МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ

КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Знамени федеральное государственное

Бюджетное образовательное учреждение высшего образования

МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И

ИНФОРМАТИКИ

Кафедра математической кибернетики и информационных технологий

Лабораторная работа № 2 по теме:

«Основы объектно-ориентированного программирования»

Выполнил: студент группы БИБ1902 \_\_\_\_\_\_\_Арустамян А.Б.

Проверил: \_\_\_\_\_\_\_\_Херсонский А. В.

Москва 2021

**Цель работы:**

В данной лабораторной работе необходимо использовать классы по одному на файл, чтобы описать, как эти объекты работают.

**Задачи:**

1. Создайте новый класс Point3d для представления точек в трехмерном Евклидовом пространстве.
2. Добавьте новый метод distanceTo, который в качестве параметра принимает другой объект Point3d, вычисляет расстояние между двумя точками с точность двух знаков после запятой и возвращает полученное значение.
3. Создайте другой класс под названием Lab1, который будет создавать три точки в 3 пространстве и считать площадь треугольника по ним.
4. Скомпилируйте оба исходных файла вместе.

**Ход работы:**

**Class Main**

package com.company;  
  
import javax.swing.\*;  
  
public class Main {  
  
 public static void main(String[] args) {  
 GUI app = new GUI();  
 app.setVisible(true);  
  
 }  
}

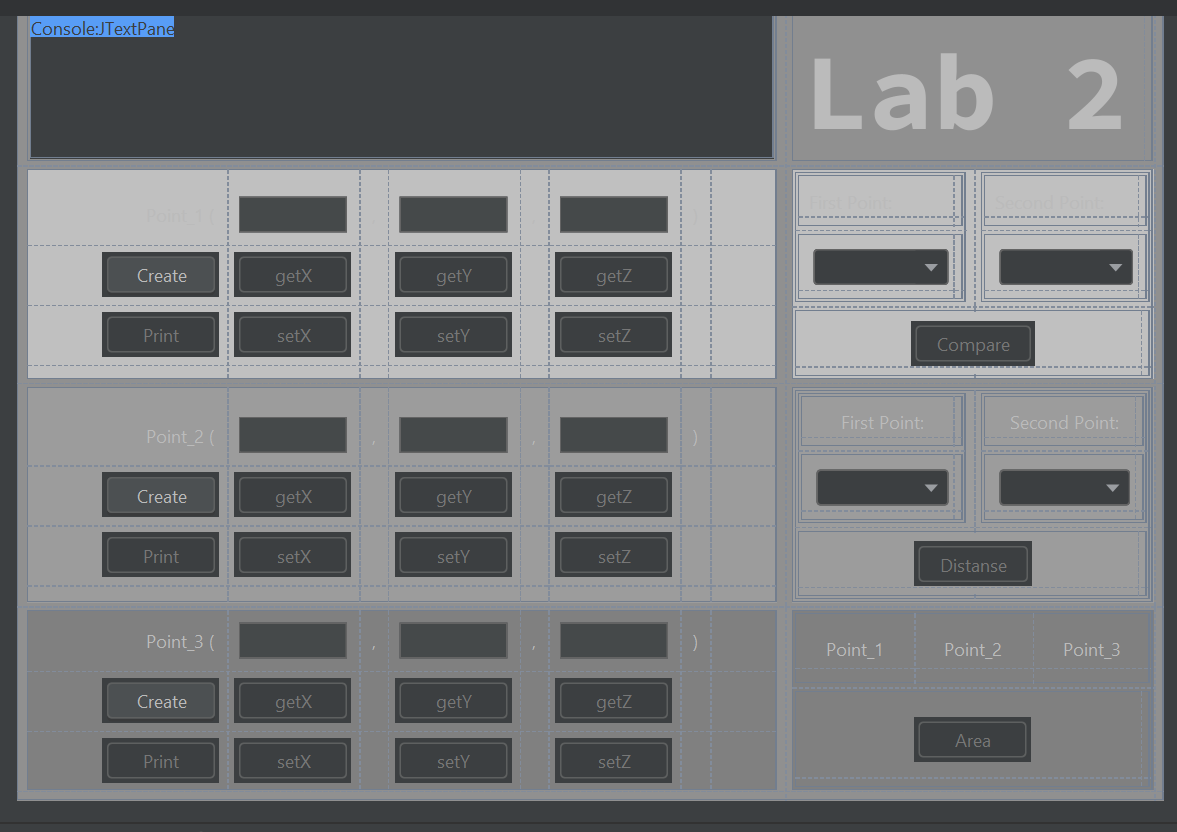
**Class Point3D**

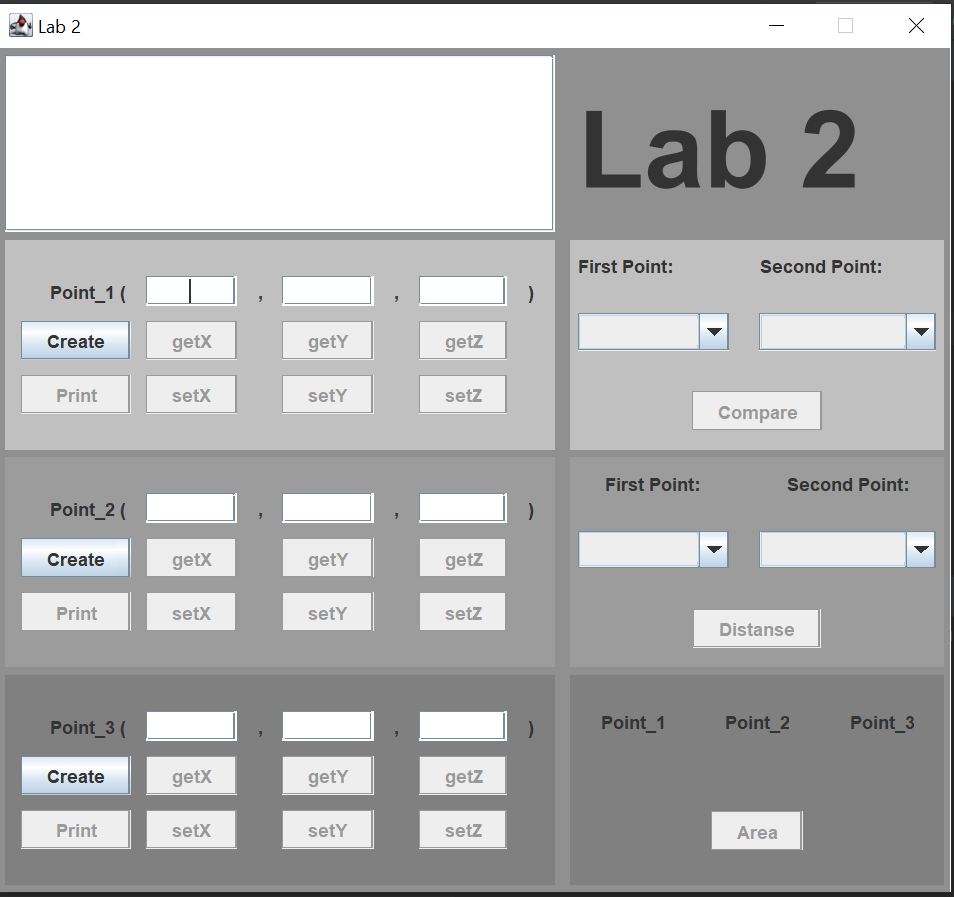
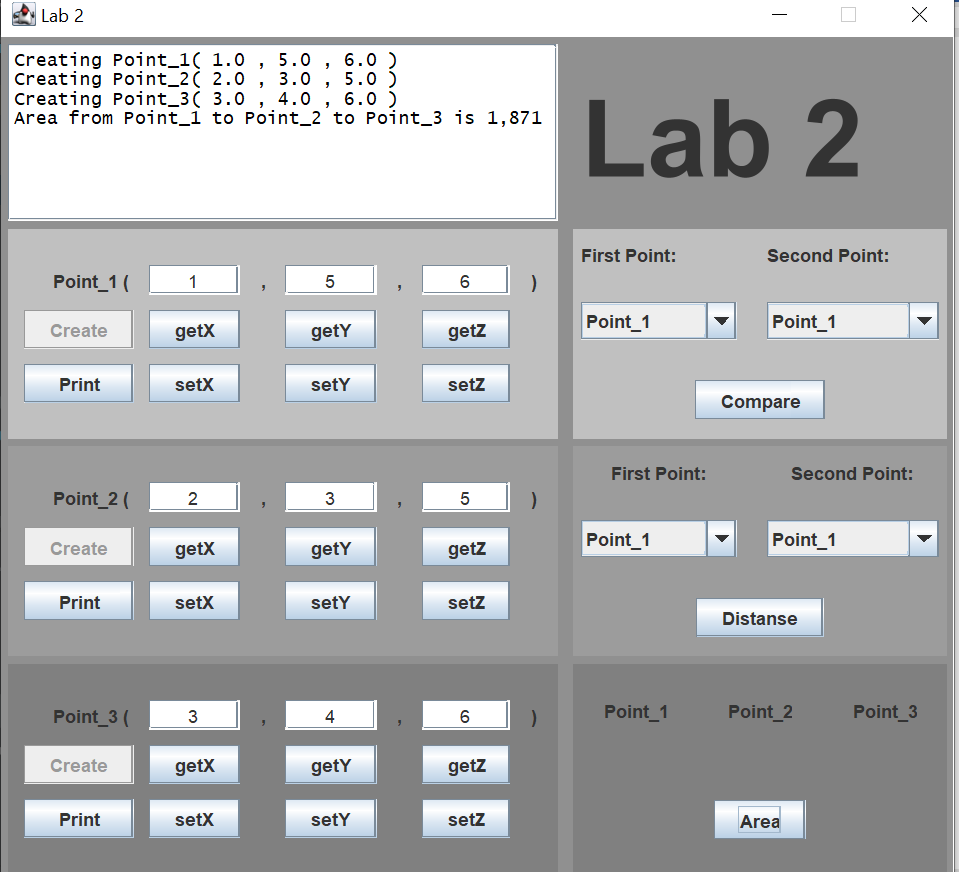
package com.company;  
  
import java.lang.Math;  
  
public class Point3D {  
 private double xCoord;  
 private double yCoord;  
 private double zCoord;  
  
 public Point3D ( double x, double y, double z) {  
 xCoord = x;  
 yCoord = y;  
 zCoord = z;  
 }  
  
 public Point3D () {  
 this(0,0,0);  
 }  
  
 public double getX () {  
 return xCoord;  
 }  
  
 public double getY () {  
 return yCoord;  
 }  
  
 public double getZ () {  
 return zCoord;  
 }  
  
 public void setX ( double val) {  
 xCoord = val;  
 }  
  
 public void setY ( double val) {  
 yCoord = val;  
 }  
  
 public void setZ ( double val) {  
 zCoord = val;  
 }  
  
 public boolean compare (Point3D secondPoint){  
 if ((this.getX() == secondPoint.getX())&&(this.getY() == secondPoint.getY())&&(this.getZ() == secondPoint.getZ()))  
 return true;  
 else  
 return false;  
 }  
  
 public String distanceTo (Point3D secondPoint){  
 double distance = Math.*sqrt*((Math.*pow*((this.getX() - secondPoint.getX()),2) + Math.*pow*((this.getY() - secondPoint.getY()),2) + Math.*pow*((this.getZ() - secondPoint.getZ()),2)));  
 return String.*format*("%.3f", distance);  
 }  
  
 public static String area (Point3D firstPoint, Point3D secondPoint, Point3D thirdPoint){  
 double P = (Double.*parseDouble*(firstPoint.distanceTo(secondPoint).replace(",", "."))+Double.*parseDouble*(secondPoint.distanceTo(thirdPoint).replace(",", "."))+Double.*parseDouble*(thirdPoint.distanceTo(firstPoint).replace(",", "."))) / 2;  
 double area = Math.*sqrt*(P\*(P - Double.*parseDouble*(firstPoint.distanceTo(secondPoint).replace(",", ".")))\*(P - Double.*parseDouble*(secondPoint.distanceTo(thirdPoint).replace(",", ".")))\*(P - Double.*parseDouble*(thirdPoint.distanceTo(firstPoint).replace(",", "."))));  
 return String.*format*("%.3f", area);  
 }

**Class GUI**

public class GUI extends JFrame {  
 private JPanel rootPanel;  
 private JTextField P\_1\_X;  
 private JTextField P\_1\_Y;  
 private JTextField P\_1\_Z;  
 private JButton P\_1\_Create;  
 private JButton P\_1\_getX;  
 private JButton P\_1\_Print;  
 private JButton P\_1\_SetX;  
 private JButton P\_1\_getY;  
 private JButton P\_1\_SetY;  
 private JButton P\_1\_getZ;  
 private JButton P\_1\_SetZ;  
 private JComboBox C\_M\_1;  
 private JComboBox C\_M\_2;  
 private JButton compareButton;  
 private JButton areaButton;  
 private JPanel main;  
 private JPanel ConsolePanel;  
 private JPanel Point1Panel;  
 private JPanel TitlePanel;  
 private JPanel ComparePanel;  
 private JPanel DistansePanel;  
 private JPanel AreaPanel;  
 private JPanel Point2Panel;  
 private JPanel Point3Panel;  
 private JLabel P\_1\_L\_1;  
 private JLabel P\_1\_L\_2;  
 private JLabel P\_1\_L\_3;  
 private JLabel P\_1\_L\_4;  
 private JLabel TitleLable;  
 private JPanel CompareButtonPanel;  
 private JPanel ComparePoint\_1\_Panel;  
 private JPanel ComparePoint\_2\_Panel;  
 private JPanel C\_L\_1\_P;  
 private JPanel C\_M\_1\_P;  
 private JLabel C\_L\_1;  
 private JPanel C\_L\_2\_P;  
 private JPanel C\_M\_2\_P;  
 private JLabel C\_L\_2;  
 private JPanel D\_Panel;  
 private JPanel D\_B\_Panel;  
 private JPanel D\_P\_1\_Panel;  
 private JPanel D\_P\_2\_Panel;  
 private JButton DistanseButton;  
 private JLabel F\_P\_L;  
 private JPanel F\_P\_L\_Panel;  
 private JPanel F\_P\_M\_Panel;  
 private JComboBox F\_P\_M;  
 private JPanel S\_P\_L\_Panel;  
 private JLabel S\_P\_L;  
 private JPanel S\_P\_M\_Panel;  
 private JComboBox S\_P\_M;  
 private JPanel AreaButtonPanel;  
 private JPanel AreaLablePanel;  
 private JLabel A\_P\_1\_L;  
 private JLabel A\_P\_2\_L;  
 private JLabel A\_P\_3\_L;  
 private JLabel P\_2\_L\_1;  
 private JLabel P\_2\_L\_2;  
 private JLabel P\_2\_L\_3;  
 private JLabel P\_2\_L\_4;  
 private JTextField P\_2\_X;  
 private JTextField P\_2\_Y;  
 private JTextField P\_2\_Z;  
 private JButton P\_2\_Create;  
 private JButton P\_2\_getX;  
 private JButton P\_2\_getY;  
 private JButton P\_2\_getZ;  
 private JButton P\_2\_Print;  
 private JButton P\_2\_SetX;  
 private JButton P\_2\_SetY;  
 private JButton P\_2\_SetZ;  
 private JLabel P\_3\_L\_1;  
 private JLabel P\_3\_L\_2;  
 private JLabel P\_3\_L\_3;  
 private JLabel P\_3\_L\_4;  
 private JTextField P\_3\_X;  
 private JTextField P\_3\_Y;  
 private JTextField P\_3\_Z;  
 private JButton P\_3\_Create;  
 private JButton P\_3\_getX;  
 private JButton P\_3\_getY;  
 private JButton P\_3\_getZ;  
 private JButton P\_3\_SetZ;  
 private JButton P\_3\_SetY;  
 private JButton P\_3\_SetX;  
 private JButton P\_3\_Print;  
 private JScrollPane ConsoleScroll;  
 private JTextPane Console;  
 private JPanel LabelPanel;  
 private int pointCounter = 0;  
 Point3D firstPoint;  
 Point3D secondPoint;  
 Point3D thirdPoint;  
  
 public GUI () {  
 add(rootPanel);  
 setTitle("Lab 2");  
 setSize(650,600);  
 setResizable(false);  
 P\_1\_X.setHorizontalAlignment(JTextField.*CENTER*);  
 P\_1\_Y.setHorizontalAlignment(JTextField.*CENTER*);  
 P\_1\_Z.setHorizontalAlignment(JTextField.*CENTER*);  
 P\_2\_X.setHorizontalAlignment(JTextField.*CENTER*);  
 P\_2\_Y.setHorizontalAlignment(JTextField.*CENTER*);  
 P\_2\_Z.setHorizontalAlignment(JTextField.*CENTER*);  
 P\_3\_X.setHorizontalAlignment(JTextField.*CENTER*);  
 P\_3\_Y.setHorizontalAlignment(JTextField.*CENTER*);  
 P\_3\_Z.setHorizontalAlignment(JTextField.*CENTER*);  
  
 P\_1\_Create.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (!(P\_1\_X.getText().trim().isEmpty()) && !(P\_1\_Y.getText().trim().isEmpty()) && !(P\_1\_Z.getText().trim().isEmpty())){  
 firstPoint = new Point3D(Double.*parseDouble*(P\_1\_X.getText()),Double.*parseDouble*(P\_1\_Y.getText()),Double.*parseDouble*(P\_1\_Z.getText()));  
 appendToPane(Console,"Creating Point\_1( "+firstPoint.getX()+" , "+firstPoint.getY()+" , "+firstPoint.getZ()+" )\n", Color.*BLACK*);  
 P\_1\_Create.setEnabled(false);  
 P\_1\_Print.setEnabled(true);  
 P\_1\_getX.setEnabled(true);  
 P\_1\_getY.setEnabled(true);  
 P\_1\_getZ.setEnabled(true);  
 P\_1\_SetX.setEnabled(true);  
 P\_1\_SetY.setEnabled(true);  
 P\_1\_SetZ.setEnabled(true);  
 pointCounter++;  
 }  
 else {  
 firstPoint = new Point3D();  
 appendToPane(Console,"Creating Point\_1( "+firstPoint.getX()+" , "+firstPoint.getY()+" , "+firstPoint.getZ()+" )\n", Color.*BLACK*);  
 P\_1\_Create.setEnabled(false);  
 P\_1\_Print.setEnabled(true);  
 P\_1\_getX.setEnabled(true);  
 P\_1\_getY.setEnabled(true);  
 P\_1\_getZ.setEnabled(true);  
 P\_1\_SetX.setEnabled(true);  
 P\_1\_SetY.setEnabled(true);  
 P\_1\_SetZ.setEnabled(true);  
 pointCounter++;  
 }  
 if (pointCounter == 2){  
 compareButton.setEnabled(true);  
 DistanseButton.setEnabled(true);  
 }  
 if (pointCounter == 3){  
 areaButton.setEnabled(true);  
 }  
 C\_M\_1.addItem("Point\_1");  
 C\_M\_2.addItem("Point\_1");  
 F\_P\_M.addItem("Point\_1");  
 S\_P\_M.addItem("Point\_1");  
 }  
 });  
 P\_1\_Print.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 appendToPane(Console,"Displaying Point\_1( "+firstPoint.getX()+" , "+firstPoint.getY()+" , "+firstPoint.getZ()+" )\n", Color.*BLACK*);  
 }  
 });  
  
 P\_1\_getX.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 appendToPane(Console,"Coordinate X of Point\_1 is: "+firstPoint.getX()+"\n", Color.*BLACK*);  
 }  
 });  
  
 P\_1\_getY.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 appendToPane(Console,"Coordinate Y of Point\_1 is: "+firstPoint.getY()+"\n", Color.*BLACK*);  
 }  
 });  
  
 P\_1\_getZ.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 appendToPane(Console,"Coordinate Z of Point\_1 is: "+firstPoint.getZ()+"\n", Color.*BLACK*);  
 }  
 });  
  
 P\_1\_SetX.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (!(P\_1\_X.getText().trim().isEmpty())) {  
 if (firstPoint.getX() != Double.*parseDouble*(P\_1\_X.getText())){  
 firstPoint.setX(Double.*parseDouble*(P\_1\_X.getText()));  
 appendToPane(Console,"Coordinate X of Point\_1 changed to: "+firstPoint.getX()+"\n", Color.*BLACK*);  
 }  
 else {  
 appendToPane(Console,"Сhanges are unnecessary!\n", Color.*BLACK*);  
 }  
 }  
 else {  
 appendToPane(Console,"Enter X coordinate!\n", Color.*RED*);  
 }  
 }  
 });  
  
 P\_1\_SetY.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (!(P\_1\_Y.getText().trim().isEmpty())) {  
 if (firstPoint.getY() != Double.*parseDouble*(P\_1\_Y.getText())){  
 firstPoint.setY(Double.*parseDouble*(P\_1\_Y.getText()));  
 appendToPane(Console,"Coordinate Y of Point\_1 changed to: "+firstPoint.getY()+"\n", Color.*BLACK*);  
 }  
 else {  
 appendToPane(Console,"Сhanges are unnecessary!\n", Color.*BLACK*);  
 }  
 }  
 else {  
 appendToPane(Console,"Enter Y coordinate!\n", Color.*RED*);  
 }  
 }  
 });  
  
 P\_1\_SetZ.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (!(P\_1\_Z.getText().trim().isEmpty())) {  
 if (firstPoint.getZ() != Double.*parseDouble*(P\_1\_Z.getText())){  
 firstPoint.setZ(Double.*parseDouble*(P\_1\_Z.getText()));  
 appendToPane(Console,"Coordinate Z of Point\_1 changed to: "+firstPoint.getZ()+"\n", Color.*BLACK*);  
 }  
 else {  
 appendToPane(Console,"Сhanges are unnecessary!\n", Color.*BLACK*);  
 }  
 }  
 else {  
 appendToPane(Console,"Enter Z coordinate!\n", Color.*RED*);  
 }  
 }  
 });  
  
 P\_2\_Create.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (!(P\_2\_X.getText().trim().isEmpty()) && !(P\_2\_Y.getText().trim().isEmpty()) && !(P\_2\_Z.getText().trim().isEmpty())){  
 secondPoint = new Point3D(Double.*parseDouble*(P\_2\_X.getText()),Double.*parseDouble*(P\_2\_Y.getText()),Double.*parseDouble*(P\_2\_Z.getText()));  
 appendToPane(Console,"Creating Point\_2( "+secondPoint.getX()+" , "+secondPoint.getY()+" , "+secondPoint.getZ()+" )\n", Color.*BLACK*);  
 P\_2\_Create.setEnabled(false);  
 P\_2\_Print.setEnabled(true);  
 P\_2\_getX.setEnabled(true);  
 P\_2\_getY.setEnabled(true);  
 P\_2\_getZ.setEnabled(true);  
 P\_2\_SetX.setEnabled(true);  
 P\_2\_SetY.setEnabled(true);  
 P\_2\_SetZ.setEnabled(true);  
 pointCounter++;  
 }  
 else {  
 secondPoint = new Point3D();  
 appendToPane(Console,"Creating Point\_2( "+secondPoint.getX()+" , "+secondPoint.getY()+" , "+secondPoint.getZ()+" )\n", Color.*BLACK*);  
 P\_2\_Create.setEnabled(false);  
 P\_2\_Print.setEnabled(true);  
 P\_2\_getX.setEnabled(true);  
 P\_2\_getY.setEnabled(true);  
 P\_2\_getZ.setEnabled(true);  
 P\_2\_SetX.setEnabled(true);  
 P\_2\_SetY.setEnabled(true);  
 P\_2\_SetZ.setEnabled(true);  
 pointCounter++;  
 }  
 if (pointCounter == 2){  
 compareButton.setEnabled(true);  
 DistanseButton.setEnabled(true);  
 }  
 if (pointCounter == 3){  
 areaButton.setEnabled(true);  
 }  
 C\_M\_1.addItem("Point\_2");  
 C\_M\_2.addItem("Point\_2");  
 F\_P\_M.addItem("Point\_2");  
 S\_P\_M.addItem("Point\_2");  
 }  
 });  
  
 P\_2\_Print.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 appendToPane(Console,"Displaying Point\_2( "+secondPoint.getX()+" , "+secondPoint.getY()+" , "+secondPoint.getZ()+" )\n", Color.*BLACK*);  
 }  
 });  
  
 P\_2\_getX.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 appendToPane(Console,"Coordinate X of Point\_2 is: "+secondPoint.getX()+"\n", Color.*BLACK*);  
 }  
 });  
  
 P\_2\_getY.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 appendToPane(Console,"Coordinate Y of Point\_2 is: "+secondPoint.getY()+"\n", Color.*BLACK*);  
 }  
 });  
  
 P\_2\_getZ.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 appendToPane(Console,"Coordinate Z of Point\_2 is: "+secondPoint.getZ()+"\n", Color.*BLACK*);  
 }  
 });  
  
 P\_2\_SetX.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (!(P\_2\_X.getText().trim().isEmpty())) {  
 if (secondPoint.getX() != Double.*parseDouble*(P\_2\_X.getText())){  
 secondPoint.setX(Double.*parseDouble*(P\_2\_X.getText()));  
 appendToPane(Console,"Coordinate X of Point\_2 changed to: "+secondPoint.getX()+"\n", Color.*BLACK*);  
 }  
 else {  
 appendToPane(Console,"Сhanges are unnecessary!\n", Color.*BLACK*);  
 }  
 }  
 else {  
 appendToPane(Console,"Enter X coordinate!\n", Color.*RED*);  
 }  
 }  
 });  
  
 P\_2\_SetY.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (!(P\_2\_Y.getText().trim().isEmpty())) {  
 if (secondPoint.getY() != Double.*parseDouble*(P\_2\_Y.getText())){  
 secondPoint.setY(Double.*parseDouble*(P\_2\_Y.getText()));  
 appendToPane(Console,"Coordinate Y of Point\_2 changed to: "+secondPoint.getY()+"\n", Color.*BLACK*);  
 }  
 else {  
 appendToPane(Console,"Сhanges are unnecessary!\n", Color.*BLACK*);  
 }  
 }  
 else {  
 appendToPane(Console,"Enter Y coordinate!\n", Color.*RED*);  
 }  
 }  
 });  
  
 P\_2\_SetZ.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (!(P\_2\_Z.getText().trim().isEmpty())) {  
 if (secondPoint.getZ() != Double.*parseDouble*(P\_2\_Z.getText())){  
 secondPoint.setZ(Double.*parseDouble*(P\_2\_Z.getText()));  
 appendToPane(Console,"Coordinate Z of Point\_2 changed to: "+secondPoint.getZ()+"\n", Color.*BLACK*);  
 }  
 else {  
 appendToPane(Console,"Сhanges are unnecessary!\n", Color.*BLACK*);  
 }  
 }  
 else {  
 appendToPane(Console,"Enter Z coordinate!\n", Color.*RED*);  
 }  
 }  
 });  
  
 P\_3\_Create.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (!(P\_3\_X.getText().trim().isEmpty()) && !(P\_3\_Y.getText().trim().isEmpty()) && !(P\_3\_Z.getText().trim().isEmpty())){  
 thirdPoint = new Point3D(Double.*parseDouble*(P\_3\_X.getText()),Double.*parseDouble*(P\_3\_Y.getText()),Double.*parseDouble*(P\_3\_Z.getText()));  
 appendToPane(Console,"Creating Point\_3( "+thirdPoint.getX()+" , "+thirdPoint.getY()+" , "+thirdPoint.getZ()+" )\n", Color.*BLACK*);  
 P\_3\_Create.setEnabled(false);  
 P\_3\_Print.setEnabled(true);  
 P\_3\_getX.setEnabled(true);  
 P\_3\_getY.setEnabled(true);  
 P\_3\_getZ.setEnabled(true);  
 P\_3\_SetX.setEnabled(true);  
 P\_3\_SetY.setEnabled(true);  
 P\_3\_SetZ.setEnabled(true);  
 pointCounter++;  
 }  
 else {  
 thirdPoint = new Point3D();  
 appendToPane(Console,"Creating Point\_3( "+thirdPoint.getX()+" , "+thirdPoint.getY()+" , "+thirdPoint.getZ()+" )\n", Color.*BLACK*);  
 P\_3\_Create.setEnabled(false);  
 P\_3\_Print.setEnabled(true);  
 P\_3\_getX.setEnabled(true);  
 P\_3\_getY.setEnabled(true);  
 P\_3\_getZ.setEnabled(true);  
 P\_3\_SetX.setEnabled(true);  
 P\_3\_SetY.setEnabled(true);  
 P\_3\_SetZ.setEnabled(true);  
 pointCounter++;  
 }  
 if (pointCounter == 2){  
 compareButton.setEnabled(true);  
 DistanseButton.setEnabled(true);  
 }  
 if (pointCounter == 3){  
 areaButton.setEnabled(true);  
 }  
 C\_M\_1.addItem("Point\_3");  
 C\_M\_2.addItem("Point\_3");  
 F\_P\_M.addItem("Point\_3");  
 S\_P\_M.addItem("Point\_3");  
 }  
 });  
  
 P\_3\_Print.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 appendToPane(Console,"Displaying Point\_3( "+thirdPoint.getX()+" , "+thirdPoint.getY()+" , "+thirdPoint.getZ()+" )\n", Color.*BLACK*);  
 }  
 });  
  
 P\_3\_getX.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 appendToPane(Console,"Coordinate X of Point\_3 is: "+thirdPoint.getX()+"\n", Color.*BLACK*);  
 }  
 });  
  
 P\_3\_getY.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 appendToPane(Console,"Coordinate Y of Point\_3 is: "+thirdPoint.getY()+"\n", Color.*BLACK*);  
 }  
 });  
  
 P\_3\_getZ.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 appendToPane(Console,"Coordinate Z of Point\_3 is: "+thirdPoint.getZ()+"\n", Color.*BLACK*);  
 }  
 });  
  
 P\_3\_SetX.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (!(P\_3\_X.getText().trim().isEmpty())) {  
 if (thirdPoint.getX() != Double.*parseDouble*(P\_3\_X.getText())){  
 thirdPoint.setX(Double.*parseDouble*(P\_3\_X.getText()));  
 appendToPane(Console,"Coordinate X of Point\_3 changed to: "+thirdPoint.getX()+"\n", Color.*BLACK*);  
 }  
 else {  
 appendToPane(Console,"Сhanges are unnecessary!\n", Color.*BLACK*);  
 }  
 }  
 else {  
 appendToPane(Console,"Enter X coordinate!\n", Color.*RED*);  
 }  
 }  
 });  
  
 P\_3\_SetY.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (!(P\_3\_Y.getText().trim().isEmpty())) {  
 if (thirdPoint.getY() != Double.*parseDouble*(P\_3\_Y.getText())){  
 thirdPoint.setY(Double.*parseDouble*(P\_3\_Y.getText()));  
 appendToPane(Console,"Coordinate Y of Point\_3 changed to: "+thirdPoint.getY()+"\n", Color.*BLACK*);  
 }  
 else {  
 appendToPane(Console,"Сhanges are unnecessary!\n", Color.*BLACK*);  
 }  
 }  
 else {  
 appendToPane(Console,"Enter Y coordinate!\n", Color.*RED*);  
 }  
 }  
 });  
  
 P\_3\_SetZ.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (!(P\_3\_Z.getText().trim().isEmpty())) {  
 if (thirdPoint.getZ() != Double.*parseDouble*(P\_3\_Z.getText())){  
 thirdPoint.setZ(Double.*parseDouble*(P\_3\_Z.getText()));  
 appendToPane(Console,"Coordinate Z of Point\_3 changed to: "+thirdPoint.getZ()+"\n", Color.*BLACK*);  
 }  
 else {  
 appendToPane(Console,"Сhanges are unnecessary!\n", Color.*BLACK*);  
 }  
 }  
 else {  
 appendToPane(Console,"Enter Z coordinate!\n", Color.*RED*);  
 }  
 }  
 });  
  
 compareButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String P1 = String.*valueOf*(C\_M\_1.getSelectedItem());  
 String P2 = String.*valueOf*(C\_M\_2.getSelectedItem());  
 if (P1 == "Point\_1"){  
 if (P2 == "Point\_1"){  
 appendToPane(Console,"Equality is "+firstPoint.compare(firstPoint)+"\n", Color.*BLACK*);  
 }  
 else if (P2 == "Point\_2") {  
 appendToPane(Console,"Equality is "+firstPoint.compare(secondPoint)+"\n", Color.*BLACK*);  
 }  
 else {  
 appendToPane(Console,"Equality is "+firstPoint.compare(thirdPoint)+"\n", Color.*BLACK*);  
 }  
 }  
 else if (P1 == "Point\_2") {  
 if (P2 == "Point\_1"){  
 appendToPane(Console,"Equality is "+secondPoint.compare(firstPoint)+"\n", Color.*BLACK*);  
 }  
 else if (P2 == "Point\_2") {  
 appendToPane(Console,"Equality is "+secondPoint.compare(secondPoint)+"\n", Color.*BLACK*);  
 }  
 else {  
 appendToPane(Console,"Equality is "+secondPoint.compare(thirdPoint)+"\n", Color.*BLACK*);  
 }  
 }  
 else{  
 if (P2 == "Point\_1"){  
 appendToPane(Console,"Equality is "+thirdPoint.compare(firstPoint)+"\n", Color.*BLACK*);  
 }  
 else if (P2 == "Point\_2") {  
 appendToPane(Console,"Equality is "+thirdPoint.compare(secondPoint)+"\n", Color.*BLACK*);  
 }  
 else {  
 appendToPane(Console,"Equality is "+thirdPoint.compare(thirdPoint)+"\n", Color.*BLACK*);  
 }  
 }  
 }  
 });  
  
 DistanseButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String P1 = String.*valueOf*(F\_P\_M.getSelectedItem());  
 String P2 = String.*valueOf*(S\_P\_M.getSelectedItem());  
 if (P1 == "Point\_1"){  
 if (P2 == "Point\_1"){  
 appendToPane(Console,"Distance from "+P1+" to "+P2+" is "+firstPoint.distanceTo(firstPoint)+"\n", Color.*BLACK*);  
 }  
 else if (P2 == "Point\_2") {  
 appendToPane(Console,"Distance from "+P1+" to "+P2+" is "+firstPoint.distanceTo(secondPoint)+"\n", Color.*BLACK*);  
 }  
 else {  
 appendToPane(Console,"Distance from "+P1+" to "+P2+" is "+firstPoint.distanceTo(thirdPoint)+"\n", Color.*BLACK*);  
 }  
 }  
 else if (P1 == "Point\_2") {  
 if (P2 == "Point\_1"){  
 appendToPane(Console,"Distance from "+P1+" to "+P2+" is "+secondPoint.distanceTo(firstPoint)+"\n", Color.*BLACK*);  
 }  
 else if (P2 == "Point\_2") {  
 appendToPane(Console,"Distance from "+P1+" to "+P2+" is "+secondPoint.distanceTo(secondPoint)+"\n", Color.*BLACK*);  
 }  
 else {  
 appendToPane(Console,"Distance from "+P1+" to "+P2+" is "+secondPoint.distanceTo(thirdPoint)+"\n", Color.*BLACK*);  
 }  
 }  
 else{  
 if (P2 == "Point\_1"){  
 appendToPane(Console,"Distance from "+P1+" to "+P2+" is "+thirdPoint.distanceTo(firstPoint)+"\n", Color.*BLACK*);  
 }  
 else if (P2 == "Point\_2") {  
 appendToPane(Console,"Distance from "+P1+" to "+P2+" is "+thirdPoint.distanceTo(secondPoint)+"\n", Color.*BLACK*);  
 }  
 else {  
 appendToPane(Console,"Distance from "+P1+" to "+P2+" is "+thirdPoint.distanceTo(thirdPoint)+"\n", Color.*BLACK*);  
 }  
 }  
 }  
 });  
  
 areaButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if(!firstPoint.compare(secondPoint) && !secondPoint.compare(thirdPoint) && !thirdPoint.compare(firstPoint)) {  
 appendToPane(Console,"Area from Point\_1 to Point\_2 to Point\_3 is "+Point3D.*area*(firstPoint, secondPoint, thirdPoint)+"\n", Color.*BLACK*);  
 }  
 else {  
 appendToPane(Console,"Enter the correct points! \n", Color.*BLACK*);  
 }  
 }  
 });  
 }  
 private void appendToPane(JTextPane tp, String msg, Color c)  
 {  
 StyleContext sc = StyleContext.*getDefaultStyleContext*();  
 AttributeSet aset = sc.addAttribute(sc.getEmptySet(), StyleConstants.*Foreground*, c);  
  
 aset = sc.addAttribute(aset, StyleConstants.*FontFamily*, "Lucida Console");  
 aset = sc.addAttribute(aset, StyleConstants.*Alignment*, StyleConstants.*ALIGN\_JUSTIFIED*);  
  
 tp.setCharacterAttributes(aset, false);  
 tp.replaceSelection(msg);  
 }  
  
  
}

**GUI form**

****

1. Компиляция и проверка программы
2.   
   ****

**Вывод:**

Благодаря этой лабораторной работе я разобрался с основами объектно-ориентированного программирования.