УДК 004

Разработка с помощью библиотеки NetCdf для Java консольного приложения для работы с метеорологическими данными

Семченко Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
студент

Аннотация
В данной статье описан процесс создания консольного приложения, которое будет находить выбранные метеорологические данные, скачивать их и расписывать, что в них находится
Ключевые слова: NetCDF, Java, библиотека, погода

Development of a console application for working with meteorological data using the NetCdf library for Java

Semchenko Regina Viktorovna
Sholom-Aleichem Priamursky State University
student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
student

Abstract

This article describes the process of creating console applications that will be in the selected meteorological data
Keywords: NetCDF, Java, library, weather

NetCDF - это набор программных библиотек и машинно-независимых форматов данных, которые поддерживают создание, доступ и совместное использование научных данных, ориентированных на массивы. Это также стандарт сообщества для обмена научными данными. Unidata Program Center поддерживает программные интерфейсы netCDF для C , C ++ , Java и Fortran . Интерфейсы программирования также доступны для Python, IDL, MATLAB, R, Ruby и Perl.

Цель данной статьи разработать консольное приложение, которое будет по запросу скачивать данные в формате «.nc», считывать их и выводить данные в консоль.

С.Н. Королев и В.С. Суетин провели разработку программного комплекса, который позволяет распаковывать и визуализировать файлы метеорологических данных второго уровня[1]. В своей работе А.А. Недоступ и А.О. Ражев рассмотрели задачи по созданию базы данных, где будут храниться топологические данные мест лова[2]. М.В. Иванчик и А.М. Иванчик выполнили разработку программы, которая преобразоваывает файлы формата «GRIB» в файлы формата «NetCDF» [3]. М.А. Зимин и И.Р. Давлиев рассмотрели в ствоей статье проблемы возникающие при работе с данными изучения атмосферных явлений [4]. В своей статье А.Г. Титов, Е.П. Гордов и И.Г. Окладников разработали приложение для веб-сервисов которые помогают при работе с данными NetCDF[5].

Для начала создадим файл с расширением “.java” и напишем основные классы, которые будут задавать вопросы в консоли, для того, чтобы узнать, что хочет пользователь (рис.1).

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class Choices {
    private final HtmlParser __parser;
    private final Map<String, String> __categories;
    private final Map<String, String> __types;
    private final Map<String, String> __sources;
    private int count = 0;

    public Choices() {
        __parser = new HtmlParser();
        __categories = new HashMap<>();
        __types = new HashMap<>();
        __sources = new HashMap<>();
        __categories.put("1", "Атмосферные осадки");
    }
```

Рисунок 1 – Варианты ответов

Далее создадим класс, который будет считывать ответ на вопрос о выборе метеорологического центра (рис.2).

```
public void selectSource() {
    count = 0;
    Scanner scanner = new Scanner(System.in);
    System.out.println("Выберите метеорологический центр: ");
    __sources.forEach((name, link) -> {
        count++;
        System.out.println(count + ")" + " " + name);

    });
    int choice = scanner.nextInt();
    __parser.parseIt(__sources.get(__sources.keySet().toArray()[choice - 1]));

}
```

Рисунок 2 – Метеорологический центр

Далее дадим пользователю список метеорологических центров и в зависимости как ответил на предыдущий вопрос пользователь, который будет задав в следующем классе, будем давать ссылки на сайты, где хранятся данные именно с выбранными данными (рис.3).

```
public void selectTypes() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Выберите тип сбора данных: ");
    __types.forEach((id, name) -> System.out.println(id + " " + name));
    int choice = scanner.nextInt();
    switch (choice) {
        case 1:
            System.out.println("Вы выбрали: " + __types.get(String.valueOf(choice)));
            __sources.put("NASA GIS", "http://research.jisao.washington.edu/data_sets/precip_eofts/");
            __sources.put("National Centers for Environmental Prediction (NCEP) Climate Prediction Center", "http://research.jisao.washington.edu/data_se
            __sources.put("University of Delaware", "http://research.jisao.washington.edu/data_sets/ud/");
            __sources.put("University of Washington", "http://www.cses.washington.edu/data/gridded.shtml");
            selectSource();
            break;
        case 2:
            System.out.println("Вы выбрали: " + __types.get(String.valueOf(choice)));
            __sources.put("NOAA NCEP Climate Prediction Center Merged Analysis of Precipitation", "http://research.jisao.washington.edu/data_sets/cmap_pr
            __sources.put("Global Precipitation Climatology Project (GPCP)", "http://research.jisao.washington.edu/data_sets/gpcp/");
            __sources.put("Legates / MSU Precipitation Climatology", "http://research.jisao.washington.edu/legates_msu/");
            __sources.put("GOES Precipitation Index (GPI)", "http://research.jisao.washington.edu/data_sets/gpi/");
            __sources.put("Outgoing Longwave Radiation", "http://research.jisao.washington.edu/data_sets/olr/index.html");
            __sources.put("Special Sensor Microwave / Imager (SSM/I)", "http://research.jisao.washington.edu/data_sets/ssmi/nesdis/");
            selectSource();
            break;
        default:
            System.out.println("Неверный выбор!");
            break;
    }
```

Рисунок 3 – Список метеоцентров

Создадим класс, где пользователю будут даваться на выбор два разных способа сбора данных и в зависимости от ответа будет воспроизведен предыдущий вопрос (рис.4).

```
public void selectCategories() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Выберите категорию: ");
    __categories.forEach((id, name) -> System.out.println(id + " " + name));
    int choice = scanner.nextInt();
    if (choice == 1) {
        System.out.println("Вы выбрали: " + __categories.get(String.valueOf(choice)));
        __types.put("1", "С помощью дождемера: ");
        __types.put("2", "С помощью спутника: ");
        selectTypes();
        //Добавить типы по итогам исследования
    }
}
```

Рисунок 4 – Выбор способа измерения

На этом с файлом закончили, создадим следующий файл, который будет отвечать за парсинг страниц метеоцентров, подключим все необходимые функции и создадим первый класс, который возвращает первые переменные (рис.5).

```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class HtmlParser {
    private final Map<String, String> map;
    private final Scanner scanner;

    public HtmlParser(){
        map = new HashMap<>();
        scanner = new Scanner(System.in);
    }
```

Рисунок 5 – Добавление библиотек

Следующим классом будет самый большой класс, в нем разместится функция считывания данных, функция построение диаграммы и полученных данных и вывод данных пользователю (рис.6-8).

```
public void parseIt(String url) {
    String urlToParse;
    int count = 0;

    // Проверяем ссылку
    if (!url.startsWith("http")){
        urlToParse = "http://" + url;
    }
    else urlToParse = url;

    // Получаем страницу
    Document doc;
    try {
        doc = Jsoup.connect( url: "" + urlToParse).get();
        String title = doc.title();
        System.out.println("_____");
        System.out.println(title);
        System.out.println("_____");
        System.out.println("\nФайлы, найденные на выбранном ресурсе: \n");

        // Получаем все элементы из документа по тэгу
        Elements links = doc.getElementsByTag( tagName: "a");
        for (Element link : links) {
            String linkHref = link.attr( attributeKey: "href");
            String linkText = link.text();
```

Рисунок 6 – Считывание данных

```
        if (linkHref.endsWith(".nc")){
            count++;
            System.out.print(count + ") ");

            if (linkHref.startsWith("https")){
                String LinK = linkHref.replaceAll( regex: "https", replacement: "http");
                System.out.println(linkText + " " + LinK);
                map.put(String.valueOf(count), LinK);
            }
            else if (linkHref.startsWith("http")) {
                System.out.println(linkText + " " + linkHref);
                map.put(String.valueOf(count), linkHref);
            }
            else if (linkHref.startsWith("ftp")){ // ftp ссылки не работают при чтении файлов
                System.out.println(linkText + " " + linkHref + " ftp ссылки недоступны для чтения");
                map.put(String.valueOf(count), linkHref);
            }
            else{
                String urlToDownload = urlToParse.substring(urlToParse.indexOf("h"), urlToParse.lastIndexOf( str: "/")) + "/" + linkHref;
                System.out.println(linkText + "\t" + urlToDownload);
                map.put(String.valueOf(count), urlToDownload.replaceAll( regex: "https", replacement: "http"));
            }
        }
    }
```

Рисунок 7 - Скачивание данных

```
            System.out.println("Для открытия файла введите номер: ");
            int answer = scanner.nextInt();
            Reader reader = new Reader(map.get(String.valueOf(answer)));
            reader.readWholeFile();
        }
        catch (IOException e) {
            System.out.println("Недействительная ссылка!");
        }
    }
}
```

Рисунок 8 – открытие данных

Файл готов, создадим еще один файл и напишем в нем функции вывода данных (рис. 9-10).

```
import org.jfree.ui.RefineryUtilities;
import ucar.ma2.Array;
import ucar.ma2.InvalidRangeException;
import ucar.nc2.NCdumpW;
import ucar.nc2.NetcdfFile;
import ucar.nc2.Variable;
import ucar.nc2.dataset.NetcdfDataset;

import java.io.IOException;
import java.util.Scanner;


public class Reader {
    private NetcdfFile nc = null;
    private final String fileName;


    public Reader(String filePath){
        fileName = filePath;
        Scanner scanner = new Scanner(System.in);
    }

    public void readWholeFile(){
        try {
            nc = NetcdfDataset.open(fileName, cancelTask: null);
        }
```

Рисунок 9 – Добавление библиотек

```
public void readWholeFile(){
    try {
        nc = NetcdfDataset.open(fileName,  cancelTask: null);
    }
    catch (IOException ioe) {
        System.out.println("При попытке открыть файл " + fileName + " произошла ошибка: " + ioe);


    } finally {
        if (null != nc) try {
          nc.close();
        } catch (IOException ioe) {
            System.out.println("При попытке закрыть файл " + fileName + " произошла ошибка: " + ioe);
        }
    }
}
```

Рисунок 10 – открытие файлов

Теперь осталось создать файл с главным классом main и вписать в него запуск основной функции, после чего приступить к тесту приложения (рис.11).

```
public class Main {
    public static void main(String[] args) {
        Choices choices = new Choices();
        choices.selectCategories();

    }

}
```

Рисунок 11 – Класс Main

Все, написание приложения готово, теперь можно опробывать его в действии. Запустим код и ответим на вопросы от программы, до построения диаграммы (рис.12-15).

```
C:\Users\79644\.jdks\corretto-1.8.0_252\bin\java.exe ...
Выберите категорию:
1 Атмосферные осадки
1
Вы выбрали: Атмосферные осадки
Выберите тип сбора данных:
1 С помощью дождемера:
2 С помощью спутника:
1
Вы выбрали: С помощью дождемера:
Выберите метеорологический центр:
1) NASA GIS
2) University of Delaware
3) National Centers for Environmental Prediction (NCEP) Climate Prediction Center
4) University of Washington
```

Рисунок 12 – Ответ на вопрос

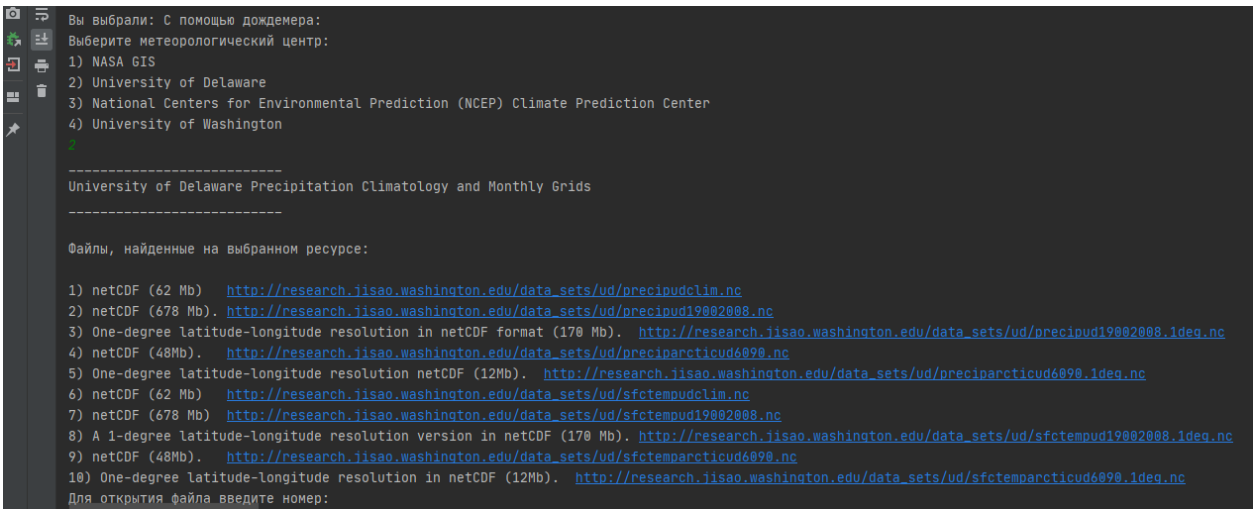

Рисунок 13 – Ответ на вопросы

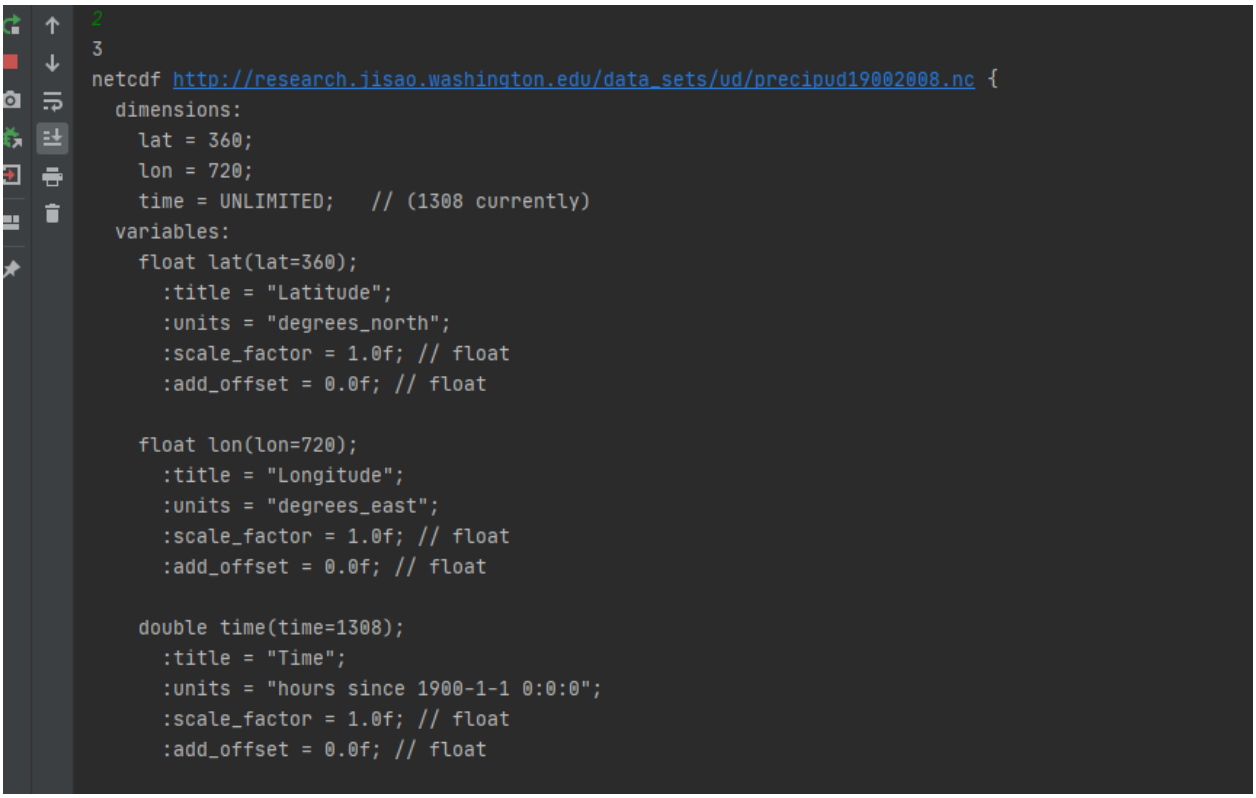

Рисунок 14 – Вывод данных в консоль

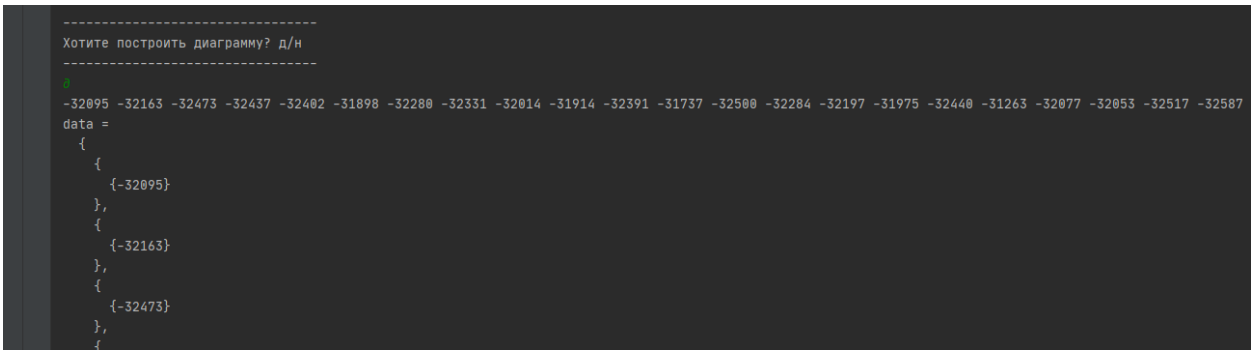

Рисунок 15 – Вывод данных диаграммы

При работе приложение вывело в консоль данные о времени, координаты места от куда считывались данные. Данные были выбраны с 1900 по 2009 год и при построении диаграммы можно заметить количество осадков в каждом год (рис.16).

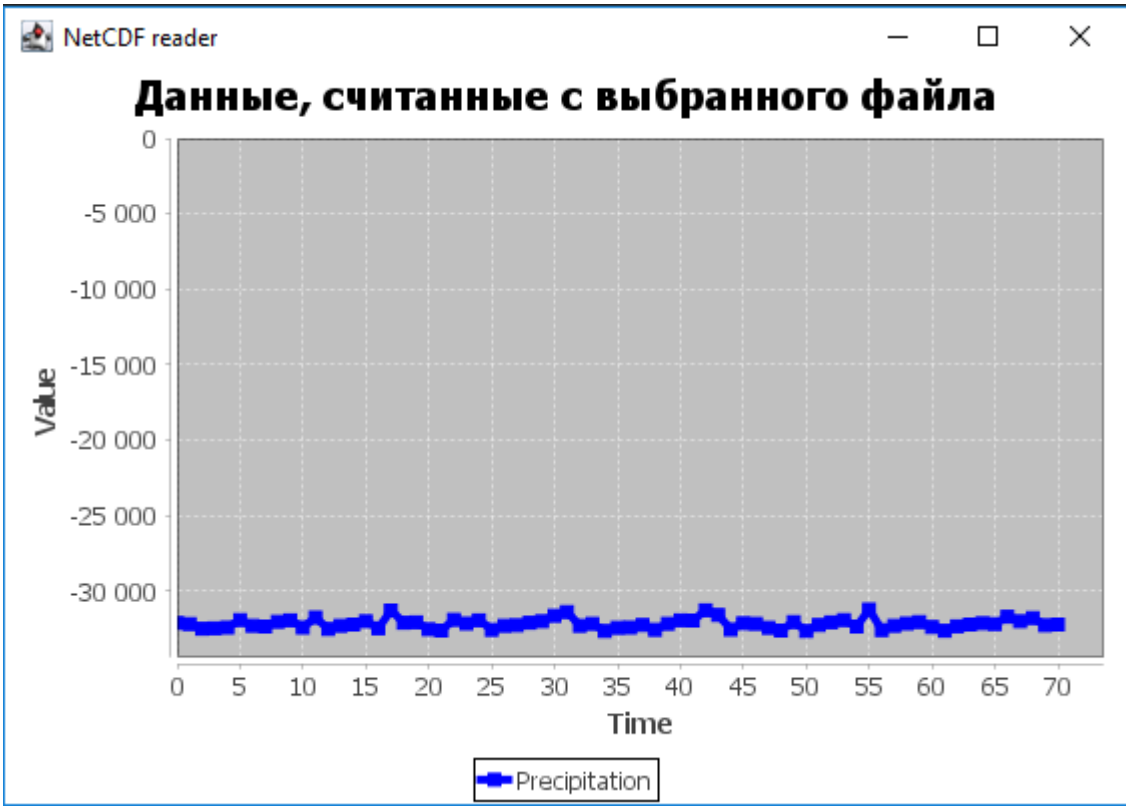

Рисунок 16 – Диаграмма с данными

Процесс создания программы завершен. С помощью данного приложения можно легко считать данные и построить диаграмму на современных данных, поменять сайты на другие регионы и решать какие-либо задачи по прогнозированию.

В данной статье было разработано консольное приложение по нахождению и считыванию метеорологических данных и построению диаграммы на основе этих данных.

Библиографический список

1. С.Н. Королев, В.С. Суетин Программный комплекс для распаковки и визуализации файлов данных дистанционного зондирования второго уровня в формате netcdf// Современные научные исследования и инновации. 2017. № 7-5 (43). С. 47-55.
2. А.А. Недоступ, О.А. Ражев Определение топологии места лова по батиметрическим картам формата netcdf // Труды Международного симпозиума «Надежность и качество». 2014. №5. С. 14-20.
3. М.В. Иванчик, А.М. Иванчик. Программа для преобразования файлов метеорологического прогноза моделью ета в netcdf формат базы данных

метеорологических прогнозов для акватории черного моря // Автоматика. Вычислительная техника. 2012. №1. С. 24-30.

4. М.А. Зимин, И.Р. Давлиев Работа с данными netcdf для анализа приземных температур на южном урале // Современные научные исследования и инновации. 2017. № 1 . С. 75-79.
5. А.Г. Титов, Е.П. Гордов, Окладников И.Г. Архитектура приложения веб-картинга для работы с данными netcdf // Труды Международного симпозиума «Надежность и качество». 2017. №2. С. 17-25.