Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

ПРАКТИКА НАВЧАЛЬНА

Звіт з практичного завдання «Графік функції»

Підготував студент

1-го року навчання

Факультету інформатики

Спеціальність «Інженерія програмного забезпечення»

3 група

Чорнокозинський Кирило Сергійович

Викладач

Олена Миколаївна  
Пєчкурова

Contents

[**1.** **Постановка задачі** 3](#_Toc9327769)

[**2.** **Структура програми, опис методів та класів** 3](#_Toc9327770)

[**3.** **Інструкція для користувача** 6](#_Toc9327771)

[**4.** **Скріншоти** 8](#_Toc9327772)

[**5.** **Труднощі у виконанні завдання** 10](#_Toc9327777)

[**6.** **Висновки** 11](#_Toc9327778)

[**7.** **Код** 12](#_Toc9327779)

­­

# **Постановка задачі**

Написати програму яка будує графік заданої функції.

Обов'язкові функції:

1) Можливість задавати початкові дані, діапазон та крок.

2) Збереження графіку в файл.

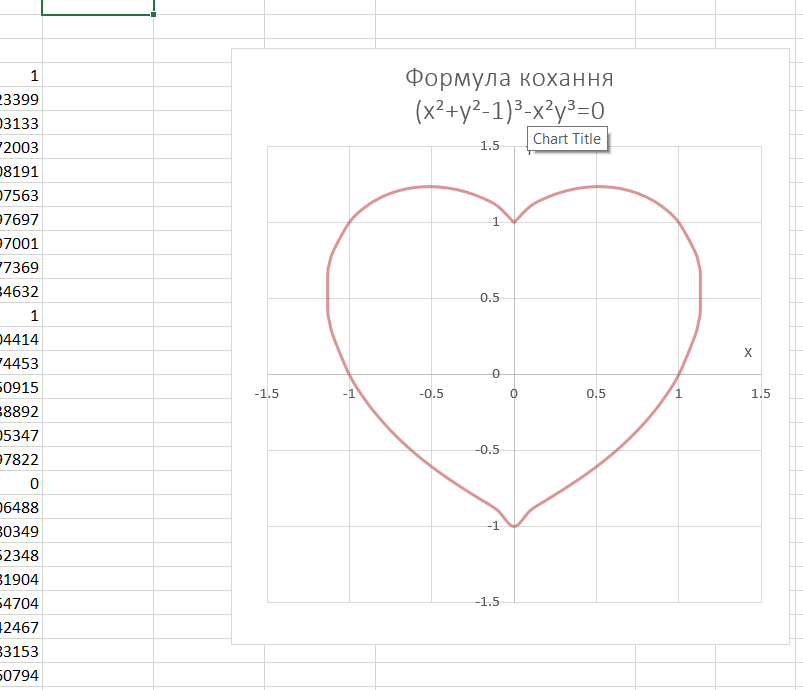


Рис. 1 - варіант графіку

# **Структура програми, опис методів та класів**

Для створення графічного інтерфейсу була використана бібліотека **javafx**.

Програма складається з трьох публічних класів:

1.Controller

2.Dot  
3.Main

* *Поля класу Main:*
* public static Stage *graphFrame*;  
  public static ArrayList<Dot> *downPart* = new ArrayList<>();  
  public static ArrayList<Dot> *upPart* = new ArrayList<>();  
  public static LineChart<Number, Number> *numberLineChart*;
* *Методи класу Controller:*
* public void start(Stage primaryStage)

Створює програму на основі файла sample.fxml.

public static void createGraph(double step, double xMin, double xMax, double yMin, double yMax)

Створює вікно з графіком та відмальовує його.

public static void recreateGraph(double step, double xMin, double xMax, double yMin, double yMax)

Обновлює графік відповідно до нових параметрів.

public static void clean()

Чистить колекції з точками.

public static void initDots(double step, double x1Range, double x2Range, double y1Range, double y2Range)

Створює колекції з точками.

public static void main(String[] args)

Запускає програму.

Інші методи цього класу менш важливі, а саме:   
сортування точок по іксам та ігрикам та перевірка стрічки на число.

* Для кнопки “Draw” - зчитуються дані з текстових полів та викликаться метод draw(), з якого розпочинаються подальші дії щодо створення графіку та виводу його на екран.
* Для кнопки “Save” – викликається метод save()(створюється файл формату .png, зі збереженою картинкою графіку).
* *Поля класу Controller:*
* @FXML  
  private Label sliderLabel;  
    
  @FXML  
  private Slider slider;  
    
  @FXML  
  private TextField xMin;  
    
  @FXML  
  private TextField xMax;  
    
  @FXML  
  private TextField yMin;  
    
  @FXML  
  private TextField yMax;  
    
  @FXML  
  private Label labelException1;  
    
  @FXML  
  private Label labelException2;  
    
  @FXML  
  private Label labelSaveException;  
    
    
  private static int *counter* = 0;  
  private static int *saveCounter* = 0;
* *Методи класу Controller:*
* public void draw(ActionEvent event)

Запускає .

Private void drawGraphic()

Зчитує те, що було введено в комірки діапазону та кроку.

public void changeLabel()

Пов’язує лейбл зі слайдером.

public void save(ActionEvent event)

Зберігає скріншот графіка.

# **Інструкція для користувача**

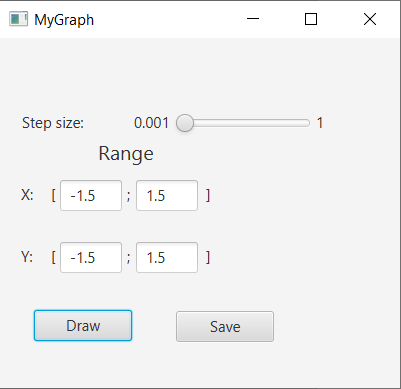


Рис. 2 - початкове вікно програми, яке зустрічає користувача

Це вікно зустрічає користувача, одразу з оптимальними параметрами.

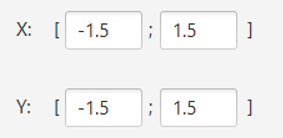


Рис. 3 - тут можна встановити діапазон видимості графіку

В цих полях можна вибрати діапазон видимості графіку.

При натисканні на кнопку Save створюється файл формату .png з скріншотом

Графіку та зберігається у дефолтну папку проекту.

При натисканні на кнопку Draw вілкривається нове вікно з графіком, якщо це можливо.

Приклади:

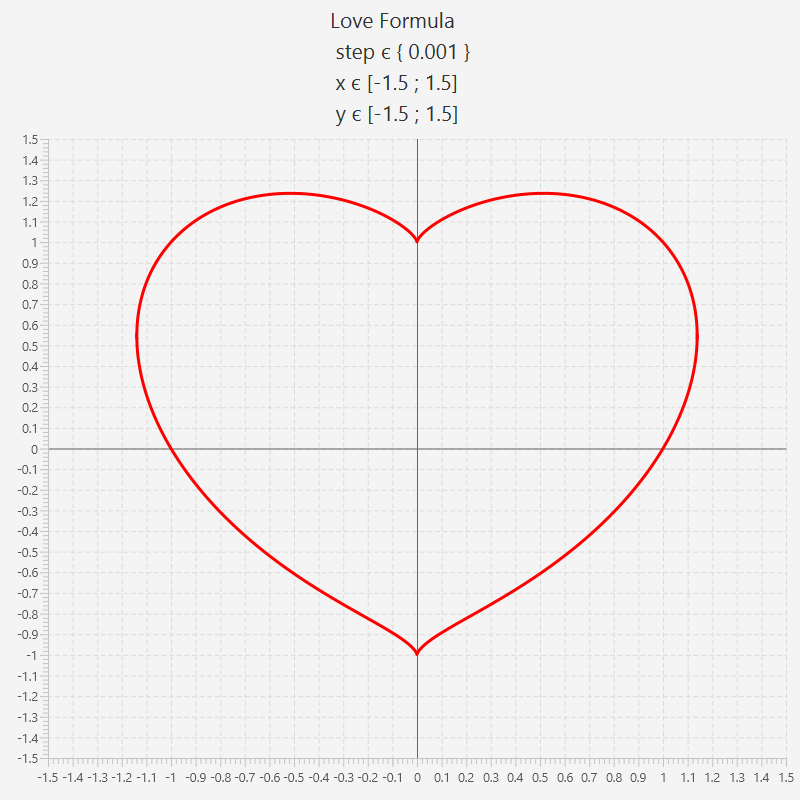


Рис. 4 – приклад 1

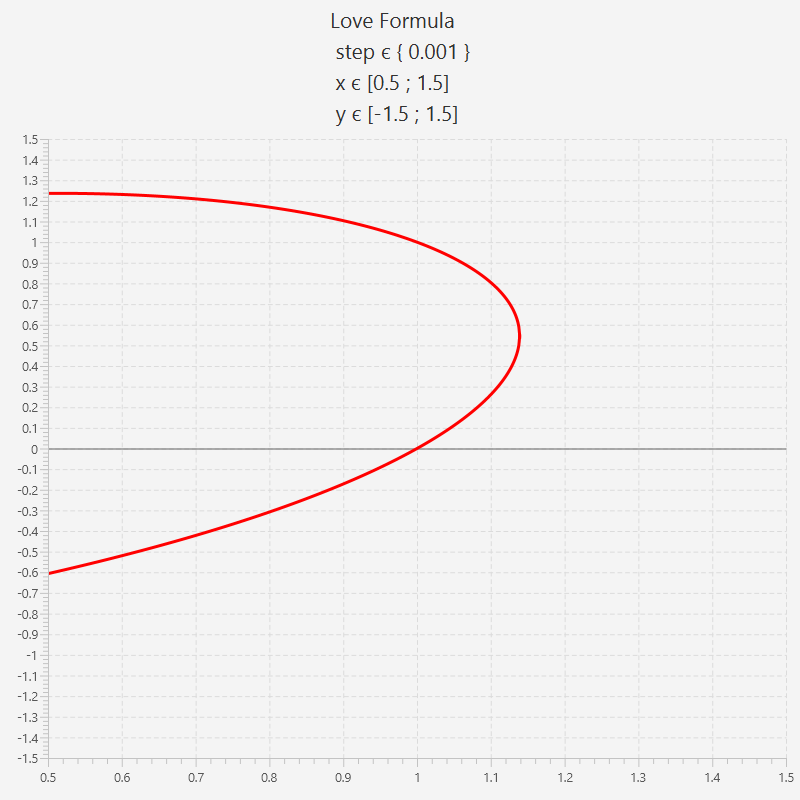
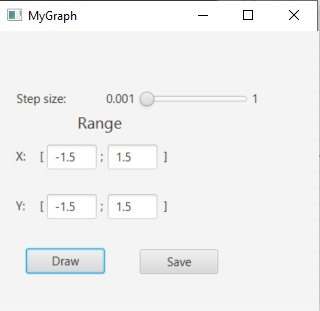
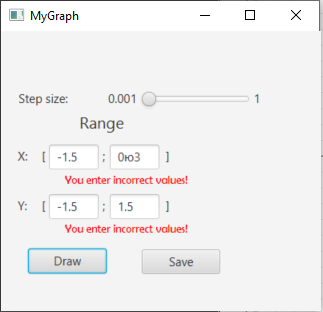
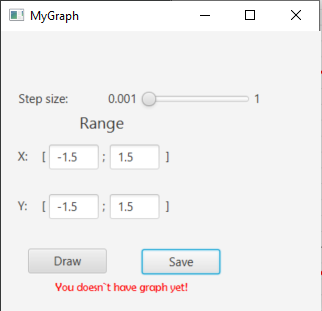


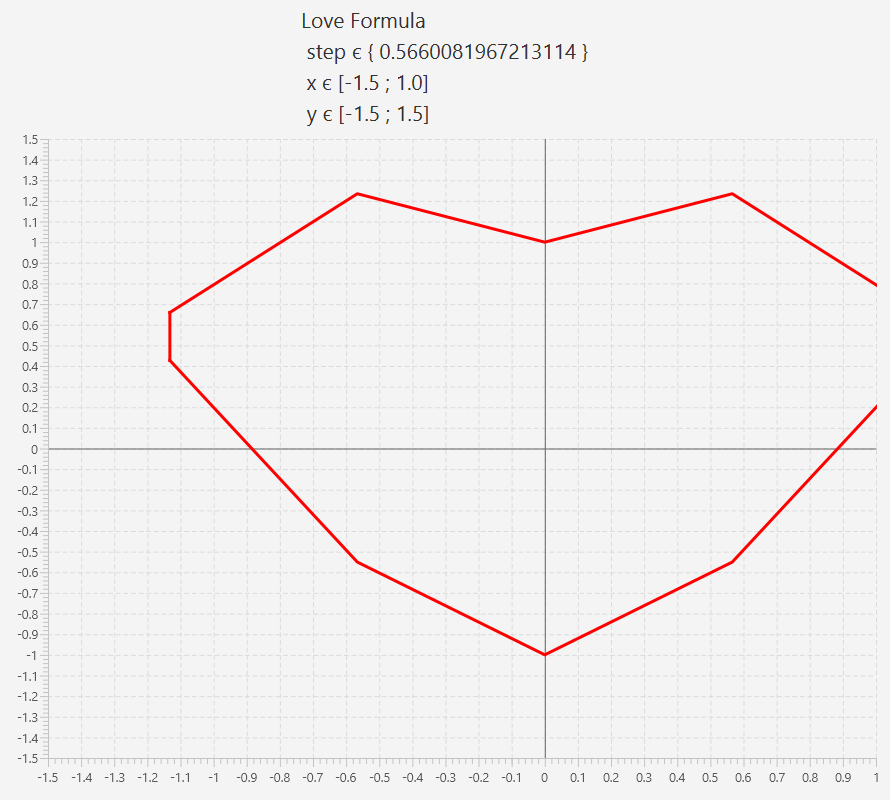
Рис. 5 – приклад 2

1. **Скріншоти**







****

1. **Труднощі у виконанні завдання**

Малюючи цей графік, я використовував компонент бібліотеки javafx, а саме LineChart.  
Він сортує або лише по ігрику, або лише по іксу, що довгий час не давало мені з’єднувати точки правильно.

Але я поділив точки на дві частини: верхня і нижня.  
А потім відсортувати з’єднання по іксу.  
Так я досяг правильного графіка.

1. **Висновки**

Виконання цього завдання дало мені змогу поглибити свої знання Java, а саме використання графічного інтерфейсу (javafx), збереження в файли, тренувало мої здібності логічного мислення.

# **Код**

1. public class Main extends Application {  
    public static Stage *graphFrame*;  
    public static ArrayList<Dot> *downPart* = new ArrayList<>();  
    public static ArrayList<Dot> *upPart* = new ArrayList<>();  
    public static LineChart<Number, Number> *numberLineChart*;  
     
    @Override  
    public void start(Stage primaryStage) throws Exception {  
    Parent root = FXMLLoader.*load*(getClass().getResource("sample.fxml"));  
    primaryStage.setTitle("MyGraph");  
    primaryStage.setScene(new Scene(root, 400, 350));  
    primaryStage.show();  
    }  
     
     
    public static void createGraph(double step, double xMin, double xMax, double yMin, double yMax) {  
    *graphFrame* = new Stage();  
    *graphFrame*.setTitle("lOVE");  
    *recreateGraph*(step, xMin, xMax, yMin, yMax);  
    }  
     
    public static void recreateGraph(double step, double xMin, double xMax, double yMin, double yMax) {  
    NumberAxis xN = new NumberAxis();  
    NumberAxis yN = new NumberAxis();  
    xN.setAutoRanging(false);  
    yN.setAutoRanging(false);  
    xN.setTickUnit(0.1);  
    yN.setTickUnit(0.1);  
    xN.setLowerBound(xMin);  
    xN.setUpperBound(xMax);  
    yN.setLowerBound(yMin);  
    yN.setUpperBound(yMax);  
    *numberLineChart* = new LineChart<Number, Number>(xN, yN);  
    *numberLineChart*.setTitle("Love Formula " + "\n" + " step є { " + step + " }" + "\n" + " x є [" + xMin + " ; " + xMax + "] " + "\n" + " y є [" + yMin + " ; " + yMax + "]");  
    *numberLineChart*.setLegendVisible(false); // !!!!  
    Scene scene = new Scene(*numberLineChart*, 800, 800);  
    *graphFrame*.setScene(scene);  
    *graphFrame*.show();  
    }  
     
    public static void clean() {  
    *downPart* = new ArrayList<>();  
    *upPart* = new ArrayList<>();  
     
    }  
     
    public static void initDots(double step, double x1Range, double x2Range, double y1Range, double y2Range) {  
    for (double x = 0; x < 1.5; x += step) {  
    double y1 = (1 / 2.0) \* ((Math.*pow*(x, 2 / 3.0)) + Math.*sqrt*(Math.*pow*(x, 4 / 3.0) + 4 \* (1 - x \* x)));  
    double y2 = (1 / 2.0) \* ((Math.*pow*(x, 2 / 3.0)) - Math.*sqrt*(Math.*pow*(x, 4 / 3.0) + 4 \* (1 - x \* x)));  
    if (!Double.*isNaN*(y1))  
    *downPart*.add(new Dot(x, Math.*min*(y1, y2)));  
     
    if (!Double.*isNaN*(y2))  
    *upPart*.add(new Dot(x, Math.*max*(y1, y2)));  
    }  
     
    if (x1Range < 0) {  
    for (double x = 0; x < 1.5; x += step) {  
     
    double y1 = (1 / 2.0) \* ((Math.*pow*(x, 2 / 3.0)) + Math.*sqrt*(Math.*pow*(x, 4 / 3.0) + 4 \* (1 - x \* x)));  
    double y2 = (1 / 2.0) \* ((Math.*pow*(x, 2 / 3.0)) - Math.*sqrt*(Math.*pow*(x, 4 / 3.0) + 4 \* (1 - x \* x)));  
     
    if (!Double.*isNaN*(y1))  
    *upPart*.add(new Dot(-x, Math.*max*(y1, y2)));  
    if (!Double.*isNaN*(y2))  
    *downPart*.add(new Dot((-x), Math.*min*(y1, y2)));  
    }  
    }  
    }  
     
     
    public static void main(String[] args) {  
    *launch*(args);  
    }  
     
     
    public static void dotSort(ArrayList<Dot> list) {  
    Collections.*sort*(list, new Comparator<Dot>() {  
    @Override  
    public int compare(Dot o1, Dot o2) {  
    return Double.*compare*(o1.getX(), o2.getX());  
    }  
    });  
     
    }  
     
    public static void dotSortY(ArrayList<Dot> list) {  
    Collections.*sort*(list, new Comparator<Dot>() {  
    @Override  
    public int compare(Dot o1, Dot o2) {  
    return Double.*compare*(o1.getY(), o2.getY());  
    }  
    });  
     
    }  
     
    public static boolean isNumeric(String s) {  
    boolean numeric = true;  
     
    try {  
    Double num = Double.*parseDouble*(s);  
    } catch (NumberFormatException e) {  
    numeric = false;  
    }  
     
    if (numeric)  
    return true;  
    else  
    return false;  
    }  
     
     
     
   }

}

public class Controller {  
  
  
 @FXML  
 private Label sliderLabel;  
  
 @FXML  
 private Slider slider;  
  
 @FXML  
 private TextField xMin;  
  
 @FXML  
 private TextField xMax;  
  
 @FXML  
 private TextField yMin;  
  
 @FXML  
 private TextField yMax;  
  
 @FXML  
 private Label labelException1;  
  
 @FXML  
 private Label labelException2;  
  
 @FXML  
 private Label labelSaveException;  
  
  
 private static int *counter* = 0;  
  
  
 public void draw(ActionEvent event) {  
 boolean a = *isNumeric*(xMin.getText()) && *isNumeric*(xMax.getText()) && *isNumeric*(yMin.getText()) && *isNumeric*(yMax.getText());  
  
 if (a && Double.*parseDouble*(xMin.getText()) < Double.*parseDouble*(xMax.getText()) && Double.*parseDouble*(yMin.getText()) < Double.*parseDouble*(yMax.getText())) {  
 labelSaveException.setOpacity(0);  
 labelException2.setOpacity(0);  
 labelException1.setOpacity(0);  
 Main.*clean*();  
 if (*counter* < 1) {  
  
 Main.*createGraph*(slider.getValue(), Double.*parseDouble*(xMin.getText()), Double.*parseDouble*(xMax.getText()), Double.*parseDouble*(yMin.getText()), Double.*parseDouble*(yMax.getText()));  
 drawGraphic();  
 *counter*++;  
 } else {  
  
 drawGraphic();  
 }  
 } else {  
 labelException2.setOpacity(100);  
 labelException1.setOpacity(100);  
 }  
  
 }  
  
 public void changeLabel() {  
 sliderLabel.textProperty().bind(  
 Bindings.*format*(  
 "%.4f",  
 slider.valueProperty()  
 )  
 );  
 }  
  
 public void save(ActionEvent event) throws IOException {  
 if (*numberLineChart* != null) {  
 String timeStamp = new SimpleDateFormat("yyyyMMdd\_HH-mm-ss").format(Calendar.*getInstance*().getTime());  
 WritableImage snapShot = Main.*numberLineChart*.snapshot(null, null);  
 ImageIO.*write*(SwingFXUtils.*fromFXImage*(snapShot, null), "png",  
 new File("graph-" + timeStamp + ".png"));  
 } else {  
 labelSaveException.setOpacity(100);  
 }  
 }  
  
 private void drawGraphic() {  
 *recreateGraph*(slider.getValue(), Double.*parseDouble*(xMin.getText()), Double.*parseDouble*(xMax.getText()), Double.*parseDouble*(yMin.getText()), Double.*parseDouble*(yMax.getText()));  
  
 XYChart.Series downPart = new XYChart.Series();  
  
 Main.*initDots*(slider.getValue(), Double.*parseDouble*(xMin.getText()),  
 Double.*parseDouble*(xMax.getText()), Double.*parseDouble*(yMin.getText()),  
 Double.*parseDouble*(yMax.getText()));  
  
 for (Dot dot : Main.*downPart*) {  
 downPart.getData().add(new XYChart.Data(dot.getX(), dot.getY()));  
 }  
 XYChart.Series upPart = new XYChart.Series();  
 for (Dot dot : Main.*upPart*) {  
  
 upPart.getData().add(new XYChart.Data(dot.getX(), dot.getY()));  
 }  
 Main.*dotSort*(Main.*upPart*);  
 Main.*dotSort*(Main.*downPart*);  
 if (Main.*downPart*.size() != 0 && Main.*upPart*.size() != 0) {  
 Dot dot = Main.*upPart*.get(0);  
 if (!(Double.*parseDouble*(xMin.getText()) > -1.125)) {  
 XYChart.Series leftConnection = new XYChart.Series();  
  
 leftConnection.getData().add(new XYChart.Data(dot.getX(), dot.getY()));  
 dot = Main.*downPart*.get(0);  
 leftConnection.getData().add(new XYChart.Data(dot.getX(), dot.getY()));  
 *numberLineChart*.getData().addAll(leftConnection);  
 leftConnection.getNode().setStyle("-fx-stroke: rgba( 255,0,0 , 1.0);");  
 }  
 if (Double.*parseDouble*(xMax.getText()) >= 1.125) {  
 XYChart.Series rightConnection = new XYChart.Series();  
 dot = Main.*upPart*.get(Main.*upPart*.size() - 1);  
 rightConnection.getData().add(new XYChart.Data(dot.getX(), dot.getY()));  
 dot = Main.*downPart*.get(Main.*downPart*.size() - 1);  
 rightConnection.getData().add(new XYChart.Data(dot.getX(), dot.getY()));  
 *numberLineChart*.getData().addAll(rightConnection);  
 rightConnection.getNode().setStyle("-fx-stroke: rgba( 255,0,0 , 1.0);");  
 }  
 }  
 *numberLineChart*.setAxisSortingPolicy(javafx.scene.chart.LineChart.SortingPolicy.*X\_AXIS*);  
 *numberLineChart*.getData().addAll(upPart);  
 *numberLineChart*.getData().addAll(downPart);  
 upPart.getNode().setStyle("-fx-stroke: rgba( 255,0,0 , 1.0);");  
 downPart.getNode().setStyle("-fx-stroke: rgba( 255,0,0 , 1.0);");  
 *numberLineChart*.setCreateSymbols(false);  
 }  
}

public class Dot {  
 private Double y;  
 private Double x;  
  
 public Dot(Double x, Double y) {  
 this.y = y;  
 this.x = x;  
 }  
  
 public Double getY() {  
 return y;  
 }  
  
 public void setY(Double y) {  
 this.y = y;  
 }  
  
 public Double getX() {  
 return x;  
 }  
  
 public void setX(Double x) {  
 this.x = x;  
 }  
}