**Aplikacja Sudoku**

Aplikacja z interfejsem graficznym:

* wczytuje z pliku tekstowego początkową zawartość planszy do [sudoku](https://www.google.com/url?q=https://pl.wikipedia.org/wiki/Sudoku&sa=D&source=editors&ust=1641055727989000&usg=AOvVaw1O7AAVq9Qscl9MHLbvzxAT) (plik zawiera 9 wierszy, w każdym maksymalnie 9 cyfr z zakresu 0-9, gdzie zero oznacza puste pole - wymagające wypełnienia przez gracza, liczby 1-9 to niezmienne - wypełnione wartości)
* pozwala użytkownikowi wybrać cyfrę w każdym polu (poza niezmiennymi) i podświetla na czerwono niepoprawnie wypełniony wiersz, kolumnę lub kwadrat
* po poprawnym wypełnieniu planszy cyframi wyświetla komunikat z gratulacjami.

Szkic rozwiązania:

Klasa Run zawiera metodę main:

**package** zadanie.sudoku;

**import** javax.swing.SwingUtilities;

**public** **class** Run {

**public** **static** **void** main(String[] args) {

                SwingUtilities.*invokeLater*(()->{

                        (**new** SudokuWindow()).setVisible(**true**);

                });

        }

}

Klasa SudokuWindow:

* dziedziczy z JFrame i implementuje interfejs FieldValueChangeListener
* w konstruktorze wywołuje metody readPuzzle() i createGui()
* metoda readPuzzle() - wczytuje zawartość pliku puzzle.txt z aktualnego katalogu, np.:

**try** {

                        puzzle = Files.*readAllLines*(Paths.*get*("puzzle.txt"));

                } **catch** (IOException e) {

                        e.printStackTrace();

                }

gdzie plik puzzle.txt to np:

020000000

000600003

074080000

000003002

080040010

600500000

000010780

500009000

000000040

* metoda createGui() tworzy i zapamiętuje tablicę 9x9 komponentów klasy SudokuField:
* jeśli w puzzle.txt komponentowi odpowiadało 0, to tworzony jest obiekt VariableSudokuField
* jeśli była to cyfra 1-9, to tworzony jest obiekt FixedSudokuField(), który jako argument otrzymuje odpowiednią cyfrę
* dodatkowo oba rodzaje pól otrzymują argument “this” (obiekt implementujący FieldValueChangeListener)
* komponenty dodawane są do okna w taki sposób, by tworzyły siatkę 9x9 elementów (a dokładniej: siatkę 3x3 kwadratów zawierających 3x3 pola)
* metoda fieldsChanged() (zdefiniowana w interfejsie FieldValueChangeListener):
* zeruje stan błędu dla wszystkich 9x9 pól (setError(false))
* sprawdza wszystkie wiersze (również: kolumny i kwadraty) i jeśli w wierszu dowolna z cyfr się powtarza: zaznacza cały wiersz (setError(true))

**public** **class** SudokuWindow **extends** JFrame **implements** FieldValueChangeListener {

// ...

**public** SudokuWindow() {

// ...

        }

**private** **void** readPuzzle() {

                // ...

        }

**private** **void** createGui() {

                setLayout( **...** );

                // ...

                setSize(500,500);

        }

        @Override

**public** **void** fieldsChanged() {

                // ...

        }

}

Klasa SudokuField:

* jest abstrakcyjna, dziedziczy z JLabel
* prywatne pole int value - cyfra wpisana w polu; publiczny getter i setter oznaczony jako protected
* konstruktor(int) ustawia wartość przy pomocy settera
* metoda protected ​​setDisplayedValue(), która ustawia zawartość pola zgodnie z wartością value; jeśli value jest równe zero, to zawartość pola jest pusta
* metoda publiczna setError(boolean) ustawia tło na Color.red w przypadku true i UIManager.*getColor*("Panel.background") w przypadku braku błędu

Klasa FixedSudokuField:

* dziedziczy z SudokuField
* wywołuje konstruktor nadklasy
* wywołuje​​ setDisplayedValue()

Klasa VariableSudokuField:

* nadpisuje setValue() tak by wywoływała metodę z nadklasy a następnie setDisplayedValue()
* w konstruktorze (argument: FieldValueChangeListener listener):
* wywołuje konstruktor nadklasy z parametrem zero
* tworzy obiekt popupMenu - obiekt klasy JPopupMenu zawierający polecenia:
* Wyczyść - wpisuje wartość zero do pola i wywołuje metodę listener.fieldsChanged()
* 1..9 - wpisuje odpowiednią wartość do pola i wywołuje metodę listener.fieldsChanged()
* rejestruje odbiornik MouseListener i w metodzie mouseClicked() wyświetla menu (popupMenu), np.

                                VariableSudokuField field = VariableSudokuField.**this**;

                                popupMenu.show(field, field.getWidth()/2, field.getHeight()/2);

public class VariableSudokuField extends SudokuField {

        public VariableSudokuField(final FieldValueChangeListener listener) {

                // konstr. nadklasy(0)

                JPopupMenu popupMenu = new JPopupMenu();

                JMenuItem mi;

                popupMenu.add( mi = new JMenuItem("Wyczyść"));

                mi.addActionListener(e->{

// ...

                        listener.fieldsChanged();

                });

                // … pozostałe polecenia menu

                addMouseListener( … );

        }

        @Override

        public void setValue(int value) {

                //...

        }

}

Obraz zawierający stół

Opis wygenerowany automatycznie

Podpowiedzi:

* odstęp widoczny w menu na obrazku pomiędzy “Wyczyść” i cyframi dodaje się przy pomocy polecenia: addSeparator()
* warto skorzystać z metod setHorizontalAlignment i setVerticalAlignment żeby uzyskać odpowiednie położenie tekstu w etykietach
* odstępy między kwadratami 3x3 można uzyskać określając hgap i vgap dla odpowiedniego managera układu
* ramki widoczne na około VariableSudokuField to wynik: setBorder( BorderFactory.*createEtchedBorder*());

Aby uzyskać ocenę 5 należy opisaną powyżej aplikację rozszerzyć o (należy zrealizować dowolne dwie opcje z poniższych):

* możliwość wczytania gry z dowolnego pliku txt; zawartość pliku należy sprawdzić i jeśli nie jest poprawny (za mało linii, linie zawierają znaki inne niż 0-9) - wyświetlić odpowiedni komunikat;  
  [https://docs.oracle.com/javase/tutorial/uiswing/components/filechooser.html](https://www.google.com/url?q=https://docs.oracle.com/javase/tutorial/uiswing/components/filechooser.html&sa=D&source=editors&ust=1641055728012000&usg=AOvVaw0FrepHGcBN4iEyYABUWJga)
* jeśli dany wiersz, kolumna lub kwadrat są poprawnie wypełnione (wszystkimi cyframi 1..9), to są zaznaczane na zielono (mechanizm podobny do setError)
* po kliknięciu pola VariableSudokuField prawym klawiszem, zamiast zwykłego menu kontekstowego (popupMenu) wyświetla się menu zawierające wyłącznie cyfry, które w tym polu będą poprawne; cyfra aktualnie wpisana powinna być oznaczona, np. jako “Wpisane: X”. Podpowiedź: do konstruktora VariableSudokuField można przekazać kolejny parametr (lub dodać seter w celu ustawienia wartości) - obiekt implementujący HintSource (będzie to np. SudokuWindow.this). W interfejsie potrzebna jest jedna metoda, np. List<Integer> getValidNumbers( SudokuField ) - zwraca listę cyfr, które można wpisać w tym polu biorąc pod uwagę zawartość tablicy gry.

Na wikampie należy przesłać sprawozdanie w postaci pliku PDF, w którym:

* pokazany zostanie kod wszystkich napisanych klas (wraz z opisem, jeśli klasy są inne niż proponowane)
* zrzuty ekranu ilustrujące działanie poszczególnych funkcji programu
* opisane będą wybrane rozszerzenia aplikacji - należy opisać sposób ich realizacji i udokumentować zrzutami ekranu.