**Usage Guide**

**Create a simple geometry object:**

If the user only requires geometry objects, wich do not need to hold any further information, the three functions createSimplePoint(), createSimplePath(), and createSimplePolygon() are usable. These functions reduce the effort that is needed to create geometries because they only need the corresponding object and are only embedded in a simple placemark object.

Example:

EasyKMLCreator easyKmlCreator = new EasyKMLCreator();

Point point = **new** Point(**new** Location(-43.61725568991, 170.1426332867, 0.0));

easyKmlCreator.createSimplePoint(point);

**Create a geometry object with additional information:**

To create objects which should be equipped with additional information, they must be embedded in an instance of placemark. As placemarks are child classes within the features they offer a wide range of attributes that may be changed to enhance the geometry objects. To finally create these objects the function createPlacemark() is provided, which requires the previously created placemark instance.

Example:

EasyKMLCreator easyKmlCreator = new EasyKMLCreator();

Point point = **new** Point(**new** Location(-43.61725568991, 170.1426332867, 0.0));

Placemark placemark = **new** Placemark("Name", "Description", point);

placemark.setId("Point 1");

easyKmlCreator.createPlacemark(placemark);

**Create Styles:**

If customisation of the KML objects created is necessary the API provides the class Style, which contains four styling classes for different purposes. The user can decide which of these styling classes they want to create and store in the Style instance. The façade class provides the function createStyle(), which expects the Style object and is responsible for the creation of the XML code.

Example:

EasyKMLCreator easyKmlCreator = new EasyKMLCreator();

Style style = **new** Style();

LineStyle lineStyle = **new** LineStyle();

lineStyle.setColor(Color.***CYAN***);

style.setLineStyle(lineStyle);

easyKmlCreator.createStyle(style);

**Create a Tour:**

Touring is essentially created using the objects Tour, Playlist and TourPrimitive. In detail, the tour is a child class from the features. Furthermore, the Tour keeps a Playlist which is filled with several TourPrimitive objects that finally define the process of the Tour. After the playlist is filled with objects the Tour objects can be transferred to the façade function createTour() to create the corresponding XML code.

Example:

EasyKMLCreator easyKmlCreator = **new** EasyKMLCreator();

LookAt lookAt = **new** LookAt();

lookAt.setCoordinates(**new** Location(-39.279, 174.007, 0.0));

FlyTo flyTo = **new** FlyTo(2.0, lookAt);

Wait wait = **new** Wait(3.0);

Playlist playlist = **new** Playlist();

playlist.addPrimitiveAction(flyTo);

playlist.addPrimitiveAction(wait);

Tour tour = **new** Tour(playlist);

easyKmlCreator.createTour(tour);