CENTRE NATIONAL D'ÉTUDES SPATIALES

GENIUS

GENeration of Interface for Users of Scientific S/W

Application to PATRIUS ⇨ GENOPUS (V2.1.1)

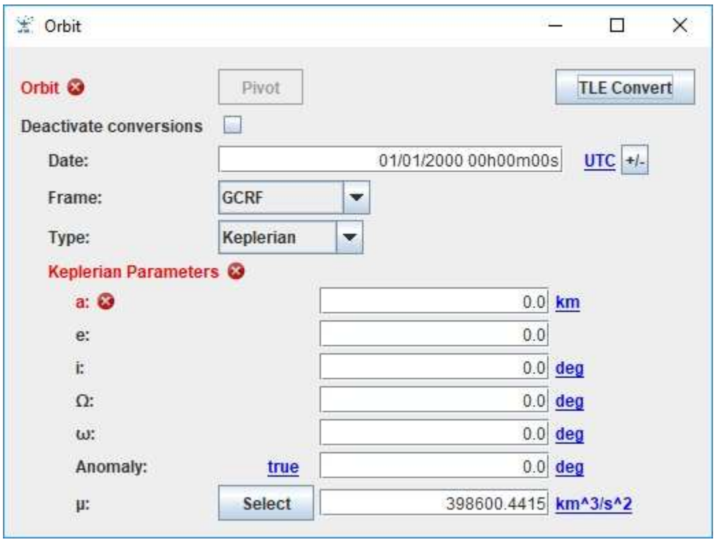
Wiki on http://genopus.cnes.fr

GENOPUS (basic principles)

- A Flight Dynamics oriented library, based on GENIUS (*for GUI*) and PATRIUS, including widgets about *Date*, *Frame*, *Orbit*, *Attitude*, *Maneuvers*, *Vehicle*, *Events*, …
- Not only data entry but also some computations (parameters conversions)
- One of the basic requirement is:
  - Any main widget, linked to a PATRIUS object will have at least two constructors:
    - One without arguments (*for using GComponentList*)
    - One with the corresponding PATRIUS object
  - Any main widget, linked to a PATRIUS object will have a *getPatriusObject* method that will return this kind of object,
- Very easy use: not more complex as for a simple real entry widget!

GENOPUS Exercise #1

- Only with a dozen of lines of code, create this GUI …



Orbit

Deactivate conversions ☐

Date: 01/01/2000 00h00m00s UTC

Frame: GCRF

Type: Keplerian

Keplerian Parameters

a: 0.0 km

e: 0.0

i: 0.0 deg

Ω: 0.0 deg

ω: 0.0 deg

Anomaly: true 0.0 deg

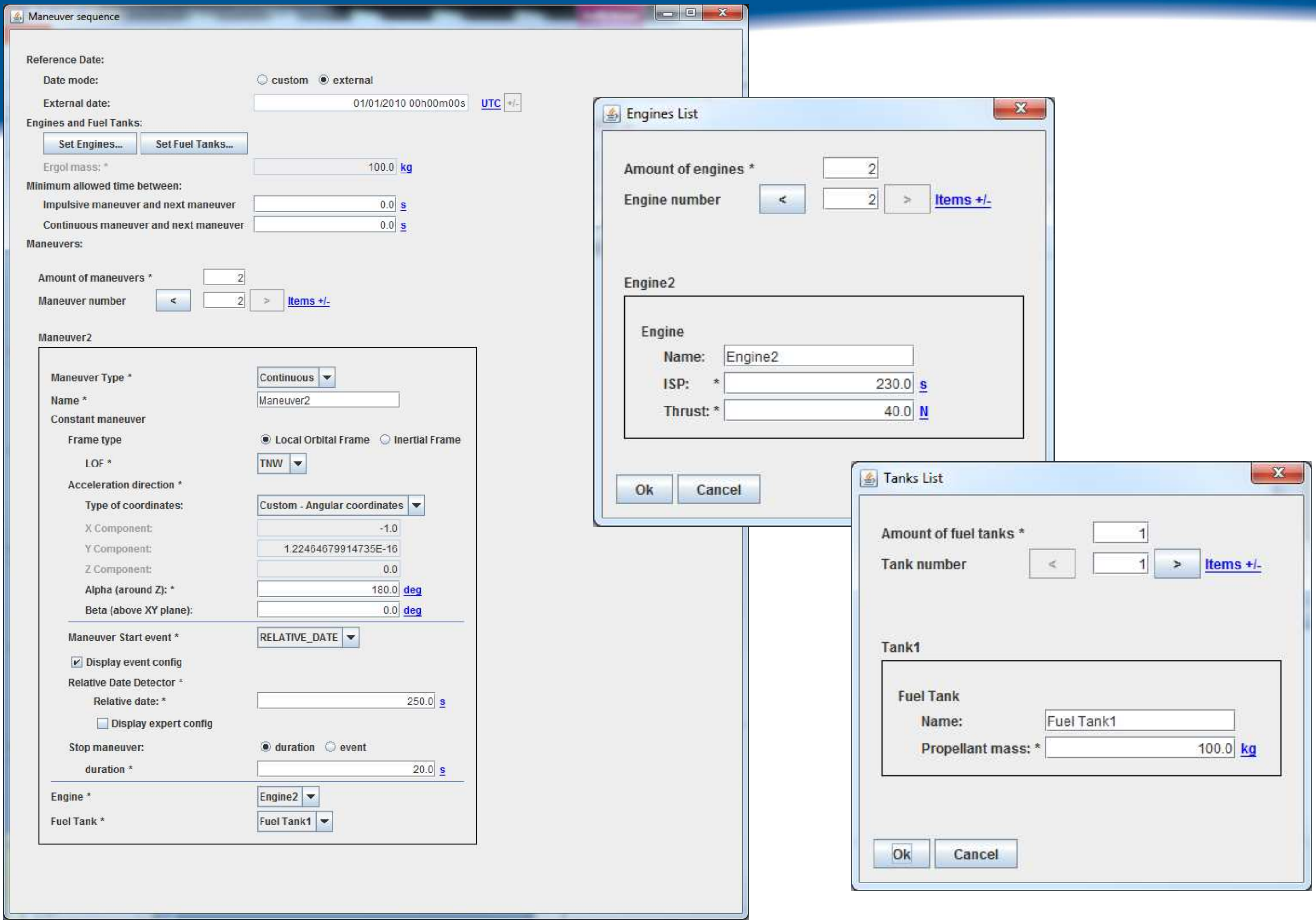
μ: 398600.4415 km^3/s^2

CENTRE NATIONAL D'ÉTUDES SPATIALES

GENOPUS

Some examples …

GENOPUS (GPManeuverSequence)



Maneuver sequence

Reference Date:

Date mode: ○ custom ● external

External date: 01/01/2010 00h00m00s UTC +/-

Engines and Fuel Tanks:

Ergol mass: * 100.0 kg

Minimum allowed time between:

Impulsive maneuver and next maneuver 0.0 s

Continuous maneuver and next maneuver 0.0 s

Maneuvers:

Amount of maneuvers * 2

Maneuver number < 2 > Items +/-

Maneuver2

Maneuver Type * Continuous

Name * Maneuver2

Constant maneuver

Frame type ● Local Orbital Frame ○ Inertial Frame

LOF * TNW

Acceleration direction *

Type of coordinates: Custom - Angular coordinates

X Component: -1.0

Y Component: 1.22464679914735E-16

Z Component: 0.0

Alpha (around Z): * 180.0 deg

Beta (above XY plane): 0.0 deg

Maneuver Start event * RELATIVE_DATE

☑ Display event config

Relative Date Detector *

Relative date: * 250.0 s

□ Display expert config

Stop maneuver: ● duration ○ event

duration * 20.0 s

Engine * Engine2

Fuel Tank * Fuel Tank1

Engines List

Amount of engines * 2

Engine number < 2 > Items +/-

Engine2

Engine

Name: Engine2

ISP: * 230.0 s

Thrust: * 40.0 N

Tanks List

Amount of fuel tanks * 1

Tank number < 1 > Items +/-

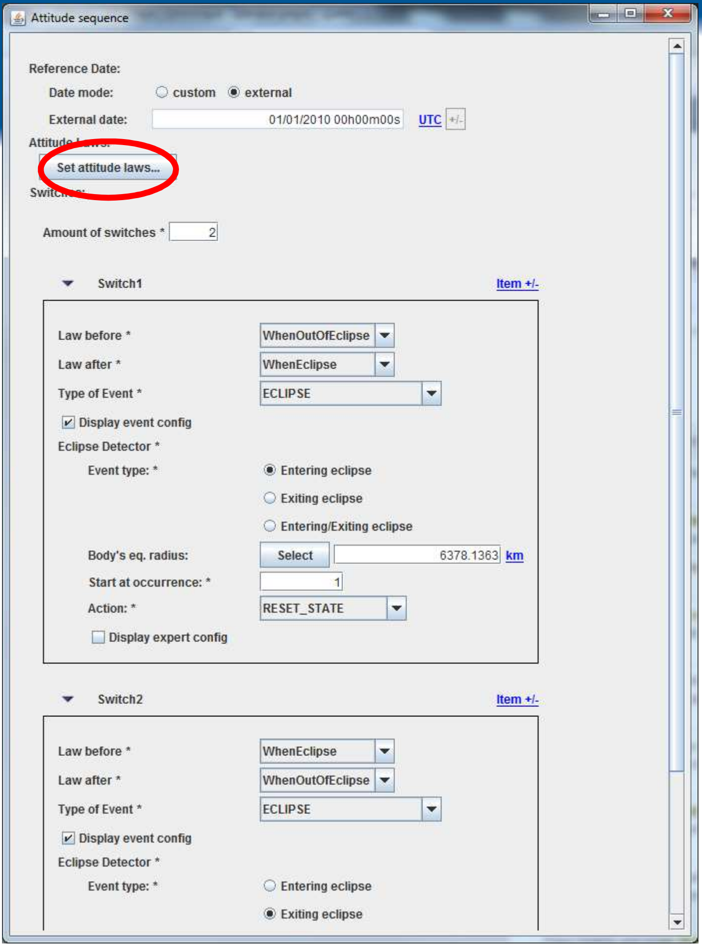
Tank1

Fuel Tank

Name: Fuel Tank1

Propellant mass: * 100.0 kg

GENOPUS (GPAttitudeSequence)



Attitude sequence

Reference Date:

Date mode: ○ custom ● external

External date: 01/01/2010 00h00m00s UTC +/-

Attitude laws:

Switches:

Amount of switches * 2

Switch1 Item +/-

Law before * WhenOutOfEclipse

Law after * WhenEclipse

Type of Event * ECLIPSE

☑ Display event config

Eclipse Detector *

Event type: * ● Entering eclipse ○ Exiting eclipse ○ Entering/Exiting eclipse

Body's eq. radius: Select 6378.1363 km

Start at occurrence: * 1

Action: * RESET_STATE

□ Display expert config

Switch2 Item +/-

Law before * WhenEclipse

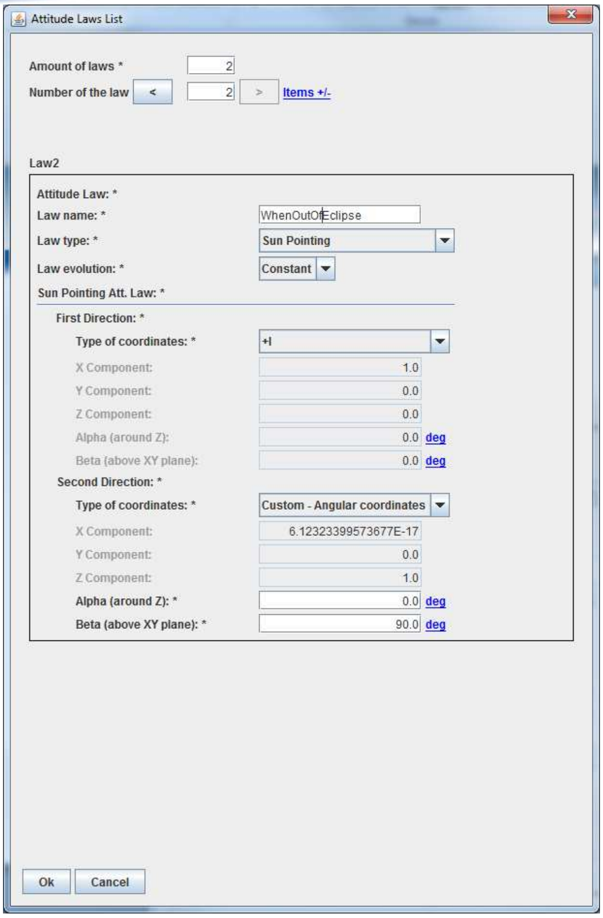
Law after * WhenOutOfEclipse

Type of Event * ECLIPSE

☑ Display event config

Eclipse Detector *

Event type: * ○ Entering eclipse ● Exiting eclipse



Attitude Laws List

Amount of laws * 2

Number of the law < 2 > Items +/-

Law2

Attitude Law: *

Law name: * WhenOutOfEclipse

Law type: * Sun Pointing

Law evolution: * Constant

Sun Pointing Att. Law: *

First Direction: *

Type of coordinates: * +I

X Component: 1.0

Y Component: 0.0

Z Component: 0.0

Alpha (around Z): 0.0 deg

Beta (above XY plane): 0.0 deg

Second Direction: *

Type of coordinates: * Custom - Angular coordinates

X Component: 6.12323399573677E-17

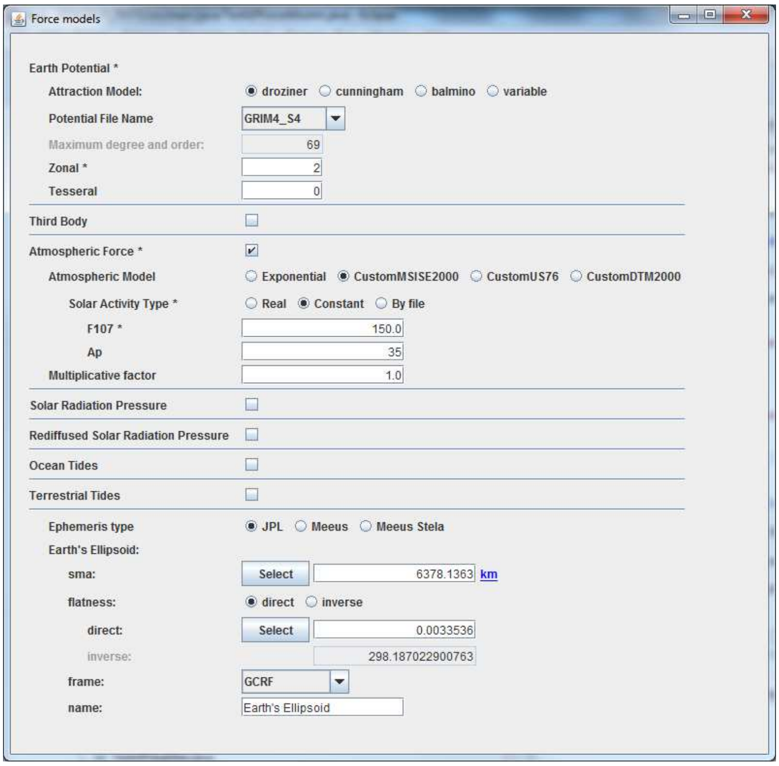
Y Component: 0.0

Z Component: 1.0

Alpha (around Z): * 0.0 deg

Beta (above XY plane): * 90.0 deg

GENOPUS (GPForceModels and GPVehicle)



Force models

Earth Potential *

Attraction Model: ● droziner ○ cunningham ○ balmino ○ variable

Potential File Name: GRIM4_S4

Maximum degree and order: 69

Zonal *: 2

Tesseral: 0

Third Body □

Atmospheric Force * ☑

Atmospheric Model: ○ Exponential ● CustomMSISE2000 ○ CustomUS76 ○ CustomDTM2000

Solar Activity Type *: ○ Real ● Constant ○ By file

F107 *: 150.0

Ap: 35

Multiplicative factor: 1.0

Solar Radiation Pressure □

Rediffused Solar Radiation Pressure □

Ocean Tides □

Terrestrial Tides □

Ephemeris type: ● JPL ○ Meeus ○ Meeus Stela

Earth's Ellipsoid:

sma: Select 6378.1363 km

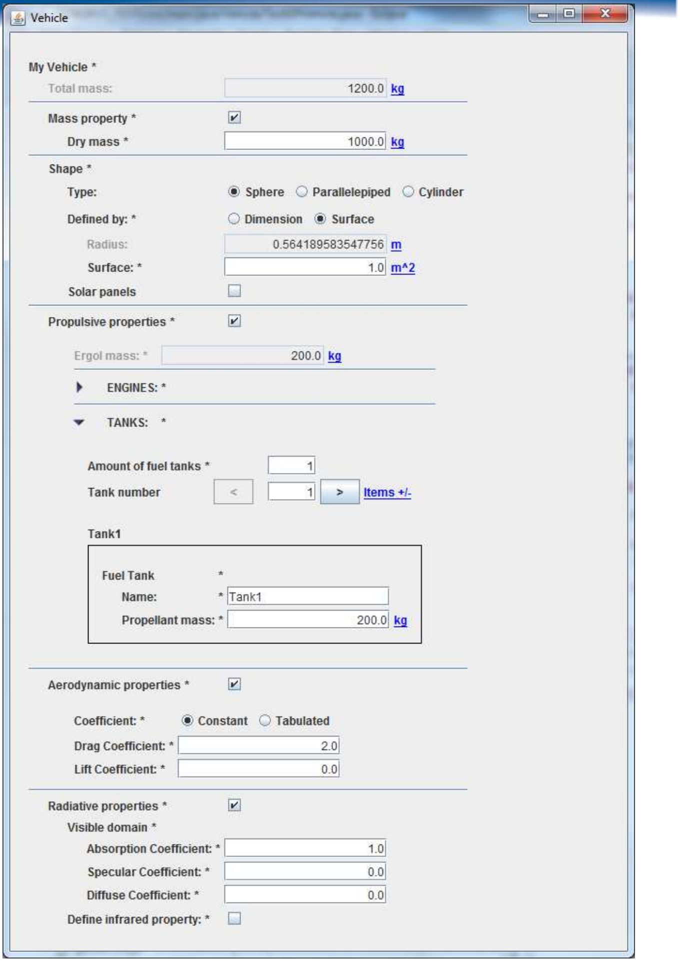
flatness: ● direct ○ inverse

direct: Select 0.0033536

inverse: 298.187022900763

frame: GCRF

name: Earth's Ellipsoid



Vehicle

My Vehicle *

Total mass: 1200.0 kg

Mass property * ☑

Dry mass *: 1000.0 kg

Shape *

Type: ● Sphere ○ Parallelepiped ○ Cylinder

Defined by: * ○ Dimension ● Surface

Radius: 0.564189583547756 m

Surface: * 1.0 m^2

Solar panels □

Propulsive properties * ☑

Ergol mass: * 200.0 kg

ENGINES: *

TANKS: *

Amount of fuel tanks *: 1

Tank number: < 1 > Items +/-

Tank1

Fuel Tank *

Name: * Tank1

Propellant mass: * 200.0 kg

Aerodynamic properties * ☑

Coefficient: * ● Constant ○ Tabulated

Drag Coefficient: * 2.0

Lift Coefficient: * 0.0

Radiative properties * ☑

Visible domain *

Absorption Coefficient: * 1.0

Specular Coefficient: * 0.0

Diffuse Coefficient: * 0.0

Define infrared property: * □

CENTRE NATIONAL D'ÉTUDES SPATIALES

GENOPUS

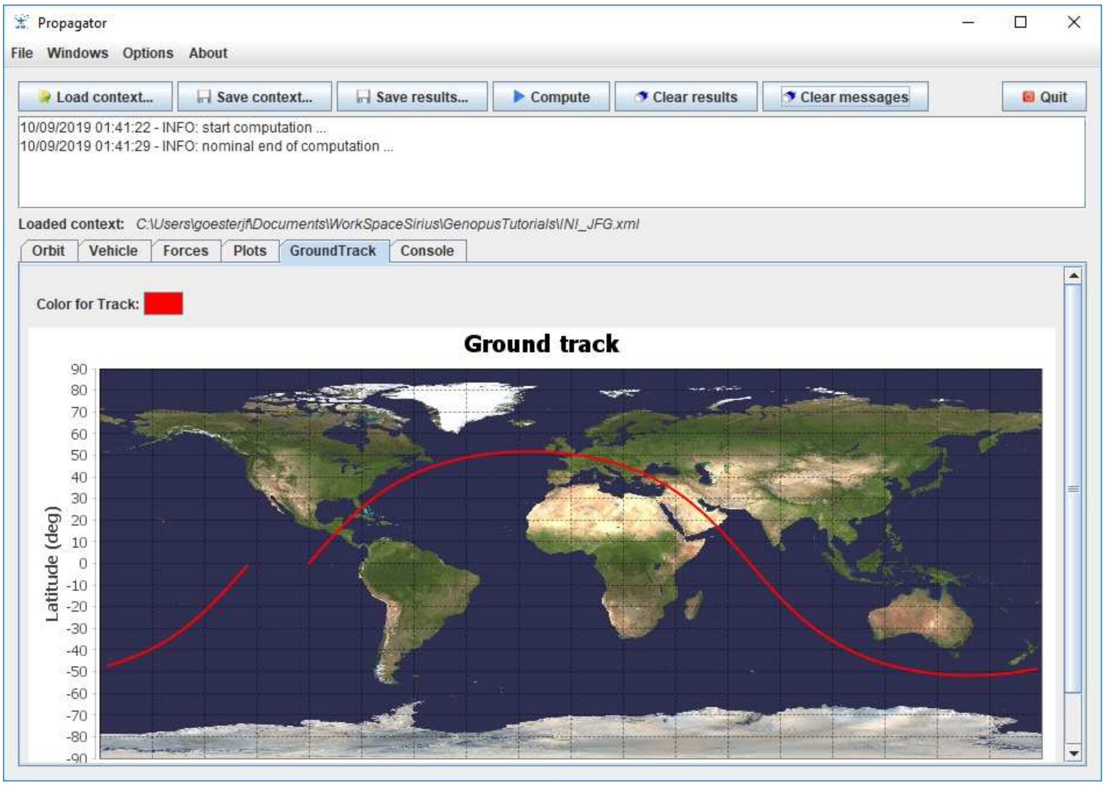
Exercise #2 : build a propagator

GENOPUS Exercise #2 (1/2)

- **The goal of this exercise is to develop an application with its *GUI* allowing to propagate an orbit using PATRIUS and considering as inputs:**
  - **Initial orbital parameters**
  - **Vehicle characteristics (only dry mass and simple aerodynamic characteristics)**
  - **Choice of force models:**
    - Only Balmino for potential
    - Atmospheric models
- **Data will be stored in INI_*suffix*.xml files (or INIT.xml by default)**
- **Results will be stored in a EPH_*suffix*.txt file (EPHEM.txt by default)**
- **Results will be displayed on the *GUI* console but also on plots and ground tracks.**
- **To do it, we will use GENIUS classes to build the main frame *(http://genius.cnes.fr/index.php/How_to_build_a_standard_application)***

WidP...aPanel.java
BatchProp.java
WidProp.java

GENOPUS Exercise #2 (2/2)



```
10/09/2019 01:41:22 - INFO: start computation ...
10/09/2019 01:41:29 - INFO: nominal end of computation ...
```

Loaded context: *C:\Users\goesterjf\Documents\WorkSpaceSirius\GenopusTutorials\INI_JFG.xml*

GENOPUS Exercise #2 / first step (1/5)

1. **Create an new Maven project using GENOPUS V2.1.1 as main dependency**
2. **Create a WidPropDataPanel class extending GDataPanelAbstract:**

- **Add, as a first step, only GPOrbit**

  ⇨ *Be careful, GPOrbit extends from GContainer and not from GPanel, so create an intermediate WidOrbit class extending GPanel*

- **Return the corresponding PATRIUS Orbit object**
- **Add a console tab (*addConsoleTab()* method)**

GENOPUS Exercise #2 / first step (2/5)

3. **Create a BatchProp class:**

- Constructor :
  - Two input parameters:
    - the name of the input data file
    - the name of the output data file
  - Initialize **PatriusDataset** (static method *addResourcesFromPatriusDataset()*)
  - Read XML file and initialize **PATRIUS Orbit** object
- In a public method (**compute()**), propagate the orbit using the *shiftedBy()* method :
  - Propagation time = about one orbital period
  - Output time step = 60s
- Write on the console: final and initial information (for example date and semi-major axis)
- Write in a EPHEM.txt file: date, semi-major axis and mean longitude for each step
- Create a main method:
  - calling this constructor with these two arguments or, by default “INIT.xml” and “EPHEM.txt”
  - calling the **compute()** method

GENOPUS Exercise #2 / first step (3/5)

4. **Create a WidProp class extending GMainFrameAbstract<WidPropagatorDataPanel>:**
   - Initialize the super constructor
   - Add a main method to display the *GUI*

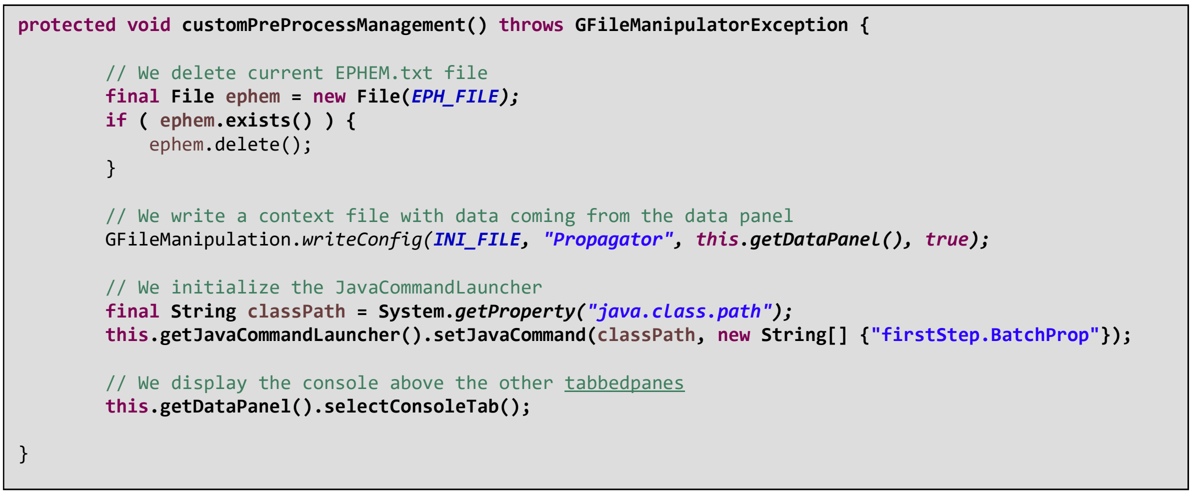
5. **Run the *GUI*:**
   - Initialize an orbit
   - Save the context file (for example in INI_FirstStep.xml) and reload it to verify it

6. **Run the batch mode with this context file and EPHEM.txt as arguments:**
   - Verify the console output and the EPHEM.txt content

7. **Fill the *customPreProcessManagement()* method in the WidProp class (see next slide):**
   - Delete the EPHEM.txt file if it exists
   - Create the INIT.xml file
   - Use the *setJavaCommand()* method with **Batchprop** class
   - Switch on the console tab

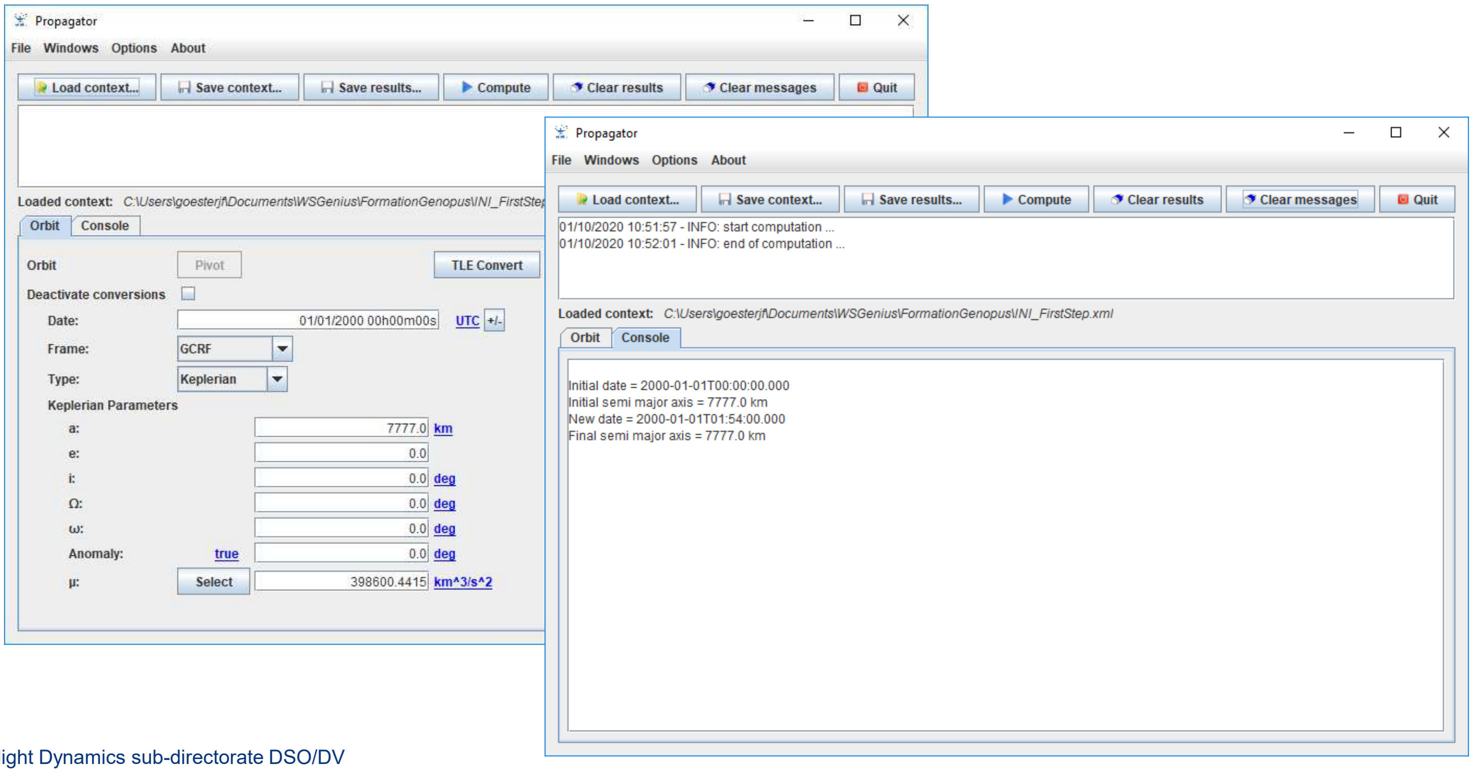
7. **Run the *GUI* again, load the previous context and execute the computation directly.**

GENOPUS Exercise #2 / second step (3/4)



```
protected void customPreProcessManagement() throws GFileManipulatorException {

        // We delete current EPHEM.txt file
        final File ephem = new File(EPH_FILE);
        if ( ephem.exists() ) {
            ephem.delete();
        }

        // We write a context file with data coming from the data panel
        GFileManipulation.writeConfig(INI_FILE, "Propagator", this.getDataPanel(), true);

        // We initialize the JavaCommandLauncher
        final String classPath = System.getProperty("java.class.path");
        this.getJavaCommandLauncher().setJavaCommand(classPath, new String[] {"firstStep.BatchProp"});

        // We display the console above the other tabbedpanes
        this.getDataPanel().selectConsoleTab();

}
```

GENOPUS Exercise #2 / first step (4/4)



Loaded context: C:\Users\goesterjf\Documents\WSGenius\FormationGenopus\INI_FirstStep

Orbit

Deactivate conversions □

Date: 01/01/2000 00h00m00s UTC +/-

Frame: GCRF

Type: Keplerian

Keplerian Parameters

a:	7777.0	km
e:	0.0	
i:	0.0	deg
Ω:	0.0	deg
ω:	0.0	deg
Anomaly: true	0.0	deg
μ:	398600.4415	km^3/s^2

```
01/10/2020 10:51:57 - INFO: start computation ...
01/10/2020 10:52:01 - INFO: end of computation ...
```

Loaded context: C:\Users\goesterjf\Documents\WSGenius\FormationGenopus\INI_FirstStep.xml

```
Initial date = 2000-01-01T00:00:00.000
Initial semi major axis = 7777.0 km
New date = 2000-01-01T01:54:00.000
Final semi major axis = 7777.0 km
```

GENOPUS Exercise #2 / second step (1/3)

1. Add GPVehicle and GPForceModels tabs in WidPropData class:

```
// Creating a vehicle widget (only with dry mass and simple aerodynamic properties)
widVehicle =  new GPVehicle("Vehicle characteristics", true, false, true, false, false);

// Creating a force model widget (only with potential [Balmino] and atmosphere)
AttractionModelsEnum[] attractionModelsAvailable = { AttractionModelsEnum.BALMINO };
widForces =  new GPForceModels("Models", AttractionModelsEnum.BALMINO, attractionModelsAvailable,
                               false, true, false, false, false, false);
```

2. Add also getter for PATRIUS Vehicle and ForceModelsData objects

   ⇨*Be careful to create an Assembly from vehicle before creating force PATRIUS object*

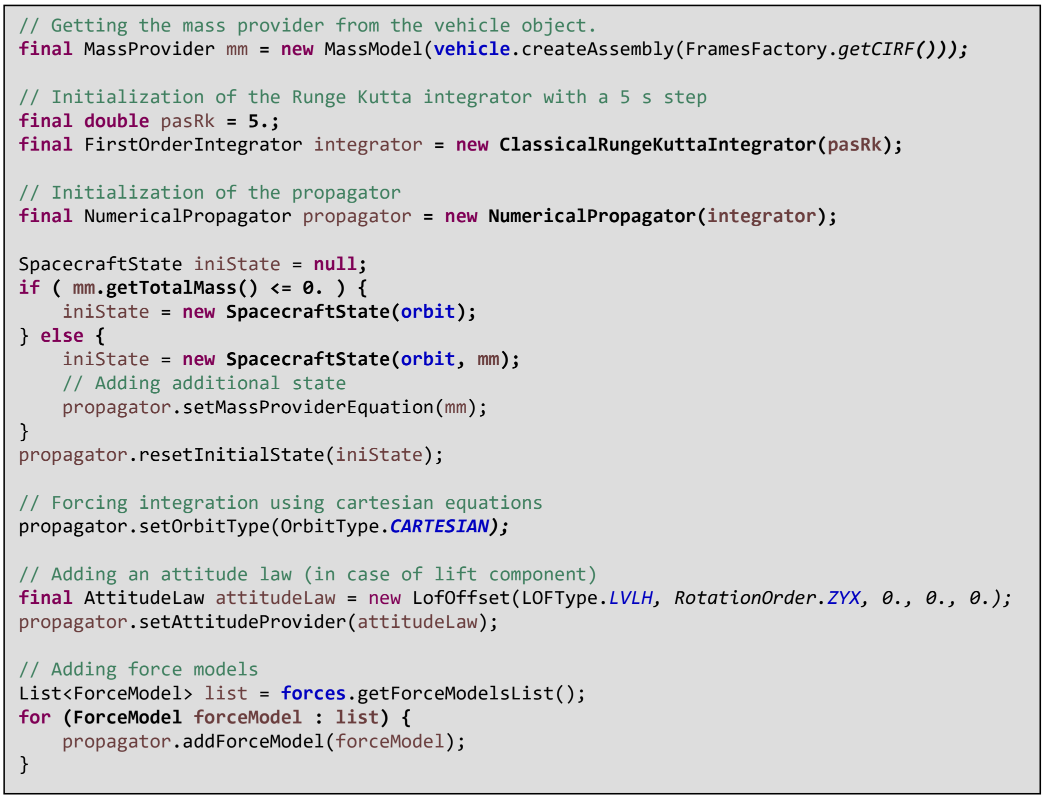
```
final Assembly assembly = getVehicle().createAssembly(FramesFactory.getCIRF());
return widForces.getPatriusObject(assembly);
```

3. Initialize these objects in the constructor of the BatchProp class

4. Build a PATRIUS propagator then propagate the trajectory (see next slides)

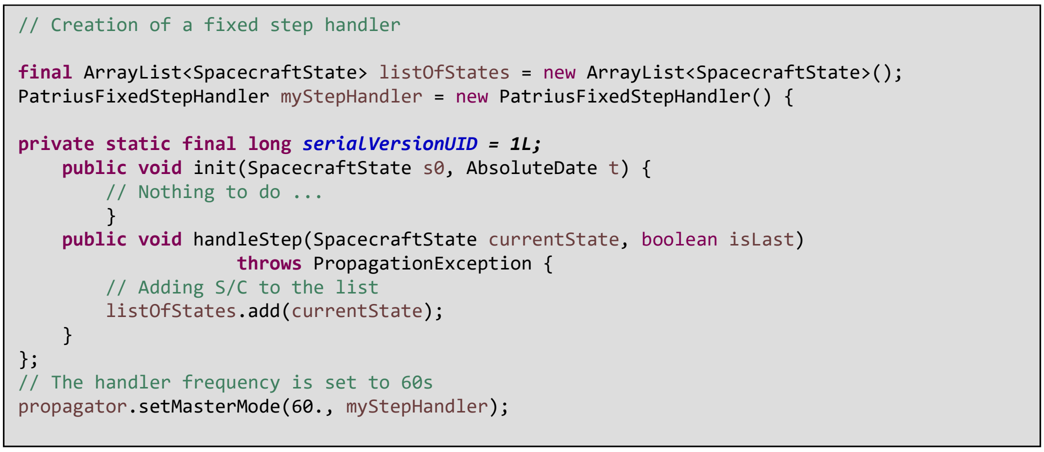
GENOPUS Exercise #2 / second step (2/3)

Patrius propagator initialization (1/2)



```java
// Getting the mass provider from the vehicle object.
final MassProvider mm = new MassModel(vehicle.createAssembly(FramesFactory.getCIRF()));

// Initialization of the Runge Kutta integrator with a 5 s step
final double pasRk = 5.;
final FirstOrderIntegrator integrator = new ClassicalRungeKuttaIntegrator(pasRk);

// Initialization of the propagator
final NumericalPropagator propagator = new NumericalPropagator(integrator);

SpacecraftState iniState = null;
if ( mm.getTotalMass() <= 0. ) {
    iniState = new SpacecraftState(orbit);
} else {
    iniState = new SpacecraftState(orbit, mm);
    // Adding additional state
    propagator.setMassProviderEquation(mm);
}
propagator.resetInitialState(iniState);

// Forcing integration using cartesian equations
propagator.setOrbitType(OrbitType.CARTESIAN);

// Adding an attitude law (in case of lift component)
final AttitudeLaw attitudeLaw = new LofOffset(LOFType.LVLH, RotationOrder.ZYX, 0., 0., 0.);
propagator.setAttitudeProvider(attitudeLaw);

// Adding force models
List<ForceModel> list = forces.getForceModelsList();
for (ForceModel forceModel : list) {
    propagator.addForceModel(forceModel);
}
```

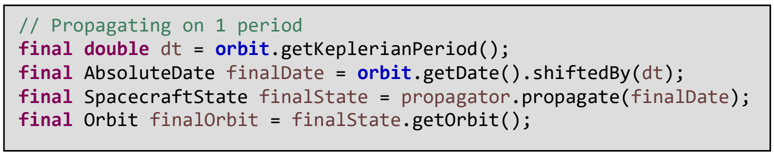
GENOPUS Exercise #2 / second step (3/3)



```java
// Creation of a fixed step handler

final ArrayList<SpacecraftState> listOfStates = new ArrayList<SpacecraftState>();
PatriusFixedStepHandler myStepHandler = new PatriusFixedStepHandler() {

private static final long serialVersionUID = 1L;
    public void init(SpacecraftState s0, AbsoluteDate t) {
        // Nothing to do ...
        }
    public void handleStep(SpacecraftState currentState, boolean isLast)
                throws PropagationException {
        // Adding S/C to the list
        listOfStates.add(currentState);
    }
};
// The handler frequency is set to 60s
propagator.setMasterMode(60., myStepHandler);
```

Patrius propagator initialization (2/2)

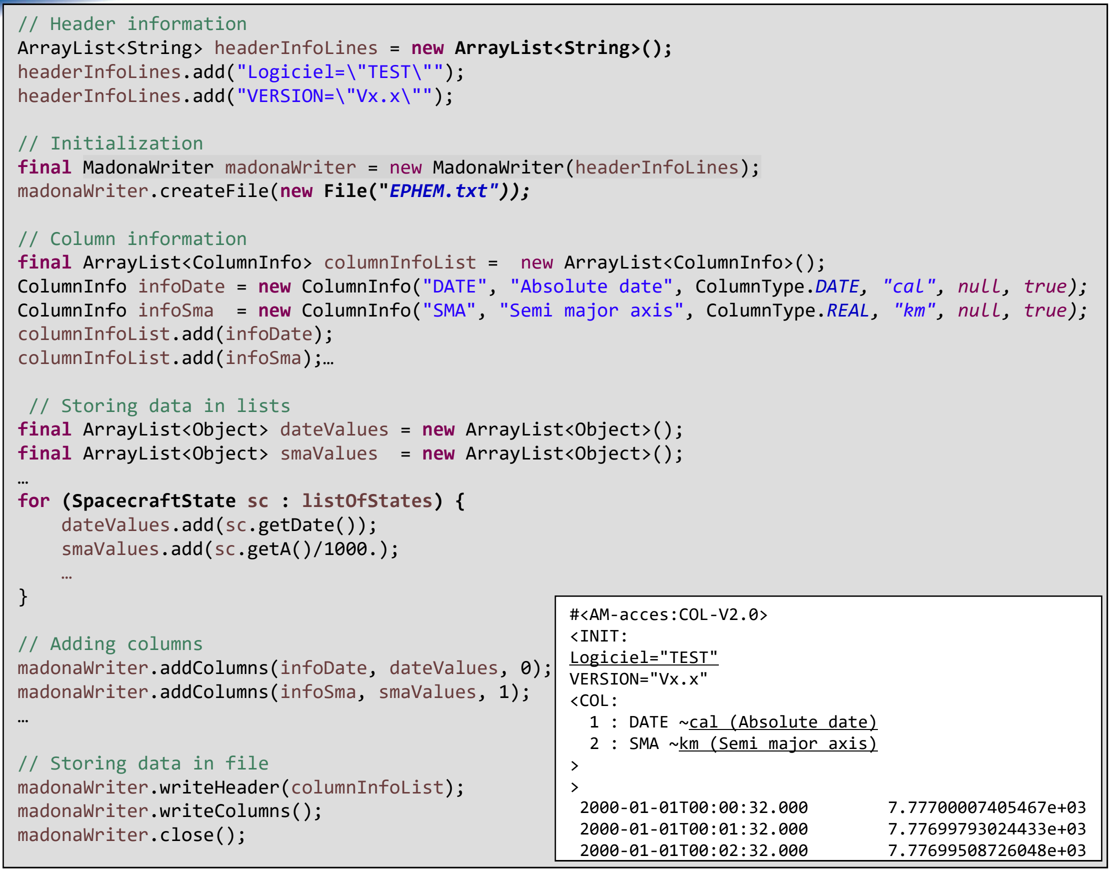
Propagation …



```java
// Propagating on 1 period
final double dt = orbit.getKeplerianPeriod();
final AbsoluteDate finalDate = orbit.getDate().shiftedBy(dt);
final SpacecraftState finalState = propagator.propagate(finalDate);
final Orbit finalOrbit = finalState.getOrbit();
```

GENOPUS Exercise #2 / third step

- Write the EPHEM.txt file using MADONA/XML methods, including date and semi-major axis

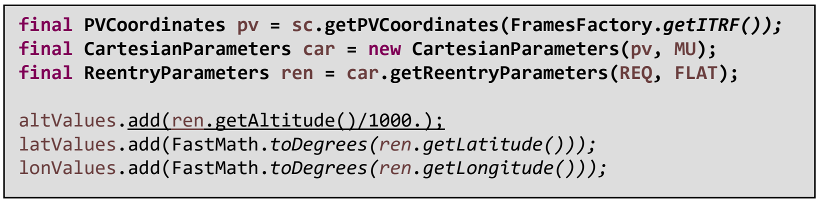


```
// Header information
ArrayList<String> headerInfoLines = new ArrayList<String>();
headerInfoLines.add("Logiciel=\"TEST\"");
headerInfoLines.add("VERSION=\"Vx.x\"");

// Initialization
final MadonaWriter madonaWriter = new MadonaWriter(headerInfoLines);
madonaWriter.createFile(new File("EPHEM.txt"));

// Column information
final ArrayList<ColumnInfo> columnInfoList =  new ArrayList<ColumnInfo>();
ColumnInfo infoDate = new ColumnInfo("DATE", "Absolute date", ColumnType.DATE, "cal", null, true);
ColumnInfo infoSma  = new ColumnInfo("SMA", "Semi major axis", ColumnType.REAL, "km", null, true);
columnInfoList.add(infoDate);
columnInfoList.add(infoSma);…

 // Storing data in lists
final ArrayList<Object> dateValues = new ArrayList<Object>();
final ArrayList<Object> smaValues  = new ArrayList<Object>();
…
for (SpacecraftState sc : listOfStates) {
    dateValues.add(sc.getDate());
    smaValues.add(sc.getA()/1000.);
    …
}

// Adding columns
madonaWriter.addColumns(infoDate, dateValues, 0);
madonaWriter.addColumns(infoSma, smaValues, 1);
…

// Storing data in file
madonaWriter.writeHeader(columnInfoList);
madonaWriter.writeColumns();
madonaWriter.close();
```

```
#<AM-acces:COL-V2.0>
<INIT:
Logiciel="TEST"
VERSION="Vx.x"
<COL:
  1 : DATE ~cal (Absolute date)
  2 : SMA ~km (Semi major axis)
>
>
 2000-01-01T00:00:32.000        7.77700007405467e+03
 2000-01-01T00:01:32.000        7.77699793024433e+03
 2000-01-01T00:02:32.000        7.77699508726048e+03
```

GENOPUS Exercise #2 / fourth step (1/2)

1. In **BatchProp** class, add altitude, latitude and longitude in the EPHEM.txt file



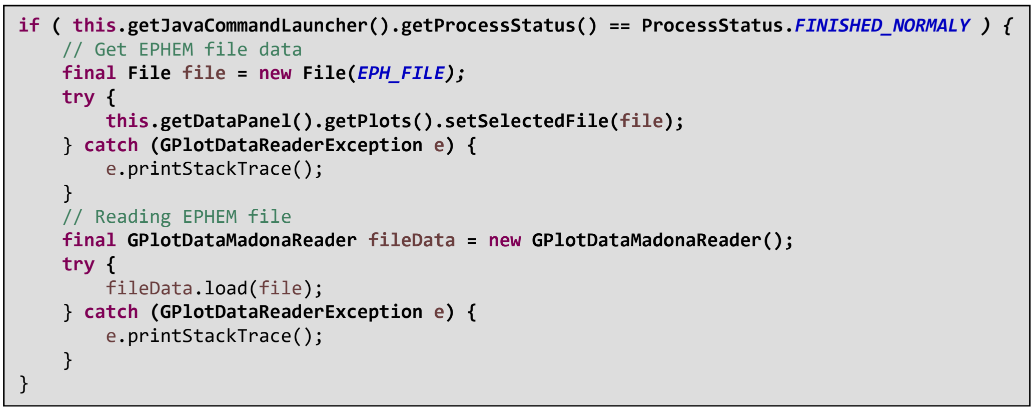
```
final PVCoordinates pv = sc.getPVCoordinates(FramesFactory.getITRF());
final CartesianParameters car = new CartesianParameters(pv, MU);
final ReentryParameters ren = car.getReentryParameters(REQ, FLAT);

altValues.add(ren.getAltitude()/1000.);
latValues.add(FastMath.toDegrees(ren.getLatitude()));
lonValues.add(FastMath.toDegrees(ren.getLongitude()));
```

2. Add **GPlotPanel** and **GGroundPlotpanel** tabs in **WidPropData** class

⇨ Do not forget to create getters …

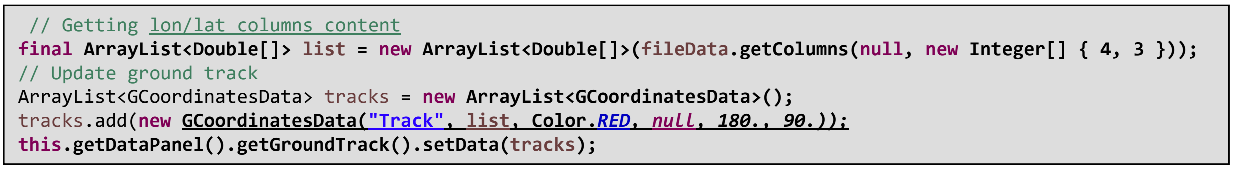
GENOPUS Exercise #2 / fourth step (2/2)

3. In **WidProp** class, fill the ***customPostProcessManagement()*** method to plot with **GPlotPanel**:



```java
if ( this.getJavaCommandLauncher().getProcessStatus() == ProcessStatus.FINISHED_NORMALY ) {
    // Get EPHEM file data
    final File file = new File(EPH_FILE);
    try {
        this.getDataPanel().getPlots().setSelectedFile(file);
    } catch (GPlotDataReaderException e) {
        e.printStackTrace();
    }
    // Reading EPHEM file
    final GPlotDataMadonaReader fileData = new GPlotDataMadonaReader();
    try {
        fileData.load(file);
    } catch (GPlotDataReaderException e) {
        e.printStackTrace();
    }
}
```

4. Then, get lat/lon data and refresh the ground track adding these lines in the ***customPostProcessManagement()*** method :



```java
 // Getting lon/lat columns content
final ArrayList<Double[]> list = new ArrayList<Double[]>(fileData.getColumns(null, new Integer[] { 4, 3 }));
// Update ground track
ArrayList<GCoordinatesData> tracks = new ArrayList<GCoordinatesData>();
tracks.add(new GCoordinatesData("Track", list, Color.RED, null, 180., 90.));
this.getDataPanel().getGroundTrack().setData(tracks);
```

GENOPUS Exercise #2 / fifth step

- In order to have only one main method :
  - Create a **Main** class with a main method
    - If no arguments execute the *GUI* mode
    - If arguments, execute the batch mode
  - Remove the main classes of the **WidProp** and the **BatchProp** class
- Export as an executable jar, considering this main method as the entry point : you will obtain your autonomous and potable propagator !

⇨ ***Possible also using Maven instructions…***