Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

**Розрахунково-графічна робота**

з дисципліни «Крос-платформне програмування»

На тему «Java Beans»

Виконав студент 2 курсу групи КС-21

Безрук Юрій Русланович

Перевірив: доц. Споров О.Є. ...............

Харків – 2020

ЗМІСТ

[ВСТУП 3](#_Toc42129961)

[РОЗДІЛ 1 ОБРОБКА XML 4](#_Toc42129962)

[РОЗДІЛ 2 ПЕРШИЙ КОМПОНЕНТ 8](#_Toc42129963)

[РОЗДІЛ 3 ДРУГИЙ КОМПОНЕНТ 14](#_Toc42129964)

[РОЗДІЛ 4 ОБ’ЄДНАННЯ КОМПОНЕНТІВ В GUI 19](#_Toc42129965)

[ВИСНОВКИ 23](#_Toc42129966)

[ДОДАТОК А ЛІСТИНГИ КЛАСІВ ДЛЯ DOM XML 24](#_Toc42129967)

# ВСТУП

В ході виконання розрахунково-графічної роботи необхідно ознайомитися з основами компонентної технології JavaBeans, з правилами створення JavaBeans-компонентів, основами їх налаштування і використання у середовищах розробки.

На практиці для виконання цих умов необхідно створити два компоненти JavaBeans для представлення зашумленого набору експериментальних даних, що розглядався у попередніх заняттях.

Використовуючи створені компоненти необхідно написати додаток з графічним інтерфейсом користувача, призначений для перегляду та редагування вказаних наборів даних, збережених у XML-файлах. При створенні додатку слід скористуватися візуальним редактором графічного інтерфейсу з обраного середовища розробки.

Таблиця з даними повинна не тільки відображати їх, а й мати можливість додавати, видаляти и редагувати відповідні дані і одразу відтворювати зміни на графіку.

Створений додаток потрібно упакувати в виконуючий jar-архів.

# **РОЗДІЛ 1 ОБРОБКА XML**

Для виконання поставленої задачі необхідно використати створені дещо раніше класи для обробки XML-документів та представлення табличних даних. Коротко опишемо основний їх функціонал.

Клас Date інкапсулює у собі елемент XML-документу з даними, і має три поля: координати х та у, і рядкове представлення дати, відповідної до цих координат. Відповідно, клас також має аксесори до цих полів.

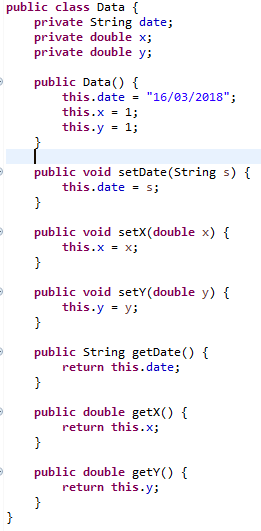


Рисунок 1.1 - клас Data

Клас DataSheet являє собою набір даних, представлений таблицею першого компоненту, тобто список точок з відповідним функціоналом по їх додаванню, редагуванню та усуненню. Полями класу є ім’я таблиці даних та перелік об’єктів класу Data.

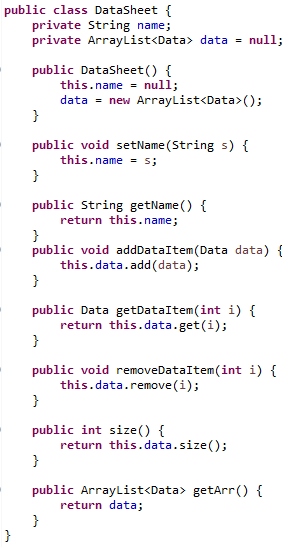


Рисунок 1.2 – клас DataSheet

Створену таблицю даних необхідно заповнювати з XML-файлу. Для цього використовуємо клас XMLSAXParser, який, з використанням SAX-парсеру XML-документів буде аналізувати файл, ім’я якого йому передається. У класі присутнє статичне поле об’єкту DataSheet для збереження зчитаних даних та статичний метод getParse(), у якому створюється фабрика парсеру, обробник, файловий потік з XML-файлом та сам парсер, після чого виконується обробка документу, а таблиця збережених даних повертається до класу, що викликав метод. Обробник документу для парсеру опишемо далі. Також клас має метод getDataSheet(), який повертає збережену таблицю без зчитування з файлу.

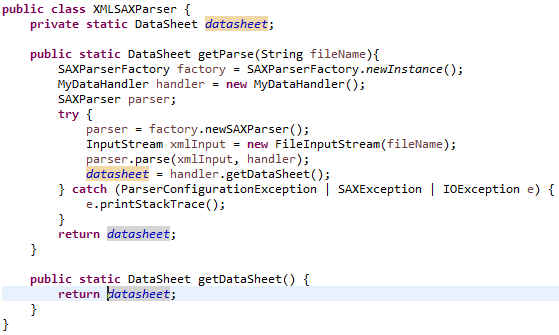


Рисунок 1.3 - клас XMLSAXParser

Тепер опишемо обробник, який був використаний у попередньому класі – MyDataHandler. У класі перевизначаємо методи суперкласу – відповідні реакції на події під час аналізу файлу таким чином, щоб створювалися об’єкти Data і додавалися у таблицю даних. Також додамо метод getDataSheet(), що буде повертати створену таблицю даних.

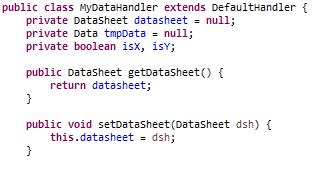


Рисунок 1.4 – поля та аксесори класу MyHandler

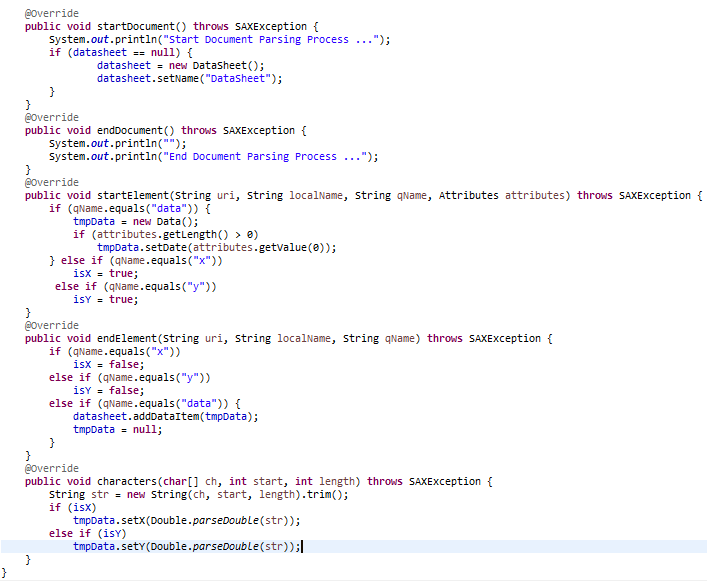


Рисунок 1.5 – методи-обробники елементів файлу

Також будемо використовувати клас XMLDOMParser, який за допомогою DOM-парсеру буде записувати дані таблиці до документу, якщо користувач бажатиме їх зберегти. Його статичний метод викликатиметься з класу DataSheetToXML, який передаватиме його створений у пам’яті за допомогою класу DOMDataSheet документ. Код цих класів наведено у додатку А.

# РОЗДІЛ 2 ПЕРШИЙ КОМПОНЕНТ

Класи, для взаємодії з документами поміщаємо у пакет xml. А для компонентів створюємо новий пакет mybeans. Розміщуємо у ньому описані вище класи Data і DataSheet.

Після цього приступаємо до створення першого компоненту JavaBean, призначеного для табличного відображення даних і роботи з ними. Для цього створюємо візуальний клас DataSheetTablе, що розширює стандартний клас JPanel. Для зручності розміщення компонентів стандартний менеджер компоновки замінюємо на BorderLayout. В південну область (BorderLayout.SOUTH) додаємо панель для кнопок додавання і видалення рядків таблиці з менеджером компоновки FlowLayout, після чого додаємо кнопки і налаштовуємо їх властивості. В центральну область панелі DataSheetTable додаємо панель прокрутки JScrollPane і встановлюємо відтворення вертикальних і горизонтальних полос прокрутки. На саму ж панель прокрутки поміщаємо таблицю JTable. Об’єкт таблиці відтворює дані, які він отримує від іншого об’єкту, що їх зберігає, реалізуючого інтерфейс TableModel. Цей об’єкт створюємо окремо, у ньому будемо зберігати табличні дані. Основні методи, пов’язані з обробкою даних таблиці необхідно перевизначити під обробку об’єкту DataSheet. У результаті клас матиме вигляд:



Рисунок 2.1 – клас DataSheetTableModel, частина 1

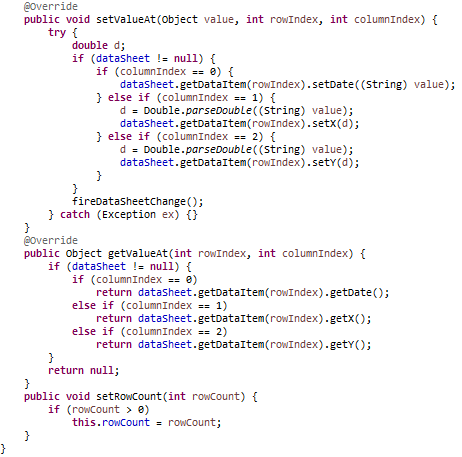


Рисунок 2.2 – клас DataSheetTableModel, частина 2

Після цього, можемо остаточно завершити створення класу DataSheetTable, додавши до його конструктору команди додавання таблиці і встановлення її моделі (даних). У приведеному на рисунці 2.3 коді показуємо остаточний вигляд класу, який ми допишемо після обробників подій.



Рисунок 2.3 – клас DataSheetTable

Оскільки далі компонент необхідно буде запакувати у виконуючий архів, слід вказати пов’язаним з ним компонентам, якщо у результаті дій користувача зміниться стан «сховища даних», що знаходиться в моделі таблиці DataSheetTableModel. Для цього створюємо подію у нашій системі.

Спочатку створюємо клас події, який має наслідувати клас EventObject і мати у назві постфікс Event. Отже, назвемо його DataSheetChangeEvent. Він не буде зберігати ніякої додаткової інформації, а просто буде маркером того, що змінилося «сховище». В конструкторі події вказуємо її джерело – компонент, у якому ця подія трапилась.

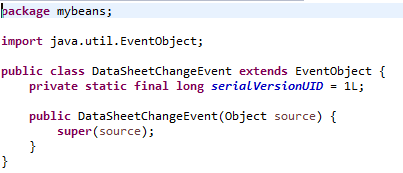


Рисунок 2.4 – клас DataSheetChangeEvent

Далі описуємо інтерфейс слухача події. Він має бути реалізованим клієнтами, які заінтересовані у відстеженні цієї події і бути насідником EventListener, який не має жодного методу. Визначаємо в інтерфейсі метод dataChanged(), який буде викликатися при зміні «сховища». Цьому методу у якості аргументу передаватиметься об’єкт події, яку ми описали вище. Таким чином, якщо реалізувати інтерфейс слухача можна буде дізнатися необхідні подробиці про подію.

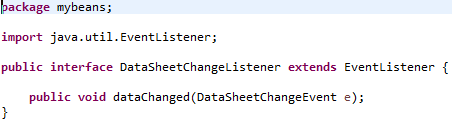


Рисунок 2.5 – клас DataSheetChangeListener

Далі долучаємо підтримку події до класу компоненту, що генерує даний тип події, DataSheetTableModel. Додаємо до класу перелік слухачів події і відповідні методи взаємодії з ним. При виникненні події всі слухачі з цього переліку отримають необхідну інформацію про неї. Також доповнюємо методи, що змінюють «сховище даних» для того, щоб усі заінтересовані компоненти могли відреагувати на ці зміни. У результаті додатковий код до класу матиме вигляд:

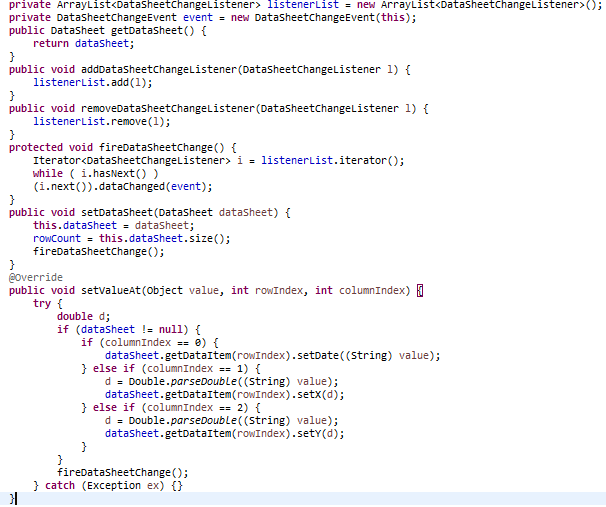


Рисунок 2.6 – додатковий код до класу DataSheetTableModel

Після цього додаємо виклики обробників подій, зображені на рисунку 2.3 у клас DataSheetTable. На цьому створення першого компоненту завершується.

# РОЗДІЛ 3 ДРУГИЙ КОМПОНЕНТ

Другий компонент призначений для графічного відтворення інформації про точки. Для його реалізації у тому ж пакеті створюємо клас DataSheetGraph, що наслідує клас JPanel. Вказуємо у ньому властивості: колір, дельта по координатам х та у і логічну змінну, що вказує на з’єднаність / роз’єднаність точок, і поле-таблицю даних, все це з відповідними методами доступу. Також створюємо конструктор, де вказуємо початковий розмір вікна и викликаємо метод ініціалізації властивостей.

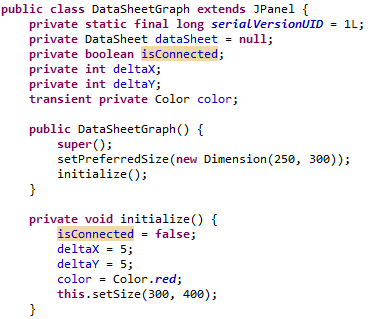


Рисунок 3.1 – початок створення класу DataSheetGraph

Для зручності роботи визначаємо методи, що обчислюють максимальне і мінімальне значення величин Х та Y, які зберігаються у сховищі.

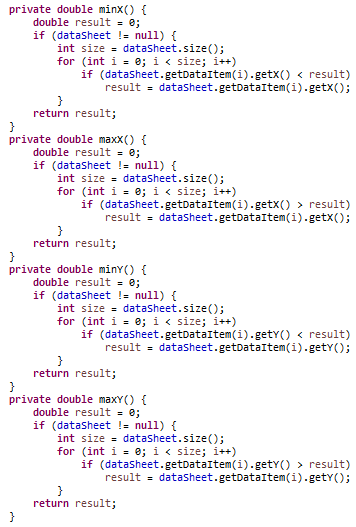


Рисунок 3.2 – методи обчислення максимуму і мінімуму х та у

Перевизначаємо метод paintComponent класу JComponent:

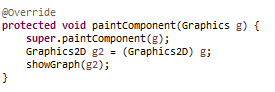


Рисунок 3.3 – метод paintComponent

Після цього створюємо метод showGraph, який і виконує малювання графіку. Спочатку задаємо змінним максимумів і мінімум їх значення з урахуванням дельти. Далі визначаємо коефіцієнти перетворення і положення початку координат, заповнюємо область графіку білим кольором, створюємо сітку для осей Х та Y за допомогою циклічного малювання паралельних ліній. Далі відтворюємо безпосередньо координатні осі і наносимо точки при умові, що їх сховище не порожнє. Остаточний вигляд методу:

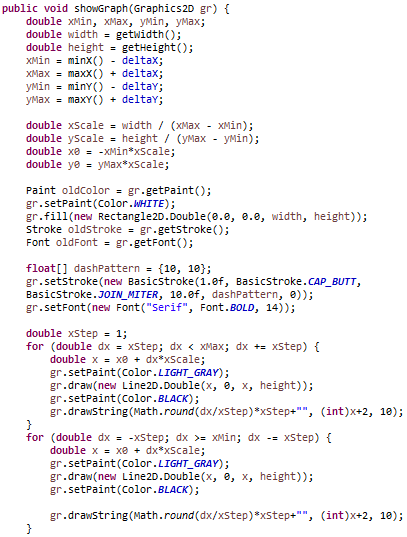


Рисунок 3.4 – метод showGraph, частина 1

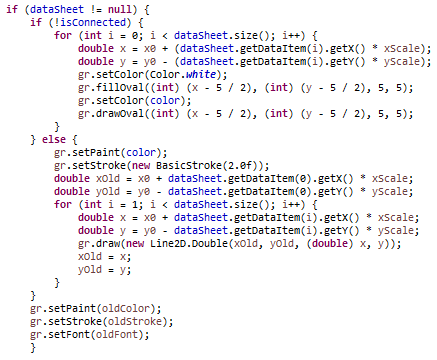
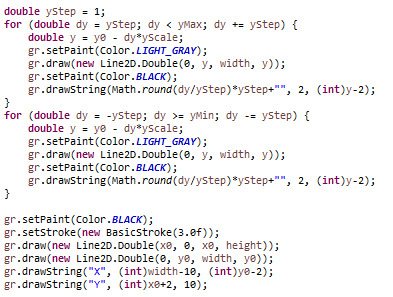


Рисунок 3.5 – метод showGraph, частина 2

Додатково створимо клас DataSheetGraphBeenInfo, який буде описувати компонент для того, щоб на панелі інструментів не відтворювалися «зайві» властивості. Цей клас повинен наслідувати клас SimpleBeanInfo. У ньому обмежуємо відтворювані властивості:

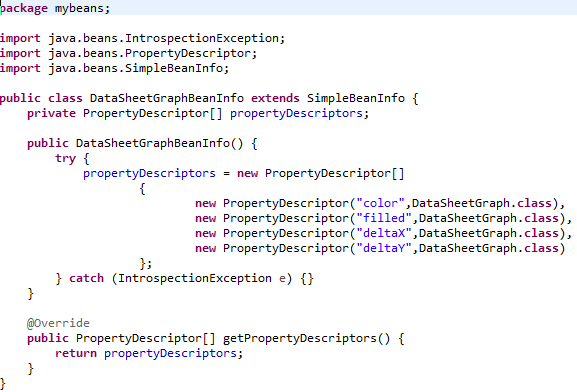


Рисунок 3.6 – клас DataSheetGraphBeenInfo

Таким чином, два компоненти JavaBeans згідно завдання створені.

# РОЗДІЛ 4 ОБ’ЄДНАННЯ КОМПОНЕНТІВ В GUI

Після того, як компоненти з пакету mybeans і класи для роботи с XML-документами з пакету xml створено, необхідно створити головне вікно додатку. Для цього у новому пакеті myApplication створимо клас Test, який буде наслідувати JFrame та додамо йому метод main(), звідки й будемо запускати додаток. Додамо йому поле «сховища», заборонимо змінювати розмір и встановимо менеджер компоновки BorderLayout. У південній області розміщуємо панель кнопок управління додатком. У східну область додаємо компонент-графік, а у західну – компонент-таблицю. Сховище створюється з однією строчкою за замовчуванням. Після створення компоненту DataSheetTable встановлюємо йому сховище і додаємо слухача подій. Для забезпечення можливості вибору файлу для читання чи запису (робота з XML-файлами) необхідно додати стандартний компонент JFileChooser, якому встановлюємо папку вибору за замовчуванням як корінну папку проекту. Далі ствоюємо обробники подій для кнопок: завершення роботи з додатком, очищення таблиці з даними, збереження даних у файлі і відкриття файлу з даними. Також додаємо чекбокс для з’єднання усіх точок лінією.

Роботу додатку перевіряємо у методі main().

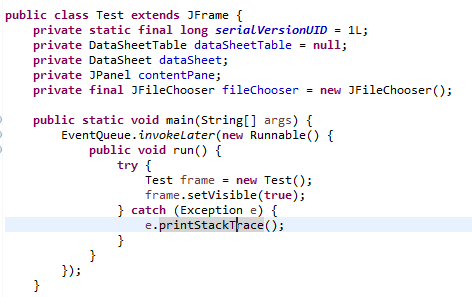


Рисунок 4.1 – поля класу та метод main()

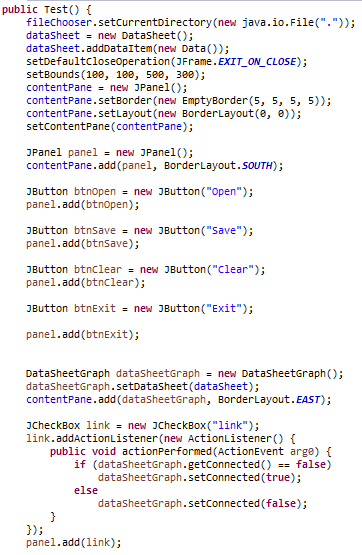


Рисунок 4.2 – конструктор класу. Ініціалізація змінних та чекбокс «зв’язати»

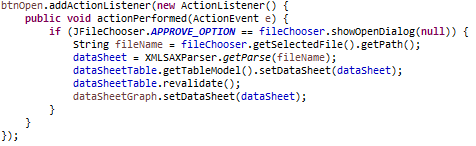


Рисунок 4.3 – конструктор класу. Ініціалізація таблиці даних да слухачі до кнопок «вихід», «очистити», «зберегти», «відкрити»

Тепер графічний інтерфейс створено. Запаковуємо його у виконуючий jar-файл JAVA\_BEANS\_GUI і запускаємо для тестування.

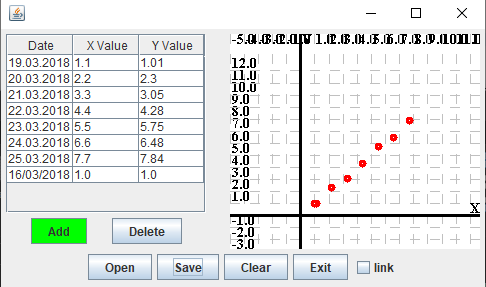


Рисунок 4.4 – графічний інтерфейс користувача

# ВИСНОВКИ

Таким чином, під час виконання розрахунково графічної роботи були створені два графічних компоненти JavaBeans для представлення таблиці, що зберігає зашумлений набір даних за координатами та графіку, що їх відтворяє.

Компоненти були об’єднані у графічний інтерфейс для зручної взаємодії з користувачем, до них були додані відповідні кнопки-активності. Система дозволяє не тільки редагувати і представляти дані, ай завантажувати їх з зовнішнього XML-файлу, або, навпаки, зберігати дані до файлу.

В ході виконання роботи було проведено ознайомлення з основами компонентної технології JavaBeans, з правилами створення JavaBeans-компонентів, основами їх налаштування і використання в середовищах розробки.

# ДОДАТОК А ЛІСТИНГИ КЛАСІВ ДЛЯ DOM XML

Лістинг 1 – клас XmlDOMParser

**public class XmlDOMParser {**

**public static Document *docum*;**

**public static Document getParse(String s){**

**DocumentBuilderFactory dbf = DocumentBuilderFactory.*newInstance*();**

**File xmlFile = new File(s);**

**dbf.setValidating(true);**

**Document doc = null;**

**try {**

**DocumentBuilder db = dbf.newDocumentBuilder();**

**doc = db.parse(xmlFile);**

**} catch (ParserConfigurationException e) { e.printStackTrace();**

**} catch (SAXException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**} catch (IOException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**return doc;**

**}**

**public static Document createDoc() {**

**DocumentBuilderFactory dbf = DocumentBuilderFactory.*newInstance*();**

**dbf.setValidating(true);**

**try {**

**DocumentBuilder db = dbf.newDocumentBuilder();**

***docum* = db.newDocument();**

**} catch (ParserConfigurationException e) { e.printStackTrace();**

**}**

**return *docum*;**

**}**

**public static void CastToXML(String fileName, Document doc) throws TransformerException,**

**FileNotFoundException {**

**TransformerFactory transFactory = TransformerFactory.*newInstance*();**

**Transformer transformer = transFactory.newTransformer();**

**DOMSource source = new DOMSource(doc);**

**File newXMLFile = new File(fileName);**

**FileOutputStream fos = new FileOutputStream(newXMLFile);**

**StreamResult result = new StreamResult(fos);**

**transformer.transform(source, result);**

**}**

**private static void stepThrough(Node start) {**

**System.*out*.println(start.getNodeName() + " = " + start.getNodeValue());**

**if (start instanceof Element) {**

**NamedNodeMap startAttr = start.getAttributes();**

**for (int i = 0; i < startAttr.getLength(); i++) {**

**Node attr = startAttr.item(i);**

**System.*out*.println(" Attribute: " + attr.getNodeName() + " = "**

**+ attr.getNodeValue());**

**}**

**}**

**for (Node child = start.getFirstChild(); child != null;**

**child = child.getNextSibling()) {**

***stepThrough*(child);**

**}**

**}**

**public static void processDocument(Document doc) {**

**Element rootEl = doc.getDocumentElement();**

**System.*out*.println("Root element: " + rootEl.getNodeName());**

**System.*out*.println("Child elements: ");**

***stepThrough*(rootEl);**

**}**

**public static void getSelectInfa(Document doc) {**

**NodeList nl1 = doc.getDocumentElement().getElementsByTagName("x");**

**NodeList nl2 = doc.getDocumentElement().getElementsByTagName("y");**

**if (nl1.getLength() == nl2.getLength()) {**

**for (int i = 0; i < nl1.getLength(); i++) {**

**System.*out*.println(nl1.item(i).getNodeName() + " "**

**+ nl1.item(i).getTextContent() + "\t"**

**+ nl2.item(i).getNodeName() + " "**

**+ nl2.item(i).getTextContent());**

**}**

**}**

**}**

**public static void main(String[] args) throws FileNotFoundException, TransformerException{**

**Document doc = *getParse*("MyXML.xml");**

**DOMDataSheet dsh = new DOMDataSheet(doc);**

**dsh.addElement(dsh.newElement("19.04.2018", 19, 4));**

***CastToXML*("NewXML.xml",dsh.getDoc());**

**}**

**}**

Лістинг 2 –клас DOMDataSheet

**public class DOMDataSheet {**

**private Document doc;**

**public DOMDataSheet(Document doc) {**

**super();**

**this.doc = doc;**

**Element element = this.doc.createElement("root");**

**doc.appendChild(element);**

**}**

**public DOMDataSheet() {**

**this(null);**

**}**

**public Document getDoc() {**

**return doc;**

**}**

**public void setDoc(Document doc) {**

**this.doc = doc;**

**}**

**public int numData() {**

**return doc.getDocumentElement().getElementsByTagName("data").getLength();**

**}**

**public double getX(int pos) {**

**String s = doc.getDocumentElement().getElementsByTagName("x").item(pos).getTextContent();**

**return Double.*parseDouble*(s);**

**}**

**public void setX(int pos, double val) {**

**doc.getDocumentElement().getElementsByTagName("x").item(pos).setTextContent(val+"");**

**}**

**public double getY(int pos) {**

**String s = doc.getDocumentElement().getElementsByTagName("y").item(pos).getTextContent();**

**return Double.*parseDouble*(s);**

**}**

**public void setY(int pos, double val) {**

**doc.getDocumentElement().getElementsByTagName("y").item(pos).setTextContent(val+"");**

**}**

**public Element newElement(String date, double x, double y) {**

**Element data = doc.createElement("data");**

**Attr attr = doc.createAttribute("date");**

**attr.setValue(date.trim());**

**data.setAttributeNode(attr);**

**Element elemX = doc.createElement("x");**

**elemX.appendChild(doc.createTextNode(x+""));**

**data.appendChild(elemX);**

**Element elemY = doc.createElement("y");**

**elemY.appendChild(doc.createTextNode(y+""));**

**data.appendChild(elemY);**

**return data;**

**}**

**public void addElement(Element data) {**

**this.doc.getDocumentElement().appendChild(data);**

**}**

**public void removeElement(int pos) {**

**Node el = doc.getDocumentElement().getElementsByTagName("data").item(pos);**

**doc.getDocumentElement().removeChild(el);**

**}**

**public void insertElementAt(int pos, Node nd) {**

**Node el = doc.getDocumentElement().getElementsByTagName("data").item(pos);**

**doc.getDocumentElement().insertBefore(nd, el);**

**}**

**public void replaceElementAt(int pos, Node nd) {**

**Node el = doc.getDocumentElement().getElementsByTagName("data").item(pos);**

**doc.getDocumentElement().replaceChild(nd, el);**

**}**

**}**

Лістинг 3 –клас DataSheetToXML

**public class DataSheetToXML {**

**private static DOMDataSheet *DOMdataSheet*;**

**public static void ToXML(DataSheet datasheet, String s) throws FileNotFoundException, TransformerException{**

***DOMdataSheet* = new DOMDataSheet(XmlDOMParser.*createDoc*());**

**int j = 0;**

**Iterator<Data> i = datasheet.getArr().iterator();**

**while ( i.hasNext() ) {**

**Data dat = datasheet.getDataItem(j);**

***DOMdataSheet*.addElement(*DOMdataSheet*.newElement(dat.getDate(),dat.getX(), dat.getY()));**

**i.next();**

**j++;**

**}**

**XmlDOMParser.*CastToXML*(s, *DOMdataSheet*.getDoc());**

**}**

**}**