**ЛАБОРАТОРНА РОБОТА № 2**

з дисципліни

«Розробка програмного забезпечення на Java»

Шейко Р. О.

Групи КІ-21-2

**Завдання на лабораторну роботу**

1. Доопрацювати інтерфейс програми. Для всіх панелей оформити рамки різних

типів.

2. Додати в програму перевірку значень, що вводяться. При введенні неприпус-

тимих символів у рядок результату вивести напис «Помилка!!!".

3. Розробити додаток, аналогічний реалізованому додатку, з переліку:

* вага (мг, г, кг, т, ц);
* кількість інформації (Б, КБ, МБ, ГБ, ТБ);
* площа (мм, см, дц, м, км);
* час (нс, мс, с, хв, год);
* і тому подібні величини.

4. Одну з груп радіокнопок в другому додатку замінити компонентом JList,

іншу – JComboBox.

5. Доопрацювати додаток аналогічно проекту з попередньої роботи.

1. Доробимо програму. Увесь код у додатку А і Б.

Інтерфейс програми повинен виглядає ось так:

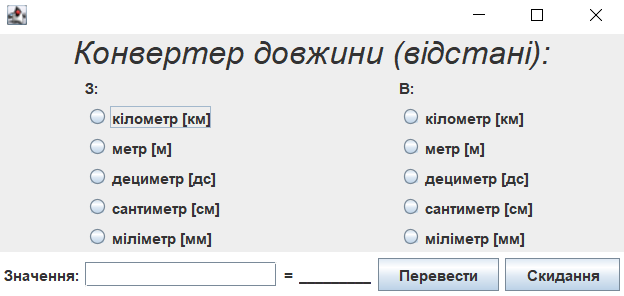


Рис 1. Інтерфейс програми

Було додано конвертацію у типи:

* Метр[м]
* Дециметр[дм]
* Сантиметр[см]
* Міліметр[мм]

Протестуємо:

|  |  |
| --- | --- |
|  |  |
| Рис 2а. Тестування переведення | Рис 2б. Тестування переведення |

1. Додамо до калькулятору перевірку для введених значень:

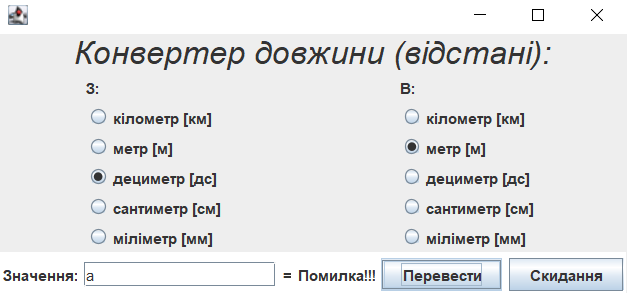


Рис 3. Ілюстрація роботи перевірки

1. Розробимо другий додаток:

* вага (мг, г, кг, т, ц);

Увесь код знаходиться у додатках В і Г.

1. Було замінено на JList і JComboBox.

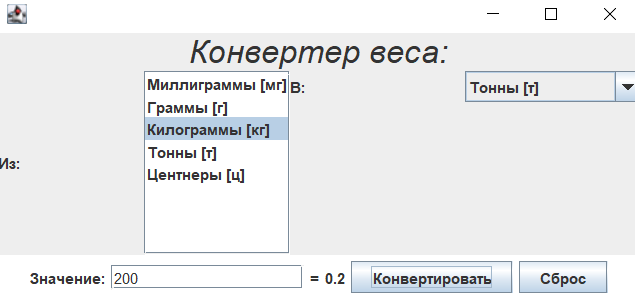


Рис 4. Ілюстрація додатку

Функціонал додатку не був змінений. Перевірка працює. Була змінена мова додатку. Зміна на JList та JComboBox проілюстровано у лістингу 1.

JList<String> list1 = new JList<>(listModel1);

list1.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

pan1.add(new JScrollPane(list1));

frame.add(pan1, c);

JPanel pan2 = new JPanel();

c.gridx = 1;

c.gridy = 1;

pan2.setLayout(new GridLayout(1, 1));

JLabel l\_out = new JLabel("В:");

pan2.add(l\_out);

JComboBox<String> comboBox = new JComboBox<>(Names);

pan2.add(comboBox);

frame.add(pan2, c);

Лістинг 1.

1. Удосконалити проект: по можливості позбавитись повторюваності коду, за-класти можливості подальшого розвитку проекту (заміна інтерфейсу користувача, бізнес-логіки і т.д.), оптимізувати код. Удосконалення виконувати над копією проекту. Звернути увагу на такі поняття, як інтерфейс, посилання на метод і т.д.

Для початку в проекті були винесені головні налаштування у клас Main. Зміни зображені нижче.

public static int width = 600;

public static int height = 450;

public static Font font = new Font("Arial", Font.ITALIC, 26);

Також було додано константу Names.

public static final String[] Names = {

"Миллиграммы [мг]",

"Граммы [г]",

"Килограммы [кг]",

"Тонны [т]",

"Центнеры [ц]"};

Також після цього було оптимізовано код й додано цикл додавання всіх елементів:

for (int i = 0; i < Names.length; i++)

listModel1.addElement(Names[i]);

JList<String> list1 = new JList<>(listModel1);

Й оптимізовано запис у JComboBox:

JComboBox<String> comboBox = new JComboBox<>(Names);

Додано новий метод getType(String).

**Висновок**

В процесі розробки двох конвертерів з використанням бібліотеки Swing було проведено наступні кроки:

**1. Створення інтерфейсу:** Було створено інтерфейс за допомогою бібліотеки Swing. Це включало в себе створення вікна, розміщення елементів інтерфейсу, таких як мітки (JLabel), кнопки (JButton), поля для вводу (JTextField), списки (JList) та комбіновані списки (JComboBox).

**2. Налаштування інтерфейсу:** Для зручності користувачів було додано мітки, що пояснюють функціональність кожного елемента інтерфейсу. Використовувалися шрифти для покращення зовнішнього вигляду додатків.

**3. Взаємодія з об'єктами інтерфейсу**: Було налаштовано обробники подій для кнопок "Перевести" та "Скидання". При натисканні кнопки "Перевести" введене значення переводилось між обраними одиницями вимірювання, і результат виводився на екран. При натисканні кнопки "Скидання" введені дані очищалися.

**4. Оптимізація:** Була здійснена невелика оптимізація коду для забезпечення кращої продуктивності та стабільності додатків. Зокрема, виправлено помилку, яка виникала при передачі значення з JList та JComboBox в метод `convert`. Ця помилка була вирішена шляхом правильного отримання вибраного значення і передачі його в метод.

У результаті цієї роботи ви створили два функціональні додатки конвертера, які дозволяють користувачам конвертувати значення між різними одиницями вимірювання. Ви також набули досвіду роботи з інтерфейсами Swing та взаємодії з об'єктами у вашому додатку. Тепер ви можете вдосконалити ці додатки або створити нові з використанням ваших навичок у розробці програмного забезпечення на Java.

Додаток А

*Лістинг файлу «Сonverter.java»*

import javax.swing.\*;

import javax.swing.plaf.basic.BasicButtonListener;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

// Press Shift twice to open the Search Everywhere dialog and type `show whitespaces`,

// then press Enter. You can now see whitespace characters in your code.

public class Converter {

public static void main(String[] args) {

JFrame frame = new JFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(600, 450);

frame.setLocationRelativeTo(null);

frame.setLayout(new GridBagLayout());

GridBagConstraints c = new GridBagConstraints();

JLabel l1 = new JLabel("Конвертер довжини (відстані):");

l1.setFont(new Font("Arial", Font.ITALIC, 26));

c.gridx = 0; c.gridy = 0;

c.anchor = GridBagConstraints.NORTH;

c.gridheight = 1; c.gridwidth = 2;

c.weightx = 1.0; c.weighty = 0.0;

frame.add(l1, c);

JPanel pan1 = new JPanel();

c.fill = GridBagConstraints.EAST;

c.gridx = 0; c.gridy = 1;

c.gridheight = 1; c.gridwidth = 1;

c.weightx = 1.0; c.weighty = 0.5;

pan1.setLayout(new GridLayout(6,1));

JLabel l\_in = new JLabel("З:");

pan1.add(l\_in);

final ButtonGroup group1 = new ButtonGroup();

JRadioButton rb1 = new JRadioButton("кілометр [км]");

rb1.setActionCommand("km");

JRadioButton rb2 = new JRadioButton("метр [м]");

rb2.setActionCommand("m");

JRadioButton rb3 = new JRadioButton("дециметр [дс]");

rb3.setActionCommand("dc");

JRadioButton rb4 = new JRadioButton("сантиметр [см]");

rb4.setActionCommand("sm");

JRadioButton rb5 = new JRadioButton("міліметр [мм]");

rb5.setActionCommand("mm");

group1.add(rb1);

group1.add(rb2);

group1.add(rb3);

group1.add(rb4);

group1.add(rb5);

pan1.add(rb1);

pan1.add(rb2);

pan1.add(rb3);

pan1.add(rb4);

pan1.add(rb5);

frame.add(pan1, c);

JPanel pan2 = new JPanel();

c.gridx = 1;

c.gridy = 1;

pan2.setLayout(new GridLayout(6,1));

JLabel l\_out = new JLabel("B:");

pan2.add(l\_out);

final ButtonGroup group2 = new ButtonGroup();

JRadioButton rb6 = new JRadioButton("кілометр [км]");

rb6.setActionCommand("km");

JRadioButton rb7 = new JRadioButton("метр [м]");

rb7.setActionCommand("m");

JRadioButton rb8 = new JRadioButton("дециметр [дс]");

rb8.setActionCommand("dc");

JRadioButton rb9 = new JRadioButton("сантиметр [см]");

rb9.setActionCommand("sm");

JRadioButton rb10 = new JRadioButton("міліметр [мм]");

rb10.setActionCommand("mm");

group2.add(rb6);

group2.add(rb7);

group2.add(rb8);

group2.add(rb9);

group2.add(rb10);

pan2.add(rb6);

pan2.add(rb7);

pan2.add(rb8);

pan2.add(rb9);

pan2.add(rb10);

frame.add(pan2, c);

JPanel pan3 = new JPanel();

pan3.setBackground(Color.WHITE);

c.fill = GridBagConstraints.BOTH;

c.gridx = 0; c.gridy = 2;

c.gridwidth = 2;

c.weightx = 1.0; c.weighty = 0.0;

JLabel l\_value = new JLabel("Значення:");

pan3.add(l\_value);

final JTextField tf\_value = new JTextField(15);

pan3.add(tf\_value);

JLabel l\_equal = new JLabel("=");

pan3.add(l\_equal);

final JLabel l\_result = new JLabel("\_\_\_\_\_\_\_\_\_");

pan3.add(l\_result);

JButton result = new JButton("Перевести");

pan3.add(result);

JButton reset = new JButton("Скидання");

pan3.add(reset);

Conversion convert = new Conversion();

result.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent e) {

float result = 0;

try{

result =

convert.convert(group1.getSelection().getActionCommand(),

group2.getSelection().getActionCommand(),

Float.parseFloat(tf\_value.getText()) );

l\_result.setText(Float.toString(result));

}

catch (Exception NullPointerException){

l\_result.setText("Помилка!!!");

}

}});

reset.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

group1.clearSelection();

group2.clearSelection();

tf\_value.setText("");

l\_result.setText("\_\_\_\_\_\_\_\_\_");

}});

frame.add(pan3, c);

frame.pack();

frame.setVisible(true);

}

}

*Додаток Б*

*Лістинг файлу «Conversion.java»*

public class Conversion {

private float km\_conv(float value, String output\_type){

switch(output\_type){

case "km": break;

case "m" : {value \*= 1000; break;}

case "sm": {value \*= 100000; break;}

case "mm": {value \*= 1000000; break;}

case "dc": {value \*= 10000; break;}

}

return value;

}

private float m\_conv(float value, String output\_type){

switch(output\_type){

case "km": {value /= 1000; break;}

case "m" : { break;}

case "dc": {value \*= 10; break;}

case "sm": {value \*= 100; break;}

case "mm": {value \*= 1000; break;}

}

return value;

}

private float dc\_conv(float value, String output\_type){

switch(output\_type){

case "km": {value \*= 0.0001; break;}

case "m" : { value \*= 0.1; break;}

case "dc": { break; }

case "sm": {value \*= 10; break;}

case "mm": {value \*= 100; break;}

}

return value;

}

private float sm\_conv(float value, String output\_type){

switch(output\_type){

case "km": { value \*= 0.00001 ; break;}

case "m" : { value \*= 0.01; break;}

case "dc": { value \*= 0.1; break; }

case "sm": { break;}

case "mm": { value \*= 10; break;}

}

return value;

}

private float mm\_conv(float value, String output\_type){

switch(output\_type) {

case "km": { value \*= 0.000001 ; break;}

case "m" : { value \*= 0.001; break;}

case "dc": { value \*= 0.01; break; }

case "sm": { value \*= 0.1; break;}

case "mm": { break;}

}

return value;

}

public float convert(String input\_type, String output\_type,

float value){

float res = 0;

switch(input\_type){

case "km": {res = km\_conv(value, output\_type); break;}

case "m": {res = m\_conv(value, output\_type); break;}

case "dc": {res = dc\_conv(value, output\_type); break;}

case "sm": {res = sm\_conv(value, output\_type); break;}

case "mm": {res = mm\_conv(value, output\_type); break;}

}

return res; }

}

*Додаток В*

*Лістинг файлу «Main.java»*

import javax.swing.\*;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class Main {

public static int width = 600;

public static int height = 450;

public static Font font = new Font("Arial", Font.ITALIC, 26);

public static final String[] Names = {

"Миллиграммы [мг]",

"Граммы [г]",

"Килограммы [кг]",

"Тонны [т]",

"Центнеры [ц]"};

public static void main(String[] args) {

JFrame frame = new JFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(width, height);

frame.setLocationRelativeTo(null);

frame.setLayout(new GridBagLayout());

GridBagConstraints c = new GridBagConstraints();

JLabel l1 = new JLabel("Конвертер веса:");

l1.setFont(font);

c.gridx = 0;

c.gridy = 0;

c.anchor = GridBagConstraints.NORTH;

c.gridheight = 1;

c.gridwidth = 2;

c.weightx = 1.0;

c.weighty = 0.0;

frame.add(l1, c);

JPanel pan1 = new JPanel();

c.fill = GridBagConstraints.EAST;

c.gridx = 0;

c.gridy = 1;

c.gridheight = 1;

c.gridwidth = 1;

c.weightx = 1.0;

c.weighty = 0.5;

pan1.setLayout(new GridLayout(1, 1));

JLabel l\_in = new JLabel("Из:");

pan1.add(l\_in);

DefaultListModel<String> listModel1 = new DefaultListModel<>();

for (int i = 0; i < Names.length; i++)

listModel1.addElement(Names[i]);

JList<String> list1 = new JList<>(listModel1);

list1.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

pan1.add(new JScrollPane(list1));

frame.add(pan1, c);

JPanel pan2 = new JPanel();

c.gridx = 1;

c.gridy = 1;

pan2.setLayout(new GridLayout(1, 1));

JLabel l\_out = new JLabel("В:");

pan2.add(l\_out);

JComboBox<String> comboBox = new JComboBox<>(Names);

pan2.add(comboBox);

frame.add(pan2, c);

JPanel pan3 = new JPanel();

pan3.setBackground(Color.WHITE);

c.fill = GridBagConstraints.BOTH;

c.gridx = 0;

c.gridy = 2;

c.gridwidth = 2;

c.weightx = 1.0;

c.weighty = 0.0;

JLabel l\_value = new JLabel("Значение:");

pan3.add(l\_value);

final JTextField tf\_value = new JTextField(15);

pan3.add(tf\_value);

JLabel l\_equal = new JLabel("=");

pan3.add(l\_equal);

final JLabel l\_result = new JLabel("\_\_\_\_\_\_\_\_\_");

pan3.add(l\_result);

JButton result = new JButton("Конвертировать");

pan3.add(result);

JButton reset = new JButton("Сброс");

pan3.add(reset);

Conversion convert = new Conversion();

list1.addListSelectionListener(new ListSelectionListener() {

@Override

public void valueChanged(ListSelectionEvent e) {

if (!e.getValueIsAdjusting()) {

String selectedUnit1 = list1.getSelectedValue();

// Здесь обработайте выбор единицы веса из первой группы

}

}

});

comboBox.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String selectedUnit2 = comboBox.getSelectedItem().toString();

// Здесь обработайте выбор единицы веса из второй группы

}

});

result.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

double result = 0;

try {

String inputUnit = getType(list1.getSelectedValue());

String outputUnit = getType(comboBox.getSelectedItem().toString());

result = convert.convert(inputUnit, outputUnit, Double.parseDouble(tf\_value.getText()));

l\_result.setText(Double.toString(result));

} catch (Exception ex) {

l\_result.setText(ex.getMessage());

}

}

});

reset.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

list1.clearSelection();

comboBox.setSelectedIndex(0);

tf\_value.setText("");

l\_result.setText("\_\_\_\_\_\_\_\_\_");

}

});

frame.add(pan3, c);

frame.pack();

frame.setVisible(true);

}

public static String getType(String type){

switch (type)

{

case "Миллиграммы [мг]": return "mg";

case "Граммы [г]": return "g";

case "Килограммы [кг]": return "kg";

case "Тонны [т]": return "t";

case "Центнеры [ц]": return "c";

}

return "error";

}

}

*Додаток Г*

*Лістинг файлу «Conversion.java»*

public class Conversion {

public static double convert(String inputType, String outputType, double value) {

switch (inputType) {

case "mg":

return convertFromMilligrams(value, outputType);

case "g":

return convertFromGrams(value, outputType);

case "kg":

return convertFromKilograms(value, outputType);

case "t":

return convertFromTonnes(value, outputType);

case "c":

return convertFromCentners(value, outputType);

default:

throw new IllegalArgumentException("Неподдерживаемый тип входных данных");

}

}

private static double convertFromMilligrams(double value, String outputType) {

switch (outputType) {

case "mg":

return value;

case "g":

return value / 1000;

case "kg":

return value / 1\_000\_000;

case "t":

return value / 1\_000\_000\_000;

case "c":

return value / 10\_000\_000;

default:

throw new IllegalArgumentException("Неподдерживаемый тип выходных данных");

}

}

private static double convertFromGrams(double value, String outputType) {

switch (outputType) {

case "mg":

return value \* 1000;

case "g":

return value;

case "kg":

return value / 1000;

case "t":

return value / 1\_000\_000;

case "c":

return value / 10\_000;

default:

throw new IllegalArgumentException("Неподдерживаемый тип выходных данных");

}

}

private static double convertFromKilograms(double value, String outputType) {

switch (outputType) {

case "mg":

return value \* 1\_000\_000;

case "g":

return value \* 1000;

case "kg":

return value;

case "t":

return value / 1000;

case "c":

return value / 10;

default:

throw new IllegalArgumentException("Неподдерживаемый тип выходных данных");

}

}

private static double convertFromTonnes(double value, String outputType) {

switch (outputType) {

case "mg":

return value \* 1\_000\_000\_000;

case "g":

return value \* 1\_000\_000;

case "kg":

return value \* 1\_000;

case "t":

return value;

case "c":

return value \* 10;

default:

throw new IllegalArgumentException("Неподдерживаемый тип выходных данных");

}

}

private static double convertFromCentners(double value, String outputType) {

switch (outputType) {

case "mg":

return value \* 10\_000\_000;

case "g":

return value \* 10\_000;

case "kg":

return value \* 100;

case "t":

return value / 10;

case "c":

return value;

default:

throw new IllegalArgumentException("Неподдерживаемый тип выходных данных");

}

}

}