



Министерство науки и высшего образования Российской  
Федерации  
Федеральное государственное бюджетное образовательное  
учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ «Информатика и системы  
управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные  
технологии»

**Лабораторная работа № 2**

**Тема** Преобразование изображений: перенос, масштабирование, поворот,  
комбинированные преобразования.

**Студент** Буртелов Н.Н.

**Группа** ИУ7-43Б

**Оценка (баллы)** \_\_\_\_\_

**Преподаватель** Куров А.В.

Москва.  
2020 г.

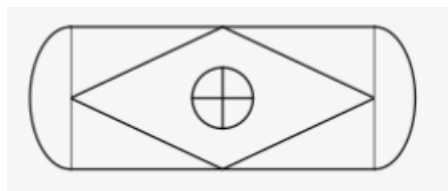
**ЦЕЛЬ РАБОТЫ:** познакомиться с назначением, областями применения, сущностью операций преобразования; научиться разрабатывать программы, осуществляющие преобразования изображений на плоскости.

**РЕЗУЛЬТАТ:** должна быть разработана программа, реализующая операции преобразования. Выбор операции, задание параметров, определяющих каждую операцию, должно выполняться с помощью меню или с помощью системы кнопок и окон для ввода параметров. Должны быть реализованы функции вывода исходного изображения, текущего изображения, возврата к предыдущему изображению. Каждое преобразование должно применяться к текущему изображению.

## Описание условия задачи.

Нарисовать исходный рисунок, осуществить его перенос, масштабирование поворот.

Исходный рисунок:



## Технические характеристики.

Используемый язык программирования для реализации поставленной задачи: Java, версия 12.

## Описание работы алгоритма построения изображения.

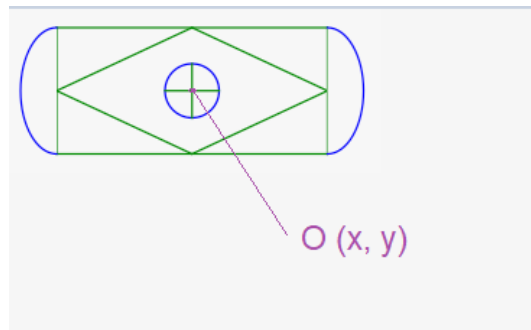
Для того, чтобы вывести изображение исходного рисунка необходимо первым делом выбрать геометрические объекты, из которых состоит рисунок. Для реализации достаточно выбрать объекты: отрезок, дуга эллипса.

Зеленым цветом выделены объекты, которые строятся с помощью отрезков.

Синим цветом выделены объекты, которые строятся с помощью дуги эллипса (дугу окружности можно считать частным случаем дуги эллипса).

За центр изображения принята точка **O**. На рисунке линией указана ее позиция.

Детальная реализация объектов приведена в описании работ подпрограмм.



## **Описание работы программы:**

0. Вывод исходного изображения.

1. Выбор пункта меню.

2. Ввод данных

2.1 Проверка ввода данных

3. Выполнение пункта (или пунктов) меню.

3.1 Если выбран процесс масштабирования.

3.1.1 Произвести масштабирование объектов, из которых состоит изображение.

3.1.2 Сохранение текущих позиций объектов.

3.2 Если выбран процесс переноса.

3.2.1 Произвести перенос объектов, из которых состоит изображение.

3.2.2 Сохранение текущих позиций объектов.

3.3 Если выбран процесс поворота.

3.3.1 Произвести поворот объектов, из которых состоит изображение.

3.3.2 Сохранение текущих позиций объектов.

4. Вывод преобразованного изображения.

### **Входные данные:**

1. Строка, содержащая пару чисел для переноса изображения.
2. Строка, содержащая пару чисел для масштабирования изображения.
3. Строка, содержащая градусную меру для осуществления поворота изображения.

### **Выходные данные:**

1. Графическое представление изображения.

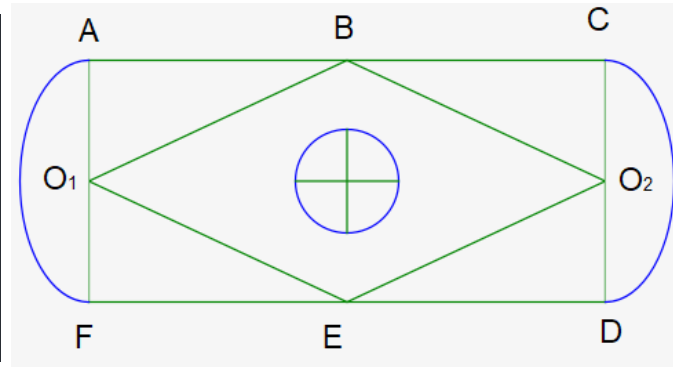
### **Аварийные ситуации**

1. Некорректный ввод строк, содержащих параметры переноса: недопустимые символы, по текущим данным изображение выходит за границы поля (700 \* 600 пикселей).
2. Некорректный ввод строк, содержащих параметры масштабирования: недопустимые символы, по текущим данным изображение выходит за границы поля (700 \* 600 пикселей)
3. Некорректный ввод строк, содержащих параметры поворота: недопустимые символы.

## Код программы.

### Глобальные и начальные параметры:

```
private double[] start = new double[] {350, 270};  
double[] arr = new double[] {275, 235, 425, 305};  
double[] arrArcOne = new double[] {275, 270};  
double[] arrArcTwo = new double[] {425, 270};  
double radMinX = 15;  
double radMinY = 15;
```



**start** – массив координат центра изображения.

**arr** – массив, содержащий координаты точек A, C, D, F; зная координаты этих точек можно найти координаты точек B, E, O<sub>1</sub>, O<sub>2</sub>.

Т.к.  $AB = BC = FE = ED = 75$  единиц.  $AO_1 = O_1F = CO_2 = O_2D = 35$  единиц.

Отсюда **arrArcOne** – массив, содержащий координаты точки O<sub>1</sub>; **arrArcTwo** – массив, содержащий координаты точки O<sub>2</sub>. **radMinX** и **radMinY** – параметры горизонтального и вертикального радиусов соответственно, по ним строятся линии и две дуги в центре рисунка.

## 1. Функция построения отрезка.

Параметры метода: **arrPlace** – массив, содержащий координаты вершин отрезка; **color** – цвет линии.

Возвращаемое значение: **line** (отрезок (объект)).

Для построения отрезка, достаточно знать координаты двух точек - его вершин, после лишь соединить две точки отрезком.

```
Line drawLine(double[] arrPlace){  
    Line line = new Line(arrPlace[0], arrPlace[1], arrPlace[2], arrPlace[3]);  
    line.setStroke(color); //цвет отрезка  
    line.setStrokeWidth(1); //толщина линии  
  
    return line;  
}
```

## 2. Функция перемещения отрезка.

Параметры метода: **arrPlace** – массив, содержащий координаты вершин отрезка; **arrOffset** – массив, содержащий параметры смещения изображения по оси **X** и оси **Y**.

Возвращаемое значение: **line** (отрезок (объект)).

Для того, чтобы переместить объект необходимо к координатам его вершин прибавить параметры смещения. Соответственно к параметрам по оси **X** прибавляются смещения по оси **X**, к параметрам по оси **Y** прибавляются смещения по оси **Y**.

Для переноса точки из позиции с координатами (X,Y) в позицию с координатами (X1,Y1) надо к координате X добавить DX, а к координате Y - DY единиц.

Формулы, по которой приводятся преобразования:

$$X1 = X + DX$$

$$Y1 = Y + DY.$$

Положительное значение DX означает перемещение точки вправо по горизонтали, отрицательное - влево; положительное значение DY - перемещение вниз по вертикали, отрицательное – вверх.

```
Line moveLine(double[] arrPlace, double[] arrOffset) {  
    double dxOne, dyOne;  
  
    dxOne = arrOffset[0];  
    dyOne = arrOffset[1];  
  
    Line line = new Line( v: arrPlace[0] + dxOne, v1: arrPlace[1] + dyOne,  
        v2: arrPlace[2] + dxOne, v3: arrPlace[3] + dyOne);  
  
    line.setStroke(color); //цвет отрезка  
    line.setStrokeWidth(1); //толщина линии  
  
    return line;  
}
```



### 3. Функция масштабирования отрезка.

Параметры метода: **arrPlace** – массив, содержащий координаты вершин отрезка; **size** – массив, содержащий параметры масштабирования по **X** и по **Y**; **centerPlace** – массив, содержащий координаты центра масштабирования.

Масштабирование может быть и неоднородным (коэффициенты вдоль осей абсцисс и ординат различны), в этом случае пропорции рисунка изменяются.

Возвращаемое значение: **line** (отрезок (объект)).

Координаты промасштабированной точки определяются из следующих выражений:

$$\begin{aligned} X1 &= X * KX + (1 - KX) * XM \\ Y1 &= Y * KY + (1 - KY) * YM \end{aligned} \quad (6)$$

где X, Y - координаты исходной точки;

X1, Y1 - координаты промасштабированной точки;

XM, YM - координаты центра масштабирования;

KX, KY - коэффициенты масштабирования.

```
Line scaleLine(double[] arrPlace, double[] size, double[] centerPlace) {  
    double x1, y1, x2, y2;  
  
    x1 = arrPlace[0] * size[0] + (1 - size[0]) * centerPlace[0];  
    x2 = arrPlace[2] * size[0] + (1 - size[0]) * centerPlace[0];  
  
    y1 = arrPlace[1] * size[1] + (1 - size[1]) * centerPlace[1];  
    y2 = arrPlace[3] * size[1] + (1 - size[1]) * centerPlace[1];  
  
    Line line = new Line(x1, y1, x2, y2);  
  
    line.setStroke(color);  
    line.setStrokeWidth(1);  
  
    return line;  
}
```

#### 4. Функция поворота отрезка.

Параметры метода: **arrPlace** – массив, содержащий координаты вершин отрезка; **angle** – угол поворота; **centerPlace** – массив, содержащий координаты центра, относительно которого будет осуществляться поворот.

Возвращаемое значение: **line** (отрезок (объект)).

Для выполнения поворота надо указать величину угла, на который необходимо осуществить поворот, и координаты точки, которая берется за центр вращения. Если исходную точку А с координатами (X,Y) по дуге окружности с центром в точке С с координатами (X<sub>с</sub>,Y<sub>с</sub>) поворачивают на угол t, то координаты (X1,Y1) повернутой точки могут быть записаны в следующем виде:

$$X1 = X_c + (X - X_c) \cdot \cos(t) + (Y - Y_c) \cdot \sin(t)$$

$$Y1 = Y_c + (Y - Y_c) \cdot \cos(t) - (X - X_c) \cdot \sin(t)$$

Угол поворота лежит обычно в пределах от 0 до 360 , другие углы поворота также допустимы, однако поворот при этих углах эквивалентен повороту при углах из указанного диапазона.

```
Line rotateLine(double[] arrPlace, double angle, double[] centerPlace) {
    double x1, y1, x2, y2;

    x1 = centerPlace[0] + (arrPlace[0] - centerPlace[0]) * Math.cos(angle)
        + (arrPlace[1] - centerPlace[1]) * Math.sin(angle);
    x2 = centerPlace[0] + (arrPlace[2] - centerPlace[0]) * Math.cos(angle)
        + (arrPlace[3] - centerPlace[1]) * Math.sin(angle);
    y1 = centerPlace[1] + (arrPlace[1] - centerPlace[1]) * Math.cos(angle)
        + (arrPlace[0] - centerPlace[1]) * Math.sin(angle);
    y2 = centerPlace[1] + (arrPlace[3] - centerPlace[1]) * Math.cos(angle)
        + (arrPlace[2] - centerPlace[1]) * Math.sin(angle);

    Line line = new Line(x1, y1, x2, y2);

    line.setStroke(color);
    line.setStrokeWidth(1);

    return line;
}
```

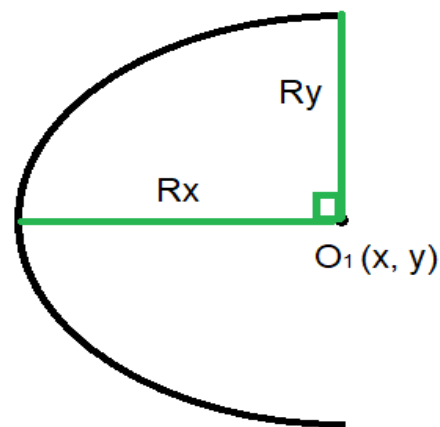
## 5. Функция построения дуги эллипса.

Параметры метода: **dataArr** – массив, содержащий координаты центра эллипса, относительно от которого будет строиться дуга; **angle** – угол, от которого будет строиться дуга; **angleEnd** – угол, до которого будет строиться дуга; **radArr** – массив, содержащий параметры горизонтального и вертикального радиусов.

Т.е. для построения подобной дуги эллипса достаточно знать координаты центра  $O_1(x, y)$ , горизонтальный радиус **Rx**, вертикальный радиус **Ry**.

В подпрограмме **dataArr[0]** – **x**, **dataArr[1]** – **y**; **radArr[0]** – **Rx**, **radArr[1]** – **Ry**.

Для построения дуги, изображенной на рисунке **angle** = 90 градусов, **angleEnd** = 180 градусов. **angleEnd** – это такой угол, который отсчитывается от **angle**, т.е. **angleEnd** разворачивает дугу в 180 градусов.



Возвращаемое значение: **arc** (дуга (объект)).

```
Arc drawArc(double[] dataArr, double angle, double angleEnd, double[] radArr) {  
    Arc arc = new Arc();  
  
    arc.setCenterX(dataArr[0]); // координаты X центра  
    arc.setCenterY(dataArr[1]); // координаты Y центра  
    arc.setRadiusX(radArr[0]); // горизонтальный радиус  
    arc.setRadiusY(radArr[1]); // вертикальный радиус  
    arc.setStartAngle(angle); // начальный угол  
    arc.setLength(angleEnd); //конечный угол  
  
    return arc;  
}
```

## 6. Функция перемещения дуги эллипса.

Параметры метода: **dataArr** – массив, содержащий координаты центра эллипса, относительно от которого будет строиться дуга; **angle** – угол, от которого будет строиться дуга; **angleEnd** – угол, до которого будет строиться дуга; **radArr** – массив, содержащий параметры горизонтального и вертикального радиусов; **moveArr** - массив, содержащий параметры смещения изображения по оси **X** и оси **Y**;

Для переноса дуги достаточно переместить центр построения этой дуги. Для переноса точки из позиции с координатами (X,Y) в позицию с координатами (X1,Y1) надо к координате X добавить DX, а к координате Y - DY единиц.

Формула расчета новых координат точки:

$$X1 = X + DX, Y1 = Y + DY.$$

В программе: **moveArr[0]** – DX, **moveArr[1]** – DY;

Возвращаемое значение: **pane** (панель компоновки).

```
Pane moveArc(Pane pane, double[] dataArr, double angle, double angleEnd, double[] radArr, double[] moveArr) {
    dataArr[0] += moveArr[0]; // меняем X центра
    dataArr[1] += moveArr[1]; // меняем Y центра
    this.arrStart[0] += moveArr[0];
    this.arrStart[1] += moveArr[1];

    pane.getChildren().add(drawArc(dataArr, angle, angleEnd, radArr)); // строим дугу эллипса

    return pane;
}
```

## 7. Функция масштабирования дуги эллипса.

Параметры метода: **dataArr** – массив, содержащий координаты центра эллипса, относительно от которого будет строиться дуга; **angle** – угол, от которого будет строиться дуга; **angleEnd** – угол, до которого будет строиться дуга; **radArr** – массив, содержащий параметры горизонтального и вертикального радиусов; **sizeArr** - массив, содержащий параметры масштабирования изображения по оси **X** и оси **Y**.

Координаты промасштабированной точки определяются из следующих выражений:

$$X1 = X * KX + (1 - KX) * XM$$

$$Y1 = Y * KY + (1 - KY) * YM$$

где  $X, Y$  - координаты исходной точки;

$X1, Y1$  - координаты промасштабированной точки;

$XM, YM$  - координаты центра масштабирования;

$KX, KY$  - коэффициенты масштабирования.

В программе: **sizeArr[0]** –  $KX$ , **sizeArr[1]** –  $KY$ ; **arrStart[0]** –  $XM$ , **arrStart[1]** –  $YM$ ;

Для масштабирования дуги достаточно переместить центр построения этой дуги и изменить параметры горизонтального и вертикального радиусов.

Возвращаемое значение: **pane** (панель компоновки).

```
Pane scaleArc(Pane pane, double[] dataArr, double angle, double angleEnd, double[] radArr, double[] sizeArr)
{
    radArr[0] *= sizeArr[0]; //rx
    radArr[1] *= sizeArr[1]; //ry
    dataArr[0] = dataArr[0] * sizeArr[0] + (1 - sizeArr[0]) * arrStart[0]; // X координата центра дуги
    dataArr[1] = dataArr[1] * sizeArr[1] + (1 - sizeArr[1]) * arrStart[1]; // Y координата центра дуги

    pane.getChildren().add(drawArc(dataArr, angle, angleEnd, radArr));

    return pane;
}
```

## 8. Функция поворота дуги эллипса.

Параметры метода: **dataArr** – массив, содержащий координаты центра эллипса, относительно от которого будет строиться дуга; **angle** – угол, от которого будет строиться дуга; **angleEnd** – угол, до которого будет строиться дуга; **radArr** – массив, содержащий параметры горизонтального и вертикального радиусов; **rotAngle** – угол поворота.

Если исходную точку А с координатами (X,Y) по дуге окружности с центром в точке С с координатами (X<sub>с</sub>,Y<sub>с</sub>) поворачивают на угол t, то координаты (X1,Y1)

повернутой точки могут быть записаны в следующем виде:

$$X1 = X_c + (X - X_c) * \cos t + (Y - Y_c) * \sin t$$

$$Y1 = Y_c + (Y - Y_c) * \cos t - (X - X_c) * \sin t$$

Для поворота дуги достаточно переместить центр построения этой дуги и изменить параметры стартового угла, от которого начнется построение дуги.

Возвращаемое значение: pane (панель компоновки).

```
Pane rotateArc(Pane pane, double[] dataArr, double angle, double angleEnd, double[] radArr, double rotAngle) {
    angle += rotAngle;
    double x1, y1;

    x1 = arrStart[0] + (dataArr[0] - arrStart[0]) * Math.cos(angle)
        + (dataArr[1] - dataArr[1]) * Math.sin(angle);

    y1 = arrStart[1] + (dataArr[1] - arrStart[1]) * Math.cos(angle)
        + (dataArr[0] - arrStart[1]) * Math.sin(angle);

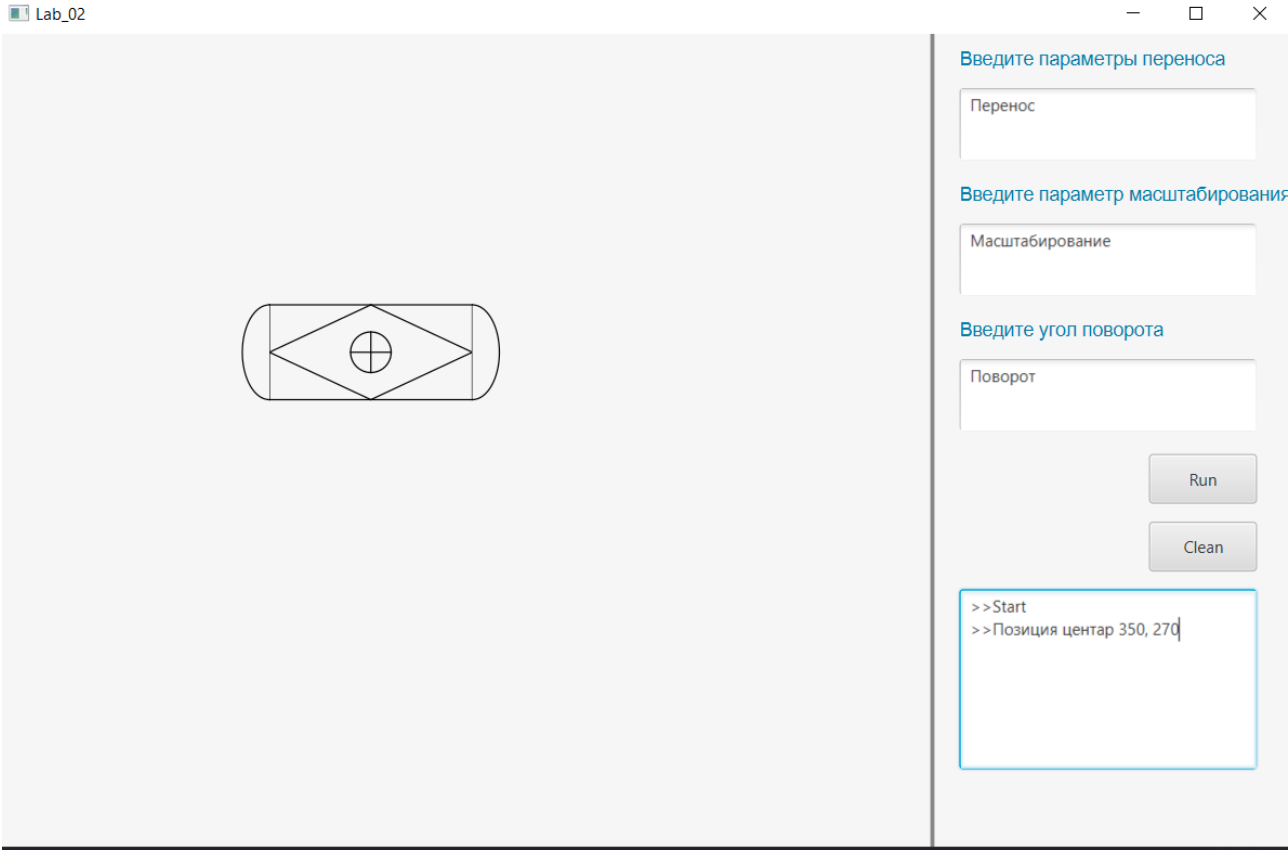
    dataArr[0] = x1;
    dataArr[1] = y1;

    pane.getChildren().add(drawArc(dataArr, angle, angleEnd, radArr));

    return pane;
}
```

# Пример работы программы:

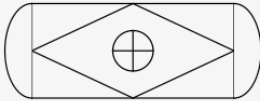
## Начальный интерфейс программы:



Пример 1.

Перенос изображения

Lab\_02



Введите параметры переноса

150, 150

Введите параметр масштабирования

Масштабирование

Введите угол поворота

Поворот

Run

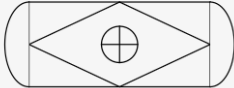
Clean

>>Start  
>>Позиция центр 350, 270  
>>  
>>Перенос  
>>по x +150, по y +150

Пример 2.

Перенос изображения

Lab\_02



Введите параметры переноса

0, 50

Введите параметр масштабирования

Масштабирование

Введите угол поворота

Поворот

Run

Clean

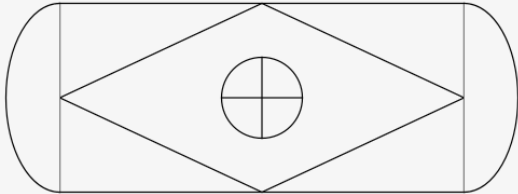
>>Start  
>>Позиция центра 350, 270  
>>  
>>Перенос  
>>по x +150, по y +150  
>>  
>>Перенос  
>>по x +0, по y +50



Пример 3.

Масштабирование изображения

Lab\_02



Введите параметры переноса

Перенос

Введите параметр масштабирования

2

Введите угол поворота

Поворот

Run

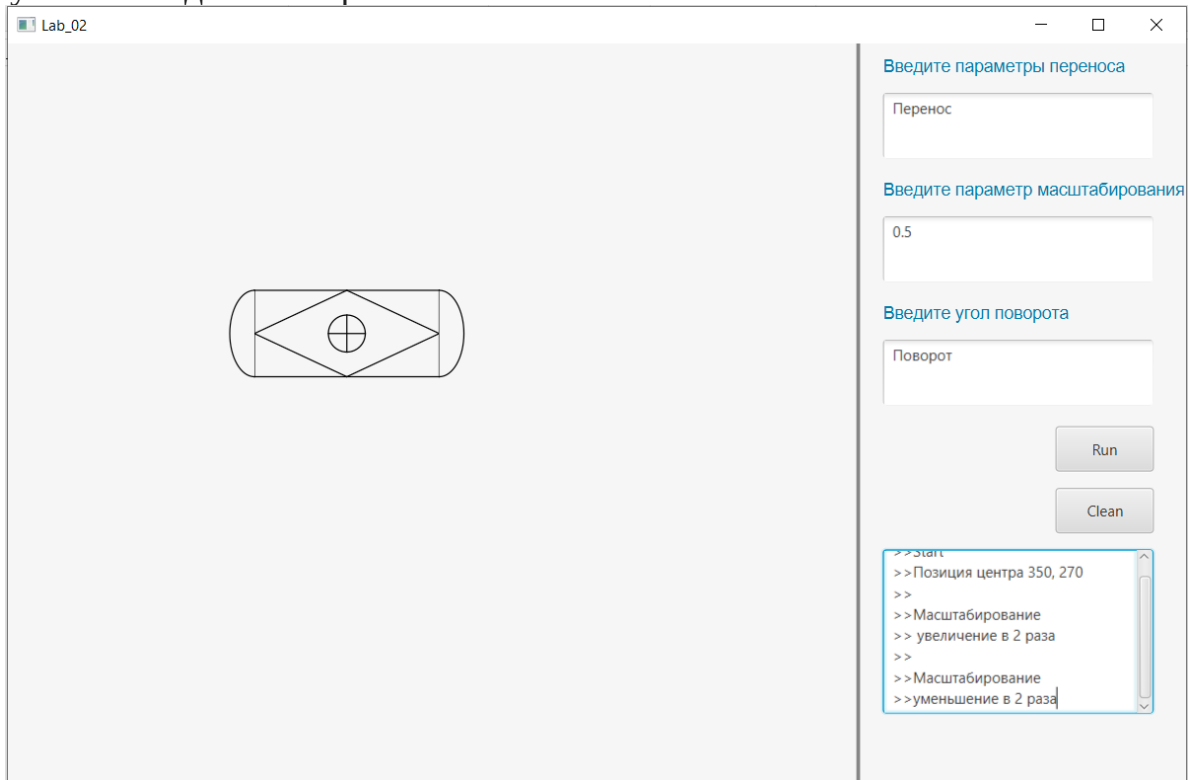
Clean

>>Start  
>>Позиция центра 350, 270  
>>  
>>Масштабирование  
>>увеличение в 2 раза

## Пример 4.

### Масштабирование изображения.

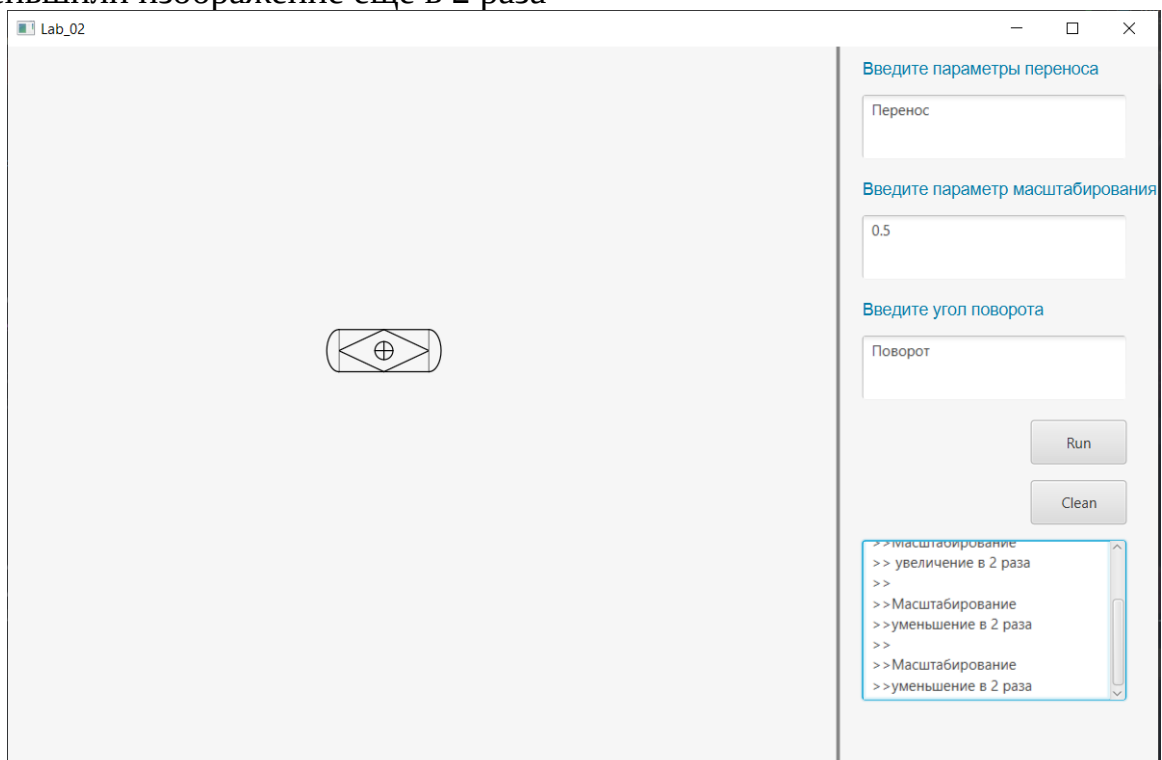
Увеличенное изначально изображение в 2 раза мы уменьшаем в 2 раза и получаем исходное изображение.



## Пример 5.

### Масштабирование изображения.


Уменьшили изображение еще в 2 раза



Пример 6.

Поворот изображения

Lab\_02



Введите параметры переноса

Перенос

Введите параметр масштабирования

Масштабирование

Введите угол поворота

90

Run

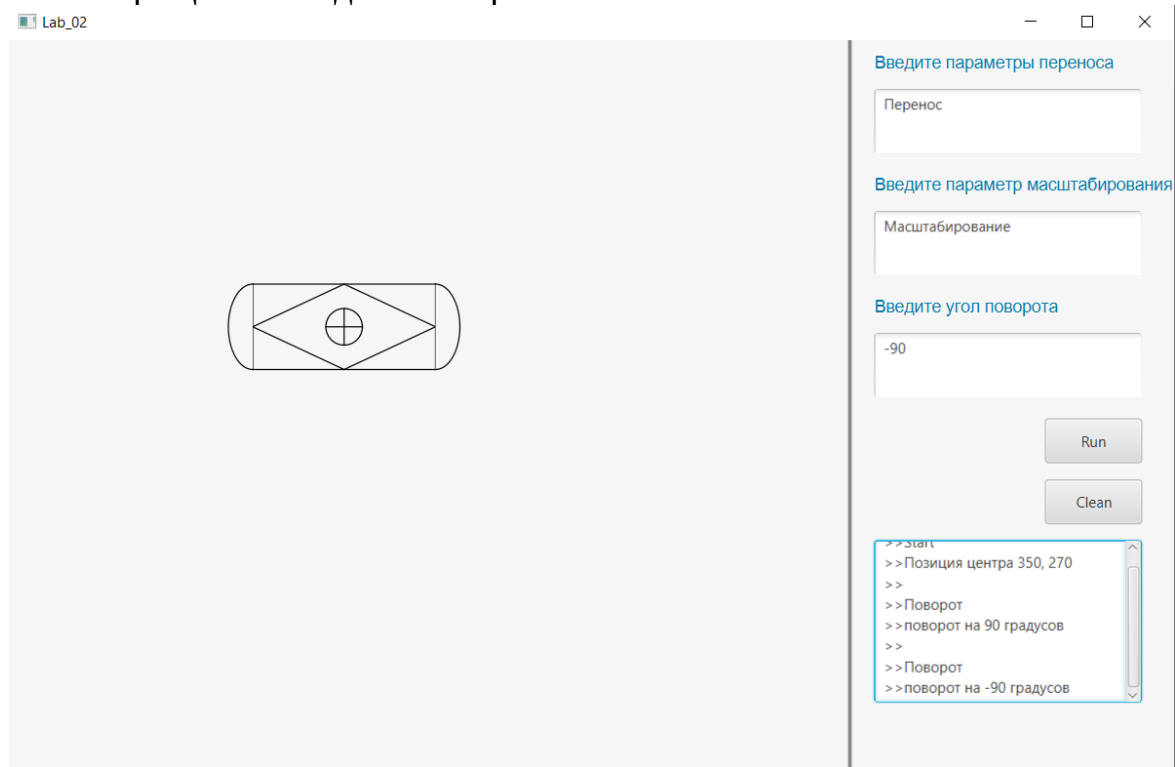
Clean

>>Start  
>>Позиция центра 350, 270  
>>  
>>Поворот  
>>поворот на 90 градусов

## Пример 7.

### Поворот изображения.

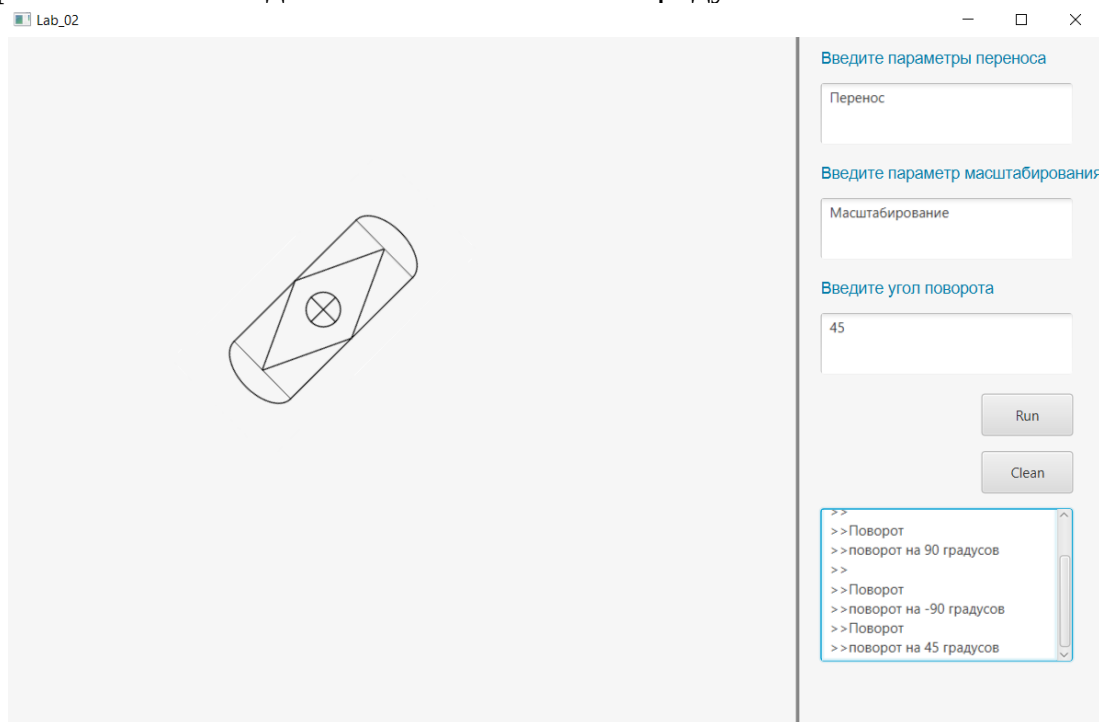
Поворачиваем изображение на 90 градусов в противоположенную сторону. Тем самым возвращаем исходное изображение.



## Пример 8.

### Поворот изображения

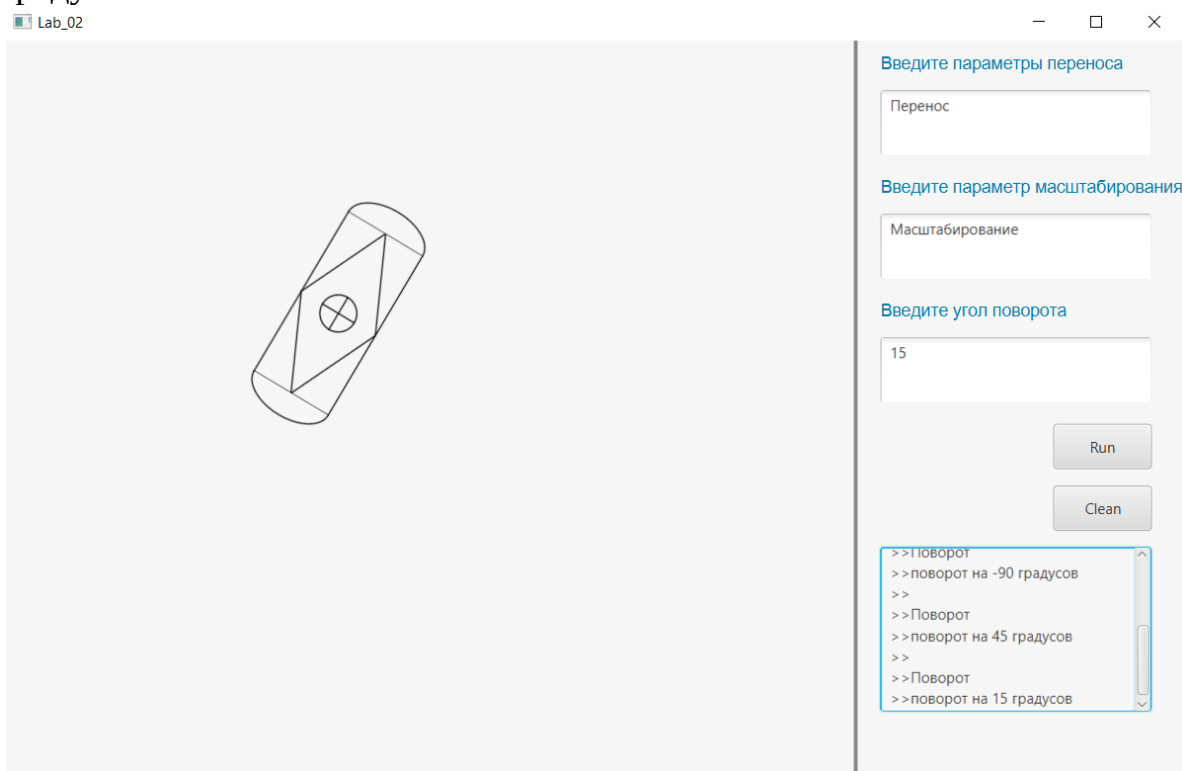
Поворачиваем из исходного положения на 45 градусов.



## Пример 9.

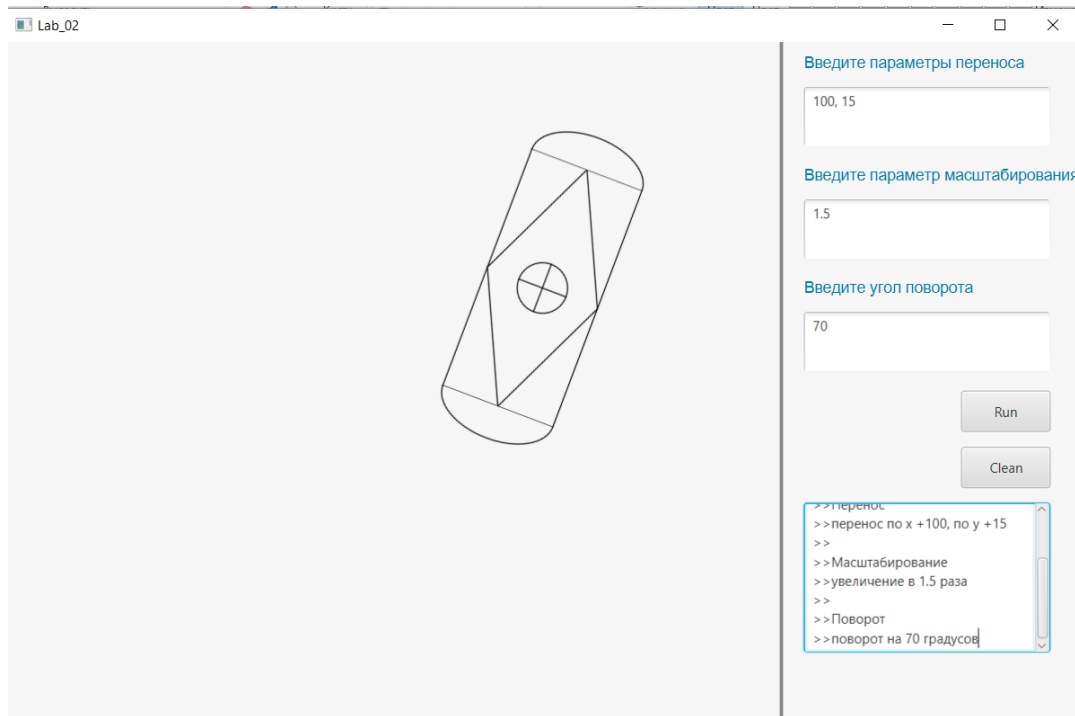
### Поворот изображения

Еще поворачиваем изображение на 15 градусов, это равносильно повороту на 60 градусов из начального положения.



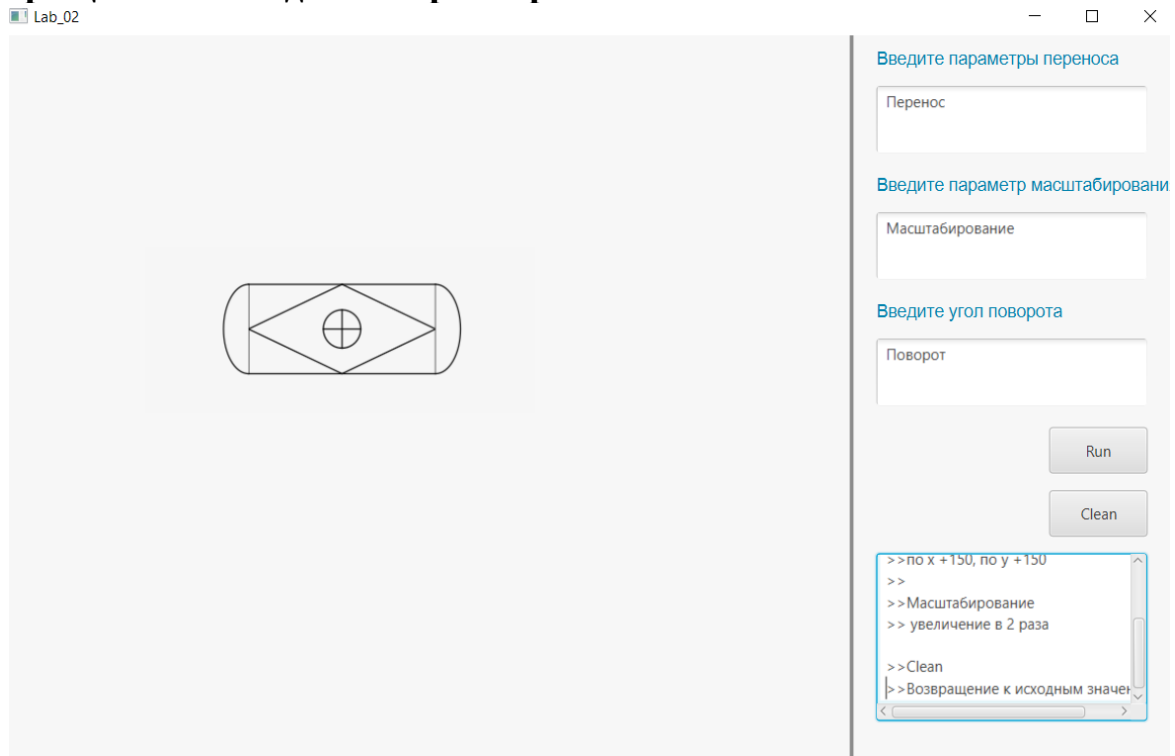
## Пример 10.

### Комбинированные преобразования.



## Пример 11.

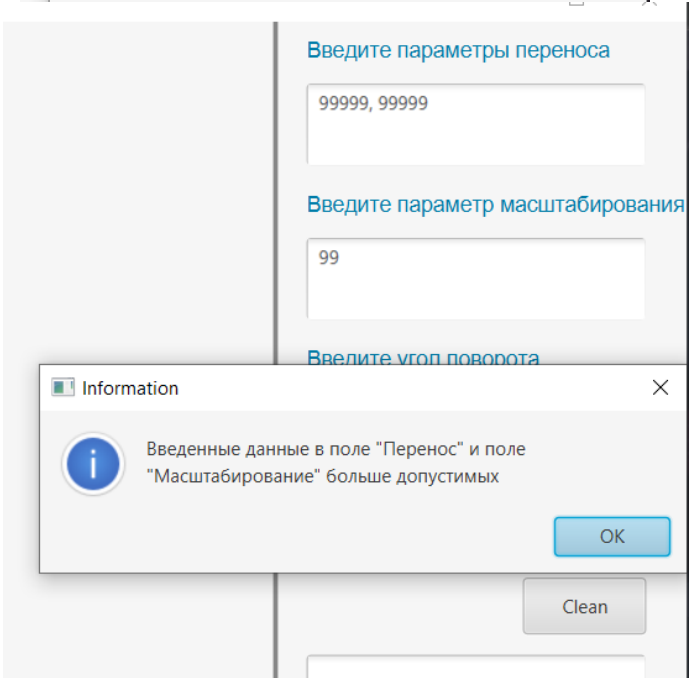
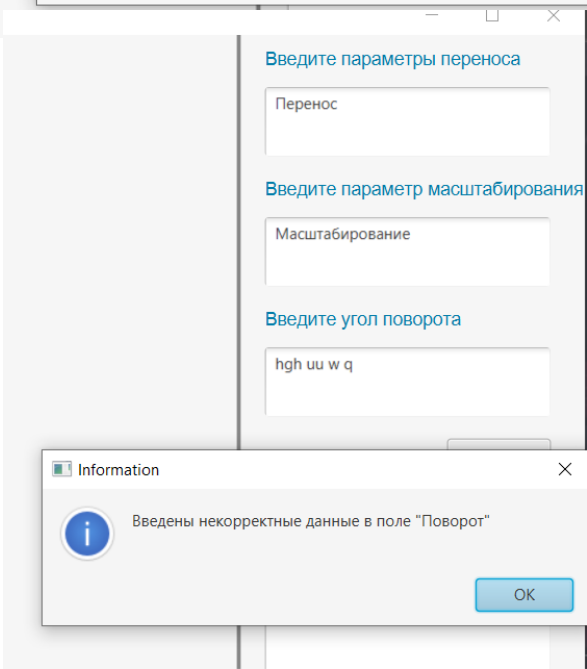
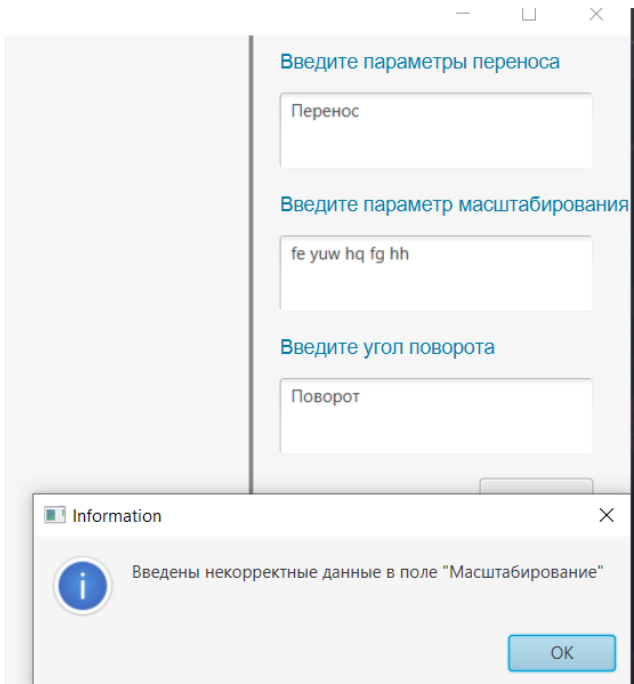
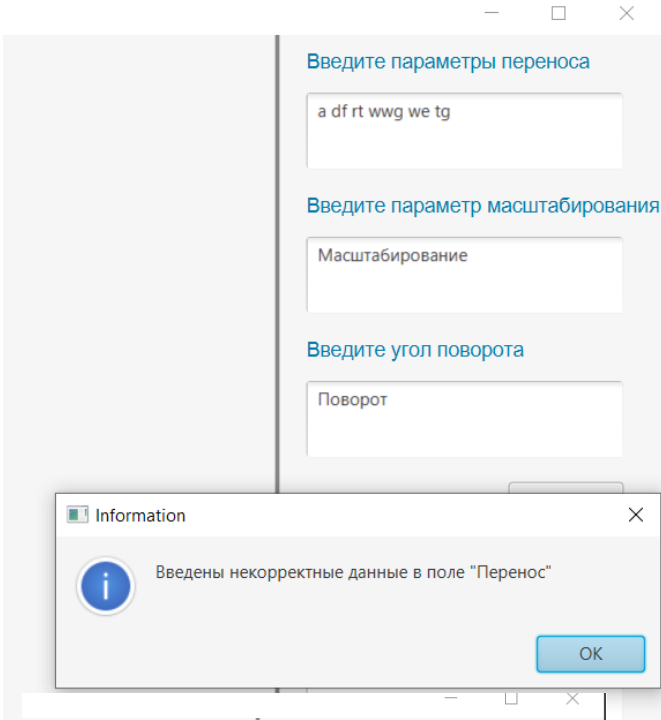
### Возвращение к исходным параметрам



При нажатии на кнопку **Clean** изображение становится исходным.

Пример 12, 13, 14, 15

Аварийные ситуации.



## **Вывод.**

В результате проделанной работы познакомился с назначением, областями применения, сущностью операций преобразования; научился разрабатывать программы, осуществляющие преобразования изображений на плоскости.