Reading and writing netcdf scalar data with Java

Java class example adapter for netcdf scalar time series

```
package netcdfexample.reading;

import nl.wldelft.util.TextUtils;
import ucar.ma2.Array;
import ucar.ma2.ArrayDouble;
import ucar.ma2.ArrayFloat;
import ucar.ma2.DataType;
import ucar.ma2.InvalidRangeException;
import ucar.nc2.Attribute;
import ucar.nc2.Dimension;
import ucar.nc2.NetcdfFile;
import ucar.nc2.NetcdfFileWriteable;
import ucar.nc2.Variable;
import ucar.nc2.dataset.NetcdfDataset;
import ucar.nc2.units.DateUnit;

import java.io.File;
import java.io.IOException;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

public class NetcdfScalarAdapter {

    private static final String DISCHARGE_VARIABLE_NAME = "discharge";
    public static final String LON = "lon";
    public static final String LAT = "lat";
    public static final String Y = "y";
    public static final String X = "x";
    public static final String TIME = "time";

    public static void main(String[] args) throws Exception {

        if (args.length != 1)
            throw new Exception("Specify only the netcdf run file as argument");

        File runFile = new File(args[0]);
        if (!runFile.exists())
            throw new Exception("Can not find run file specified as argument " + runFile);


        //Reading netcdf run file
        Path runPath = runFile.getParentFile().toPath();
        NetcdfDataset netcdfRunFileDataset = new NetcdfDataset(NetcdfDataset.openFile(runFile.getAbsolutePath(),
null));

        Variable startTimeVar = netcdfRunFileDataset.findVariable("start_time");
        //Start and end time can be used for model specific settings
        double startDateTime = startTimeVar.readScalarDouble();
        double endDateTime = netcdfRunFileDataset.findVariable("end_time").readScalarDouble();
        Attribute units = startTimeVar.findAttribute("units");
        String tunitsString = units.getStringValue();
        DateUnit referenceUnit = new DateUnit(tunitsString);
        //The time of date origin can be used to determine the real time values
        long currentReferenceTime = referenceUnit.getDateOrigin().getTime();
        //The variable work_dir specifies the working directory relative to the directory of the run file
        String relativeWorkDirString = netcdfRunFileDataset.findVariable("work_dir").readScalarString().trim();
        String workDir;
        //A point "." means the work dir is the same as the directory of the run file
        if (TextUtils.equals(relativeWorkDirString, ".")) {
            workDir = runPath.toString();
        } else {
```

```
            workDir = runPath.resolve(new File(relativeWorkDirString).toPath()).toString();
        }

        //Reading variable that specifies input netcdf files
        Variable inputNetcdfVar = netcdfRunFileDataset.findVariable("input_netcdf_files");
        Array inputNetcdfArray = inputNetcdfVar.read();
        Object inputTimeSeriesFilesNDArrayObject = inputNetcdfArray.copyToNDJavaArray();
        char[][] inputTimeSeriesFilesChar = (char[][]) inputTimeSeriesFilesNDArrayObject;

        //Converting 2 dimensional char array to list of strings
        List<String> inputTimeSeriesFilesList = new ArrayList<>();
        for (char[] chArray : inputTimeSeriesFilesChar) {
            inputTimeSeriesFilesList.add(String.valueOf(chArray).trim());
        }
        //Done reading netcdf run file

        //Processing input files
        for (String inputFilePath : inputTimeSeriesFilesList) {
            //The input netcdf files are also specified as a relative path compared to the directory of the run
file
            File inputFileRelativePath = new File(inputFilePath);
            String absoluteFilePath = runPath.resolve(inputFileRelativePath.toPath()).toString();
            File inputNetcdfFile = new File(absoluteFilePath);
            //Should not happen often in practice because run file only specifies input netcdf files that are
actually exported
            //But could happen when running adapter with older run info file
            if (!inputNetcdfFile.exists())
                throw new RuntimeException("Input file does not exist: " + absoluteFilePath);
            processDischargeScalarStationsData(absoluteFilePath, new File(workDir), tunitsString);
        }
    }

    //Example how to read Scalar data from netcdf file exported by FEWS
    public static void processDischargeScalarStationsData(String absoluteFilePath, File workDir, String
tunitsString) throws Exception {
        NetcdfFile netcdfDatasetInputFile = new NetcdfDataset(NetcdfDataset.openFile(absoluteFilePath, null));
        try {
            //Reading discharge variable
            Variable dischargeVariable = netcdfDatasetInputFile.findVariable(DISCHARGE_VARIABLE_NAME);
            if (dischargeVariable == null)
                throw new Exception("Tide variable '" + DISCHARGE_VARIABLE_NAME + "' not found in example.nc");
            Array waterLevelArray = dischargeVariable.read();
            float[][] floatValues = (float[][]) waterLevelArray.copyToNDJavaArray();

            //Reading time values
            //First dimension is time (second is stations)
            Dimension timeDimension = dischargeVariable.getDimension(0);
            Variable timeVariable = netcdfDatasetInputFile.findVariable(timeDimension.getName());
            long[] convertedTimeArray = readTimes(timeVariable);
            //Reading latitude values
            Variable latitudeVariable = netcdfDatasetInputFile.findVariable(LAT);
            Array latitudeArray = latitudeVariable.read();
            double[] latitudeValues = (double[]) latitudeArray.copyTo1DJavaArray();
            //Reading longitude values
            Variable longitudeVariable = netcdfDatasetInputFile.findVariable(LON);
            Array longitudeArray = longitudeVariable.read();
            double[] longitudeValues = (double[]) longitudeArray.copyTo1DJavaArray();
            //Reading y values
            Variable yVariable = netcdfDatasetInputFile.findVariable(Y);
            Array yArray = yVariable.read();
            double[] yValues = (double[]) yArray.copyTo1DJavaArray();
            //Reading x values
            Variable xVariable = netcdfDatasetInputFile.findVariable(X);
            Array xArray = xVariable.read();
            double[] xValues = (double[]) xArray.copyTo1DJavaArray();

            File outputNetcdfFile = new File(workDir, "scalar.nc");
            NetcdfFileWriteable scalarNetcdf = NetcdfFileWriteable.createNew(outputNetcdfFile.getPath(), false);
            try {
                writeNetcdfScalar(scalarNetcdf, floatValues, tunitsString, convertedTimeArray, latitudeValues,
longitudeValues, xValues, yValues);
```

```
            } finally {
                scalarNetcdf.close();
            }
        } finally {
            netcdfDatasetInputFile.close();
        }
    }

    //Example how to write netcdf for FEWS
    private static void writeNetcdfScalar(NetcdfFileWriteable netcdfFile, float[][] floatValues, String
timeUnitsString, long[] convertedTimeArray, double[] latitudeValues, double[] longitudeValues, double[]
xValues, double[] yValues) throws IOException, InvalidRangeException {

        Dimension timeDimension = netcdfFile.addDimension(TIME, floatValues.length);
        Dimension stationDimension = netcdfFile.addDimension("stations", floatValues[0].length);

        ArrayList<Dimension> timeDimensions = new ArrayList<>();
        timeDimensions.add(timeDimension);
        addNetcdfVariable(netcdfFile, timeDimensions, TIME, DataType.DOUBLE, "time", "time", timeUnitsString,
"T", -999, "y x");

        ArrayList<Dimension> stationDimensions = new ArrayList<>();
        stationDimensions.add(stationDimension);

        addNetcdfVariable(netcdfFile, stationDimensions, X, DataType.DOUBLE, "projection_x_coordinate", "x
coordinate according to CH1903", "degrees_east", "X", Integer.MIN_VALUE, null);
        addNetcdfVariable(netcdfFile, stationDimensions, Y, DataType.DOUBLE, "projection_y_coordinate", "y
coordinate according to CH1903", "degrees_north", "Y", Integer.MIN_VALUE, null);
        addNetcdfVariable(netcdfFile, stationDimensions, LAT, DataType.DOUBLE, "latitude", "latitude",
"degrees_north", "Y", Integer.MIN_VALUE, null);
        addNetcdfVariable(netcdfFile, stationDimensions, LON, DataType.DOUBLE, "longitude", "longitude",
"degrees_east", "X", Integer.MIN_VALUE, null);

        ArrayList<Dimension> dischargeDimensions = new ArrayList<>();
        dischargeDimensions.add(timeDimension);
        dischargeDimensions.add(stationDimension);

        addNetcdfVariable(netcdfFile, dischargeDimensions, DISCHARGE_VARIABLE_NAME, DataType.DOUBLE,
"discharge", "discharge", "m", null, -999, "y x");

        //First define all variable and dimensions, then create the netcdf, after creation values can be
written to variables
        netcdfFile.create();

        //Write values to variable
        writeTime(netcdfFile, convertedTimeArray);
        writeDoubleVariable1D(netcdfFile, xValues, X);
        writeDoubleVariable1D(netcdfFile, yValues, Y);
        writeDoubleVariable1D(netcdfFile, latitudeValues, LAT);
        writeDoubleVariable1D(netcdfFile, longitudeValues, LON);
        writeDischarge(netcdfFile, floatValues);
    }

    private static void addNetcdfVariable(NetcdfFileWriteable netcdfFile, ArrayList<Dimension>
stationDimensions, String variableName, DataType dataType, String standardName, String longName, String units,
String axis, int fillValue, String coordinates) {
        netcdfFile.addVariable(variableName, dataType, stationDimensions);
        if (standardName != null) netcdfFile.addVariableAttribute(variableName, "standard_name", standardName);
        if (longName != null) netcdfFile.addVariableAttribute(variableName, "long_name", longName);
        if (units != null) netcdfFile.addVariableAttribute(variableName, "units", units);
        if (axis != null) netcdfFile.addVariableAttribute(variableName, "axis", axis);
        if (fillValue != Integer.MIN_VALUE) netcdfFile.addVariableAttribute(variableName, "_FillValue",
fillValue);
        if (coordinates != null) netcdfFile.addVariableAttribute(variableName, "coordinates", coordinates);
    }

    private static void writeDischarge(NetcdfFileWriteable netcdfFile, float[][] floatValues) throws
IOException, InvalidRangeException {
        ArrayFloat.D2 dischargeArray = new ArrayFloat.D2(floatValues.length, floatValues[0].length);
        for (int i = 0; i < floatValues.length; i++) {
            for (int j = 0; j < floatValues[0].length; j++) {
```

```
                float value = floatValues[i][j];
                if (Float.isNaN(value)) value = -999f;
                dischargeArray.set(i, j, value);
            }
        }
        netcdfFile.write(DISCHARGE_VARIABLE_NAME, dischargeArray);
    }

    private static void writeDoubleVariable1D(NetcdfFileWriteable netcdfFile, double[] latValues, String
variableName) throws IOException, InvalidRangeException {
        ArrayDouble.D1 latArray = new ArrayDouble.D1(latValues.length);
        for (int i = 0; i < latValues.length; i++) {
            latArray.set(i, latValues[i]);
        }
        netcdfFile.write(variableName, latArray);
    }

    private static void writeTime(NetcdfFileWriteable netcdfFile, long[] convertedTimeArray) throws
IOException, InvalidRangeException {
        ArrayDouble.D1 timeArray = new ArrayDouble.D1(convertedTimeArray.length);
        for (int i = 0; i < convertedTimeArray.length; i++) {
            timeArray.set(i, convertedTimeArray[i]);
        }
        netcdfFile.write(TIME, timeArray);
    }

    private static long[] readTimes(Variable timeVariable) throws IOException {
        if (timeVariable == null) throw new RuntimeException("Time variable not present");

        //read times.
        Array timeArray = timeVariable.read();
        double[] times = (double[]) timeArray.get1DJavaArray(Double.class);

        //convert times.
        long[] convertedTimes = new long[times.length];
        DateUnit dateUnit = readTimeUnit(timeVariable);
        if (dateUnit == null) {
            throw new IOException("Invalid date time unit string has been coded in the file: '" + timeVariable.
getUnitsString() +
                    "'.  Unit string should be for example: 'seconds since 2012-01-30 00:00:00'");

        }
        for (int i = 0; i < times.length; i++) {
            Date date = dateUnit.makeDate(times[i]);
            if (date != null) {
                convertedTimes[i] = date.getTime();
            } else {//if date is null.
                convertedTimes[i] = 0;
            }
        }
        return convertedTimes;
    }

    public static DateUnit readTimeUnit(Variable timeVariable) {
        try {
            String unitString = timeVariable.getUnitsString();

            //if present, replace . by : in the timeZone specification in the unitString,
            //e.g. change "days since 2011-09-19 06:0:0.0 0.00" to "days since 2011-09-19 06:0:0.0 0:00".
            //This is a workaround implemented for FEWS-6544.
            String[] parts = unitString.split("\\s+");
            if (parts.length > 0 && parts[parts.length - 1].matches(".?\\d{1,2}\\.\\d{2}")) {
                //if a timeZone specification is present, then it is always the last part, see
                //http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.5/cf-conventions.html#time-coordinate
                parts[parts.length - 1] = parts[parts.length - 1].replaceFirst("\\.", ":");
                StringBuilder buffer = new StringBuilder(parts[0]);
                for (int n = 1; n < parts.length; n++) {
                    buffer.append(' ').append(parts[n]);
                }
                unitString = buffer.toString();
            }
```

```
            return new DateUnit(unitString);
        } catch (Exception e) {
            //if the given variable does not have a unit of time.
            return null;
        }
    }

}
```