

# Translation of Some Star Catalogs to the XEphem Format

Richard J. Mathar\*

*Max-Planck Institute of Astronomy, Königstuhl 17, 69117 Heidelberg, Germany*

(Dated: October 7, 2021)

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, (v) the SKY2000 catalog, (vi) the [2MASS](#) catalog, (vii) the Third Reference Catalog of Bright Galaxies (RC3), (viii) the General Catalogue of Variable Stars (GCVS v. 5). (ix) the Naval Observatory CCD Astrograph Catalog (UCAC4). (x) the USNO catalog digitized by the PMM (USNO 2A).  
[viXra:1802.0035]

PACS numbers: 95.10.Km, 95.80.+p

## I. AIM AND SCOPE

The Java library provided 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 4.0.2`.

The envisioned use is that the translated star catalogs can be included in the display by using the `Data→Files` menu of the `xephem` graphical user interface (GUI), and by clicking on the `Files` menu 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/cds2xephem/*.java
```

Optionally the output can be packed into a single `jar` file with

```
jar cvf Cds2XEphem.jar  
de/mpg/mpia/cds2xephem/*.java  
de/mpg/mpia/cds2xephem/*.class
```

which produces `Cds2XEphem.jar`, and then replacing the dot in the calls `java -cp . args` mentioned further down by `java -cp Cds2XEphem.jar args`.

Astronomers who do not have a Java development kit may get this file from <https://www.mpia.de/~mathar/progs/Cds2XEphem.jar> or <https://build.opensuse.org/package/show/home:rjmathar/cds2xephem>. This is also the simplest way to obtain the source code.

Which program of the library 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.cds2xephem.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. If this for an openSUSE installation of XEphem via <https://software.opensuse.org/download.html?project=Education&package=xephem>, the output directory could be `/usr/lib/xephem/catalogs` instead, to share catalogs for all users.

The commets data base can be downloaded in the XEphem format from <https://www.minorplanetcenter.net/iau/Ephemerides/Comets/SoftwareComets.html>; 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

---

\* <http://www.mpia-hd.mpg.de/~mathar>

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 program `Hip2Xeph` accepts 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.cds2xephem.Hip2Xeph
hip_main.dat > ~/.xephem/hipparcos.edb
```

```
java -cp . de.mpg.mpia.cds2xephem.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.

### 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.cds2xephem.Tyc22Xeph
[-a] suppl_1.dat > ~/.xephem/tycho2.edb
```

```
java -cp . de.mpg.mpia.cds2xephem.Tyc22Xeph
[-a] suppl_2.dat >> ~/.xephem/tycho2.edb
```

```
java -cp . de.mpg.mpia.cds2xephem.Tyc22Xeph
[-a] tyc2.dat.00 >> ~/.xephem/tycho2.edb
```

```
java -cp . de.mpg.mpia.cds2xephem.Tyc22Xeph
[-a] tyc2.dat.01 >> ~/.xephem/tycho2.edb
```

...

```
java -cp . de.mpg.mpia.cds2xephem.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.cds2xephem.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> or <http://www.astro.gsu.edu/wds/wdstext.html>; our software supports only the non-frame versions. The command argument should be names of one or more uncompressed data files.

```
java -cp . de.mpg.mpia.cds2xephem.Wds2Xeph
wdsweb_summ2.txt > ~/.xephem/WDS.edb
```

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

```
java -cp . de.mpg.mpia.cds2xephem.Wds2Xeph
wdsnewerweb1.txt > ~/.xephem/WDS.edb
```

```
java -cp . de.mpg.mpia.cds2xephem.Wds2Xeph
wdsnewerweb2.txt >> ~/.xephem/WDS.edb
```

```
java -cp . de.mpg.mpia.cds2xephem.Wds2Xeph
wdsnewerweb3.txt >> ~/.xephem/WDS.edb
```

```
java -cp . de.mpg.mpia.cds2xephem.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.cds2xephem.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 requires that at least one of the decompressed catalogs `ppm1.dat` and `ppm2.dat` is in the local file system, and that one or both of them are the command line arguments:

```
java -cp . de.mpg.mpia.cds2xephem.Ppm2Xeph
ppm[12].dat > ~/.xephem/ppm.edb
```

### F. SKY2000 Catalog

The SKY2000 catalog can be downloaded from <http://cdsarc.u-strasbg.fr/vizier/ftp/cats/V/145/sky2kv5.dat.gz>. The program requires that the decompressed catalog `sky2kv5.dat` is in the local file system and becomes the command line argument:

```
java -cp . de.mpg.mpia.cds2xephem.Sky2k2Xeph
sky2kv5*.dat > ~/.xephem/SKY2000.edb
```

## G. 2MASS Catalog

The 2MASS Catalogue of infrared objects can be downloaded from <https://irsa.ipac.caltech.edu/2MASS/download/allsky/>. Only the contents of the Point Source Catalog (PSC) is rendered by our program. The program requires that the decompressed catalog—at least the area of the sky of interest—is in the local file system.

A command line option of either `-J` or `-H` or `-K` indicates which magnitude should be transported to the XEphem file. If the option is absent, H-magnitudes are used.

A command line option of `-m` followed by a floating point number indicates a limiting magnitude which applies a filter on the catalog; stars weaker than that magnitude are not converted to the XEphem format. If the option is absent, no such magnitude filter is applied.

The first command line argument is the name of the *directory* which contains files named `psc_a??` and `psc_b??`. Each of these files is approximately 1.7 GB of size. If a file is missing in this directory that is needed according to the remaining command line arguments, an error message says which of these required files is not readable.

The second command line argument is the pointing center's right ascension, either expressed as a floating point number in the range 0 to 24.0, or as a string in the `HH:MM:SS.sss` format.

The third command line argument is the pointing center's declination, either expressed as a floating point number in the range  $-90$  to  $+90$  or as a string in the `+DD:MM:SS.sss` format.

The last command line argument is a positive cone search radius in units of arcseconds. Only objects not further than this distance away from the pointing center are converted.

```
java -cp . de.mpg.mpia.cds2xephem.Tmass2Xeph
[-J|-H|-K] [-m mag] dirname ra dec coneas
> ~/.xephem/2mass.edb
```

So the difference to the other scanners is that only a portion of the catalog is rendered by one call.

## H. RC3 Catalog

The Third Reference Catalog of Bright Galaxies (RC3) can be downloaded from <http://cdsarc.u-strasbg.fr/vizier/ftp/cats/VII/155/rc3.gz>. The program requires that the decompressed catalog `rc3` is in the local file system and is the command line argument:

```
java -cp . de.mpg.mpia.cds2xephem.Rc32Xeph
rc3 > ~/.xephem/rc3.edb
```

## I. GCVS Catalog

The General Catalogue of Variable Stars can be downloaded from <http://www.sai.msu.su/gcvs/gcvs/>

[gcvs5/htm/](http://www.sai.msu.su/gcvs/gcvs/). The program requires that the decompressed catalog `gcvs5.txt` is in the local file system and is the command line argument:

```
java -cp . de.mpg.mpia.cds2xephem.Gcvs2Xeph
gcvs5.txt > ~/.xephem/GCVS.edb
```

The maximum magnitude of the GCVS is submitted to the XEphem file. If no such magnitude is in the GCVS, no magnitude is submitted to the XEphem file, with the result that the object is implicitly considered to have a magnitude of 0.

## J. UCAC4 Catalog

The Astrograph Catalogue of can be downloaded in pieces as binary files from <ftp://cdsarc.u-strasbg.fr/ftp/cats/I/322A/UCAC4/u4b/>. The program requires that at least one of these `ziii` files is in the local file system and is the command line argument:

```
java -cp . de.mpg.mpia.cds2xephem.Ucac42Xeph
[-m mag] ziii [zjjj] ... > ~/.xephem/ucac4.edb
```

If more than one of these zone files is in the arguments, the program runs through them in sequence to produce the standard output. For a given region of interest at declination  $\delta$ , the index *iii* of the zone file is  $1 + \lfloor (\delta + 90^\circ) / 0.2^\circ \rfloor$ .

A command line option of `-m` followed by a floating point number indicates a limiting magnitude which applies a filter on the catalog; stars weaker than that magnitude are not converted to the XEphem format. If the option is absent, no such magnitude filter is applied.

## K. USNO Catalog

The USNO Catalogue (version A2) of can be downloaded as binary files from <https://cdsarc.u-strasbg.fr/ftp/I/252/>, but that compressed binary format is not supported here.

The variant which contains roughly 10 percent of these stars in another binary format, usually called SA2, can be downloaded from <http://catalogs.astro.uni-altai.ru/usnosa/> or <https://ftp.imcce.fr/pub/catalogs/USNO-SA2/>.

The program requires that at least one of these `zoneiii.cat` files is in the local file system and is the command line argument:

```
java -cp . de.mpg.mpia.cds2xephem.UsnoA22Xeph
[-m mag] [-a] [-B] [-R] zoneiii.cat
[zonejjj.cat ...] > ~/.xephem/usno.edb
```

If more than one of these zone files is in the arguments, the program runs through them in sequence to produce the standard output.

A command line option of `-m` followed by a floating point number indicates a limiting magnitude which applies a filter on the catalog; stars weaker than that magnitude are not converted to the XEphem format. If the option is absent, no such magnitude filter is applied. As

the binary files represent each object by 12 bytes, the expanded output needs approximately 3 times as much disk space as the combined input files if no such magnitude limit is applied.

The command line option `-a` indicates that all files (after applying the magnitude cut) are converted; if the option is absent only stars with the cleared `Q` flag in the catalogue are rendered.

The options `-B` or `-R` indicate that the red or the blue plates represent the magnitude of the object. If both or none of these options is used, the mean of the two magnitudes is used for comparison with the magnitude cut and used in the output.

## L. 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.cds2xephem.MpcObscode [-v] 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.

The output order of the observatories is the same order as in the ObsCodes.html source file.

The option `-v` may be used to generate additional lines in the format of a wikipedia page listing the Observatory Codes.

Another use is a merge-sort of files which already have the XEphem format:

```
java -cp . de.mpg.mpia.cds2xephem.MpcObscode -s [-u] file1sites [file2sites ...]
```

The option `-s` means that the one or more files with observatory site information is/are sorted (with higher priority by geographic latitude, with lower priority by geographic longitude). The optional `-u` lets the program remove duplicates, which means, sites where longitude and latitude are the same are printed only once, by first occurrence in the source files.

## Appendix A: Source Code

### 1. File de/mpg/mpia/cds2xephem/Mpc2Xeph.java

```
package de.mpg.mpia.cds2xephem ;

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/cds2xephem/*.java
 * java -cp . de.mpg.mpia.cds2xephem.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 Cds2XEphem.jar de.mpg.mpia.cds2xephem.Mpc2Xeph [-a] MPCORB.DAT") ;
    }
    else
    {
        if ( argv[0].startsWith("-a") )
            withProvi = true ;
        String catfname = argv[argv.length-1] ;

        Mpc2Xeph orb = new Mpc2Xeph(catfname) ;
        System.out.println("# generated from "+catfname + " with " + Mpc2Xeph.class.getName()
            + " " + java.time.Clock.systemUTC().instant() ) ;
        System.out.println("# [viXra:1802.0035] Richard J. Mathar") ;
        orb.toXephem(withProvi) ;
    }
}

} /* main */
} /* Mpc2Xeph */

```

## 2. File de/mpg/mpia/cds2xephem/Hip2Xeph.java

```

package de.mpg.mpia.cds2xephem ;

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)
    {
        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/cds2xephem/*.java
 * java -cp . de.mpg.mpia.cds2xephem.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 Cds2XEphem.jar de.mpg.mpia.cds2xephem.Hip2Xeph hip_main.dat") ;
    }
    else
    {
        if ( argv[0].startsWith("-a") )
            allT1 = true ;
        String catfname = argv[argv.length-1] ;

        Hip2Xeph hip = new Hip2Xeph(catfname) ;
        System.out.println("# generated from "+catfname + " with " + Hip2Xeph.class.getName()
            + " " + java.time.Clock.systemUTC().instant() ) ;
        System.out.println("# [viXra:1802.0035] Richard J. Mathar") ;

        hip.toXephem(allT1) ;
    }

} /* main */
} /* Hip2Xeph */

```

### 3. File de/mpg/mpia/cds2xephem/Tyc22Xeph.java

```

package de.mpg.mpia.cds2xephem ;

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.
*/
static String hexRep(double deg)
{
    double degabs = Math.abs(deg) ;
    /* We'll write 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)
{
    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/cds2xephem/*.java
 * java -cp . de.mpg.mpia.cds2xephem.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 Cds2XEphem.jar de.mpg.mpia.cds2xephem.Tyc22Xeph [-a] suppl_[12].dat tyc") ;
    }
    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]) ;
            System.out.println("# generated from "+argv[i] + " with " + Tyc22Xeph.class.getName()
                + " " + java.time.Clock.systemUTC().instant() ) ;
            System.out.println("# [viXra:1802.0035] Richard J. Mathar") ;

            tyc.toXephem(allT1) ;
        }
    }
} /* main */

} /* Tyc22Xeph */

```

#### 4. File de/mpg/mpia/cds2xephem/Wds2Xeph.java

```

package de.mpg.mpia.cds2xephem ;

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 M2 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") ;
    /* decide 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/cds2xephem/*.java
 * java -cp . de.mpg.mpia.cds2xephem.Wds2XEphem 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 Cds2XEphem.jar de.mpg.mpia.cds2xephem.Wds2XEphem wdsweb_summ2.txt") ;
    }
    else
    {
        /* loop over all catalogues mentioned in the command line
        */
        for(int i=0 ; i < argv.length ; i++)
        {
            Wds2XEphem orb = new Wds2XEphem(argv[i]) ;
            System.out.println("# generated from "+argv[i] + " with " + Wds2XEphem.class.getName()
                + " " + java.time.Clock.systemUTC().instant() ) ;
            System.out.println("# [viXra:1802.0035] Richard J. Mathar") ;

            orb.toXephem() ;
        }
    }
} /* main */

} /* Wds2XEphem */

```

## 5. File de/mpg/mpia/cds2xephem/Ppm2XEphem.java

```

package de.mpg.mpia.cds2xephem ;

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()
    {
        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/cds2xephem/*.java
 * java -cp . de.mpg.mpia.cds2xephem.Ppm2Xeph ppm[12].dat > ~/.xephem/Ppm.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 Cds2Xeph.jar de.mpg.mpia.cds2xephem.Ppm2Xeph ppm[12].dat") ;
    }
    else
    {
        for(int i=0 ; i < argv.length ; i++)
        {
            Ppm2Xeph pp = new Ppm2Xeph(argv[i]) ;
            System.out.println("# generated from "+argv[i] + " with " + Ppm2Xeph.class.getName()
                               + " " + java.time.Clock.systemUTC().instant() ) ;
            System.out.println("# [viXra:1802.0035] Richard J. Mathar") ;
        }
    }
}

```

```

        pp.toXephem() ;
    }
}

} /* main */

} /* Ppm2Xeph */

```

## 6. File de/mpg/mpia/cds2xephem/Sky2k2Xeph.java

```

package de.mpg.mpia.cds2xephem ;

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

/**
 * A translator of the positions of the SKY2000 star catalogue to the XEphem format
 * @since 2018-09-07
 * @author R. J. Mathar
 */
class Sky2k2Xeph
{
    /**
     * *****
     * The ASCII file of the main catalog.
     * sky2kv4n.dat or sky2kv4ra.dat taken from
     * http://tdc-www.harvard.edu/catalogs/sky2k.html.
     */
    String catfile ;

    /**
     * Ctor specifying the ASCII file with the catalog stars, one per line.
     * @param catfname The name of an existing SKY2000 star catalog file.
     */
    Sky2k2Xeph(final String catfname)
    {
        catfile = catfname ;
    } /* ctor */

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

                /* Each line is supposed to have 520 bytes, the last being f_mag3. But most of them
                 * seem to contain only 390 bytes (??), some even 304 bytes,
                 * so http://tdc-www.harvard.edu/catalogs/sky2kv4.readme seems to be outdated or missing
                 * some major information.
                 */
                if ( lin.length() < 243)
                {
                    continue ;
                }
            }
        }
    }
}

```

```

}

String idNo = lin.substring(27,35).trim() ;

String hd = null;
try
{
    hd = lin.substring(35,41).trim() ;
}
catch(Exception ex)
{
}

String sao = null ;
try
{
    sao = lin.substring(43,49).trim() ;
}
catch(Exception ex)
{
}

String raH = lin.substring(118,120).trim() ;
String raM = lin.substring(120,122).trim() ;
String raS = lin.substring(122,129).trim() ;
String sigde = lin.substring(129,130).trim() ;
String deD = lin.substring(130,132).trim() ;
String deM = lin.substring(132,134).trim() ;
String deS = lin.substring(134,140).trim() ;
String pmra = lin.substring(149,157).trim() ;
String pmde = lin.substring(157,165).trim() ;

String vmag = lin.substring(232,238).trim() ;
if ( vmag.length() == 0)
    /* if observed visual magnitute is absent, use derived magnitude */
    vmag = lin.substring(238,243).trim() ;

String sptype = new String() ;
if ( lin.length() >= 339)
    sptype = lin.substring(336,339).trim() ;

System.out.print("S2K "+idNo) ;
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(", "
    + raH + ":" + raM + ":" + raS) ;

if ( pmra.length() > 0)
{
    /* 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) ;

    /* need cosine of declination-> skip sign of declination here */
    /* 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("|" + String.format("%.1f",pmRaMas)) ;
    }

    System.out.print(", " + sigde + deD+ ":" + deM + ":" + deS) ;

    if ( pmde.length() > 0)
    {
        /* Factor 1000 for as-> mas
        */
        double pmDecMas = 1.e3*Double.parseDouble(pmde) ;

        System.out.print("|" + 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/cds2xephem/*.java
 * jar cvf Cds2XEphem.jar de/mpg/mpia/cds2xephem*.java de/mpg/mpia/cds2xephem*.class
 * java -cp . de.mpg.mpia.cds2xephem.Sky2k2Xeph sky2kv*.dat > ~/.xephem/SKY2000.edb
 * java -cp Cds2XEphem.jar de.mpg.mpia.cds2xephem.Sky2k2Xeph sky2kv*.dat > ~/.xephem/SKY2000.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 Cds2XEphem.jar de.mpg.mpia.cds2xephem.Sky2k2Xeph sky2kv4*.dat") ;
    }
    else
    {
        for(int i=0 ; i < argv.length ; i++)
        {
            System.out.println("# "+argv[i]) ;
            Sky2k2Xeph pp = new Sky2k2Xeph(argv[i]) ;
            System.out.println("# generated from "+argv[i] + " with " + Sky2k2Xeph.class.getName()
                +" " + java.time.Clock.systemUTC().instant() ) ;
            System.out.println("# [viXra:1802.0035] Richard J. Mathar") ;

            pp.toXephem() ;
        }
    }
}

} /* main */

} /* Sky2k2Xeph */

```

## 7. File de/mpg/mpia/cds2xephem/Rc32Xeph.java

```

package de.mpg.mpia.cds2xephem ;

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

/**
 * A translator of the positions of the RC3 galaxy catalogue to the XEphem format
 * @since 2018-09-07
 * @author R. J. Mathar
 */
class Rc32Xeph
{
    /**
     * The ASCII file of the main catalog taken from
     * http://cdsarc.u-strasbg.fr/viz-bin/Cat?cat=VII%2F155&target=http&
     */
    String catfile ;

    /**
     * Ctor specifying the ASCII file with the catalog stars, one per line.
     * @param catfname The name of an existing RC3 catalog file.
     */
    Rc32Xeph(final String catfname)
    {
        catfile = catfname ;
    } /* ctor */

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

                /* Each line is supposed to have 363 bytes, the last being V3K
                */
                if ( lin.length() < 363)
                {
                    continue ;
                }

                String raH = lin.substring(0,2).trim() ;
                String raM = lin.substring(2,4).trim() ;
                String raS = lin.substring(4,8).trim() ;
                String sigde = lin.substring(9,10).trim() ;
                String deD = lin.substring(10,12).trim() ;
                String deM = lin.substring(12,14).trim() ;
                String deS = lin.substring(14,16).trim() ;
                String nam = lin.substring(62,74).trim() ;
                String pgc = lin.substring(105,116).trim() ;
                String typ = lin.substring(117,124).trim() ;
                String logd25 = lin.substring(151,155).trim() ;
            }
        }
    }
}

```

```

String logr25 = lin.substring(161,165).trim() ;
String pa = lin.substring(185,188).trim() ;
String bt = lin.substring(189,194).trim() ;
if ( bt.length() == 0 )
    bt = lin.substring(200,205).trim() ;

System.out.print(pgc) ;
if ( nam.length() > 0 )
    System.out.print("|" +nam) ;

System.out.print(",f") ;
/* render the spiral calaxies as G and all others as H
*/
if ( typ.length() < 2 )
    System.out.print("|H") ;
else if ( typ.substring(1,2).compareTo("S") == 0 )
    System.out.print("|G") ;
else
    System.out.print("|H") ;

System.out.print(", " + raH + ":" + raM + ":" + raS) ;

System.out.print(", " + sigde + deD+ ":" + deM + ":" + deS) ;

/* field 5, use the B -magnitude here... */
System.out.print(", " + bt) ;

/* field 6, default 2000 */
System.out.print(",") ;

/* field 7, factor 0.1 because logd25 is with 0.1 arcmin, and factor 60 for arcsec */
double d25 = 0.0 ;
double r25 = 0.0 ;
try {
    d25 = 6.0*Math.pow(10.,Double.parseDouble(logd25)) ;
    r25 = d25/ Math.pow(10.,Double.parseDouble(logr25)) ;
}
catch(Exception ex)
{
    /* in case one of these fields was empty */
    d25 = 0.0 ;
    r25 = 0.0 ;
}

System.out.print(", " + String.format("%.2f",d25) + "|" + String.format("%.2f",r25)) ;
if ( pa.length() == 0 )
    System.out.print("|" + 0.) ;
else
    System.out.print("|" + pa) ;

System.out.println() ;
}
}
catch (Exception ex)
{
}
}
}
} /* toXephem */

/**
 * @param argv Vector of the command line arguments
 * Usage:
 * javac -cp . de/mpg/mpia/cds2xephem/*.java

```

```

* jar cvf Cds2XEphem.jar de/mpg/mpia/cds2xephem*.java de/mpg/mpia/cds2xephem*.class
* java -cp . de.mpg.mpia.cds2xephem.Rc32Xeph rc3 > ~/.xephem/rc3.edb
* java -cp Cds2XEphem.jar de.mpg.mpia.cds2xephem.Rc32Xeph rc3 > ~/.xephem/rc3.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 Cds2XEphem.jar de.mpg.mpia.cds2xephem.Rc32Xeph rc3") ;
    }
    else
    {
        for(int i=0 ; i < argv.length ; i++)
        {
            System.out.println("# "+argv[i]) ;
            Rc32Xeph pp = new Rc32Xeph(argv[i]) ;
            System.out.println("# generated from "+argv[i] + " with " + Rc32Xeph.class.getName()
                +" " + java.time.Clock.systemUTC().instant() ) ;
            System.out.println("# [viXra:1802.0035] Richard J. Mathar") ;

            pp.toXephem() ;
        }
    }

} /* main */
} /* Rc32Xeph */

```

## 8. File de/mpg/mpia/cds2xephem/Gcvs2Xeph.java

```

package de.mpg.mpia.cds2xephem ;

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

/**
 * A translator of the positions of the General catalogue of Variable Stars to the XEphem format
 * @since 2018-09-10
 * @author R. J. Mathar
 */
class Gcvs2Xeph
{
    /**
     * The ASCII file of the GCVS v5.1 catalog taken from
     * http://www.sai.msu.su/gcvs/gcvs/gcvs5/htm/
     */
    String catfile ;

    /**
     * Ctor specifying the ASCII file with the catalog stars, one per line.
     * @param catfname The name of an existing GCVS catalog file.
     */
    Gcvs2Xeph(final String catfname)
    {
        catfile = catfname ;
    } /* ctor */

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

```



```

void toXephem()
{
    try
    {
        FileReader istream = new FileReader(catfile) ;
        BufferedReader dstream = new BufferedReader(istream) ;
        int linno = 1 ;
        for(;;linno++)
        {
            String lin = dstream.readLine() ;
            if ( lin == null)
                break ;

            /* Each line is supposed to have 235 bytes, the last being V3K
            * Skip the first 2 header lines.
            */
            if ( lin.length() < 235 || linno < 3)
            {
                continue ;
            }

            String nam = lin.substring(8,18).trim() ;
            String raH = lin.substring(20,22).trim() ;
            String raM = lin.substring(22,24).trim() ;
            String raS = lin.substring(24,29).trim() ;
            String sigde = lin.substring(30,31).trim() ;
            String deD = lin.substring(31,33).trim() ;
            String deM = lin.substring(33,35).trim() ;
            String deS = lin.substring(35,39).trim() ;

            String mag = lin.substring(53,59).trim() ;
            /*
            * Just copying over the entire specification is not possible because some
            * of them contains commas, which will severely confuse the xephem parser.
            */
            String spec = lin.substring(137,139).trim() ;
            String pmra = lin.substring(179,185).trim() ;
            String pmdec = lin.substring(186,192).trim() ;

            /* skip fields with unknown coordinates */
            if ( raH.length() < 2)
                continue ;

            System.out.print(nam) ;

            System.out.print(",f|V") ;
            if ( spec.length() > 0)
                System.out.print("|" + spec) ;

            System.out.print(", " + raH + ":" + raM + ":" + raS) ;
            if ( pmra.length() > 0)
            {
                /* factor 100 for conversion as/yr to mas/yr
                * Does the catalog define this with or without the cos(delta) ???
                * We assume this is without the factor in the GCVS so we need to multiply here.
                */
                double mas = 1000.0*Double.parseDouble(pmra) ;

                /* need cosine of declination-> skip sign of declination here */
                /* declination in degrees */
                double decDeg = Double.parseDouble(deD) + Double.parseDouble(deM)/60.0
                    + Double.parseDouble(deS)/3600.0 ;
                mas *= Math.cos(Math.toRadians(decDeg)) ;
            }
        }
    }
}

```

```

        System.out.print("|" + String.format("%.2f",mas)) ;
    }

    System.out.print(", " + sigde + deD+ ":" + deM + ":" + deS) ;
    if ( pmdec.length() > 0)
    {
        /* factor 100 for conversion as/yr to mas/yr
        */
        double mas = 1000.0*Double.parseDouble(pmdec) ;
        System.out.print("|" + String.format("%.2f",mas)) ;
    }

    /* field 5, use the maximum magnitude here... */
    System.out.print(", " + mag) ;

    /* field 6, default 2000 */
    System.out.print(",") ;

    System.out.println() ;
}
}
catch (Exception ex)
{
    System.err.println(ex) ;
    ex.printStackTrace() ;
}
} /* toXephem */

/**
 * @param argv Vector of the command line arguments
 * Compilation:
 * javac -cp . de/mpg/mpia/cds2xephem/*.java
 * jar cvf Cds2XEphem.jar de/mpg/mpia/cds2xephem*.java de/mpg/mpia/cds2xephem*.class
 * Usage:
 * java -cp . de.mpg.mpia.cds2xephem.Gcvs2Xeph gcvs5.txt > ~/.xephem/GCVS.edb
 * java -cp Cds2XEphem.jar de.mpg.mpia.cds2xephem.Gcvs2Xeph gcvs5.txt > ~/.xephem/GCVS.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.cds2xephem.Gcvs2Xeph gcvs5.txt") ;
    }
    else
    {
        for(int i=0 ; i < argv.length ; i++)
        {
            Gcvs2Xeph pp = new Gcvs2Xeph(argv[i]) ;
            System.out.println("# generated from "+argv[i] + " with " + Gcvs2Xeph.class.getName()
                + " " + java.time.Clock.systemUTC().instant() ) ;
            System.out.println("# [viXra:1802.0035] Richard J. Mathar") ;
            pp.toXephem() ;
        }
    }
}

} /* main */

} /* Gcvs2Xeph */

```

## 9. File de/mpg/mpia/cds2xephem/Tmass2Xeph.java

```

package de.mpg.mpia.cds2xephem ;

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

/**
 * A translator of the positions of the 2MASS point source catalog (PSC) to the XEphem format
 * @since 2019-03-29
 * @author R. J. Mathar
 */
class Tmass2Xeph
{
    /**
     * @brief The directory which contains the ps[ab]_??? catalog files.
     * This specifies the directory, and which of the files are accessed depends
     * on the pointing center and the field cone radius.
     */
    File catdir ;

    /**
     * @brief RA of pointing center in radians.
     */
    double ra ;

    /**
     * @brief DEC of pointing center in radians
     */
    double dec ;

    /**
     * @brief cone search radius in radians
     */
    double cone ;

    /**
     * @brief Infrared band enumerated as 0=J, 1=H, 2=K.
     */
    int band ;

    /**
     * @brief limiting magnitude in the band.
     * Stars weaker than this magnitude are not copied to the XEphem file.
     */
    float mag ;

    /**
     * @brief list of the possible file patterns of the PSC files
     * These patterns are fixed by the standard distribution of the catalog.
     * The constant prefix "psc_" is omitted here.
     */
    static String[] pscPtt = {
        "aaa", "aab", "aac", "aad", "aae", "aaf", "aag", "aah", "aai", "aaj",
        "aak", "aal", "aam", "aan", "aao", "aap", "aaq", "aar", "aas", "aat",
        "aau", "aav", "aaw", "aax", "aay", "aaz", "aba", "abb", "abc", "abd",
        "abe", "abf", "abg", "abh", "abi", "abj", "abk", "abl", "abm", "abn",
        "abo", "abp", "abq", "abr", "abs", "abt", "abu", "abv", "abw", "abx",
        "aby", "abz", "aca", "acb", "acc", "acd", "ace", "baa", "bab", "bac",
        "bad", "bae", "baf", "bag", "bah", "bai", "baj", "bak", "bal", "bam",
        "ban", "bao", "bap", "baq", "bar", "bas", "bat", "bau", "bav", "baw",
    }
}

```

```

"bax", "bay", "baz", "bba", "bbb", "bbc", "bbd", "bbe", "bbf", "bbg",
"bbh", "bbi"
} ;

/**
 * @brief declination ranges (low and high) in the 92 PSC files in degrees.
 * Note that these numbers are in exactly the order associated with the pscPtt[] .
 */

static double[] pscLim = {
-89.992844, -74.5, /* psc_aaa */
-74.599998, -70.400002, /*psc_aab*/
-70.499992, -67.900002, /*psc_aac*/
-67.999992, -66.0, /* psc_aad */
-66.099991, -64.400002, /*psc_aae*/
-64.499992, -63.100002, /*psc_aaf*/
-63.199997, -62.0, /*psc_aag*/
-62.099998, -61.0, /*psc_aah*/
-61.099998, -59.900002, /*psc_aai*/
-59.999996, -58.900002, /*psc_aaj*/
-58.999996, -57.900002, /*psc_aak*/
-57.999996, -56.800018, /*psc_aal*/
-56.899998, -55.800003, /*psc_aam*/
-55.899998, -54.700001, /*psc_aan*/
-54.799999, -53.600002, /*psc_aao*/
-53.699997, -52.5, /*psc_aap*/
-52.599998, -51.400002, /*psc_aaq*/
-51.499996, -50.200001, /*psc_aar*/
-50.299999, -49.100002, /*psc_aas*/
-49.099998, -47.900002, /*psc_aat*/
-47.999996, -46.700001, /*psc_aau*/
-46.799999, -45.5, /*psc_aav*/
-45.599998, -44.300003, /*psc_aaw*/
-44.399998, -43.100002, /*psc_aax*/
-43.199997, -41.800007, /*psc_aay*/
-41.899998, -40.600002, /*psc_aaz*/
-40.699997, -39.300003, /*psc_aba*/
-39.399998, -38.100002, /*psc_abb*/
-38.199997, -36.900002, /*psc_abc*/
-36.999943, -35.600002, /*psc_abd*/
-35.699997, -34.400002, /*psc_abe*/
-34.499996, -33.300003, /*psc_abf*/
-33.399998, -32.100002, /*psc_abg*/
-32.199997, -31.0, /*psc_abh*/
-31.099995, -29.800011, /*psc_abi*/
-29.899998, -28.800001, /*psc_abj*/
-28.899996, -27.6, /*psc_abk*/
-27.699999, -26.5, /*psc_abl*/
-26.599998, -25.300001, /*psc_abm*/
-25.399998, -24.1, /*psc_abn*/
-24.199999, -22.9, /*psc_abo*/
-22.999994, -21.6, /*psc_abp*/
-21.699999, -20.4, /*psc_abq*/
-20.499998, -19.1, /*psc_abr*/
-19.199999, -17.800007, /*psc_abs*/
-17.899998, -16.5, /*psc_abt*/
-16.599998, -15.100003, /*psc_abu*/
-15.199999, -13.700001, /*psc_abv*/
-13.799997, -12.3, /*psc_abw*/
-12.399998, -10.8, /*psc_abx*/
-10.899999, -9.3, /*psc_aby*/
-9.399999, -7.8, /*psc_abz*/
-7.899999, -6.3, /*psc_aca*/
-6.399999, -4.8, /*psc_acb*/

```

```

-4.899989,-3.2, /*psc_acc*/
-3.299998,-1.500014, /*psc_acd*/
-1.599999,-2.0E-6, /*psc_ace*/
0.0,1.699999, /*psc_baa*/
1.600001,3.399998, /*psc_bab*/
3.3,5.099998, /*psc_bac*/
5.0,6.799999, /*psc_bad*/
6.7,8.499999, /*psc_bae*/
8.4,10.199999, /*psc_baf*/
10.1,11.899999, /*psc_bag*/
11.800001,13.699996, /*psc_bah*/
13.600002,15.399999, /*psc_bai*/
15.3,17.099998, /*psc_baj*/
17.0,18.899998, /*psc_bak*/
18.800001,20.699999, /*psc_bal*/
20.6,22.499998, /*psc_bam*/
22.4,24.299999, /*psc_ban*/
24.200003,26.199999, /*psc_bao*/
26.1,28.099998, /*psc_bap*/
28.000004,29.999998, /*psc_baq*/
29.900002,31.999981, /*psc_bar*/
31.9,33.899998, /*psc_bas*/
33.800003,35.899982, /*psc_bat*/
35.800003,37.899998, /*psc_bau*/
37.800003,39.999996, /*psc_bav*/
39.900002,42.099998, /*psc_baw*/
42.0,44.299999, /*psc_bax*/
44.200005,46.499996, /*psc_bay*/
46.400002,48.599998, /*psc_baz*/
48.5,50.799999, /*psc_bba*/
50.700001,53.099998, /*psc_bbb*/
53.0,55.299999, /* bbc */
55.200005,57.599998, /*psc_bbd*/
57.5,60.099998, /*psc_bbe*/
60.0,62.799999, /*psc_bbf*/
62.700001,66.199997, /*psc_bbg*/
66.100006,72.199974, /*psc_bbh*/
72.100006,89.990128 /*psc_bbi*/
    };

/**
 * @brief Ctor specifying the directory.
 * Contains the information that is needed to search the declination ranges in the files.
 * @param cat The name of an existing directory with the psc_??? files.
 */
Tmass2Xeph(final String cat)
{
    catdir = new File(cat);
} /* ctor */

/**
 * @brief Ctor specifying the directory.
 * Contains all the information that is needed to start a single query.
 * @param cat The name of an existing directory with psc_* files.
 * @param rahrs Center of field RA in hours
 * @param decdeg Center of field DEC in degrees
 * @param conearcs Cone field of view radius in arcseconds
 * @param band index of the infrared band.
 * @param mag Limiting magnitude
 */
Tmass2Xeph(final String cat, final double rahrs, final double decdeg, final double conearcs,
            int band, float mag )
{
    catdir = new File(cat);

```

```

    /* convert all position arguments to radians */
    dec = Math.toRadians(decdeg) ;
    ra = Math.toRadians(rahms*15.0) ;
    cone = Math.toRadians(conearcs/3600.0) ;
    this.band = band ;
    this.mag = mag ;
} /* ctor */

/**!*****
 * @brief distance between two coordinates along great circle.
 * @param ra object RA in degrees
 * @param dec object DEC in degrees
 * @return Distance from this pointing center to the object in radians.
 * @todo Speed this up by storing the fixed cosines and sines of the pointing center.
 */
double distance(double radeg, double decdeg)
{
    radeg = Math.toRadians(radeg) ;
    decdeg = Math.toRadians(decdeg) ;
    double c = Math.sin(dec)*Math.sin(decdeg) + Math.cos(dec)*Math.cos(decdeg)*Math.cos(ra-radeg) ;
    return Math.acos(c) ;
}

/**
 * @brief Convert one 2MASS line to XEphem
 * @param fields The fields (60?) obtained by splitting the catalog line at the vertical bars.
 * See <a href="https://irsa.ipac.caltech.edu/2MASS/download/allsky/format_psc.html">Data fields</a>
 */
void toXephem1(String[] fields)
{
    if ( fields.length < 15)
        /* wrong format */
        return ;

    /* check distance to pointing.. fields[0-1] are ra and dec degrees
    */
    double[] obj =new double[2] ;
    obj[0] = Double.valueOf(fields[0]) ;
    obj[1] = Double.valueOf(fields[1]) ;
    final double dist = distance(obj[0],obj[1]) ;
    if ( dist <= cone)
    {
        /* magnitude of this object
        */
        float m ;
        /* in cone search area: write to stdout unless magnitude weaker than limit */
        switch(band)
        {
        case 0:
            m = Float.parseFloat(fields[6]) ; break;
        case 2:
            m = Float.parseFloat(fields[14]) ; break;
        case 1:
        default:
            m = Float.parseFloat(fields[10]) ; break;
        }
        if ( m >= mag)
            return ;

        System.out.print("2MASSJ"+ fields[5].trim()) ;
        String rahms = Tyc22Xeph.hexRep(obj[0]/15.0) ;
        String dedms = Tyc22Xeph.hexRep(obj[1]) ;
        System.out.print(",f|S") ;
        System.out.print(", " + rahms) ;
    }
}

```

```

        System.out.print(", " + dedms) ;
        System.out.print(", " + m) ;

        System.out.print(",") ;
        System.out.print(",") ;
        System.out.println() ;
    }

} /* toXephem1 */

/**
 * @brief scan the input file and emit the XEphem file to stdout.
 */
void toXephem()
{
    /* Translate pointing cone into a file name list of PSC files.
     * First describe range in degrees.
     */
    double[] declim = new double[2] ;
    declim[0] = Math.max(Math.toDegrees(dec-cone),-90.0) ;
    declim[1] = Math.min(Math.toDegrees(dec+cone),90.0) ;
    for(int f=0 ; f < pscPtt.length; f++)
    {
        if ( declim[0] > pscLim[2*f+1]+1.e-6 || declim[1] < pscLim[2*f]-1.e-6 )
        {
            /* skip file, not in cone range: Either lower cone rim
             * larger than file range or upper rim smaller than file
             */
            continue ;
        }

        File psc = new File(catdir,"psc_"+ pscPtt[f]) ;
        String pscNam = psc.getName() ;
        /* to avoid scanning files multiple times, use a 'chmod -r psc_aa[a-b]' or similar wildcard
         */
        if ( psc.isFile() && psc.canRead() && pscNam.indexOf("psc_") >= 0 && pscNam.length() == 7)
        {
            try
            {
                FileReader istream = new FileReader(psc) ;
                BufferedReader dstream = new BufferedReader(istream) ;
                for(;;)
                {
                    String lin = dstream.readLine() ;
                    if ( lin == null)
                        break ;
                    /* split lines along vertical bars
                     */
                    final String[] fields = lin.split("[|]") ;
                    toXephem1(fields) ;
                }
            }
            catch (Exception ex)
            {
                System.err.println(ex) ;
                ex.printStackTrace() ;
            }
        }
        else
        {
            System.err.println(pscNam + " not readable") ;
        }
    }
}

```

```

    } /* end loop over 50+ PSC files */
} /* toXephem */

/*!*****
 * @brief Convert declination representation to degrees
 * @param dec Either a floating point string (in units of degrees) or the +-DD:MM:SS.sss format
 * @return the equivalent value in degrees
 */
static double dec2deg(final String dec)
{
    if (dec.indexOf(":") >= 0 )
    {
        final String[] dms = dec.split(":") ;
        int d=0 ;
        int m=0 ;
        double s=0 ;
        if ( dms.length >= 1)
        {
            d = Integer.valueOf(dms[0]) ;
        }
        if ( dms.length >= 2)
        {
            m = Integer.valueOf(dms[1]) ;
        }
        if ( dms.length >= 3)
        {
            s = Double.valueOf(dms[2]) ;
        }
        double deg = Math.abs(d)+ m/60.0 + s/3600.0 ;
        return (dms[0].indexOf("-") >= 0 ) ? -deg : deg;
    }
    else
        return Double.valueOf(dec) ;
} /* dec2deg */

/** convert right ascension representation to hours
 * @param ra Either a floating point string (in units of hours) or the +HH:MM:SS.sss format
 * @return the equivalent value in hours
 */
static double ra2hrs(final String ra)
{
    if (ra.indexOf(":") >= 0 )
    {
        String[] hms = ra.split(":") ;
        int h=0 ;
        int m=0 ;
        double s=0 ;
        if ( hms.length >= 1)
        {
            h = Integer.valueOf(hms[0]) ;
        }
        if ( hms.length >= 2)
        {
            m = Integer.valueOf(hms[1]) ;
        }
        if ( hms.length >= 3)
        {
            s = Double.valueOf(hms[2]) ;
        }
        double hrs = Math.abs(h)+ m/60.0 + s/3600.0 ;
        return (hms[0].indexOf("-") >= 0 ) ? -hrs : hrs;
    }
    else
        return Double.valueOf(ra) ;
}

```



```

} /* ra2hrs */

/*****
 * @brief Scan the declination limits in all readable files.
 * This is a developers method to detect which psc_* files comprise which declination bands,
 * and is used to fill in the pscLim[] values.
 * make Cds2XEphem.jar
 * java -cp . de.mpg.mpia.cds2xephem.Tmass2Xeph -d ~/work/tmp/2mass
 */
void scan()
{
    /* loop over all "psc_" files in the directory
    */
    File[] pscs = catdir.listFiles() ;
    for(int f =0 ; f < pscs.length ; f++)
    {
        File psc = pscs[f] ;
        String pscNam = psc.getName() ;
        /* to avoid scanning files multiple times, use a 'chmod -r psc_aa[a-b]' o rsimilar wildcard
        */
        if ( psc.isFile() && psc.canRead() && pscNam.indexOf("psc_") >= 0 && pscNam.length() == 7)
        {
            double[] lims = new double[2] ;
            lims[0] = 100.0 ;
            lims[1] = -100.0 ;
            try
            {
                FileReader istream = new FileReader(psc) ;
                BufferedReader dstream = new BufferedReader(istream) ;
                for(;;)
                {
                    String lin = dstream.readLine() ;
                    if ( lin == null)
                        break ;
                    /* split lines along vertical bars
                    */
                    final String[] fields = lin.split("[|]") ;

                    if ( fields.length >= 2)
                    {
                        lims[0] = Math.min(lims[0], Double.valueOf(fields[1])) ;
                        lims[1] = Math.max(lims[1], Double.valueOf(fields[1])) ;
                    }
                }
            }
            catch (Exception ex)
            {
                System.err.println(ex) ;
                ex.printStackTrace() ;
            }
            System.out.println("" + lims[0] + "," + lims[1]+", /*" + psc.getName() + "*/" ) ;
        }
    }
} /* scan */

/*****
 * Print a usage hint on stderr.
 * @since 2019-04-01
 */
static public void usage()
{
    System.err.println("Error: command line argument missing") ;
    System.err.println("Usage: java -cp Cds2XEphem.jar de.mpg.mpia.cds2xephem.Tmass2Xeph [-J|-H|-K] [-m magnitude]
} /* usage */

```

```

/**
 * @param argv Vector of the command line arguments
 * Compilation:
 * javac -cp . de/mpg/mpia/cds2xephem/*.java
 * Usage:
 * java -cp . de.mpg.mpia.cds2xephem.Tmass2Xeph [-J|-H|-K] [-m magnitude] catdir ra dec arcsecCone
 */
static public void main(String argv[])
{
    if ( argv.length < 4 )
        usage() ;
    else
    {
        boolean devel = false ;
        /* 0=J, 1=H, 2=K */
        int band=1 ;
        /* not limit to magnitude by default */
        float mag= 99 ;
        String dir =null;
        String ra =null;
        String dec =null;
        String cone =null;
        /* can all command line arguments (file names, bands or the -d option)
        */
        for(int i=0 ; i < argv.length ; i++)
        {
            if ( argv[i].startsWith("-d") )
                devel = true ;
            else if ( argv[i].startsWith("-J") )
                band = 0 ;
            else if ( argv[i].startsWith("-H") )
                band = 1 ;
            else if ( argv[i].startsWith("-K") )
                band = 2 ;
            else if ( argv[i].startsWith("-m") )
            {
                mag = Float.valueOf(argv[i+1]) ;
                i++ ;
            }
            else if ( devel )
            {
                /* render the catalogs in the order of the command line
                */
                dir = argv[i] ;
            }
            else if ( i == argv.length-1)
                cone = argv[i] ;
            else if ( i == argv.length-2)
                dec = argv[i] ;
            else if ( i == argv.length-3)
                ra = argv[i] ;
            else if ( i == argv.length-4)
                dir = argv[i] ;
        }
        if ( devel)
        {
            Tmass2Xeph tm = new Tmass2Xeph(dir) ;
            tm.scan() ;
        }
        else if ( dir == null || ra == null || dec == null || cone == null)
            usage() ;
        else
        {

```

```

        double RA=ra2hrs(ra) ;
        double DEC = dec2deg(dec) ;
        double fov = Double.valueOf(cone) ;
        Tmass2Xeph tm = new Tmass2Xeph(dir,RA,DEC,fov,band,mag) ;
        System.out.println("# generated from "+ dir + " with " + Tmass2Xeph.class.getName()
            + " " + java.time.Clock.systemUTC().instant() ) ;
        System.out.println("# [viXra:1802.0035] Richard J. Mathar") ;

        tm.toXephem() ;
    }
}

} /* main */

} /* Tmass2Xeph */

```

## 10. File de/mpg/mpia/cds2xephem/Ucac42Xeph.java

```

package de.mpg.mpia.cds2xephem ;

import java.lang.Math ;
import java.io.FileInputStream ;
import java.io.BufferedInputStream ;
import java.nio.ByteBuffer ;
import java.nio.ByteOrder ;

/**
 * A translator of the UCAC4 Catalog to the XEphem format
 * @since 2020-08-28
 * @since 2020-09-22 Added option -m to discard entries with weak magnitudes.
 * @author R. J. Mathar
 */
class Ucac42Xeph
{
    /** number of bytes in each unformatted line of the binary file
     */
    static int BYTES_BINLINE = 78 ;

    /**
     * The binary file of one zone of the catalog.
     * This is one of the ftp://cdsarc.u-strasbg.fr/ftp/cats/I/322A/UCAC4/u4b/z*
     * where the wild card (star) is a 3-digit number from 001 to 900.
     */
    String orbfile ;

    /**
     * Ctor specifying the binary one per zone
     * @param catfname The name of an existing UCAC4 file.
     */
    Ucac42Xeph(final String catfname)
    {
        orbfile = catfname ;
    } /* ctor */

    /**
     * @brief Convert 1 to 4 bytes to integer.
     * @param star collection of (signed) bytes representing a stream of integers
     * @param off 0-based offset into start[] with LSB
     * @param len Number of bytes in start[] representing the int.
     * This is typically 4 for 32-bit integers, 2 for 16-bit integers.
     */
    int by2int(byte[] star, int off, int len)

```

```

{
    int out =0 ;
    ByteBuffer bb = ByteBuffer.wrap(star,off,len) ;
    bb.order(ByteOrder.LITTLE_ENDIAN) ;
    switch(len)
    {
    case 1:
        return bb.get() ;
    case 2:
        return bb.getShort() ;
    case 4:
        return bb.getInt() ;
    default:
        return 0 ;
    }
}

/**
 * @brief scan the input file and emit the XEphem file to stdout.
 * @param mag A limit magnitude. Stars weaker than this are not emitted.
 */
void toXephem(float mag)
{
    try
    {
        FileInputStream istream = new FileInputStream(orbfile) ;
        BufferedInputStream dstream = new BufferedInputStream(istream) ;
        byte[] star = new byte[BYTES_BINLINE] ;
        System.out.println("# " + orbfile) ;
        for(int starNo=0;starNo++)
        {
            int byts = dstream.read(star, 0,BYTES_BINLINE) ;
            if ( byts < BYTES_BINLINE)
                break ;

            /* byts[0..3] = ra, [4..7]=spd, [8..9] = magm, [10..11] = maga,
             * 12=sigmag, 13=objt, 14=cdf, 15=sigra, 16=sigdc, 17=na1, 18=nu1,
             * 18=cu1, [19..20]=cepra, [21..22]=cepdc, [23..24]=pmra2, [25..26]=pmdc2,
             * 27=sigpmr, 28=sigpmd, [29..33]=pts_key, [34..35]=j_m, [35..37]=h_m,
             * [38..39]=k_m, [40..42]=icqflg(3), [43.45]=q2mflg(3), [46..55]=apasm(5),
             * [56..60]=apase(5), 61=gcflg, [62..65]=mcf, 66=leda, 67=x2m, [68..71]=rnm,
             * [72..73]=zn2, [74..77]=rnz (total of 78 bytes per object)
             */

            /* RA at j2000 in mas. Note integers are little endian in the binary, and star[] are signed bytes
             */
            int ra =by2int(star, 0,4);

            /* south pole distance in mas
             */
            int sdp =by2int(star, 4,4) ;
            /* declination in mas, 90*3600*1000 offset
             */
            sdp -= 324000000 ;

            /* proper motion RA*cos(dec0) in 0.1 mas/yr
             */
            int pmrac =by2int(star, 24,2) ;

            /* proper motion dec in 0.1 mas/yr
             */
            int pmdc =by2int(star, 26,2) ;

            /* star id number

```

```

    */
    int rnm =by2int(star, 68,4) ;

    /* fit model magnitude in millimag
    */
    int mmag = by2int(star,8,2) ;

    if ( 0.001f*mmag <= mag)
    {
        System.out.print("UCAC"+rnm) ;
        System.out.print(",f|S") ;

        /* convert RA to hrs */
        double raHrs = ra/1000.0/3600.0/15.0 ;
        /* print as HH:MM:SS string
        */
        System.out.print(", " + Tyc22Xeph.hexRep(raHrs) ) ;

        /* proper motion in RA
        */
        System.out.print("|" + pmrac/10.0) ;

        /* convert DEC to degrees */
        double decDeg = sdp/1000.0/3600.0 ;
        /* print as DD:MM:SS string
        */
        System.out.print(", " + Tyc22Xeph.hexRep(decDeg) ) ;

        /* proper motion in DEC
        */
        System.out.print("|" + pmdec/10.0) ;

        /* magnitude */
        System.out.print(", " + mmag/1000.0) ;

        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/cds2xephem/*.java
 * java -cp . de.mpg.mpia.cds2xephem.Ucac42Xeph [-m magnitude] UCAC4/u4b/ziii [UCAC$/u4b/zjjj ...]>> ~/.xephem/Ucac4
 */
static public void main(String argv[])
{
    if ( argv.length == 0 )
    {
        System.err.println("Error: command line argument missing") ;
        System.err.println("Usage: java -cp Cds2XEphem.jar de.mpg.mpia.cds2xephem.Ucac42Xeph [-m magnit] ziii [zjjj]") ;
    }
    else
    {
        /* no limit to magnitude by default
        */
        float mag = 99 ;
    }
}

```

```

    /* loop over all zone files mentioned in the command line
    */
    for(int i=0 ; i < argv.length ; i++)
    {
        if ( argv[i].startsWith("-m") )
        {
            i++ ;
            mag = Float.valueOf(argv[i]) ;
        }
        else
        {
            Ucac42Xeph orb = new Ucac42Xeph(argv[i]) ;
            System.out.println("# generated from "+argv[i] + " with " + Ucac42Xeph.class.getName()
                +" " + java.time.Clock.systemUTC().instant() ) ;
            System.out.println("# [viXra:1802.0035] Richard J. Mathar" ) ;

            orb.toXephem(mag) ;
        }
    }
}

} /* main */

} /* Ucac42Xeph */

```

## 11. File de/mpg/mpia/cds2xephem/UsnoA22Xeph.java

```

package de.mpg.mpia.cds2xephem ;

import java.lang.Math ;
import java.io.File ;
import java.io.FileInputStream ;
import java.io.BufferedReader ;
import java.io.BufferedInputStream ;
import java.io.IOException ;
import java.io.InputStream ;
import java.io.InputStreamReader ;
import java.io.OutputStream ;
import java.io.OutputStreamWriter ;
import java.nio.ByteBuffer ;
import java.nio.ByteOrder ;

/**
 * A translator of the USNO A2 Catalog or USNO SA2 to the XEphem format
 * @since 2021-10-07
 * @author R. J. Mathar
 */
class UsnoA22Xeph
{
    /** number of bytes in each unformatted line of the binary file
     * For each target 3 values, each one integer (4 bytes)
     */
    static int BYTES_BINLINE = 12 ;

    /**
     * *****
     * The binary file of one zone of the catalog.
     * This is one of the https://ftp.imcce.fr/pub/catalogs/USNO-SA2/zone\*.cat
     * or http://catalogs.astro.uni-altai.ru/usnosa/zone\*.cat
     * where the wild card (star) is a 4-digit number from 0000 to 1725 (and a multiple of 25).
     */
    String orbfile ;

    /**
     * Ctor specifying the binary one per zone
     * @param catfname The name of an existing USNO 2A or USNO SA2.0 file.
     */
    UsnoA22Xeph(final String catfname)

```

```

{
    orbfile = catfname ;
} /* ctor */

/**
 * @brief Convert 1 to 4 bytes to integer.
 * This is the big-endian variant of the Ucac42Xeph class
 * @param star collection of (signed) bytes representing a stream of integers
 * @param off 0-based offset into start[] with LSB
 * @param len Number of bytes in start[] representing the int.
 * This is typically 4 for 32-bit integers, 2 for 16-bit integers.
 */
int by2int(byte[] star, int off, int len)
{
    int out =0 ;
    ByteBuffer bb = ByteBuffer.wrap(star,off,len) ;
    bb.order(ByteOrder.BIG_ENDIAN) ;
    switch(len)
    {
    case 1:
        return bb.get() ;
    case 2:
        return bb.getShort() ;
    case 4:
        return bb.getInt() ;
    default:
        return 0 ;
    }
}

/**
 * @brief scan the input file and emit the XEphem file to stdout.
 * @param limmag A limit magnitude. Stars weaker than this are not emitted.
 * @param all Emits all stars, even if the magnitudes have been flagged as bad in the catalog
 * @param magB if true, the blue magnitude is placed into the output
 * @param magR if true, the red magnitude is placed into the output
 * If neither magB nor magR or both are given, the mean magnitude of both plates is used.
 */
void toXephem(float limmag, boolean all, boolean magB, boolean magR)
{
    try
    {
        FileInputStream istream = new FileInputStream(orbfile) ;
        BufferedInputStream dstream = new BufferedInputStream(istream) ;
        final byte[] star = new byte[BYTES_BINLINE] ;
        System.out.println("# " + orbfile) ;

        /* remove directory name, skip the "zone" and extract next 4 integers
        */
        final String zon = (new File(orbfile)).getName().substring(4,8) ;
        for(int starNo=0;starNo++)
        {
            int byts = dstream.read(star, 0,BYTES_BINLINE) ;
            if ( byts < BYTES_BINLINE)
                break ;

            /* byts[0..3] = ra, [4..7]=dec, [8..11] = mag,
            * (total of 12 bytes per object)
            */

            /* RA at j2000 in hours. Note integers are big endian in the binary, and star[] are signed bytes
            */
            int ra = by2int(star, 0,4) ;

```

```

/* south pole distance in degrees
*/
int dec = by2int(star, 4,4) ;

/* red and blue magnitude and flags
* Format SQFFFBRRR (decimal)
*/
int mag = Math.abs(by2int(star, 8,4)) ;
int magr = mag % 1000 ;
mag /= 1000 ;
int magb = mag % 1000 ;
mag /= 1000000 ;
int qflag = mag %2 ;

if ( all || (qflag == 0) )
{
    /* preserve red or blue or mean magnitude
    */
    if ( magR && ! magB)
        mag = magr ;
    else if ( ! magR && magB)
        mag = magb ;
    else
        mag = (magr+magb)/2 ;

    /* the integers were actually deci-magnitudes
    */
    if ( mag/10.0 <= limmag)
    {
        System.out.print("USNO"+ "-" +zon+"-"+(starNo+1)) ;
        System.out.print(",f|S") ;

        /* convert RA to hrs */
        double raHrs = ra/100.0/3600.0/15.0 ;
        /* print as HH:MM:SS string
        */
        System.out.print(", " + Tyc22Xeph.hexRep(raHrs) ) ;

        /* convert south pole distance to degrees and DEC */
        double decDeg = dec/100.0/3600.0 -90.0 ;
        /* print as DD:MM:SS string
        */
        System.out.print(", " + Tyc22Xeph.hexRep(decDeg) ) ;

        /* magnitude */
        System.out.print(", " + mag/10.0) ;

        System.out.println() ;
    }
}
}
dstream.close() ;
}
catch (Exception ex)
{
    System.err.println(ex) ;
    ex.printStackTrace() ;
}
} /* toXephem */

/**
 * @param argv Vector of the command line arguments
 * Usage:
 * javac -cp . de/mpg/mpia/cds2xephem/*.java

```



```

* java -cp . de.mpg.mpia.cds2xephem.UsnoA22Xeph [-m magnitude] [-a] [-B] [-R] usno/zoneiiii.cat [usno/zonejjjj.cat
*/
static public void main(String argv[])
{
    if ( argv.length == 0 )
    {
        System.err.println("Error: command line argument missing") ;
        System.err.println("Usage: java -cp Cds2XEphem.jar de.mpg.mpia.cds2xephem.UsnoA22Xeph [-m magnit] [-B] [-R]
    }
    else
    {
        /* no limit to magnitude by default
        */
        float mag = 99 ;
        boolean all = false ;
        boolean magR = false ;
        boolean magB = false ;

        /* loop over all zone files mentioned in the command line
        */
        for(int i=0 ; i < argv.length ; i++)
        {
            if ( argv[i].startsWith("-m") )
            {
                i++ ;
                mag = Float.valueOf(argv[i]) ;
            }
            else if ( argv[i].startsWith("-a") )
                all = true ;
            else if ( argv[i].startsWith("-B") )
                magB = true ;
            else if ( argv[i].startsWith("-R") )
                magR = true ;
            else
            {
                UsnoA22Xeph orb = new UsnoA22Xeph(argv[i]) ;
                System.out.println("# generated from "+argv[i] + " with " + UsnoA22Xeph.class.getName()
                    + " " + java.time.Clock.systemUTC().instant() ) ;
                System.out.println("# [viXra:1802.0035] Richard J. Mathar") ;

                orb.toXephem(mag,all,magB,magR) ;
            }
        }
    }

} /* main */

} /* UsnoA22Xeph */

```

## 12. File de/mpg/mpia/cds2xephem/MpcObscod.java

```

package de.mpg.mpia.cds2xephem ;

import java.io.BufferedReader ;
import java.io.FileReader ;
import java.nio.charset.Charset ;
import java.util.Vector ;
import java.util.Collections ;

/** 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.
    * @param verb If true, add comment lines as used in the english wikipedia
    *   of Observatory Codes.
    */
    void toXephem(boolean verb)
    {
        /* Reference ellipsoid is the WGS84 system.
        */
        Geoid wgs84 = new Geoid() ;
        try
        {
            FileReader istream = new FileReader(cfile) ;
            BufferedReader dstream = new BufferedReader(istream) ;
            for(;;)
            {
                String lin = dstream.readLine() ;
                if ( lin == null)
                    break ;
                toXephem(lin,wgs84,verb) ;
            }
        }
        catch (Exception ex)
        {
            System.err.println(ex) ;
            ex.printStackTrace() ;
        }
    }

    /**
    * @brief Scan the set of input files and emit the XEphem file to stdout.
    *   The input file must follow the XEphem conventions (using semicolons as delimiters etc)
    * @param optind The first valid index into argv which is an input file name
    * @param argv The collection of input file names to be merged and sorted.
    * @param uniq If true, print only one representative if the long/latitude has duplicates.
    * @since 2021-07-28
    */
    static void sort(final int optind, final String[] argv, final boolean uniq)
    {
        Vector<XephSite> sites = new Vector<XephSite>() ;
        for(int i=optind ; i < argv.length ; i++)
        {
            try
            {
                FileReader istream = new FileReader(argv[i]) ;
                BufferedReader dstream = new BufferedReader(istream) ;

```

```

        for(;;)
        {
            String lin = dstream.readLine() ;
            if ( lin == null)
                break ;
            /* skip comments and lines with less than 4 semicolons
            */
            if ( !lin.startsWith("#") && lin.length() > 4)
            {
                XephSite thisS = new XephSite(lin) ;
                if ( !uniq || ! sites.contains(thisS) )
                    sites.add( thisS ) ;
            }
        }
    }
    catch (Exception ex)
    {
        System.err.println(ex) ;
        ex.printStackTrace() ;
    }
}

Collections.sort(sites) ;

for( XephSite loc : sites)
    System.out.println( loc) ;
}

/*****
 * @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 += String.format("%3d",d) + " " + String.format("%2d",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.
* @param wgs84 The description of the Earth geoid (equatorial radiuse, flattening...)
* @param verb If true, emit a comment line suitable for the wikipedia list
*/
protected void toXephem(String ascline, Geoid wgs84, boolean verb)
{
    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) ;

        /* there may be & in the HTML code ...
        */
        descr = descr.replaceAll("&","&") ;

        try
        {
            /* geographic longitude, degrees
            */
            double longi = Double.parseDouble(longS) ;
            if ( longi > 180.0 )
                longi -= 360.0 ;
            /* geographic longitude, 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 latDeg = lTude(geodetic[1],false) ;
            String lonDeg = lTude(geodetic[0],true) ;
            if ( verb )
            {
                String cmt = "# geo:"
                    + String.format("%.5f",Math.toDegrees(geodetic[1])) + ","
                    + String.format("%.5f",Math.toDegrees(geodetic[0])) ;
            }
        }
    }
}

```

```

        String wiki = "{{Coord" ;
        wiki += "|" + latDeg.replaceAll("\\s+", "|")
            + "|" + lonDeg.replaceAll("\\s+", "|")
            + "}}";
        System.out.println(cmt+ " " + wiki) ;
    }

    String out = code + " " + String.format("%-50s",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.cds2xephem.MpcObscod [-v] directory/Obsc*.html
 * java -cp . de.mpg.mpia.cds2xephem.MpcObscod -s [-u] sitefile [sitefile ...]
 * </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.
 * The option -v triggers the a list of observatory codes as used in the
 * wikipedia article is also generated as comments.
 * The option -s assumes that the final arguments are file names of observatory sites
 * following the XEphem convents and prints them sorted along geographic latitude to stdout.
 */
static public void main(String[] argv)
{
    boolean verb = false ;
    boolean sort = false ;
    boolean uniq = false ;
    int optind=0 ;
    for( ; optind < argv.length ; optind++)
    {
        if ( argv[optind].startsWith("-v") )
            verb = true ;
        else if ( argv[optind].startsWith("-s") )
            sort = true ;
    }

```

```

        else if ( argv[optind].startsWith("-u") )
            uniq = true ;
        else
            break ;
    }

    if ( argv.length -optind == 0 )
    {
        System.err.println("Error: command line argument missing") ;
        System.err.println("Usage: java -cp Cds2XEphem.jar de.mpg.mpia.cds2xephem.MpcObscod [-v] ObsCodes.html") ;
        System.err.println("Usage: java -cp Cds2XEphem.jar de.mpg.mpia.cds2xephem.MpcObscod -s [-u] sitefile [sitefile]") ;
    }
    else
    {
        String cfname = argv[argv.length-1] ;

        System.out.println("# generated from "+cfname + " with " + MpcObscod.class.getName()
            +" " + java.time.Clock.systemUTC().instant() ) ;
        System.out.println("# [viXra:1802.0035] Richard J. Mathar") ;

        if ( sort )
        {
            MpcObscod.sort(optind,argv,uniq) ;
        }
        else
        {
            MpcObscod cod = new MpcObscod(cfname) ;
            cod.toXephem(verb) ;
        }
    }
} /* main */
} /* MpcObscod */

```

### 13. File de/mpg/mpia/cds2xephem/Geoid.java

```

package de.mpg.mpia.cds2xephem ;

import java.lang.Math ;

/** Definition of an oblate reference ellipsoid.
 * The class converts 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 */

/** Default ctor with WGS84 Earth 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 used.
*
* @param xyz The x, y and z Cartesian coordinates of the point relative to the geocenter in meters.
* Must have the same units as the major axis of the constructor.
* @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+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 dzsqr = Math.hypot(D,xyz[2]) ;

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

    return coord ;
}

```

```

} /* toSpherical */

} /* Geoid */

```

#### 14. File de/mpg/mpia/cds2xephem/XephSite.java

```

package de.mpg.mpia.cds2xephem ;

/**
 * A site of an observatory (name and WGS84 coordinates).
 * @since 2021-07-28
 * @author R. J. Mathar
 */
class XephSite implements Comparable<XephSite>
{
    /** length reserved for the name in the beautified (aligned) output
    */
    static String desiFmt = "%-60s" ;

    /**
    * The designation(s).
    */
    String desi ;

    /**
    * The geographic latitude in the DD MM SS.ss [N|S] format
    */
    String lat ;
    /**
    * The geographic latitude in degrees in the range -90..+90
    */
    double latD ;

    /**
    * The geographic longitude in the DD MM SS.ss [E|W] format
    */
    String lon ;
    /**
    * The geographic longitude in degrees in the range -180..+180
    */
    double lonD ;

    /**
    * The altitude above the geoid
    */
    String alt ;

    /**
    * The time zone
    */
    String tzo ;

    /**
    * Ctor specifying a line of the XEphem site type.
    * @param xeline The line with three semicolons.
    */
    XephSite(final String xeline)
    {
        /* split components.
        * There may &amp; residuals of the MPC HTML pages still linger in the description.

```



```

*/
String [] fields = xeline.replaceAll("&","&").split(";",5) ;

desi = fields[0].trim() ;
lat = fields[1].trim() ;
lon = fields[2].trim() ;
alt = fields[3].trim() ;
tzo = fields[4].trim() ;

String[] dms = lat.split("\\s+",4) ;
latD = Double.parseDouble(dms[0])
      + Double.parseDouble(dms[1]) /60.0
      + Double.parseDouble(dms[2]) /3600.0 ;
if ( dms[3].equals("S") )
    latD *= -1 ;

/* beware: the original XEphem sites list has an observatory
* where the initial 0 of the longitude is missing....
*/
dms = lon.split("\\s+",4) ;
if ( dms.length >= 4 )
{
    lonD = Double.parseDouble(dms[0])
          + Double.parseDouble(dms[1]) /60.0
          + Double.parseDouble(dms[2]) /3600.0 ;
    if ( dms[3].equals("W") )
        lonD *= -1 ;
}
else if ( dms.length == 3 && dms[2].length() == 1 )
{
    lonD = Double.parseDouble(dms[0]) /60.0
          + Double.parseDouble(dms[1]) /3600.0 ;
    if ( dms[2].equals("W") )
        lonD *= -1 ;
}
} /* ctor */

/** Simple comparison. Equality means the distances are the same, one by one.
* @param oth The LCF string to compare with.
* @return True if each of the distances matches with the distance in oth.
*/
public boolean equals(final Object oth)
{
    if ( this == oth )
        return true;
    else if ( oth instanceof XephSite )
        return ( compareTo((XephSite)oth) == 0 ) ? true : false ;
    else
        return false;
} /* equals */

/** Compare by geographic latitude and longitude
* @param oth The second location
* @return -1 if the first location is closer to the south pole.
* On equal latitudes -1 if the first location is more to the west.
* If latitudes and longitudes are the same (within 5 m), return zero.
*/
public int compareTo(final XephSite oth)
{
    if ( latD < oth.latD-1.e-6 )
        return -1 ;
    else if ( latD > oth.latD+1.e-6 )
        return 1 ;

```

```
    else if ( lonD < oth.lonD-1.e-6)
        return -1 ;
    else if ( lonD > oth.lonD+1.e-6)
        return 1 ;
    else
        return 0 ;
}

public String toString()
{
    String s = String.format(desiFmt,desi) + " ";
    s += " " + String.format("%13s",lat) + " ";
    s += " " + String.format("%14s",lon) + " ";
    s += " " + String.format("%5s",alt) + " ";
    s += " " + tzo ;
    return s ;
}

} /* XEphSite */
```

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