БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра информационных систем управления

Специальность “Прикладная информатика”

**ОТЧЕТ**

по учебной практике (вариант 7)

Студент

группы 12Б, 2 курс \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Р. Д. Доскоч

*(подпись)*

Руководитель,

ст.реподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.А. Марушко

*(подпись)*

Минск

2021

Оглавление

**Задание 11**

**Задание 23**

**Задание 34**

**Задание 46**

**Задание 58**

**Задание 610**

**Задание 712**

**Задание 813**

**Задание 915**

**Задание 1017**

**Задание 1118**

**Задание 1221**

**Литература24**

ЗАДАНИЕ 1

Постановка задачи:

* Изобразить сектор круга, вращающийся в плоскости экрана вокруг своего центра по часовой стрелке.
* Установить в программе толщину границы фигуры, цвет фигуры, цвет границы фигуры.

Особенности реализации:

Создаю Новую Форму на нее добавляю панель с параметрами сектора

На панели запускаю таймер и ActionListener для вращения сектора. Вращаю сектор через изменение угла и функции AffineTransform.getRotateInstance(angle).

**public** **void** **draw**(Graphics2D g) {

g.setStroke(**new** BasicStroke(width, BasicStroke.CAP\_SQUARE, BasicStroke.JOIN\_BEVEL));

g.translate(**250**, **250**);

g.transform(AffineTransform.getRotateInstance(angle));

angle += (**float**) Math.PI / **24**f;

g.setColor(fillColor);

g.fill(toDraw);

g.setColor(contourColor);

g.draw(toDraw);

}

Результат работы программы:

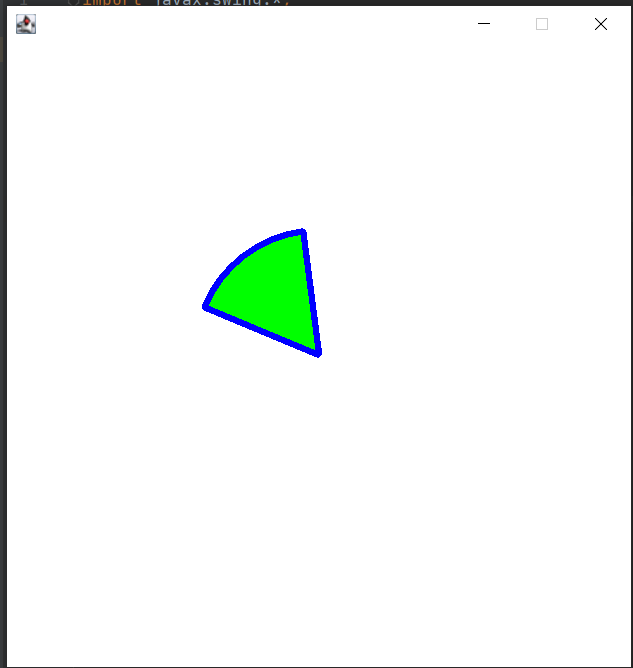


Рисунок 1 - скриншот работы программы

ЗАДАНИЕ 2

Постановка задачи:

- для изображения указанной в задании фигуры создать класс, реализующий интерфейс

Shape;

- создайте указанный фильтр изображения; при тестировании выведите фигуру без

фильтра и с фильтром (аналогично фильтрам из примеров);

Фигура (дорожный знак): надпись 80 в окружности, цвет окружности и символа –

красный, цвет фона – серый с градиентной заливкой справа-налево

Фильтр: Brighten

Особенности реализации:

Рисуем основную фигуру:

**public** **void** **draw**(Graphics2D ig, Component c)

{

BufferedImage bimage = **new** BufferedImage(

WIDTH, HEIGHT, BufferedImage.TYPE\_INT\_RGB);

Graphics2D g = bimage.createGraphics();

g.setPaint(**new** GradientPaint(

**0**, **0**, Color.WHITE, WIDTH, **0**, Color.GRAY));

g.fillRect(**0**, **0**, WIDTH, HEIGHT);

g.setColor(Color.RED);

g.setStroke(**new** BasicStroke(**40**));

g.drawOval(**50**, **50**, WIDTH - **100**, HEIGHT - **100**);

g.setColor(Color.white);

g.fillOval(**70**, **70**, WIDTH - **140**, HEIGHT - **140**);

Font font = **new** Font("Serif", Font.BOLD, **17**);

Font bigfont = font.deriveFont(

AffineTransform.getScaleInstance(**13.0**, **13.0**));

GlyphVector gv = bigfont.createGlyphVector(

g.getFontRenderContext(), "80");

Shape shapeEight = gv.getGlyphOutline(**0**);

Shape shapeZero = gv.getGlyphOutline(**1**);

g.setStroke(**new** BasicStroke(**5.0f**));

g.translate(**140**, **320**);

g.setPaint(Color.RED);

g.fill(shapeEight);

g.fill(shapeZero);

ig.drawImage(**new** RescaleOp(**1.25f**, **0**, **null**)

.filter(bimage, **null**), **0**,**0**, c);

ig.drawString("Brighten", **80**, **13**);

}

Результат работы программы:



Рисунок 2 - Скриншот окна приложения

ЗАДАНИЕ 3

Постановка задачи:

- Разработайте пользовательский класс Shape реализующий рисование указанной

алгебраической линии.

- Разработайте пользовательский класс Stroke для отображения указанного контура

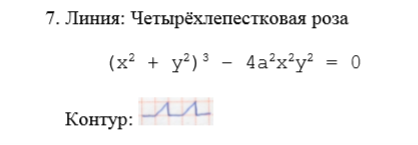


Рисунок 3 - Вариант задания

Особенности реализации:

Рисуем основную фигуру:

**private** **void** **createrose**() {

moveTo(**450**, **250**);

**for** (**var** angle = **0**; angle < **2**\*PI; angle += toRadians(**1**))**{**

**this.**lineTo(

translateX(xByAngle(angle)),

translateY(yByAngle(angle))

);

}

closePath();

}

**private** **double** **translateX**(**double** x) { **return** x + startX; }

**private** **double** **translateY**(**double** y) { **return** y + startY; }

**private** **double** **yByAngle**(**double** angle) {

**return** radius \* Math.cos(angle \* **2**) \* Math.sin(angle);

}

**private** **double** **xByAngle**(**double** angle) {

**return** radius \* Math.cos(angle \* **2**) \* Math.cos(angle);

}

Отображаем необходимый контур:

**public** Shape **createStrokedShape**(Shape shape) {

GeneralPath path = **new** GeneralPath();

**float**[] coord = **new** **float**[**2**];

**float**[] prev = **new** **float**[**2**];

**double** got = **0**;

**for** (PathIterator iter = shape.getPathIterator(**null**); !iter.isDone(); iter.next()) {

**int** type = iter.currentSegment(coord);

**switch** (type) {

**case** PathIterator.SEG\_MOVETO:

path.moveTo(coord[**0**], coord[**1**]);

**break**;

**case** PathIterator.SEG\_LINETO:

**double** x1 = prev[**0**];

**double** y1 = prev[**1**];

**double** dx = coord[**0**] - x1;

**double** dy = coord[**1**] - y1;

**double** length = Math.sqrt(dx \* dx + dy \* dy);

**double** cos = dx / length;

**double** sin = dy / length;

x1 += cos \* got;

y1 += sin \* got;

length -= got;

got = **0**;

**double** step = **4**;

**boolean** isStraight = **true**;

**while** (got < length) {

x1 += cos \* step;

y1 += sin \* step;

got += step;

**if** (!isStraight) {

path.lineTo(x1 + sin \* step, y1 - cos \* step);

}

path.lineTo(x1, y1);

isStraight = !isStraight;

}

got -= length;

**break**;

}

prev[**0**] = coord[**0**];

prev[**1**] = coord[**1**];

}

**return** stroke.createStrokedShape(path);

}

Результат работы программы:

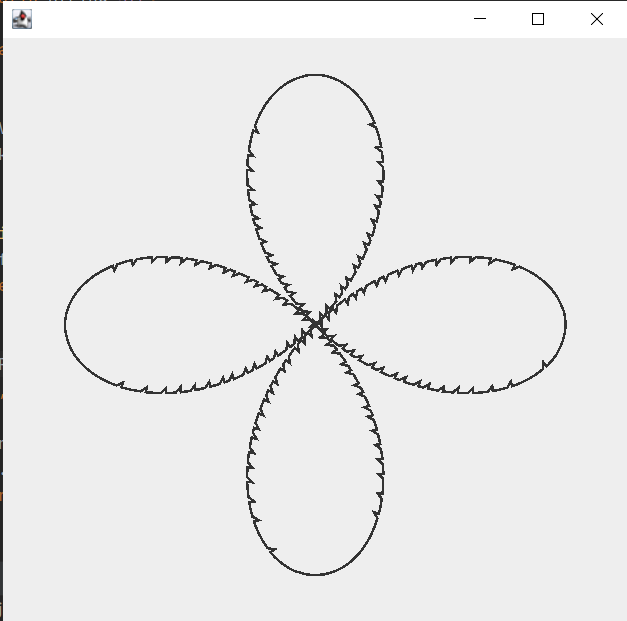


Рисунок 4 - Скриншот окна приложения

ЗАДАНИЕ 4

Постановка задачи:

Модифицируйте вашу программу следующим образом. В демонстрационное приложение добавьте возможность печати небольшого отчёта о решении задания №3. Отчёт должен содержать следующее:

- рисунок с подписью алгебраической линии вашего задания

- исходный текст класса Shape, реализующий рисование указанной алгебраической линии.

При печати используйте режим альбомной ориентации страницы и двустороннюю печать.

Особенности реализации:

С помощью PrinterJob “печатаем” отчёт о работе, т.е. сохраняем его в pdf файл.

**public** **HardcopyWriter**(Frame frame, String name,**int** size, **double** lM,

**double** rM, **double** tM, **double** bM)

{

Toolkit toolkit = frame.getToolkit(); // get Toolkit from Frame

**synchronized** (printProps)

{

PageAttributes pa = **new** PageAttributes();

pa.setOrientationRequested(PageAttributes.

OrientationRequestedType.LANDSCAPE);

JobAttributes ja = **new** JobAttributes();

ja.setSides(JobAttributes.SidesType.TWO\_SIDED\_LONG\_EDGE);

job = toolkit.getPrintJob(frame, name, ja, pa);

type = pa.getOrientationRequested() ==

PageAttributes.OrientationRequestedType.PORTRAIT ? **0** : **1**;

}

pageSize = job.getPageDimension(); // query the page size

pagedPi = job.getPageResolution(); // query the page resolution

x0 = (**int**) (lM \* pagedPi);

y0 = (**int**) (tM \* pagedPi);

width = pageSize.width - (**int**) ((lM + rM) \* pagedPi) - **200**;

height = pageSize.height - (**int**) ((tM + bM) \* pagedPi);

font = **new** Font("Arial", Font.PLAIN, size);

metrics = frame.getFontMetrics(font);

lineHeight = metrics.getHeight();

lineAscent = metrics.getAscent();

charWidth = metrics.charWidth('0');

**switch** (type) {

**case** **0**:

chars\_per\_line = width / charWidth - **1**;

lines\_per\_page = height / lineHeight - **10**;

**break**;

**case** **1**:

chars\_per\_line = width / charWidth - **1**;

lines\_per\_page = height / lineHeight - **27**;

**break**;

}

df.setTimeZone(TimeZone.getDefault());

time = df.format(**new** Date());

**this**.jobName = name;

**this**.fontSize = size;

}

Результат работы программы:

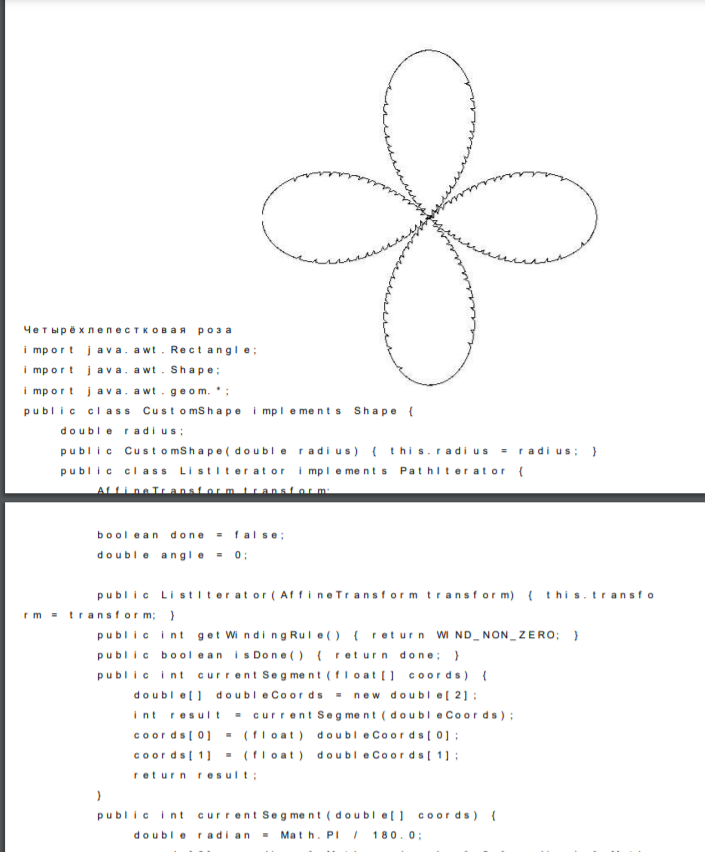
}

Рисунок 5 - Скриншот pdf файла

ЗАДАНИЕ 5

Постановка задачи:

- Разработайте систему классов/интерфейсов для предметной области Вашего варианта

задания. Данные необходимо упорядочить по атрибутам/свойствам товаров, предметов и т.п. в виде дерева.

- Разработайте графическое приложение для ввода/отображения данных Вашего варианта задания. При отображении структуры данных в виде дерева реализуйте интерфейс javax.swing.Tree.TreeModel. Листья дерева отображайте в виде таблицы, для

этого реализуйте интерфейс javax.swing.table.TableModel.

Тематика: Видеотека.

Особенности реализации:

Реализуем интерфейс TableModel:

**class** **CatalogTableModel** **implements** TableModel {

**private** Set<TableModelListener> listeners = **new** HashSet<>();

**private** List<MovieNode> infoNodes = **new** ArrayList<>();

**private** **static** **final** String[] columnNames = **new** String[]

{"Genre", "Name", "Year", "Price"};

**private** **static** **final** Class<?>[] columnTypes = **new** Class[]

{String.class, String.class, Integer.class, Float.class};

CatalogTableModel() {}

CatalogTableModel(List<MovieNode> al) {

setInfoNodes(al);

}

**private** **void** **setInfoNodes**(List<MovieNode> movieNodes) {

infoNodes = movieNodes;

}

**@Override**

**public** **void** **addTableModelListener**(TableModelListener l) {

listeners.add(l);

}

**@Override**

**public** Class<?> getColumnClass(**int** columnIndex) {

**return** columnTypes[columnIndex];

}

**@Override**

**public** **int** **getColumnCount**() {

**return** columnNames.length;

}

**@Override**

**public** String **getColumnName**(**int** columnIndex) {

**return** columnNames[columnIndex];

}

**@Override**

**public** **int** **getRowCount**() {

**return** infoNodes.size();

}

**@Override**

**public** Object **getValueAt**(**int** rowIndex, **int** columnIndex) {

MovieNode busNode = infoNodes.get(rowIndex);

**switch** (columnIndex) {

**case** **0**:

**return** busNode.genre;

**case** **1**:

**return** busNode.name;

**case** **2**:

**return** busNode.year;

**case** **3**:

**return** busNode.price;

}

**return** "";

}

**@Override**

**public** **boolean** **isCellEditable**(**int** rowIndex, **int** columnIndex) {

**return** **false**;

}

**@Override**

**public** **void** **removeTableModelListener**(TableModelListener l) {

listeners.remove(l);

}

**@Override**

**public** **void** **setValueAt**(Object aValue, **int** rowIndex, **int** columnIndex){ }

}

Результат работы программы:

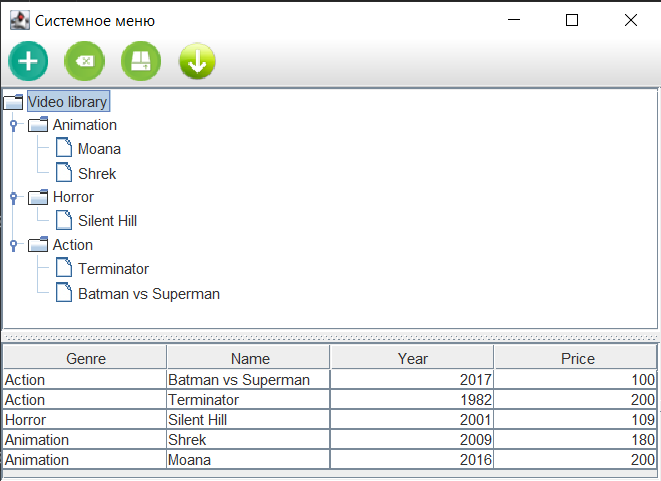


Рисунок 6 - Скриншот окна приложения

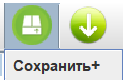
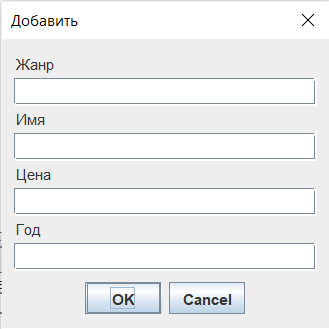
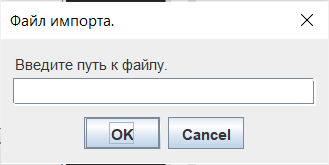


Рисунок 10 - Опция импорт из файла

Рисунок 8 - Опция Сохранить

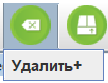
 Рисунок 9 - Опция Удалить

Рисунок 7 - Опция Добавить.

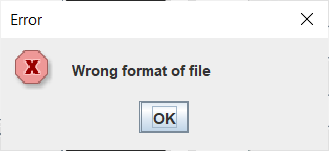
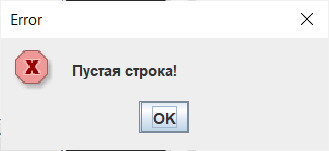
 

Рисунок 11 - Возможная ошибка импорта из файла Рисунок 12 - Возможная ошибка опции создания

ЗАДАНИЕ 6

Постановка задачи:

Для выполнения задания используется ваш вариант решения задания №3. Модифицируйте вашу программу следующим образом.

Создайте тестовое приложение, добавьте в ваш класс рисования алгебраической линии

возможность «перетаскивание» (drag-and-drop). Реализуйте необходимые интерфейсы в классе и в приложении для демонстрации «перетаскивания» алгебраической линии между несколькими копиями тестового приложения.

Особенности реализации:

Используем интерфейсы DragGestureListener, DragSourceListener, DropTargetListener, MouseListener, MouseMotionListener.

Определяем тип действия:

**public** **void** **dragGestureRecognized**(DragGestureEvent e) {

**if** (!dragMode) **return**;

MouseEvent inputEvent = (MouseEvent) e.getTriggerEvent();

**int** x = inputEvent.getX();

**int** y = inputEvent.getY();

Rectangle rectangle = **new** Rectangle(

x - LINE\_WIDTH, y - LINE\_WIDTH,

LINE\_WIDTH \* **2**, LINE\_WIDTH \* **2**);

**for** (CustomShape witch : witches) {

**if** (witch.intersects(rectangle)) {

customShape = witch;

CustomShape dragScribble = (CustomShape) witch.clone();

dragScribble.translate(-x, -y);

Cursor cursor;

**switch** (e.getDragAction()) {

**case** DnDConstants.ACTION\_COPY:

cursor = DragSource.DefaultCopyDrop;

**break**;

**case** DnDConstants.ACTION\_MOVE:

cursor = DragSource.DefaultMoveDrop;

**break**;

**default**:

**return**;

}

e.startDrag(cursor, dragScribble, **this**);

**return**;

}

}

}

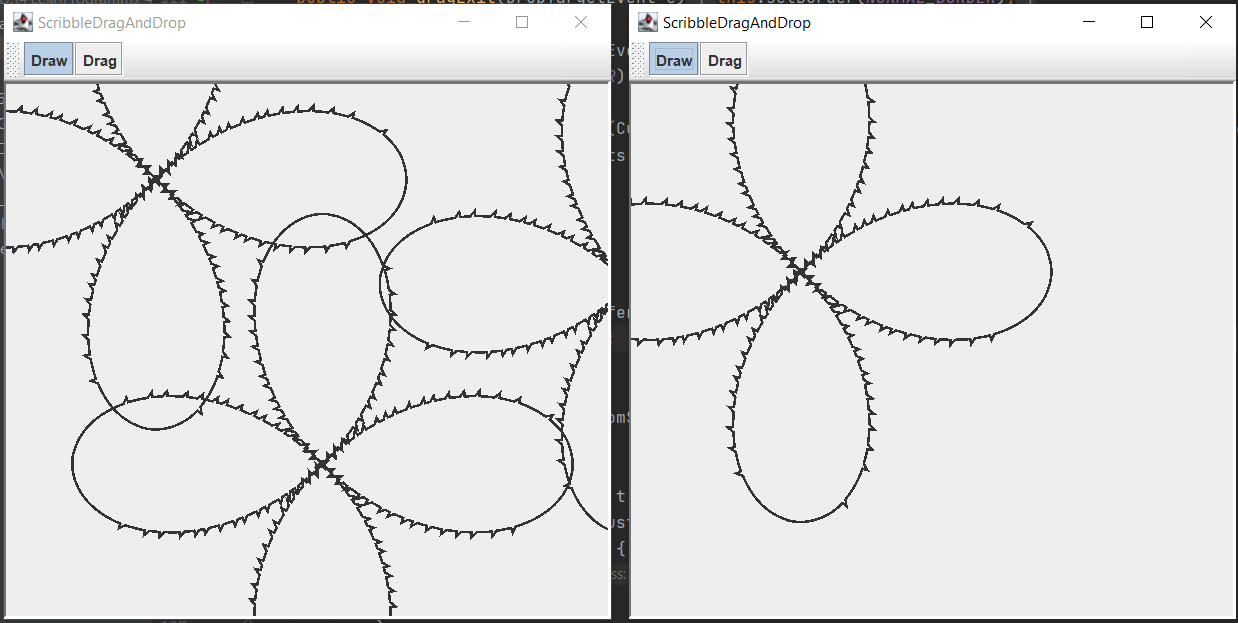


Рисунок 13 - Скриншот работы приложения

ЗАДАНИЕ 7

Постановка задачи:

- Исследовать предложенную предметную область, спроектировать структуру базы

данных объектов выбранной предметной области (из не менее чем 2-х таблиц объектов).

- Разработайте графическое приложение для создания/ввода/отображения БД Вашего

варианта задания. Содержимое БД отображайте в виде таблиц.

Особенности реализации:

Передаем sql запросы в БД с помощью методов. Например:

connection = DriverManager.getConnection(connect);

statement = connection.createStatement();

String selectSQL = "SELECT \* FROM MyVideotekaFirst1";

resultSet = statement.executeQuery(selectSQL);

Получаем информацию из таблиц:

**public** **static** **void** **createNewTable**(String tableName) **throws** ClassNotFoundException {

// SQL statement for creating a new table

String sql = "create table if not exists "+ tableName +"(\n" +

"\tid integer primary key,\n" +

"\tgenre TEXT,\n" +

"\tname TEXT,\n" +

"\tprice TEXT,\n" +

"\tyear TEXT\n" +

");";

Class.forName("org.sqlite.JDBC");

**try** (Connection conn = DriverManager.getConnection(url);

Statement stmt = conn.createStatement())

{

stmt.execute(sql);

} **catch** (SQLException e) {

System.out.println(e.getMessage());

}

}

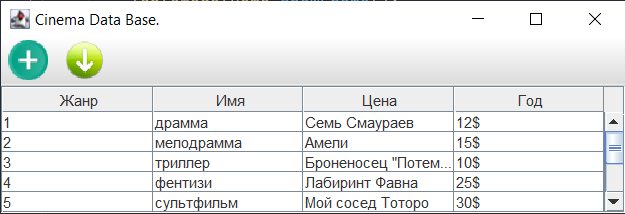


Рисунок 14 - главное меню приложения

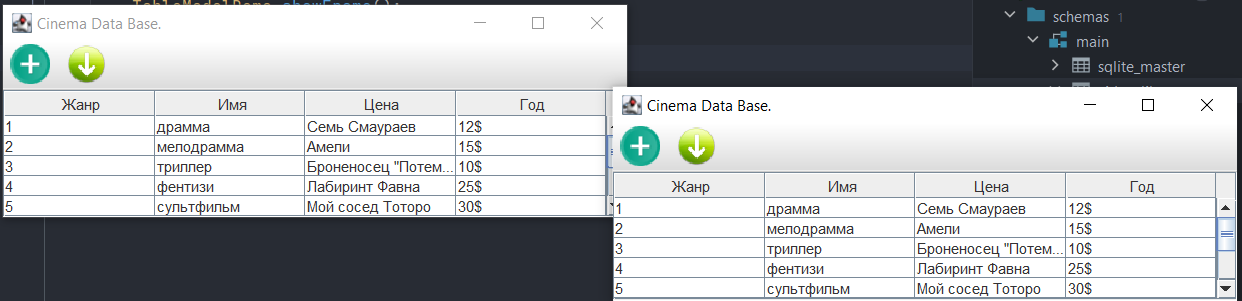


Рисунок 15 Опция Создать

ЗАДАНИЕ 8

Постановка задачи:

Разработайте простой компонент вашего варианта задания на базе класса Canvas.

Создайте файл манифеста и упакуйте компонент вместе с исходным кодом разработанных классов. При разработке поместите все ваши классы в пакет:

bsu.fpmi.educational\_practice

Создайте тестовое приложение в NetBeans с использованием вашего компонента.

Прямоугольная заливка со скруглёнными углами. Свойства: ширина, высота и цвет.

Особенности реализации:

Задаём цвет и размер:

**@Override**

**public** **void** **paint**(Graphics g) {

Graphics2D g2 = (Graphics2D) g;

g2.setPaint(Color.RED);

g2.draw(**new** RoundRectangle2D.Double(x, y,

width, height, arcWidth, arcHeight));

g2.fillRoundRect(x, y, width, height, arcWidth, arcHeight);

}

В файле манифеста указываем наличие компонента:

Manifest-Version: 1.0

X-COMMENT: Main-Class will be added automatically by build

Name: MainClass.class

Java-Bean: true

Результат работы программы:

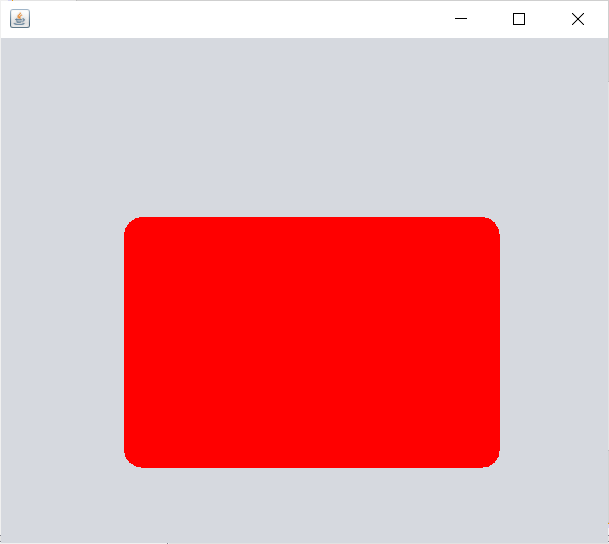
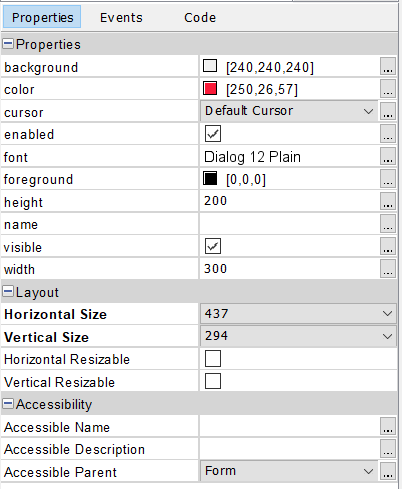
 

Рисунок 16 – скриншот программы Рисунок 17 - Настройки get set

ЗАДАНИЕ 9

Постановка задачи:

Разработайте компонент вашего варианта задания. Создайте файл манифеста и

упакуйте компонент вместе с исходным кодом разработанных классов. При разработке

поместите все ваши классы в пакет: bsu.fpmi.educational\_practice2019

Компонент должен реализовать класс BeanInfo с информацией о компоненте.

Создайте тестовое приложение в NetBeans с использованием вашего компонента.

Интерфейсные компоненты с реализацией собственного события AcсeptEvent.

Определить также интерфейс слушателя вашего события AcсeptListener:

Передавать слушателю события информацию о том, в результате чего произошло

событие.

Однострочный статический текст и кнопка. Свойства: текст, текст кнопки, символ

подтверждения. Событие генерируется при нажатии на кнопку или вводе символа.

Особенности реализации:

Передаем сообщения компоненту с помощью ActionListener. AnswerEvent получает ID кнопки, на которую мы нажали.

**public** **void** **initialize**() {

setLayout(**new** FlowLayout(FlowLayout.LEFT));

setSize(**400**, **300**);

setResizable(**false**);

String[] items = {"One", "Two", "Three", "Four", "Five"};

var indexItemToDelete = **new** AtomicInteger();

var labelDeleteIndex = **new** JLabel("Current Index to delete: 0");

var comboBox = **new** JComboBox<>(items);

comboBox.addActionListener(e -> {

indexItemToDelete.set(comboBox.getSelectedIndex());

labelDeleteIndex.setText("Current Index to delete: " + indexItemToDelete.incrementAndGet());

});

add(comboBox);

var bAdd = **new** JButton("Add new");

bAdd.addActionListener(e -> {

JPanel panel = **new** JPanel(**new** GridLayout(**0**, **1**));

panel.add(**new** Label("Enter New Item Text"));

var text = **new** JTextField("Enter Text...");

panel.add(text);

var result = JOptionPane.showConfirmDialog(

**null**,

panel, "Добавить", JOptionPane.OK\_CANCEL\_OPTION,

JOptionPane.PLAIN\_MESSAGE);

**if** (result == JOptionPane.OK\_OPTION)

comboBox.insertItemAt(text.getText(), **0**);

});

add(bAdd);

add(labelDeleteIndex);

setLocationRelativeTo(**null**);

setVisible(**true**);

}

Результат работы программы:

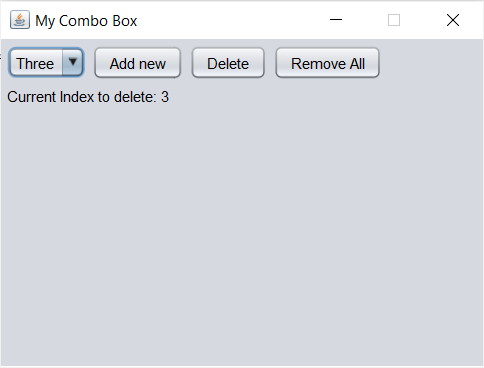
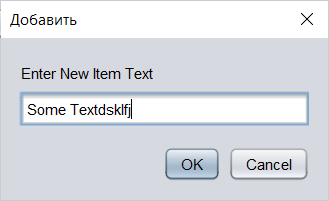
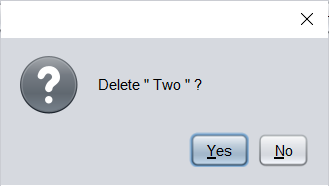
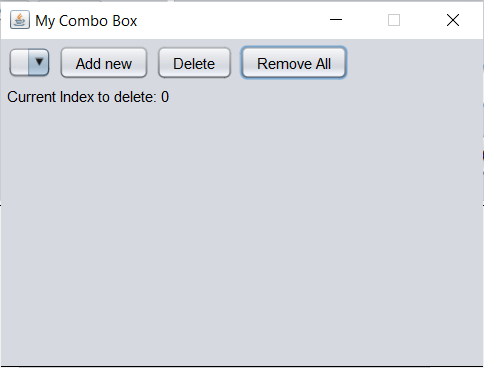
 

Рисунок 18 - скриншот выполнения программы Рисунок 19 - Опция Добавить

 Рисунок 20 - Опция удалить Рисунок 21 - Результат действия Опции удалить все

ЗАДАНИЕ 10

Постановка задачи:

Создаём собственный редактор для каждого свойства компонента.Каждый

редактор ограничивает возможные значения свойства,предоставляя выбор из списка

трёх – пяти допустимых значений (т.е.определяем методы getTags()).

Регистрируем редакторы в классе BeanInfo компонента.

Особенности реализации:

**public** **class** **ButtonAddTextEditor** **extends** PropertyEditorSupport{

**protected** String value;

**@Override**

**public** String[] **getTags**()

{

**return** **new** String[] {

"ADD", "add", "add new", "Add new Record" };

}

**@Override**

**public** **void** **setAsText**(String s) {

**this**.value = s;

**super**.setAsText(s);

}

**@Override**

**public** String **getJavaInitializationString**()

{

**return** "\"" + value + "\""; }

}

Регистрация всех Bean:

**static** PropertyDescriptor **prop**(String name, String description) {

**try** {

PropertyDescriptor p = **new** PropertyDescriptor(name, MyComboBox.class);

p.setShortDescription(description);

**return** p;

}

**catch**(IntrospectionException e) { **return** **null**; }

}

**static** PropertyDescriptor[] props = {

prop("ButtonAddText", "add new record"),

prop("ButtonDeleteText", "delete by ID"),

prop("ButtonRemoveAllText", "Remove all records"),

};

**static** {

props[**0**].setPropertyEditorClass(ButtonAddTextEditor.class);

props[**1**].setPropertyEditorClass(ButtonDeleteTextEditor.class);

props[**2**].setPropertyEditorClass(ButtonRemoveAllTextEditor.class);

}

Результат работы программы:

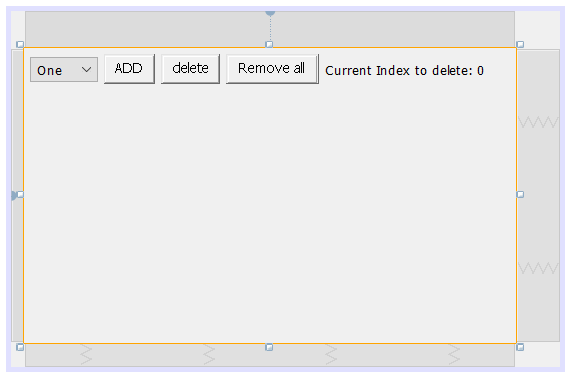


Рисунок 22 - Скриншот окна

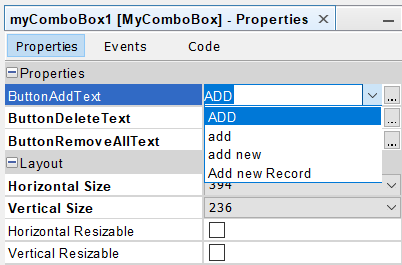


Рисунок 23 - Собственный редактор для свойств компонента

ЗАДАНИЕ 11

Постановка задачи:

Создать сервлет и взаимодействующие с ним пакеты Java-классов и HTML-

документов, выполняющие действия для решения вашего варианта задания.

Представить решение в виде web-приложения.

Вывод фрагментов текстов шрифтами, различного размера. Размер шрифта и количество строк вводятся пользователем.

Реализация шаблона MVC

Подключение к серверу:

**public** **class** **Program**

{

**public** **static** **void** **Main**(**string**[] args)

{

CreateHostBuilder(args).Build().Run();

}

**public** **static** IHostBuilder **CreateHostBuilder**(**string**[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder => webBuilder.UseStartup<Startup>(););

}

Логика переноса строк текста:

**public** **static** List<**string**> Divider(**this** **string** str, **int** blockLength) {

**var** blocks = **new** List<**string**>(str.Length / blockLength + **1**);

**for** (**var** i = **0**; i < str.Length; i += blockLength)

{

**if** (str.Length - i > blockLength)

blocks.Add(str.Substring(i, blockLength));

**else**

blocks.Add(str.Substring(i, str.Length - i)

+ **string**(' ', blockLength - (str.Length - i)));

}

**return** blocks;

}

Часть View главная страница:

<div class="text-center">

<h1 class="display-4">Welcome to converter</h1>

<p>Insert your text and set size to convert.</p>

<form method="post" action="~/Home/Area">

<label>Size</label><br/>

<input type="number" name="size" required/><br/>

<label>Count of rows</label><br/>

<input type="number" name="count" required/><br/>

<label>Your Text here</label><br/>

<textarea oninput="auto\_grow(this)" name="text" placeholder="Enter your text" required></textarea><br/>

<input type="submit" value="Convert it!"/>

</form>

</div>

Часть Controller:

**public** **class** **HomeController** : Controller

{

**private** **readonly** ILogger<HomeController> \_logger;

**public** **HomeController**(ILogger<HomeController> logger) => \_logger = logger;

**public** IActionResult **Index**() => View();

**public** IActionResult **Privacy**() => View();

**public** IActionResult **About**() => View();

[HttpPost]

**public** IActionResult **Area**(**string** text, **int** size, **int** count)

{

ViewBag.textBox = text.Divider(text.Length / count);

ViewBag.size = size + "px";

**return** **View**();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

**public** IActionResult **Error**() => View(

**new** ErrorViewModel {

RequestId = Activity.Current?.Id

?? HttpContext.TraceIdentifier

});

}

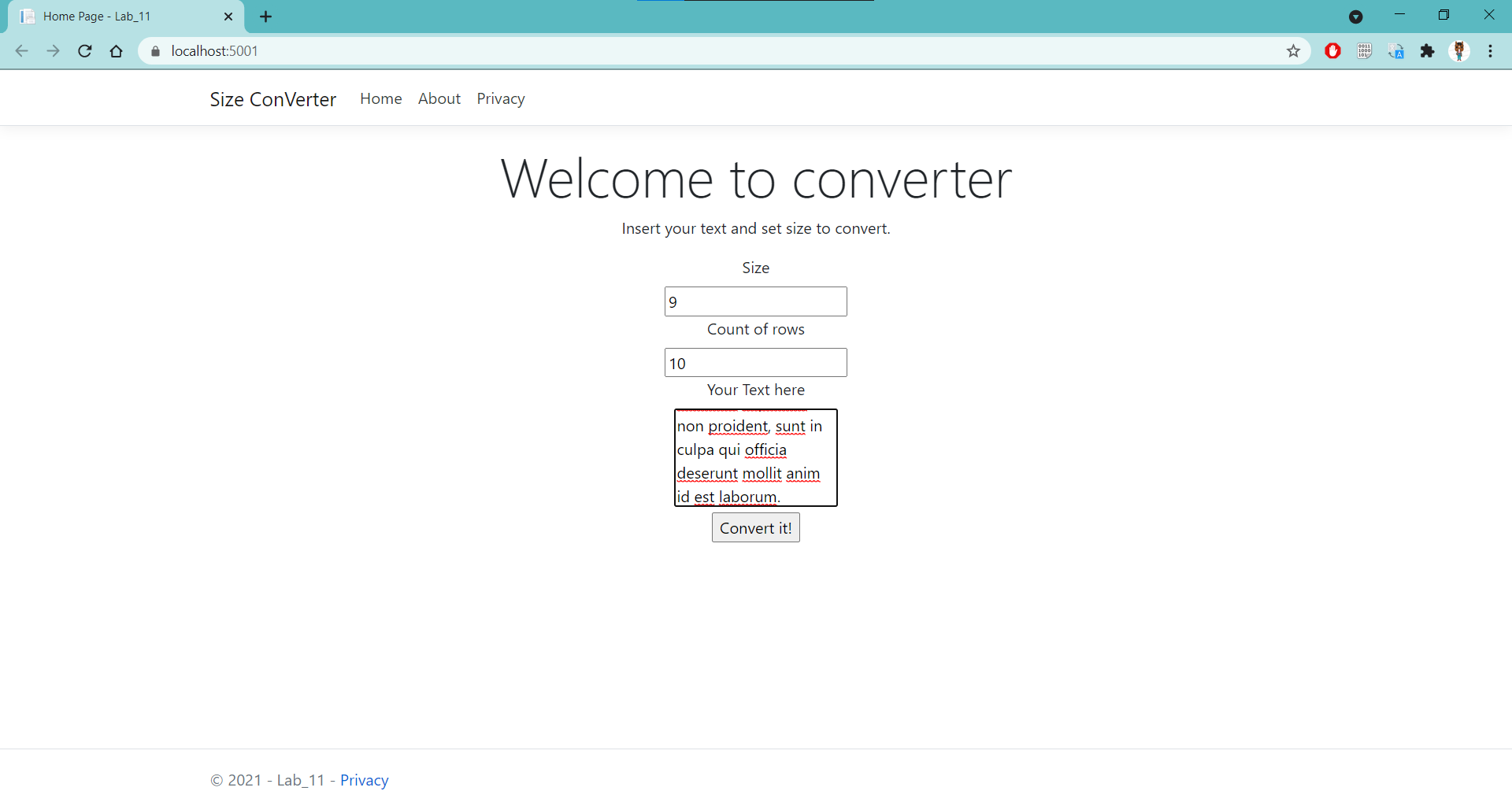


Рисунок 24 - Главная страница сайта

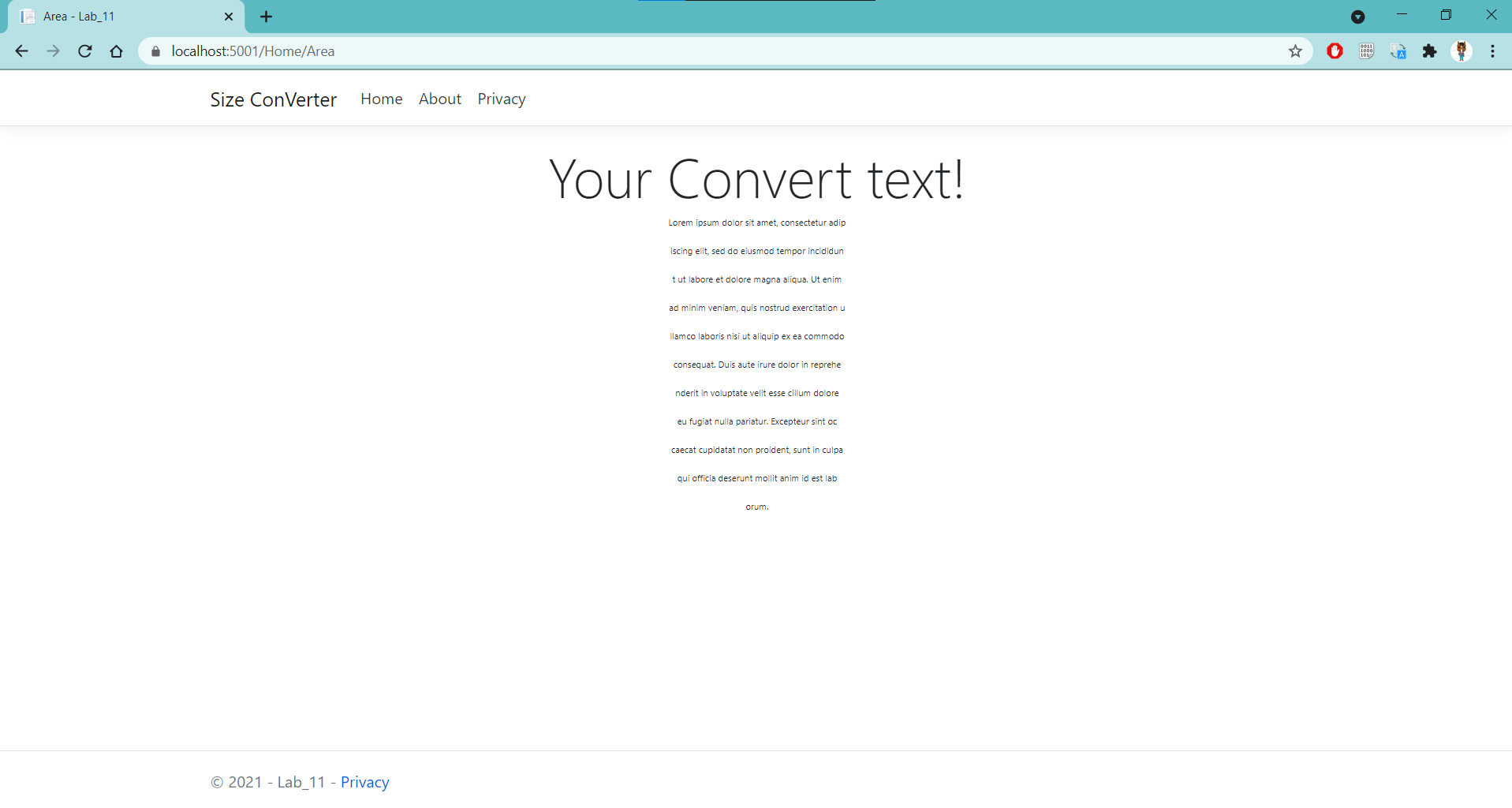


Рисунок 25 - Измененный текст приложения

ЗАДАНИЕ 12

Постановка задачи:

Создайте на основе технологии RMI клиент/серверное приложение:

Учёт рабочего времени. Сервер ведёт учёт времени работы клиентов, данные

сохраняются в файле. Клиент при запуске связывается с сервером и сообщает данные

клиента. Сервер каждые 5 мин запрашивает подтверждение у клиента, что он ещё

подключен. Если клиент не отвечает, он закончил работу.

Особенности реализации:

Сервер ищет активных клиентов.

**public** **Server**() **throws** RemoteException, MalformedURLException, InterruptedException {

LocateRegistry.createRegistry(PORT);

System.setProperty("java.rmi.server.hostname", "localhost");

Naming.rebind("rmi://" + IWorkplace.PREFIX + ":" + PORT + "/" + SERVER\_NAME, **this**);

System.out.println("Server started");

var thread = **new** Thread(() -> {

**try** {

checkClients();

} **catch** (InterruptedException | IOException e) {

e.printStackTrace();

}

});

thread.start();

thread.join();

}

**private** **void** **checkClients**() **throws** IOException, InterruptedException {

**while** (**true**) {

**for** (var client : clients)

client.receiveNewMessage("Введите любое сообщение");

Thread.sleep(**5000**);

**for** (**int** i = **0**; i < clients.size(); ) {

var client = clients.get(i);

**if** (!areWorking.get(client)) {

client.receiveNewMessage("Вы отключены от сервера");

disconnect(client);

} **else** {

areWorking.put(client, **false**);

i++;

}

}

}

}

Регистрация клиента:

**public** **Client**(InputStream inputStream)

**throws** IOException, NotBoundException

{

in = **new** BufferedReader(**new** InputStreamReader(inputStream));

String hostname = "localhost", serverName = "Server";

**boolean** success = **false**;

**while** (!success) {

port = nextPort;

**try** {

LocateRegistry.createRegistry(nextPort);

Naming.rebind(

"rmi://" + IWorkplace.PREFIX +

":" + port + "/" + CLIENT\_NAME + port

, **this**);

} **catch** (ExportException e) { nextPort++; }

**break**;

}

System.out.println("Введите свой никнейм");

**this**.name = in.readLine();

server = (IWorkplace.IChatServer) Naming.lookup(

"rmi://" + hostname + ":" + port + "/" + serverName);

server.connect(**this**);

System.out.println("Подключён к серверу");

}

Результат работы программы:

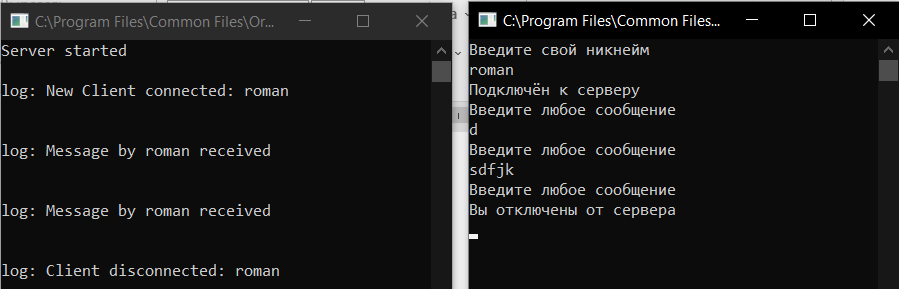


Рисунок 26 - Скриншот работы Сервера и Клиента

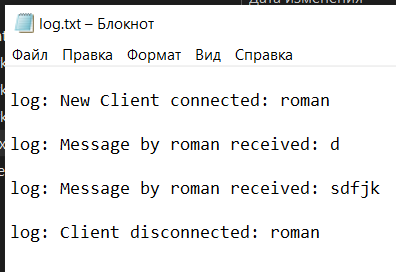


Рисунок 27 - Выходной log.txt файл

Литература

1. Jonathan Knidsen. Java 2D Graphics. O’Reilly
2. Хабибуллин И.Ш. Java 7. – СПб.: БХВ-ПИтербург, 2012
3. Г. Шилдт. Java. Полное руководство, 8-ое издание, М.: OOO И.Д. Вильямс, 2012
4. Кей С. Хорстман. Java2 Основы. Том 1. С.-Питербург. 2007
5. Кей С. Хорстман. Java2 Тонкости программирования Том 2. С.-Питербург. 2007
6. <http://docs.oracle.com/javase/8/docs/>
7. Шкарина Л., Язык SQL: учебный курс. - СПБ.: Питер, 2001. ISBN 5-318-00195-5
8. Марина Грабер. Справочное руководство по SQL. – М.: «Лори», 1887 ISBN 5-85582-022-X
9. Ульма Дж. Базы данных на паскале/ Пер. с англ. – М.: Машиностроение, 1990. ISBN 5-217-00628-5
10. И.Н.Блинов. Java Промышленное программирование. Минск «УниверсалПрес» 2007