Creating custom maps with Nutiteq's Map SDK

With following code you can easily add interactive, "slippy" mapping functions to your mobile application.

You will get more interactive results, can use client-side caching (even stored map data) without worrying about license restrictions, as compared other API-s (like Google static maps API).

Download and install Nutiteq's Map SDK

1. Download package for Java ME from Nutiteq Mobile Map API downloads. For Nokia platforms (Series 40 or S60) use maps-lib-j2me-1_0_2.zip . It includes maps_lib-1.0.0.jar as actual library, and also javadocs, and a set of sample apps. Last version number may be larger by now.
2. Include maps_lib.jar file to your project, add it to the Build Path and be sure also to Export the library.

Get your license key

Create a license key for you http://www.nutiteq.com/generate_key.html page (requires registration). Evaluation license key type is ok. Remember to enter same application and vendor name as in JAD file.

Quick overview for advanced developers

To use the library you have to create com.nutiteq.mapComponent() object and use it's methods.

1. Create class which implements MapListener
```
public class MapScreen extends Canvas implements CommandListener, MapListener {
```
2. Create map object, specify initial view location and size of map.
```
map = new MapComponent("MY_LICENSE_KEY", Mapper.instance, getWidth(), getHeight(), new WgsPoint(24.764580, 59.437420), 10);
```
3. MapListener implementation and defining is needed to get map update events from UI
```
map.setMapListener(this);
```
4. Now start mapping actions (threads)
```
map.startMapping();
```
5. Implement map painting to canvas
```
    protected void paint(Graphics g) {
        com.nutiteq.wrappers.Graphics graphicsWrapper = new com.nutiteq.wrappers.Graphics(g);
        map.paint(graphicsWrapper);
    }
```
6. Implement MapListener methods
```
    public void mapClicked(WgsPoint pnt) {
    }

    public void mapMoved() {
    }

    public void needRepaint(boolean downloadReady) {
        // This repaint call is mandatory, otherwise map is not refreshed!
        repaint();
    }
```
7. Add UI controls (keypresses, pointer) to control map, menu commands and other stuff. See full source below.

Full basic application sources

Main MIDlet class (Mapper.java)

The only mapping-specific line here is "Log.enableAll();", which turns on logging in mapping library, it makes troubleshooting easier.

```
import javax.microedition.lcdui.Display;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

import com.nutiteq.log.Log;

public class Mapper extends MIDlet {

    public static Mapper instance;
```


(C) Copyright Nokia 2013. All rights reserved.

```
    private MapScreen screen;

    public Mapper() {
        instance = this;
    }

    protected void destroyApp(boolean arg0) throws MIDletStateChangeException {
        screen = null;
        instance.notifyDestroyed();
        instance = null;
    }

    protected void pauseApp() {

    }

    protected void startApp() throws MIDletStateChangeException {
        Log.enableAll();
        if (screen == null) {
          screen = new MapScreen();
        }
        Display.getDisplay(this).setCurrent(screen);
        screen.repaint();
    }
}
```

MapScreen.java class

```
package mapper;

import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Graphics;
import javax.microedition.midlet.MIDletStateChangeException;

import com.nutiteq.MapComponent;
import com.nutiteq.components.WgsPoint;
import com.nutiteq.controls.MGMapsKeysHandler;
import com.nutiteq.listeners.MapListener;

public class MapScreen extends Canvas implements CommandListener, MapListener {
    public static MapScreen instance;
    private final MapComponent map;
    private final Command exit = new Command("Exit", Command.EXIT, 0);

    public MapScreen() {
        instance = this;

        // 1. initialize map object, also set initial location and zoom
        // Note that default map source will be OpenStreetMap (Mapnik tiles)
        map = new MapComponent("MY_LICENSE_KEY", Mapper.instance, getWidth(),
                getHeight(), new WgsPoint(24.764580, 59.437420), 10);

        // 2. define keycode handler for keypresses.
        // We use a default set (arrows and 2468 for moving, # for zoom in, *
        // for zoom out)
        // you can define own keys instead of this
        MGMapsKeysHandler keyHandler = new MGMapsKeysHandler();
        map.setControlKeysHandler(keyHandler);

        // 3. set this class to listen map events: needRepaint, mapMoved and
        // mapClicked
        map.setMapListener(this);

        // 4. show default location pointer
        map.showDefaultCursor();

        // 5. start mapping threads and tasks in the library
        map.startMapping();

        // add Exit command to the midlet menu
        addCommand(exit);
        setCommandListener(this);
    }

    protected void paint(Graphics g) {
        // 6. define to repaint of map
        com.nutiteq.wrappers.Graphics graphicsWrapper = new com.nutiteq.wrappers.Graphics(g);
        map.paint(graphicsWrapper);
    }

    // 7. Forward keypressing events to library using following methods
    protected void keyPressed(final int keyCode) {
        map.keyPressed(keyCode);
    }

    protected void keyReleased(final int keyCode) {
        map.keyReleased(keyCode);
    }

    protected void keyRepeated(final int keyCode) {
        map.keyRepeated(keyCode);
    }

    // Basic handler for Midlet menu action
    public void commandAction(Command c, Displayable d) {

        if (c == exit) {
   // 8. stop mapping properly before terminating Midlet.
   // NB! do not forget it, otherwise RMS cache index is not updated
            map.stopMapping();
            try {
                Mapper.instance.destroyApp(true);
            } catch (final MIDletStateChangeException ignore) {
            }
        }
    }

    // 9. Handlers for MapListener
    public void mapClicked(WgsPoint pnt) {
    }

    public void mapMoved() {
    }
```


(C) Copyright Nokia 2013. All rights reserved.

```
    public void needRepaint(boolean downloadReady) {
        // 10. This repaint call is mandatory, otherwise map is not refreshed!
        repaint();
    }

}
```

Extra features from the library

1. **Use Navteq MapTP map server** as source

```
map.setMap(new MapTPMap("YOUR MAPTP KEY"));
```

2. **Add a dynamic KML data source**, say Panoramio images

```
map.addKmlService(new KmlUrlReader(
  "http://www.panoramio.com/panoramio.kml?LANG=en_US.utf8&", true));
```

3. **Add individual placemarks** on top of map: points, lines, polygons

```
 // 1. first load some image for point display. It could be also custom painted
 try {
   icon = Image.createImage("/x.png");
  } catch (IOException e) {}
 // 2. add it as a point
 map.addPlace(new Place(1, "Tallinn", icon, 24.764580, 59.437420));

 // 3. add a line
 WgsPoint[] linePoints = {
  new WgsPoint(24.76382468302337, 59.44325151314919),
  new WgsPoint(24.76344295658494, 59.4462352840583),
  new WgsPoint(24.76593650384734, 59.44530921763007),
  new WgsPoint(24.76804665483925, 59.44616268729941),
  new WgsPoint(24.76810500478219, 59.443291656657) };

  final PlaceLabel lineLabel = new PlaceLabel("Label for a line");
  Line line = new Line(linePoints, new LineStyle(0xFF00FF00, 1), lineLabel);
  map.addLine(line);

 // 4. make a polygon from same linePoints. Use default style here
  map.addPolygon(new Polygon(linePoints));
```

4. **Add some visual overlays**: animated zoom indicator and download indicator:

```
map.setZoomLevelIndicator(new DefaultZoomIndicator(0, 1));
map.showZoomLevelIndicator(true);

map.enableDownloadCounter();
map.enableDownloadDisplay();
```

5. Take a look to Library developer resources to see how to **many use other features** like:

- GPS and Cell-ID positioning
- Geocoding
- Routing
- Map overlays (hybrid maps)
- Map streaming for much faster map download
- Offline mapping: preloaded maps to application package or flash drive
- many more

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.