java  SprawzenieDuzegoKwadratu.java  Spraw
4      public class SprawzenieDuzegoKwadratu { 2 usages
6          public ArrayList<String> sprawdzWiersze(int[] liczby){ 1 usage
18              }
19              licznik += 9;
20
21              //Sprawdzanie danego wiersza:
22              Arrays.fill(checkList, val: 0);
23              for(int l: sprawdzane) {
24                  checkList[l-1]+=1;
25              }
26              int[] arr={1,1,1,1,1,1,1,1,1};
27              if (Arrays.equals(checkList,arr)==false) {
28                  listaBledow.add("Bład w "+(i+1)+" duzym wierszu sudoku");
29              }
30          }
31          if(listaBledow.size()!=0) {
32              return listaBledow;
33          }
34          else {
35              return null;
36          }
37      }
38
39      public ArrayList<String> sprawdzKolumny(int[] liczby){ 1 usage
40          int[] sprawdzane=new int[9];
41          int[] checkList=new int[9];
42          ArrayList<String>listaBledow=new ArrayList<>();
43          int licznik=0;
44

```

```

Main.java  GUI.java  OdczytZpliku.java  SprawzenieDuzegoKwadratu.java  Sp
4      public class SprawzenieDuzegoKwadratu { 2 usages
59      public ArrayList<String> sprawdzKolumny(int[] liczby){ 1 usage
62
63          ArrayList<String> listaBledow=new ArrayList<>();
64          int licznik=0;
65
66          for(int i=0;i<9;i++) {
67              //Wczytanie jednej kolumny
68              int indeks = 0;
69              for (int j = licznik; j <= (licznik + 8*9); j+=9) {
70                  sprawdzane[indeks] = liczby[j];
71                  indeks++;
72              }
73              licznik += 1;
74
75              //Sprawdzanie danej kolumny:
76              Arrays.fill(checkList, val: 0);
77              for(int l: sprawdzane) {
78                  checkList[l-1]+=1;
79              }
80              int[] arr={1,1,1,1,1,1,1,1,1};
81              if (Arrays.equals(checkList,arr)==false) {
82                  listaBledow.add("Bład w "+(i+1)+" duzej kolumnie sudoku");
83              }
84          }
85          if(listaBledow.size()!=0) {
86              return listaBledow;
87          }
88          else {
89              return null;
90          }
91      }
92  }

```

Powyżej analogiczne postępowanie w metodzie dla dużych kolumn.

Plik SprawdzanieMalychKwadratow:

```
import java.util.ArrayList;
import java.util.Arrays;

public class SprawdzanieMalychKwadratow { 2 usages
    int malyKw; 12 usages
    SprawdzanieMalychKwadratow(){ 1 usage
        this.malyKw=0;
    }
    public ArrayList<String> sprawdzMaly(int num, int[] liczby){ 1 usage
        this.malyKw=num;
        int[] sprawdzane=new int[9];
        int[] checkList=new int[9];
        ArrayList<String>listaBledow=new ArrayList<>();
        int licznik=0;
        int indeks=0;
        int[] sprawdzajaca=new int[9];

        switch (malyKw){
            case 1:
                //Wczytanie pierwszego kwadratu
```

```

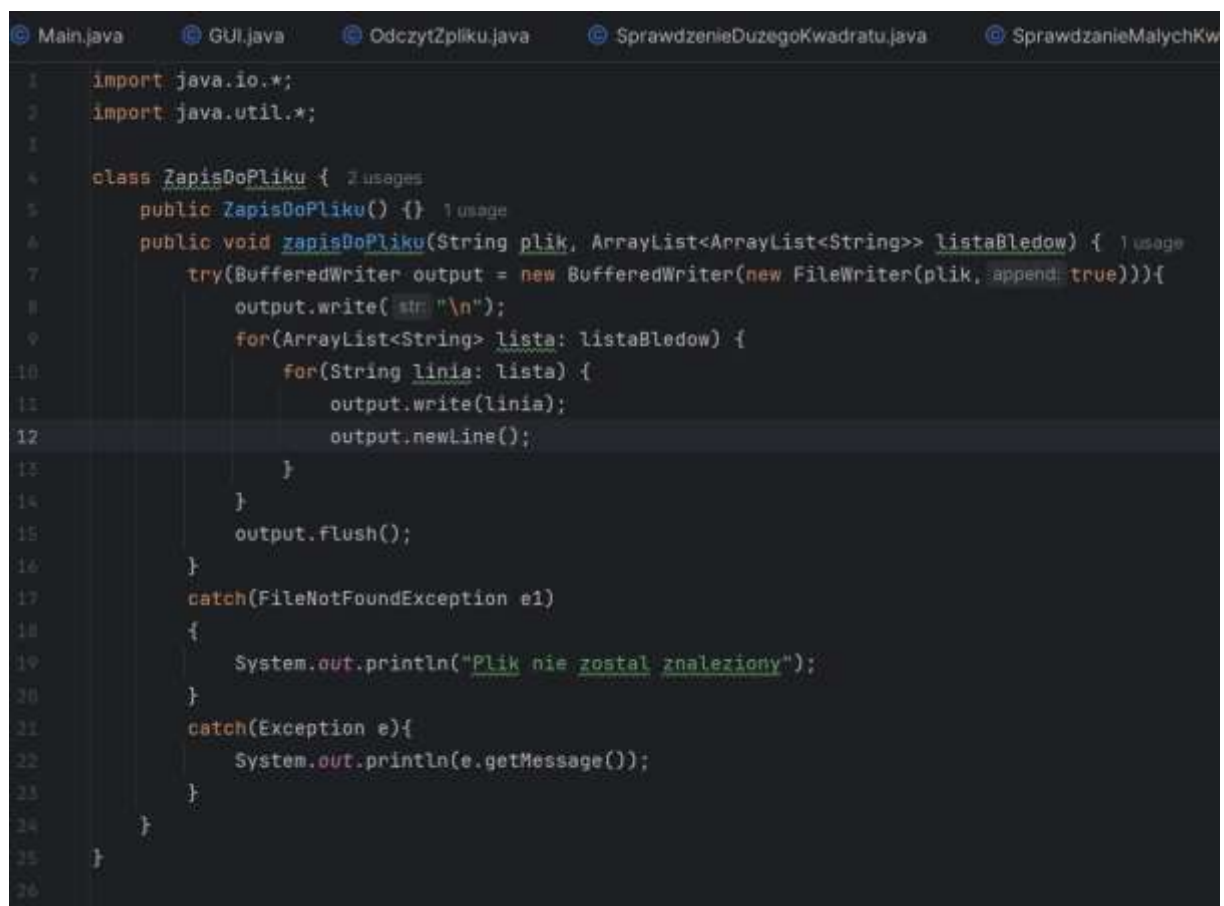
public class SprawdzanieMalychKwadratow { 2 usages
    public ArrayList<String> sprawdzMaly(int num, int[] liczby){ 1 usage
        switch (malyKw){
            case 1:
                //Wczytanie pierwszego kwadratu
                licznik=0;
                indeks = 0;
                for (int j = licznik; j <= licznik+2; j++) {
                    sprawdzane[indeks] = liczby[j];
                    indeks++;
                }
                licznik+=9;
                for (int j =licznik ; j <= licznik+2; j++) {
                    sprawdzane[indeks] = liczby[j];
                    indeks++;
                }
                licznik+=9;
                for (int j =licznik ; j <= licznik+2; j++) {
                    sprawdzane[indeks] = liczby[j];
                    indeks++;
                }

                //Sprawdzanie danego kwadratu:
                Arrays.fill(checkList, val: 0);
                for(int l: sprawdzane) {
                    checkList[l-1]++;
                }
                Arrays.fill(sprawdzajaca, val: 1);
                if (Arrays.equals(checkList,sprawdzajaca)==false) {

```

Klasa *SprawdzanieMalychKwadratow* posiada jedną metodę „*sprawdzMaly*”, która w zależności od podanego argumentu sprawdza jeden z 9 małych kwadratów. Sposób działania jest analogiczny do wcześniej przedstawionego sprawdzania wierszy i kolumn. De facto większa część tej metody to instrukcja *switch case*, która w zależności od podanego parametru bada odpowiedni mały kwadrat. Kod w odpowiednich *case*’ach różni się indeksami początkowymi, co pozwala sprawnie dobrać określony kwadrat do zbadania. Zastosowanie *Switch case* pozwoliło zamknąć sprawdzenie wszystkich 9 małych kwadratów w jednej metodzie z parametrem. W *mainie* zaś metoda „*sprawdzMaly*” jest wywoływana w pętli *for* z indeksami od 1 do 9 aby sprawdzić wszystkie małe kwadraty.

Plik ZapisDoPliku:



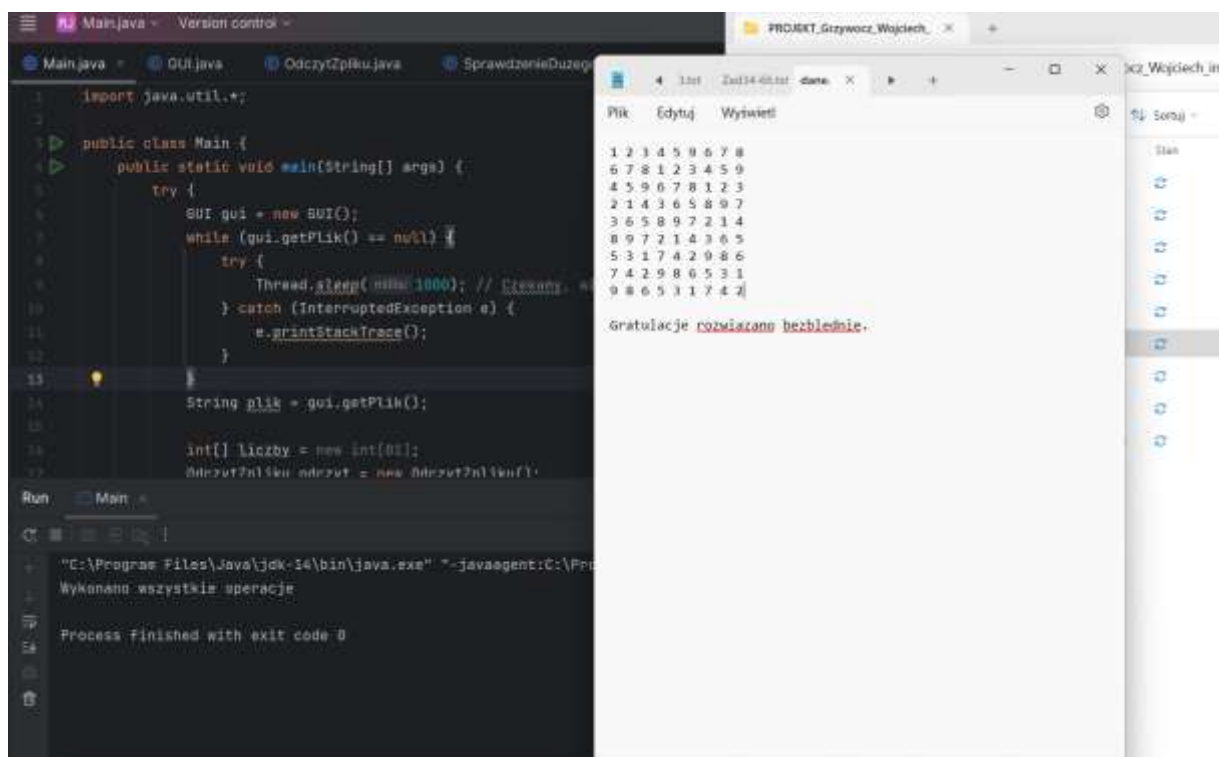
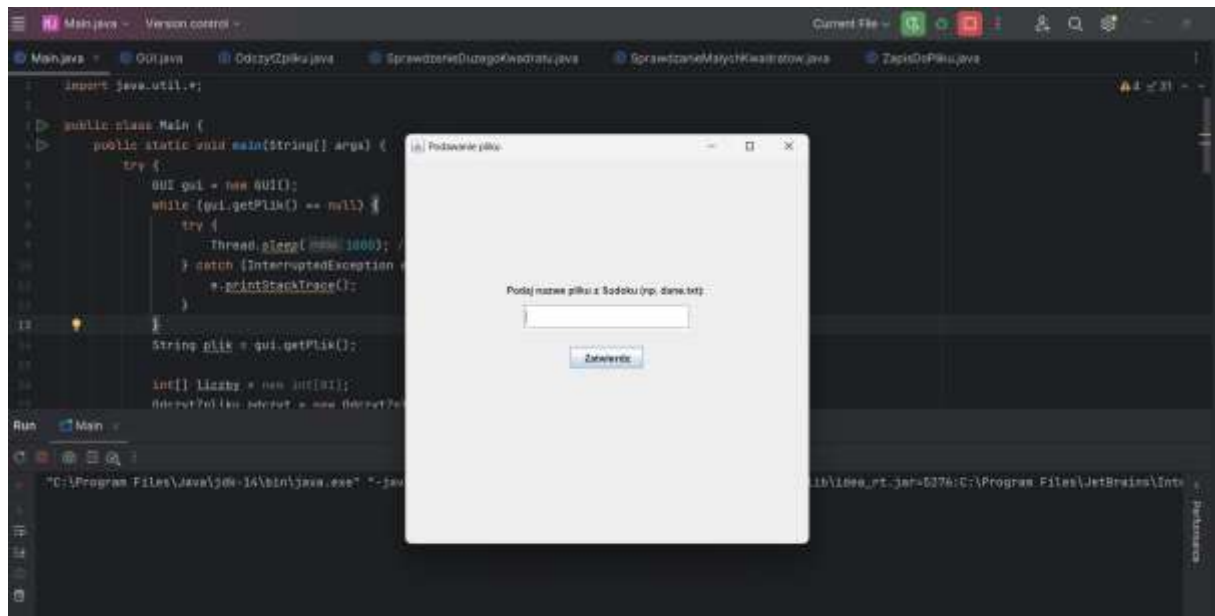
```

1  import java.io.*;
2  import java.util.*;
3
4  class ZapisDoPliku { 1usage
5      public ZapisDoPliku() {} 1usage
6      public void zapisDoPliku(String plik, ArrayList<ArrayList<String>> listaBledow) { 1usage
7          try(BufferedWriter output = new BufferedWriter(new FileWriter(plik, append: true))){
8              output.write( str "\n");
9              for(ArrayList<String> lista: listaBledow) {
10                 for(String linia: lista) {
11                     output.write(linia);
12                     output.newLine();
13                 }
14             }
15             output.flush();
16         }
17         catch(FileNotFoundException e1)
18         {
19             System.out.println("Plik nie został znaleziony");
20         }
21         catch(Exception e){
22             System.out.println(e.getMessage());
23         }
24     }
25 }
26

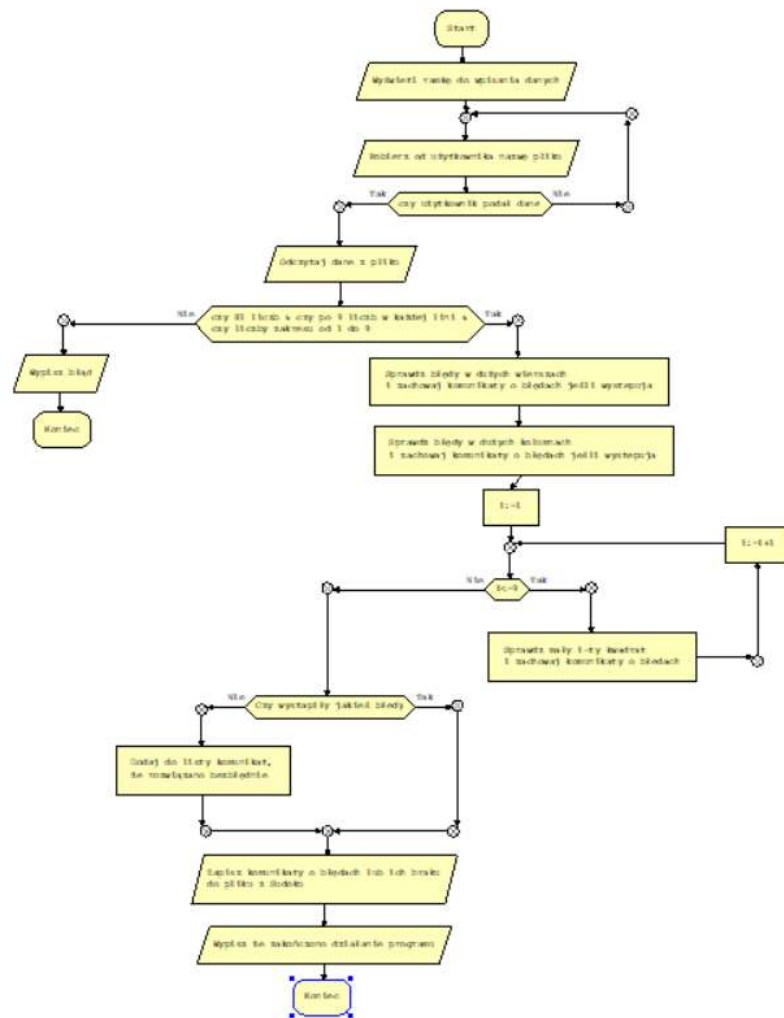
```

Klasa ta pozwala zapisać kolejne komunikaty o błędzie w formacie String do kolejnych linii w pliku z Sudoku.

Przykładowe wywołanie programu:



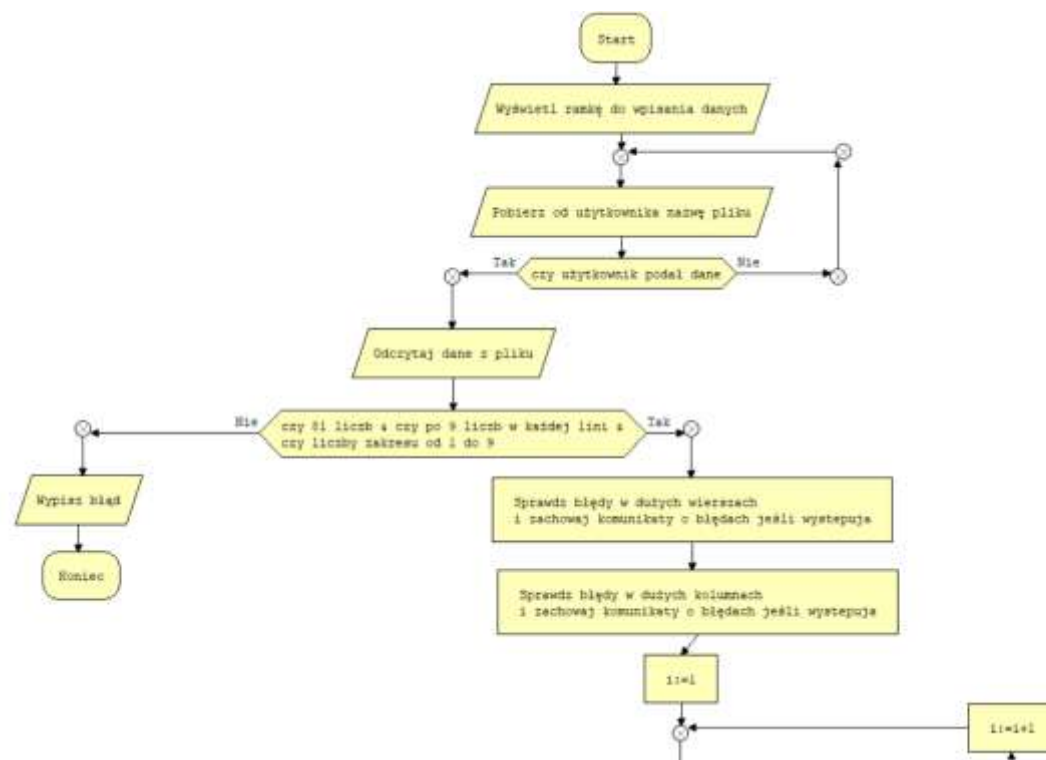
Schemat blokowy:



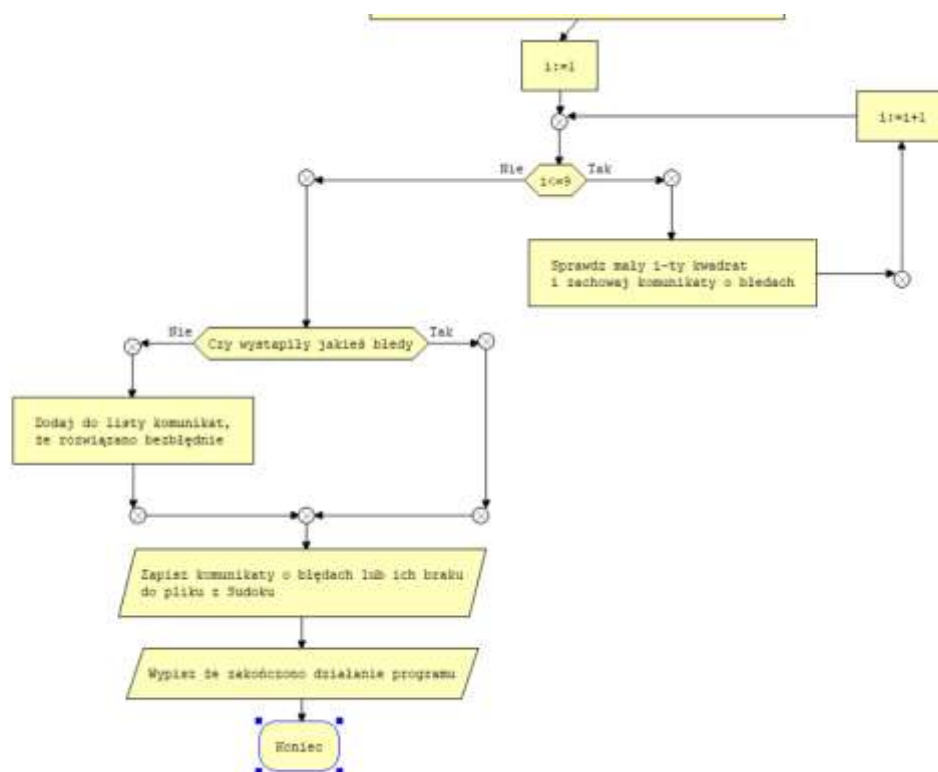
Cały schemat

W powiększeniu:

Cz.1



Cz.2



4. Instrukcja użytkownika.

Program służy do sprawdzania poprawności rozwiązania klasycznego Sudoku złożonego z 81 pól.

Po włączeniu programu użytkownik zostaje zapytany o plik txt w którym ma rozwiązany Sudoku, który zamierza sprawdzić. Po podaniu nazwy pliku txt program analizuje rozwiązanie i w tym samym pliku txt pod rozwiązaniem Sudoku zapisuje czy w rozwiązaniu pojawiły się błędy czy nie. Jeśli błędy się pojawiły program wypisuje komunikaty, dzięki którym użytkownik może zlokalizować swoje błędy. Przy czym, ze względu na dokładną analizę przeprowadzaną przez program, możliwe jest że kilka komunikatów będzie się odnosić do jednego błędu, z tego też względu wymaga się od użytkownika aby znał zasady Sudoku i posiadał zdolność analizy wyświetlanych komunikatów.

Program sprawdza za równo błędy w dużych wiersza, kolumnach jak i w małych kwadratach, wobec czego wprawionemu użytkownikowi łatwo zauważyć iż wystarczy przeanalizować komunikaty o błędach w dużych wierszach i dużych kolumnach oraz zliczyć ich ilość aby znaleźć dokładną pozycję występowania błędów.

Aby dodać plik txt z rozwiązaniem sudoku do sprawdzenia należy wkleić go do folderu projektu, a format zapisanych liczb w pliku txt musi być taki jak w pliku dane.txt, czyli powinno się w nim znajdować 81 liczb zapisanych w 9 wierszach po 9 liczb przy czym liczby są od siebie oddzielone spacjami.

5. Podsumowanie i wnioski.

Projekt pozwolił mi zwiększyć wiedzę na temat Javy i utrwalić już zdobytą wiedzę podczas zajęć na uczelni dając przekrój przez wszystkie tematy poruszane na zajęciach. Pomimo iż składnie Javy w niektórych aspektach można uznać za trudną to pisanie kodu nie sprawiało mi tak dużych kłopotów jak się spodziewałem przed rozpoczęciem projektu. Zadanie z konkursu Algorytmion udało się w pełni zrealizować dokładając nawet do niego dodatkowe funkcjonalności w postaci opisów błędów i interfejsu graficznego.

Projekt udało się zrealizować dużym nakładem pracy i daje on wiele satysfakcji, pozostawiając również możliwości na dalszy rozwój np. w formie czysto graficznej, z wyświetlaniem całej planszy sudoku w oknie javy itp. Z pewnością zachęcił mnie on do dalszego rozwijania swoich umiejętności w języku Java.

Autor: Wojciech Grzywocz