Translation of Some Star Catalogs to the XEphem Format

Richard J. Mathar*
Hoeschstr. 7, 52372 Kreuzau, Germany
(Dated: February 3, 2018)

The text lists Java programs which convert star catalogs to the format of the XEphem visualiser. The currently supported input catalog formats are (i) the data base of the orbital elements of the Minor Planet Center (MPC) of the International Astronomical Union (IAU), (ii) the data base of the Hipparcos main catalog or the variants of the Tycho-1 or Tycho-2 catalogs, (iii) the systems in the Washington Double Star catalog, (iv) the Proper Motions North and South catalogs.

PACS numbers: 95.10.Km, 95.80.+p

I. AIM AND SCOPE

The program listed here supports loading star catalogs into Downey's XEphem program to populate the chart of visible objects in sections of the sky, in the format of `xephem 3.7.7`.

The envisioned use is that the translated star catalogs can be included in the display by using the `Data`→`Files` menue of the `xephem` graphical user interface (GUI), and by clicking on the `Files` menue of the GUI that pops up.

II. COMPILATION

The Java source code files of the Appendix are copied to their subdirectories as indicated and compiled from the top directory with

```
javac -cp . de/mpg/mpia/xephem/*.java
```

Which program is used depends on the type of the input catalog as detailed in the subsequent sections.

III. INVOCATION

A. MPC Ephemerides

Once the program and the auxiliary data files are compiled according to Appendix A, the program is run with a command line of the form

```
gunzip MPCORB.DAT.gz
java -cp . de.mpg.mpia.xephem.Mpc2Xeph [-a] MPCORB.DAT > ~/.xephem/Mpc.edb
```

The mandatory command argument must be the name of the uncompressed data file of the MPC taken from `https://www.minorplanetcenter.net/data`.

The option `-a` triggers that all entries of the data file are rendered to the XEphem format; if the option is missing, only the approximately 510 thousand objects up to the blank line that separates the enumerated objects from the preliminary objects are transformed, otherwise all approximately 750 thousand objects. With the option the output contains more than 80 MBytes, without more than 50 MBytes.

The XEphem Catalog is written to standard output, and the syntax illustrates that this format is usually written into the `.xephem` subdirectory in the user's home directory.

The commets data base can immediately be downloaded in the `XEphem` format from `https://www.minorplanetcenter.net/iau/Ephemerides/Comets/Soft03Cmt.txt`; so I am not proposing a tool here for that data base.

B. Hipparcos and Tycho-1 Main Catalog

The main catalog of roughly 118 thousand stars can be downloaded from `ftp://cdsarc.u-strasbg.fr/ftp/cats/I/239/` and should be decompressed with `gunzip` such that the file `hip_main.dat` is somewhere in the local file system. Optionally the Tycho-1 catalog of more than 1 million stars can be downloaded from the same directory and should be decompressed with `gunzip` such that the file `tyc_main.dat` is somewhere in the local file system. The mandatory command argument should be the name of one of these uncompressed catalog files. The class/program `Hip2Xeph` deals with both types of catalogs; it looks at the first byte in each line of the files to branch into the appropriate action.

```
java -cp . de.mpg.mpia.xephem.Hip2Xeph hip_main.dat > ~/.xephem/hipparcos.edb
java -cp . de.mpg.mpia.xephem.Hip2Xeph [-a] tyc_main.dat > ~/.xephem/tycho1.edb
```

The XEphem Catalog is written to standard output, and the syntax illustrates that this format is usually written into the `.xephem` subdirectory in the user's home directory.

The Tycho-1 stars which have no assigned magnitude are by default not translated to the standard output. The option `-a` triggers that all Tycho-1 stars are translated; Xephem will assign a default magnitude of 1 to these.

* http://www.mpia.de/~mathar

C. Tycho-2 Catalog

The sub-catalogs of all stars can be downloaded from ftp://cdsarc.u-strasbg.fr/ftp/cats/I/259/ and should be decompressed with `gunzip` such that the files `tyc2.dat.00` —`tyc2.dat.19` of interest are somewhere in the local file system. The two supplementary files `suppl_?.dat` can be parsed as well. The mandatory command argument should be the name of one or more of these uncompressed catalog files.

```
java -cp . de.mpg.mpia.xephem.Tyc22Xeph [-a] suppl_1.dat > ~/.xephem/tycho2.edb
java -cp . de.mpg.mpia.xephem.Tyc22Xeph [-a] suppl_2.dat >> ~/.xephem/tycho2.edb
java -cp . de.mpg.mpia.xephem.Tyc22Xeph [-a] tyc2.dat.00 >> ~/.xephem/tycho2.edb
java -cp . de.mpg.mpia.xephem.Tyc22Xeph [-a] tyc2.dat.01 >> ~/.xephem/tycho2.edb
...
java -cp . de.mpg.mpia.xephem.Tyc22Xeph [-a] tyc2.dat.19 >> ~/.xephem/tycho2.edb
```

With the file name expansion mechanism of the Unices all catalog files of the current working directory can be assembled in one go:

```
java -cp . de.mpg.mpia.xephem.Tyc22Xeph [-a] suppl_?.dat tyc2.dat.?? > ~/.xephem/tycho2.edb
```

The XEphem Catalog is written to standard output, and the syntax illustrates that this format is usually written into the `.xephem` subdirectory in the user's home directory, accumulating the regions of interest by appending the sub-catalogs.

The stars which have no assigned magnitude are by default not translated to the standard output. The option `-a` triggers that all Tycho-2 stars are translated; Xephem will assign a default magnitude of 1 to these.

D. WDS Catalog

The Washington Double Star catalog can be downloaded from http://ad.usno.navy.mil/wds/wdstext.html; our software supports only the non-frame versions. The command argument should be the name of one or more uncompressed data file.

```
java -cp . de.mpg.mpia.xephem.Wds2Xeph wdsweb_summ2.txt > ~/.xephem/WDS.edb
```

One can also accumulate the 4 sub-catalogs in the style of

```
java -cp . de.mpg.mpia.xephem.Wds2Xeph wdsnewerweb1.txt > ~/.xephem/WDS.edb
java -cp . de.mpg.mpia.xephem.Wds2Xeph wdsnewerweb2.txt >> ~/.xephem/WDS.edb
java -cp . de.mpg.mpia.xephem.Wds2Xeph wdsnewerweb3.txt >> ~/.xephem/WDS.edb
java -cp . de.mpg.mpia.xephem.Wds2Xeph wdsnewerweb4.txt >> ~/.xephem/WDS.edb
```

The file name expansion mechanism of Unix-type operating systems allows us to write this concisely as

```
java -cp . de.mpg.mpia.xephem.Wds2Xeph wdsnewerweb?.txt > ~/.xephem/WDS.edb
```

The XEphem Catalog is written to standard output, and the syntax illustrates that this format is usually written into the `.xephem` subdirectory in the user's home directory.

Lines in the catalog that do not show coordinates in bytes 113–130 but just two dots are not converted.

E. PPM North and South Catalog

The PPM North catalog can be downloaded from ftp://cdsarc.u-strasbg.fr/cats/I/146/, the PPM South catalog from ftp://cdsarc.u-strasbg.fr/cats/I/193/. The program assumes that at least one of the decompressed catalogs `ppm1.dat` and `ppm2.dat` is in the local file system.

```
java -cp . de.mpg.mpia.xephem.Ppm2Xeph ppm[12].dat > ~/.xephem/ppm.edb
```

F. MPC Observatory Codes

The program suite supports the conversion of the approximately 2100 geographic locations of the MPC codes of https://www.minorplanetcenter.net/iau/lists/ObsCodes.html into the XEphem format:

```
java -cp . de.mpg.mpia.xephem.MpcObscode ObsCodes.html > ~/.xephem/xephem_sites
```

It converts the geocentric coordinates of the ObsCodes file to geodetic coordinates by the algorithm proposed by Vermeille [1].

(This is a standalone variant of my earlier decoder [2].) The inherent problem of the input data in ObsCodes.html is the wide variation in quality; if filled with zeros, some of the values of lesser known stations correspond to locations far below sea level or surpassing Karakorum peak heights. The program does not create time zone data because they are not represented in the input data.

Appendix A: Source Code

1. File de/mpg/mpia/xephem/Mpc2Xeph.java

```
package de.mpg.mpia.xephem ;

import java.io.BufferedReader ;
import java.io.FileReader ;
import java.nio.charset.Charset ;

/**
* A translator of the ephemerides of the MPC to the XEphem format
* @since 2018-01-08
* @author R. J. Mathar
*/
class Mpc2Xeph
{
    /*******************************
    * The ASCII file of the minor planets orbital elements.
    * Usually MPCORB.DAT taken from https://www.minorplanetcenter.net/data
    */
    String orbfile ;

    /**
    * Ctor specifying the ASCII file with the MPC elements, one per planet.
    * @param catfname The name of an existing MPC orbital elements file.
    */
    Mpc2Xeph(final String catfname)
    {
        orbfile = catfname ;
    } /* ctor */

    /**
    * Convert the compressed date format to the XEphem month/day/year format
    * @param epoch The MPC format of the epoch
    * @param ascii The character set of the string. Should be US-ASCII.
    * @return MM/DD/YYYY
    */
    private String epoch2Date(final String epoch, final Charset ascii)
    {
        byte[] lets = epoch.getBytes(ascii) ;
        String day = new String() ;
        /* assign I=18xx, J=19xx, K=20xx for the century
        */
        lets[0]  += 18-'I' ;
        day = "" + lets[0] ;
        /* append last two digits of the year
        */
        day += epoch.substring(1,3) ;
        /* prepend day encoded as 1=1,..., V=31
        */
        lets[4]  += 1-'1' ;
        day = "" + lets[4] + "/" + day;

        /* prepend month, 1=Jan, 12 = Dec.
        */
        lets[3]  += 1-'1' ;
        day = "" + lets[3] + "/" + day ;
        return day ;
    } /* epoch2Date */

    /**
    * Convert the compressed date format of the designation to a numerical name.
```

```
* @param fallback The MPC packed format of the designation.
* @param rdes The MPC human-readable designation
* @return The expanded/decompressed integer for the recognized objects.
*   If that field is not available, return the fallback string.
*/
private String desi2NameNum(String fallback, String rdes)
{
    /* preference to the parenthetical number in rdes
    */
    if ( rdes.startsWith("(") )
    {
        /* search for closing parenthesis
        */
        int closPar = rdes.indexOf(")") ;
        return "MPC"+rdes.substring(1,closPar) ;
    }
    else
        return fallback ;
} /* desi2NameNum */

/**
* Convert the compressed date format of the designation to an alternative (conventional) name.
* @param design The MPC packed format of the designation
*   This starts with a quasi-hexadecimal first letter and ends with 4 letters in the range 0000-9999.
* @return The conventional name for the well-known (bigger) objects.
*/
private String desi2NameConv(String rdes)
{
    /* preference to the parenthetical number in rdes
    */
    if ( rdes.startsWith("(") )
    {
        /* search for closing parenthesis and remove the contents inside the parentheses.
        */
        int closPar = rdes.indexOf(")") ;
        return rdes.substring(1+closPar).trim() ;
    }
    else
        return rdes ;
} /* desi2NameConv */

/**
* @brief scan the input file and emit the XEphem file to stdout.
* @param withProvi If true include the provisional entries.
*/
void toXephem(final boolean withProvi)
{
    /* flag whether we've already read the line with dashes
    * Lines in the file before the dashes are header contents.
    * Once the scanner has passed the header it starts taking data.
    */
    boolean readDashes = false ;

    Charset ascii = Charset.forName("US-ASCII") ;
    try
    {
        FileReader istream = new FileReader(orbfile) ;
        BufferedReader dstream = new BufferedReader(istream) ;
        for(;;)
        {
            String lin = dstream.readLine() ;
            if ( lin == null)
                /* end of file */
                break ;
```

```
            if ( lin.startsWith("--------") )
                readDashes = true ;
            else if ( lin.length() < 193)
            {
                /* empty line delimits provisional entries and recognized entries
                * Decide whether we continue beyond that line...
                */
                if ( ! withProvi && readDashes)
                    break ;
            }
            else if ( readDashes)
            {
                /* split the line into fields
                */
                String design = lin.substring(0,7).trim() ;
                String H = lin.substring(8,13).trim() ;
                String G = lin.substring(14,19).trim() ;
                String epoch = lin.substring(20,25).trim() ;
                String M = lin.substring(26,35).trim() ;
                String Peri = lin.substring(37,46).trim() ;
                String Node = lin.substring(48,57).trim() ;
                String Incl = lin.substring(59,68).trim() ;
                String e = lin.substring(70,79).trim() ;
                String n = lin.substring(80,91).trim() ;
                String a = lin.substring(92,103).trim() ;
                String rDes = lin.substring(166,194).trim() ;

                String name = desi2NameConv(rDes) ;
                System.out.print(name) ;
                String name2 = desi2NameNum(design,rDes) ;
                if ( name2.length() > 0 )
                    System.out.print("|" + name2) ;

                System.out.print(",e") ;
                System.out.print("," + Incl) ;
                System.out.print("," + Node) ;
                System.out.print("," + Peri) ;
                System.out.print("," + a) ;
                System.out.print("," + n) ;
                System.out.print("," + e) ;
                System.out.print("," + M) ;
                System.out.print("," + epoch2Date(epoch,ascii) ) ;
                System.out.print(",2000") ;
                System.out.print(",") ;
                if ( H.length() > 0 )
                    System.out.print("H"  + H) ;
                System.out.print("," + G) ;
                System.out.println() ;
            }
        }
    }
    catch (Exception ex)
    {
        System.err.println(ex) ;
        ex.printStackTrace() ;
    }
} /* toXephem */

/**
* @param argv Vector of the command line arguments
* Usage:
* javac -cp . de/mpg/mpia/xephem/*.java
* java -cp . de.mpg.mpia.xephem.Mpc2Xeph [-a] MPCORB.DAT > ~/.xephem/Mpc.edb
```

```
    * This creates a file of roughly 89 MBytes.
    */
    static public void main(String argv[])
    {
        boolean withProvi = false ;
        if ( argv.length == 0 )
        {
            System.err.println("Error: command line argument missing") ;
            System.err.println("Usage: java -cp . de.mpg.mpia.xephem.Mpc2Xeph [-a] MPCORB.DAT") ;
        }
        else
        {
            if ( argv[0].startsWith("-a") )
                withProvi = true ;
            String catfname = argv[argv.length-1] ;

            Mpc2Xeph orb = new Mpc2Xeph(catfname) ;
            orb.toXephem(withProvi) ;
        }

    } /* main */

} /* Mpc2Xeph */
```

2. File de/mpg/mpia/xephem/Hip2Xeph.java

```
package de.mpg.mpia.xephem ;

import java.io.BufferedReader ;
import java.io.FileReader ;
import java.nio.charset.Charset ;

/**
* A translator of the positions of the Hipparcos or Tycho main catalogue to the XEphem format
* @since 2018-01-09
* @author R. J. Mathar
*/
class Hip2Xeph
{
    /*******************************
    * The ASCII file of the main catalog.
    * hip_main.dat of http://vizier.u-strasbg.fr/viz-bin/VizieR?-source=I%2F239
    */
    String catfile ;

    /**
    * Ctor specifying the ASCII file with the MPC elements, one per planet.
    * @param catfname The name of an existing MPC orbital elements file.
    */
    Hip2Xeph(final String catfname)
    {
        catfile = catfname ;
    } /* ctor */

    /*******************************
    * @brief Convert a TYC1-3 catalog designation to a single string.
    *  Sequences of consecutive blanks are replaced by a single hyphen.
    * @param tycNo The string in the TYC field of the Tycho-1 catalog.
    * @return  The string with underscores instead of white space.
    * @since 2018-01-28
    */
    String tyc2name(String tyc123)
```

```
{
    /* proposal to name with dashes: ftp://cdsarc.u-strasbg.fr/ftp/cats/I/259/ReadMe
    */
    return tyc123.replaceAll("\\s+","-") ;
}

/**
* @brief scan the input file and emit the XEphem file to stdout.
* @param allT1 Include the Tycho-1 stars without magnitude.
*   This means that if the magnitude field is empty, and all =true, the star is copied to the output, else not.
*/
void toXephem(boolean allT1)
{
    Charset ascii = Charset.forName("US-ASCII") ;
    try
    {
        FileReader istream = new FileReader(catfile) ;
        BufferedReader dstream = new BufferedReader(istream) ;
        for(;;)
        {
            String lin = dstream.readLine() ;
            if ( lin == null)
                break ;

            /* hip_main.dat has 449 bytes, tyc_main has 350 bytes per line
            * Need at least the lesser of both.
            */
            if ( lin.length() < 350)
                break ;

            /* catalog type: H= hipparcos, T= tycho
            */
            String cat = lin.substring(0,1).trim() ;  // H0, T0

            /* split the line into fields
            */
            if ( cat.startsWith("H") )
            {
                String hipNo = lin.substring(8,14).trim() ; // H1
                String rahms = lin.substring(17,28).trim() ; // H3
                String dedms = lin.substring(29,40).trim() ; // H4
                String vmag = lin.substring(41,46).trim() ; // H5
                String radeg = lin.substring(51,63).trim() ; // H8
                String dedeg = lin.substring(64,76).trim() ; // H9
                String pmra = lin.substring(87,95).trim() ; // H12
                String pmde = lin.substring(96,104).trim() ; // H13
                String sptype = lin.substring(435,447).trim() ; // H76

                System.out.print("HIP "+hipNo) ;
                System.out.print(",f|S") ;
                if ( sptype.length() > 0 )
                    System.out.print("|" + sptype) ;
                System.out.print("," + rahms.replaceAll(" ",":") + "|" + pmra) ;
                System.out.print("," + dedms.replaceAll(" ",":") + "|" + pmde) ;
                System.out.print("," + vmag) ;
            }
            else
            {
                String tyc123 = lin.substring(2,14).trim() ; // T1
                String rahms = lin.substring(17,28).trim() ; // T3
                String dedms = lin.substring(29,40).trim() ; // T4
                String vmag = lin.substring(41,46).trim() ; // T5
                String radeg = lin.substring(51,63).trim() ; // T8
                String dedeg = lin.substring(64,76).trim() ; // T9
```

```
                    String pmra = lin.substring(87,95).trim() ; // T12
                    String pmde = lin.substring(96,104).trim() ; // T13
                    String hipNo = lin.substring(210,216).trim() ; // T31

                    if ( allT1 || vmag.length() >0 )
                    {
                        System.out.print("TYC "+ tyc2name(tyc123)) ;
                        if ( hipNo.length() > 0)
                            System.out.print("|HIP " + hipNo) ;
                        System.out.print(",f|S") ;
                        System.out.print("," + rahms.replaceAll(" ",":") + "|" + pmra) ;
                        System.out.print("," + dedms.replaceAll(" ",":") + "|" + pmde) ;
                        System.out.print("," + vmag) ;
                    }
                }
                System.out.print(",") ;
                System.out.print(",") ;
                System.out.println() ;
            }
        }
        catch (Exception ex)
        {
            System.err.println(ex) ;
            ex.printStackTrace() ;
        }
    } /* toXephem */

    /**
    * @param argv Vector of the command line arguments
    * Usage:
    * javac -cp . de/mpg/mpia/xephem/*.java
    * java -cp . de.mpg.mpia.xephem.Hip2Xeph hip_main.dat > ~/.xephem/Hip.edb
    */
    static public void main(String argv[])
    {
        boolean allT1 = false ;
        if ( argv.length == 0 )
        {
            System.err.println("Error: command line argument missing") ;
            System.err.println("Usage: java -cp . de.mpg.mpia.xephem.Hip2Xeph hip_main.dat") ;
        }
        else
        {
            if ( argv[0].startsWith("-a") )
                allT1 = true ;
            String catfname = argv[argv.length-1] ;

            Hip2Xeph hip = new Hip2Xeph(catfname) ;
            hip.toXephem(allT1) ;
        }

    } /* main */

} /* Hip2Xeph */
```

3. File de/mpg/mpia/xephem/Tyc22Xeph.java

```
package de.mpg.mpia.xephem ;

import java.io.BufferedReader ;
import java.io.FileReader ;
import java.nio.charset.Charset ;
```

```
/**
* A translator of the positions of the Tycho 2 main catalogue or a supplement to the XEphem format
* @since 2018-01-30
* @author R. J. Mathar
*/
class Tyc22Xeph
{
    /*******************************
    * The ASCII file of the main catalog or one of the two supplements.
    * tyc2.dat of http://vizier.u-strasbg.fr/viz-bin/VizieR?-source=I/259
    */
    String catfile ;

    /**
    * Ctor specifying the ASCII file with the Tycho2 stars, one per star.
    * @param catfname The name of an existing file.
    */
    Tyc22Xeph(final String catfname)
    {
        catfile = catfname ;
    } /* ctor */

    /*******************************
    * @brief Delete leading zeros from a string.
    * @return  The string shortened by any strain of leading zeros.
    * @since 2018-01-30
    */
    String delLeadZ(String tyc123)
    {
        return tyc123.replaceFirst("^0+","") ;
    }

    /***********************
    * @brief Convert an angle in radians to blank separated D:M:S format
    * @param deg The angle in degrees or hours.
    *  To use the function to convert hours (RA) into the usual format, divide through 15 before calling this.
    * @return A string of the form +-dd:mm:ss.ss of the value.
    */
    String hexRep(double deg)
    {
        double degabs = Math.abs(deg) ;
        /* We'll writhe D:M:S.ss with two trailing digits, rounded correctly.
        * This represents D+M/60+S.ss/3600., multiplied by 360000 as the integer 360000*D+6000*M+Sss
        * D is up to 90 degrees or up to 24 hours, so that integer is <4e^7 and fitting into the standard 32-bit value.
        */
        int degabsI = (int) (360000.*degabs+0.5) ;
        int d = degabsI/360000 ;
        /* remaining value is 6000*M+Sss */
        degabsI -= d*360000 ;
        int m = degabsI/6000 ;
        /* remaining value is +Sss */
        degabsI -= 6000*m ;
                String out = new String() ;
        /* recover any leading negative sign that was removed above */
        if ( deg < 0)
            out += "-" ;
                out += d  + ":" + m + ":" + String.format("%.2f",degabsI/100.0) ;
                return out ;
        }


    /**
    * @brief scan the input file and emit the XEphem file to stdout.
```

```
* @param allT1 If true, include even the Tycho2 stars without magnitude.
*   This means that if the magnitude field is empty, and all =true, the star is copied to the output, else not.
*/
void toXephem(boolean allT1)
{
    Charset ascii = Charset.forName("US-ASCII") ;
    try
    {
        FileReader istream = new FileReader(catfile) ;
        BufferedReader dstream = new BufferedReader(istream) ;
        for(;;)
        {
            String lin = dstream.readLine() ;
            if ( lin == null)
                break ;

            /* tyc2 has 206 bytes per line, supplements have 122 bytes per line
            */
            if ( lin.length() < 122)
                break ;

            /* flag to indicate that this is one of the two supplement files.
            */
            boolean sup = ( lin.length() > 130) ? false : true ;

            String tyc1 = lin.substring(0,4).trim() ;
            String tyc2 = lin.substring(5,10).trim() ;
            String tyc3 = lin.substring(11,12).trim() ;
            String radeg = lin.substring(15,27).trim() ;
            String dedeg = lin.substring(28,40).trim() ;
            String pmra = lin.substring(41,48).trim() ;
            String pmde = lin.substring(49,56).trim() ;
            String vmag = sup? lin.substring(96,102).trim() : lin.substring(123,129).trim() ;
            String hipNo =  sup ? lin.substring(115,121).trim() : lin.substring(142,148).trim() ;

            if ( radeg.length() > 0 && dedeg.length() > 0 && (allT1 || vmag.length() >0 ))
            {
                System.out.print("TYC "+ delLeadZ(tyc1) + "-" + delLeadZ(tyc2)+ "-" + delLeadZ(tyc3)) ;
                if ( hipNo.length() > 0)
                    System.out.print("|HIP " + hipNo) ;

                double ra = Double.parseDouble(radeg) ;
                double dec = Double.parseDouble(dedeg) ;
                String rahms = hexRep(ra/15.0) ;
                String dedms = hexRep(dec) ;
                System.out.print(",f|S") ;
                System.out.print("," + rahms) ;
                if ( pmra.length() > 0)
                    System.out.print("|" + pmra) ;
                System.out.print("," + dedms) ;
                if ( pmde.length() > 0)
                    System.out.print("|" + pmde) ;
                System.out.print("," + vmag) ;

                System.out.print(",") ;
                System.out.print(",") ;
                System.out.println() ;
            }
        }
    }
    catch (Exception ex)
    {
        System.err.println(ex) ;
        ex.printStackTrace() ;
```

```
        }
    } /* toXephem */

    /**
    * @param argv Vector of the command line arguments
    * Usage:
    * javac -cp . de/mpg/mpia/xephem/*.java
    * java -cp . de.mpg.mpia.xephem.Tyc22Xeph [-a] cyt2.dat.?? suppl_[12].dat
    */
    static public void main(String argv[])
    {
        boolean allT1 = false ;
        if ( argv.length == 0 )
        {
            System.err.println("Error: command line argument missing") ;
            System.err.println("Usage: java -cp . de.mpg.mpia.xephem.Tyc22Xeph [-a] suppl_[12].dat tyc2.dat.??") ;
        }
        else
        {
            /* can all command line arguments (file names or the -a option)
            */
            for(int i=0 ; i < argv.length ; i++)
            {
                if ( argv[i].startsWith("-a") )
                    allT1 = true ;
                else
                {
                    /* render the catalogs in the order of the command line
                    */
                    Tyc22Xeph tyc = new Tyc22Xeph(argv[i]) ;
                    tyc.toXephem(allT1) ;
                }
            }
        }

    } /* main */

} /* Tyc22Xeph */
```

4. File de/mpg/mpia/xephem/Wds2Xeph.java

```
package de.mpg.mpia.xephem ;

import java.lang.Math ;
import java.io.BufferedReader ;
import java.io.FileReader ;
import java.nio.charset.Charset ;

/**
* A translator of the Washington Double Star (WDS) Catalog to the XEphem format
* @since 2018-01-10
* @author R. J. Mathar
*/
class Wds2Xeph
{
    /*********************************
    * The ASCII file of the ASCII (no-frame) version of the catalog.
    * Usually wdsweb_summ2.txt taken from http://ad.usno.navy.mil/wds/wdstext.html
    */
    String orbfile ;

    /**
```

```
* Ctor specifying the ASCII file with the binaries, one per system.
* @param catfname The name of an existing WDS ASCII file.
*/
Wds2Xeph(final String catfname)
{
    orbfile = catfname ;
} /* ctor */

/**
* Convert a HHMMSS.ss+-DDMMSS.s string to a Xephem RA or DEC designation.
* @param coord The WDS last field with the coordinates
* @param doRA If true return RA, otherwise DEC coordinate.
*/
private static String splitCoord(String coord, boolean doRA)
{
    if ( doRA)
    {
        String s = coord.substring(0,9) ;
        return s.substring(0,2) + ":" + s.substring(2,4) + ":" + s.substring(4) ;
    }
    else
    {
        String s = coord.substring(9) ;
        return s.substring(0,3) + ":" + s.substring(3,5) + ":" + s.substring(5) ;
    }
} /* splitCoord */

/**
* Convert strings of two magnitudes to the smaller magnitude (brighter star)
* @param M1 First floating point value. magnitude of primary
* @param M1 Second floating point value. magnitude of secondary
* @return A floating point value of the smaller (=brighter) magnitude
*/
private static float commonMag(String M1, String M2)
{
    float[] mags = new float[2] ;
    mags[0] = mags[1] = 9999.9F ;
    try
    {
        mags[0] = Float.valueOf(M1).floatValue() ;
    }
    catch( Exception ex)
    {
        /* here typically if there is only a dot in the string */
    }
    try
    {
        mags[1] = Float.valueOf(M2).floatValue() ;
    }
    catch( Exception ex)
    {
        /* here typically if there is only a dot in the string */
    }
    /* take the brighter component (smaller mag)
    */
    return Float.min(mags[0],mags[1]) ;
} /* commonMag */

/**
* @brief scan the input file and emit the XEphem file to stdout.
*/
void toXephem()
{
    Charset ascii = Charset.forName("US-ASCII") ;
```

```
/* deside whether to include multiple stars or just the AB component
*/
final boolean displayD = true ;
try
{
    FileReader istream = new FileReader(orbfile) ;
    BufferedReader dstream = new BufferedReader(istream) ;
    for(;;)
    {
        String lin = dstream.readLine() ;
        if ( lin == null)
            break ;

        byte[] firstLetter = lin.getBytes(ascii) ;
        if (firstLetter.length < 130)
            /* skip header lines and html lines
            */
            continue ;

        /* convert only lines starting with a RA in the range 00 - 23
        */
        if ( firstLetter[0] >= '0' && firstLetter[0] <= '2')
        {
            /* split the line into fields
            */
            String design = lin.substring(0,10).trim() ;
            String comp = lin.substring(17,22).trim() ;
            String y1 = lin.substring(23,27).trim() ;
            String y2 = lin.substring(28,32).trim() ;
            String pa1 = lin.substring(38,41).trim() ;
            String pa2 = lin.substring(42,45).trim() ;
            String sep1 = lin.substring(47,51).trim() ;
            String sep2 = lin.substring(52,57).trim() ;
            String M1 = lin.substring(58,63).trim() ;
            String M2 = lin.substring(64,69).trim() ;
            String sptype = lin.substring(70,79).trim() ;
            String pmra1 = lin.substring(80,84).trim() ;
            String pmdec1 = lin.substring(84,88).trim() ;
            String pmra2 = lin.substring(89,93).trim() ;
            String pmdec2 = lin.substring(93,97).trim() ;
            String coord = lin.substring(112,130).trim() ;

            /* flag that indicates whether that line is tranformed
            */
            boolean take = true ;
            /* skip if no coordinates tabulated
            */
            if ( coord.startsWith(".") )
                take = false;

            /* skip if these are further components of multiple systems
            */
            if ( comp.length() > 0 && ! comp.startsWith("AB") )
                take = false;

            if ( take)
            {
                System.out.print(design) ;
                if ( displayD)
                {
                    System.out.print(",f|D") ;
                    if ( sptype.length() > 0)
                        System.out.print("|"+sptype) ;
                    System.out.print("," + splitCoord(coord,true)) ;
```

```
                        if ( pmra1.length() > 0)
                            System.out.print("|" + pmra1) ;
                        System.out.print("," + splitCoord(coord,false)) ;
                        if ( pmdec1.length() > 0)
                            System.out.print("|" + pmdec1) ;
                        System.out.print("," + commonMag(M1,M2)) ;
                        System.out.print(",2000") ;
                        System.out.print(",0|"+pa2) ;
                        System.out.println() ;
                    }
                    else
                    {
                        System.out.print(",f|B") ;
                        if ( sptype.length() > 0)
                            System.out.print("|"+sptype) ;
                        System.out.print("," + splitCoord(coord,true)) ;
                        if ( pmra1.length() > 0)
                            System.out.print("|" + pmra1) ;
                        System.out.print("," + splitCoord(coord,false)) ;
                        if ( pmdec1.length() > 0)
                            System.out.print("|" + pmdec1) ;
                        System.out.print("," + M1 + "|" + M2) ;
                        System.out.print(",2000") ;
                        System.out.print(","+y1 + "|" + sep1 + "|" + pa1) ;
                        System.out.print("|"+y2 + "|" + sep2 + "|" + pa2) ;
                        System.out.println() ;
                    }
                }
            }
        }
    }
    catch (Exception ex)
    {
        System.err.println(ex) ;
        ex.printStackTrace() ;
    }
} /* toXephem */

/**
* @param argv Vector of the command line arguments
* Usage:
* javac -cp . de/mpg/mpia/xephem/*.java
* java -cp . de.mpg.mpia.xephem.Wds2Xeph wdsweb_summ2.txt > ~/.xephem/Wds.edb
*/
static public void main(String argv[])
{
    if ( argv.length == 0 )
    {
        System.err.println("Error: command line argument missing") ;
        System.err.println("Usage: java -cp . de.mpg.mpia.xephem.Wds2Xeph wdsweb_summ2.txt") ;
    }
    else
    {
        /* loop over all catalogues mentioned in the command line
        */
        for(int i=0 ; i < argv.length ; i++)
        {
            Wds2Xeph orb = new Wds2Xeph(argv[i]) ;
            orb.toXephem() ;
        }
    }

} /* main */
```

```
} /* Wds2Xeph */
```

5. File de/mpg/mpia/xephem/Ppm2Xeph.java

```
package de.mpg.mpia.xephem ;

import java.io.BufferedReader ;
import java.io.FileReader ;
import java.nio.charset.Charset ;

/**
* A translator of the positions of the PPM North or PPM South catalogue to the XEphem format
* @since 2018-02-03
* @author R. J. Mathar
*/
class Ppm2Xeph
{
    /*******************************
    * The ASCII file of the main catalog.
    * ppm1.dat of ftp://cdsarc.u-strasbg.fr/cats/I/146 or
    * ppm2.dat of ftp://cdsarc.u-strasbg.fr/cats/I/193
    */
    String catfile ;

    /**
    * Ctor specifying the ASCII file with the catalog stars, one per line.
    * @param catfname The name of an existing PPM North or PPM South
    */
    Ppm2Xeph(final String catfname)
    {
        catfile = catfname ;
    } /* ctor */

    /**
    * @brief scan the input file and emit the XEphem file to stdout.
    */
    void toXephem()
    {
        Charset ascii = Charset.forName("US-ASCII") ;
        try
        {
            FileReader istream = new FileReader(catfile) ;
            BufferedReader dstream = new BufferedReader(istream) ;
            for(;;)
            {
                String lin = dstream.readLine() ;
                if ( lin == null)
                    break ;

                /* ppm1.dat or ppm2 has 131 bytes in principle, but the later fields may be missing.
                */
                if ( lin.length() < 100)
                    break ;

                String ppmNo = lin.substring(1,7).trim() ;
                String vmag = lin.substring(19,23).trim() ;
                String sptype = lin.substring(24,26).trim() ;
                String rahms = lin.substring(27,39).trim() ;
                String sigde = lin.substring(41,42).trim() ;
                String dedms = lin.substring(42,53).trim() ;
                String pmra = lin.substring(55,62).trim() ;
                String pmde = lin.substring(63,69).trim() ;
```

```
            String sao = null ;
            try
            {
                sao = lin.substring(101,107).trim() ;
            }
            catch(Exception ex)
            {
            }

            String hd  = null;
            try
            {
                hd = lin.substring(108,114).trim() ;
            }
            catch(Exception ex)
            {
            }

            /* need to convert proper motion in RA from seconds per year to
            * mas/yr multiplied by cos(delta)
            * Factor 15 for s -> arcsec, factor 1000 for as-> mas
            */
            double pmRaMas = 15.e3*Double.parseDouble(pmra) ;
            double pmDecMas = 1.e3*Double.parseDouble(pmde) ;

            /* need cosine of declination-> skip sign of declination here */
            String deD = lin.substring(42,44).trim() ;
            String deM = lin.substring(45,47).trim() ;
            String deS = lin.substring(48,53).trim() ;
            /* declination in degrees */
            double decDeg = Double.parseDouble(deD) + Double.parseDouble(deM)/60.0
                    + Double.parseDouble(deS)/3600.0 ;
            pmRaMas *= Math.cos(Math.toRadians(decDeg)) ;

            System.out.print("PPM "+ppmNo) ;
            if (sao != null && sao.length() > 0)
                System.out.print("|SAO " + sao) ;
            if ( hd != null && hd.length() > 0)
                System.out.print("|HD " + hd) ;

            System.out.print(",f|S") ;
            if ( sptype.length() > 0 )
                System.out.print("|" + sptype) ;
            System.out.print("," + rahms.replaceAll("\\s+",":") + "|" + String.format("%.1f",pmRaMas)) ;
            System.out.print("," + sigde + dedms.replaceAll("\\s+",":") + "|" + String.format("%.1f",pmDecMas)) ;
            System.out.print("," + vmag) ;
            System.out.print(",") ;
            System.out.print(",") ;
            System.out.println() ;
        }
    }
    catch (Exception ex)
    {
        System.err.println(ex) ;
        ex.printStackTrace() ;
    }
} /* toXephem */

/**
* @param argv Vector of the command line arguments
* Usage:
* javac -cp . de/mpg/mpia/xephem/*.java
* java -cp . de.mpg.mpia.xephem.Ppm2Xeph ppm[12].dat > ~/.xephem/Ppm.edb
*/
```

```
    static public void main(String argv[])
    {
        boolean allT1 = false ;
        if ( argv.length == 0 )
        {
            System.err.println("Error: command line argument missing") ;
            System.err.println("Usage: java -cp . de.mpg.mpia.xephem.Ppm2Xeph ppm[12].dat") ;
        }
        else
        {
            for(int i=0 ; i < argv.length ; i++)
            {
                if ( argv[0].startsWith("-a") )
                    allT1 = true ;
                Ppm2Xeph pp = new Ppm2Xeph(argv[i]) ;
                pp.toXephem() ;
            }
        }

    } /* main */

} /* Ppm2Xeph */
```

6. File de/mpg/mpia/xephem/MpcObscod.java

```
package de.mpg.mpia.xephem ;

import java.io.BufferedReader ;
import java.io.FileReader ;
import java.nio.charset.Charset ;

/** Codes of observatories of the MPC (Minor Planet Center) and geocentric locations.
* The class allows to specify the location of the observatory by
* a lazy 3-letter-code if that observatory is in the list of recognized locations.
* This is a standalone variant of the conversions proposed in arXiv:1608.040340 .
* @author Richard J. Mathar
* @since 2018-01-28
*/
public class MpcObscod
{
    /** name of the file with the MPC obscodes (HTML)
    */
    String cfile ;

    /** ctor.
    * @param cfname The name with the HTML-coded Obscode.html file
    */
    public MpcObscod(String cfname)
    {
        cfile = cfname ;
    } /* ctor */

    /**
    * @brief Scan the entire input file and emit the XEphem file to stdout.
    */
    void toXephem()
    {
        /* Reference ellipsoid is the WGS84 system.
        */
        Geoid wgs84 = new Geoid() ;
        Charset ascii = Charset.forName("US-ASCII") ;
        try
```

```
    {
        FileReader istream = new FileReader(cfile) ;
        BufferedReader dstream = new BufferedReader(istream) ;
        for(;;)
        {
            String lin = dstream.readLine() ;
            if ( lin == null)
                break ;
            toXephem(lin,wgs84) ;
        }
    }
    catch (Exception ex)
    {
        System.err.println(ex) ;
        ex.printStackTrace() ;
    }
}

/***********************
* @brief Convert longitude or latitude angle in radians to blank separated D:M:S format
* @param radians the longitude or latitude in radians
* @param isLong If true the argument is a longitude, else a latitude
* @return A string of the form dd mm ss.ss {E|W|N|S} of the value in degrees
*/
String lTude(double radians, boolean isLong)
{
    /* absolute value in degrees */
    double degabs = Math.abs(Math.toDegrees(radians)) ;

    /* We aim convert this to D:M:S.ss with two digits, rounded.
    * To accomplish rounding and to avoid outputs like 60.00 for the seconds,
    * rewrite this as an integer 360000*(D+M/60+S/3600.0).
    */
    int degabsI = (int) (360000*degabs+0.5) ;
    int d = degabsI/360000 ;
    /* residual becomes 6000*M+Sss */
    degabsI -= d*360000 ;
    int m = degabsI/6000 ;
    /* residual becomes 100*S.ss */
    degabsI -= m*6000 ;
    String out = new String() ;
    out += d  + " " + m + " " + String.format("%.2f",degabsI/100.0) ;
    if ( isLong)
        out += (radians >=0.) ? " E" : " W";
    else
        out += (radians >=0.) ? " N" : " S";
    return out ;
}

/** Decode a line in the MPC format (code and 3 floating point parameters and description).
* @param ascline A line of the ObsCodes.html file of the MPC.
*  If this line starts with the opening less-sign characteristic
*  for HTML lines, the initialization remains incomplete.
*/
protected void toXephem(String ascline, Geoid wgs84)
{
    if ( !ascline.startsWith("<") )
    {
        /* read the 4 initial pieces of the line. 3-byte code.
        * geographic longitude [deg]. Cosine of geocentric latitude.
        * Sine of geocentric latitude.
        */
        String code = ascline.substring(0,3) ;
        String longS = ascline.substring(4,13) ;
```

```
        String coslatS = ascline.substring(13,21) ;
        String sinlatS = ascline.substring(21,30) ;
        String descr = ascline.substring(30) ;

        try
        {
            /* geographic longitude, degrees
            */
            double longi = Double.parseDouble(longS) ;
            if ( longi > 180.0)
                longi -= 360.0 ;
            /* geographic latitude, radians
            */
            longi = Math.toRadians(longi) ;

            /* cosine of geocentric latitude */
            double cosphi = Double.parseDouble(coslatS) ;
            /* sine of geocentric latitude */
            double sinphi = Double.parseDouble(sinlatS) ;

            /* geocentric latitude [rad] */
            double phigc = Math.atan2(sinphi,cosphi) ;

            /* the cartesian components of the geocentric position.
            */
            double[] xyzg =new double[3] ;
            xyzg[0] = cosphi*Math.cos(longi) ;
            xyzg[1] = cosphi*Math.sin(longi) ;
            xyzg[2] = sinphi ;
            for(int i=0 ; i < xyzg.length; i++)
                xyzg[i] *= wgs84.a ;

            /* convert geocentric to geodetic
            * [0] = long, [1]=lat, [2] = altitude
            */
            double[] geodetic = wgs84.toSpherical(xyzg) ;

            String cmt = "# geo:"
                + String.format("%.5f",Math.toDegrees(geodetic[1])) + ","
                + String.format("%.5f",Math.toDegrees(geodetic[0])) ;

            String latDeg = lTude(geodetic[1],false) ;
            String lonDeg = lTude(geodetic[0],true) ;
            String wiki = "{{Coord" ;
            wiki += "|" + latDeg.replaceAll("\\s+","|")
                + "|" + lonDeg.replaceAll("\\s+","|")
                + "}}" ;
            System.out.println(cmt+ " " + wiki) ;

            String out = code + " " + descr + " ;" ;
            out += " " + latDeg + " ;" ;
            out += " " + lonDeg + " ;" ;
            out += " " + String.format("%.0f",geodetic[2]) + " ;" ;
            System.out.println(out) ;
        }
        catch (Exception ex)
        {
            /* there are some empty fields for space stations which we ignore */
        }
    }
} /* ctor */

/** Show the locations in geographic units.
* <p>
```

```
    * The main program takes the HTML version of the list as the input data base.
    * This should be a copy of <a href="http://www.minorplanetcenter.net/iau/lists/ObsCodes.html">ObsCodes.html</a> .
    * </p>
    *
    * <p>
    * The program prints the observatories on a line-by-line basis after
    * converting the geocentric coordinates to WGS84 (geodetic) coordinates.
    * </p>
    *
    * <p>
    * Note that the accuracy of the sines and cosines of the geographic latitude are
    * often too small to derive a meaningful altitude above the ellipsoid (because
    * the format scales them all with the Earth radius of roughly 6300 kilometers).
    * </p>
    *
    * Usage:
    * <pre>
    *   java -cp . de.mpg.mpia.xephem.MpcObscod directory/Obsc*.html
    * </pre>
    *
    * @see <a href="http://dc.zah.uni-heidelberg.de/obscode/q/query/form">ZAH query</a>
    *   which does the same job .
    * @param args The command line argument must be the ASCII file of
    *   the MPC codes to be converted.
    */
    static public void main(String[] argv)
    {
        if ( argv.length == 0 )
        {
            System.err.println("Error: command line argument missing") ;
            System.err.println("Usage: java -cp . de.mpg.mpia.xephem.MpcObscod ObsCodes.html") ;
        }
        else
        {
            String cfname = argv[argv.length-1] ;

            MpcObscod cod = new MpcObscod(cfname) ;
            cod.toXephem() ;
        }

    } /* main */

} /* MpcObscod */
```

7. File de/mpg/mpia/xephem/Geoid.java

```
package de.mpg.mpia.xephem ;

import java.lang.Math ;

/** Definition of an oblate reference ellipsoid.
* The class aids to convert a point given in Cartesian coordinates to geodetic longitude and latitude.
* @author Richard J. Mathar
* @since 2018-01-28
*/
public class Geoid
{
    /** Semi-major axis in meters
    */
    double a ;

    /** Inverse flattening
```

```
*/
double finv ;

/** square of first eccentricity. Roughly 0.006694 for the WGS84.
*/
double esquare ;

/** Ctor.
* @param semiMaj Length of the semimajo raxis in meters.
* @param invFlat Inverse flattening parameter
*/
public Geoid(final double semiMaj, final double invFlat)
{
    a = semiMaj ;
    finv = invFlat ;
    /* e^2 = f*(2-f) where f is the flattening
    */
    esquare = (2.0-1/finv)/finv ;
} /* ctor */

/** Ctor with WGS84 values.
* Assumes major axis 6378137.0 and inverse flattening of 298.257223563
*/
public Geoid()
{
    this(6378137.0,298.257223563) ;
} /* ctor */

/** Convert a vector xyz coordinates to geodetic longitude, latitude and altitude.
* See Vermeille J. Geod 76 (8) (2002) 451 for the algebra that is implemented.
*
* @param xyz The x, y and z coordinates of the point relative to the geocenter in meters.
* @return A vector of 3 values.
*   The [0] component is the geodetic/geocentric longitude in radians.
*   The [1] component is the geodetic latitude in radians.
*   The [2] component is the altitude above the allipsoid in meters.
*/
public double[] toSpherical(double[] xyz)
{
    /* defining equations
    * x = (N+h)*cos(phi)*cos(lambda)
    * y = (N+h)*cos(phi)*sin(lambda)
    * z = ((b^2/a^2)*N+h)*sin(phi) where semi-minor b=a*(1-f), so (b^2/a^2) = 1-e^2 = (1-f)^2 = ((finv-1)/f)^2
    * N=a/sqrt(1-e^2*sin^2 phi)
    */
    double[] coord = new double[3] ;

    /* compute lambda from tan(lambda) = y/x
    */
    coord[0] = Math.atan2(xyz[1],xyz[0]) ;

    /* sqrt (x^2+y^2) = (N+h)*cos(phi)
    */
    double rproj = Math.hypot(xyz[0],xyz[1]) ;
    final double e4 = esquare*esquare ;
    final double p = Math.pow(rproj/a,2.0) ;   // (x^2+y^2)/a^2
    final double q = (1.-esquare)*Math.pow(xyz[2]/a,2.0) ; // (1-e^2)*z^2/a^2
    final double r = (p+q-e4)/6.0 ;
    final double s = e4*p*q/4.0/Math.pow(r,3.0) ;
    final double t = Math.cbrt(1.0+s+Math.sqrt(s*(2.0+s))) ;
    final double u = r*(1.0+t+1.0/t) ;
    final double v = Math.sqrt(u*u+e4*q) ;
    final double w = esquare*(u+v-q)/2.0/v ;
    final double k = Math.sqrt(u+v+w*w)-w ;
```

```
        final double D = k*rproj/(k+esquare) ;
        final double dzsqrt = Math.hypot(D,xyz[2]) ;

        coord[2] = (k+esquare-1.0)*dzsqrt/k ;
        coord[1] = 2.*Math.atan( xyz[2]/(D+dzsqrt)) ;

        return coord ;
    } /* toSpherical */


} /* Geoid */
```

---

[1] H. Vermeille, J. Geod. **76**, 451 (2002).
[2] R. J. Mathar, arXiv:1608.04340 [astro-ph.IM] (2016).