How to implement a compass in Java ME

Overview

There are two methods for displaying the device's orientation in a compass-like style.

The Location API (JSR-179) has been implemented since S60 3rd Edition and Series 40 6th Edition. It provides classes not only for getting satellite coordinates but also for getting the device's current course and speed by using the device's GPS module.

The Orientation class on the other hand, is part of the Location API, and provides methods for accessing the terminal's physical orientation by using the device's built in compass.

Getting GPS Coordinates with a Location Provider

Here is a simple way of getting coordinates:

```
try {
 LocationProvider lp = LocationProvider.getInstance(null);
 if (lp != null) {
  Location l = lp.getLocation(300);
  if (l.isValid()) {
   Coordinates c = l.getQualifiedCoordinates();
   if (c!=null) {
    double latitude = c.getLatitude();
    double longitude = c.getLongitude();
    // do something with the coordinates
   }
  }
  else {
   System.out.println("Location is not valid!");
  }
 }
 else {
  System.out.println("LocationProvider = null!");
 }
} catch (LocationException le) { // not able to retrieve location information
 System.out.println("LocationException: " + le.getMessage());
} catch (InterruptedException ie) {
 System.out.println("InterruptedException: " + ie.getMessage());
}
```

In addition to coordinates the Location API makes it possible to get the following:

- altitude
- speed and course
- additional textual address information about a location by using AddressInfo class (Note: The support for this depends on the location method and additional information is not necessarily available for all the locations)

For getting speed and course, proper Criteria must be defined for the Location Provider. By default speed and course data is not provided. For getting them a Criteria instance with the following setting should be created:

```
Criteria criteria = new Criteria();
criteria.setSpeedAndCourseRequired(true);
LocationProvider lp = LocationProvider.getInstance(criteria);
```

Getting the terminal's course without using the Orientation class

The Orientation class is supported only on certain devices. It is possible though to use the device's GPS receiver to implement a similar functionality. The example MIDlet below shows, how to get course (and speed) from GPS without using the Orientation class and how to draw a compass-like course indicator on the screen. The course retrieved is not necessarily as reliable as a real value got from a compass.

Source code: CompassMIDlet.java

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

public class CompassMIDlet extends MIDlet {
    private LocationCanvas canvas;

    public void startApp() {
        canvas = new LocationCanvas(this);
        Display.getDisplay(this).setCurrent(canvas);
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

}
```


(C) Copyright Nokia 2013. All rights reserved.

Source code: LocationCanvas.java

```
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Font;
import javax.microedition.lcdui.Graphics;
import javax.microedition.location.Criteria;
import javax.microedition.location.Coordinates;
import javax.microedition.location.Location;
import javax.microedition.location.LocationListener;
import javax.microedition.location.LocationProvider;
import javax.microedition.location.LocationException;

public class LocationCanvas extends Canvas implements CommandListener, LocationListener,Runnable {
    private CompassMIDlet midlet;
    private Command exitCommand;
    private String latitudeString = "Latitude: ";
    private String longitudeString = "Longitude: ";
    private String courseString = "Course: ";
    private String speedString = "Speed: ";
    private Font font;
    private float course = 0;
    private float speed = 0;
    Thread t;
    Criteria criteria;
    LocationProvider lp;

    public LocationCanvas(CompassMIDlet midlet) {
        this.midlet = midlet;
        exitCommand = new Command("Exit", Command.EXIT, 1);
        this.addCommand(exitCommand);
        this.setCommandListener(this);
        font = Font.getFont(Font.FACE_SYSTEM, Font.STYLE_PLAIN, Font.SIZE_SMALL);
        t=new Thread(this);
        t.start();
    }

    private void setLocationProvider() throws LocationException
    {
        criteria = new Criteria();
        criteria.setSpeedAndCourseRequired(true);
        criteria.setCostAllowed(true);
        lp = LocationProvider.getInstance(criteria);
        lp.setLocationListener(this, -1, -1, -1);
    }
    public void paint(Graphics g) {
        // Clears screen
        g.setColor(255,255,255);
        g.fillRect(0, 0, getWidth(), getHeight());

        // Draws the strings to the screen
        g.setColor(0, 0, 0);
        g.setFont(font);
        int height = font.getHeight();
        int x = getWidth()/2- (5*height)/2;
        int y = 5*height;
        int cx = x + y/2;
        int cy = y + y/2;
        g.drawString(latitudeString, 0, 0, Graphics.TOP|Graphics.LEFT);
        g.drawString(longitudeString, 0, height, Graphics.TOP|Graphics.LEFT);
        g.drawString(courseString, 0, 2*height, Graphics.TOP|Graphics.LEFT);
        g.drawString(speedString, 0, 3*height, Graphics.TOP|Graphics.LEFT);

        // Draws the compass circles and the arrow. The arrow points at north,
        // when device is pointing at the direction of traveling.
        g.setColor(255, 0, 0);
        g.drawArc(x, y, y, y, 0, 360);
        g.setColor(0, 0, 255);
        g.drawArc(x-2, y-2, y+4, y+4, 0, 360);
        g.setColor(0, 0, 0);
        double rad = Math.toRadians(-course);
        double rad2 = Math.toRadians(-course + 165);
        double rad3 = Math.toRadians(-course - 165);
        g.fillTriangle((int)(cx + (y*Math.sin(rad))/2), (int)(cy - (y*Math.cos(rad))/2),
            (int)(cx + (y*Math.sin(rad2))/2), (int)(cy - (y*Math.cos(rad2))/2),
            (int)(cx + (y*Math.sin(rad3))/2), (int)(cy - (y*Math.cos(rad3))/2));
    }

    protected  void keyPressed(int keyCode) {
    }

    public void commandAction(Command c, Displayable d) {
        if (c == exitCommand) {
            midlet.notifyDestroyed();
        }
    }

    /**
     * Gets the coordinates, speed and course from the GPS.
     */
    private void updateLocation(Location loc) {
        if(loc.isValid())
        {
            if (lp != null) {
                Location l = loc;
                if (l.isValid()) {
                    Coordinates c = l.getQualifiedCoordinates();
                    if (c!=null) {
                        course = l.getCourse(); // moving direction is degrees
                        speed = l.getSpeed() * 3.6f; // m/s converted to km/h
                        double latitude = c.getLatitude();
                        double longitude = c.getLongitude();
                        latitudeString = "Latitude: " + latitude;
                        longitudeString = "Longitude: " + longitude;
                        courseString = "Course: " + course + '°';
                        speedString = "Speed: " + speed;
                        if (speedString.length() > 12) speedString = speedString.substring(0, 12);
                        speedString += "km/h";

                    }
                }
                else {
```


(C) Copyright Nokia 2013. All rights reserved.

```
                System.out.println("Location is not valid!");
                }
            }
            else {
                System.out.println("LocationProvider = null!");
            }
        }
        else
        {
            System.out.println("Location not valid");
        }

    }

    public void locationUpdated(LocationProvider arg0, Location loc) {
        updateLocation(loc);
        repaint();
    }

    public void providerStateChanged(LocationProvider arg0, int arg1) {

    }

    public void run() {
        try {
                setLocationProvider();
            } catch (LocationException e) {
                e.printStackTrace();
            }
    }
}
```

Screenshot

Getting the course, using the device's built in compass

The advantage of using the device's built in compass, over using the above described satellite dependent method, is greater accuracy and also the ability to use the compass functionality indoors. If the Orientation.getOrientation() method doesn't throw a LocationException, that means as minimum, the device supports the getCompassAzimuth() method for retrieving the terminal's horizontal compass azimuth in degrees relative to either magnetic or true north.

Note that when calling the getCompassAzimuth(), a NullPointerException can be thrown even if the device supports the getOrientation() method, which indicates that the device's compass requires calibration. On certain touch screen phones, opening OVI Maps and touching the compass on the bottom right corner, provides on-screen directions on how to move the phone in order to calibrate the compass.

Source code: OrientationTest.java

```
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.StringItem;
import javax.microedition.location.LocationException;
import javax.microedition.location.Orientation;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;


public class OrientationTest extends MIDlet implements CommandListener,Runnable {

    Command exitCommand;
    Orientation orientation;
    Form f;
    Thread t;
    float azimuth;
    StringItem label;

    public OrientationTest() {

    }

    protected void destroyApp(boolean unconditional)
```


(C) Copyright Nokia 2013. All rights reserved.

```
        throws MIDletStateChangeException {

    }

    protected void pauseApp() {

    }

    protected void startApp() throws MIDletStateChangeException {
        //A form with a string item is added to the display as well as an exit button
        exitCommand=new Command("Exit", Command.EXIT, 1);
        f=new Form("");
        f.addCommand(exitCommand);
        f.setCommandListener(this);
        label=new StringItem("","Orientation: "+azimuth);
        f.append(label);
        Display.getDisplay(this).setCurrent(f);
        t=new Thread(this);
        t.start();
    }

    public void commandAction(Command cmd, Displayable arg1) {
        // Exit button code
        if (cmd == exitCommand)
        {
            try {
                destroyApp(false);
            } catch (MIDletStateChangeException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            notifyDestroyed();
        }
    }

    public void run() {
        // Runs a thread every half a second with updated compass information

        while(true)
        {
            try {
                Thread.sleep(500);
            } catch (InterruptedException e1) {
                return;
            }

            try {
                orientation = Orientation.getOrientation();
                azimuth=orientation.getCompassAzimuth();
                label.setText("Orientation: "+azimuth);
                }catch(LocationException le)
                {
                    label.setText("Orientation not supported");
                    return;
                }catch (NullPointerException np)
                {
                    label.setText("Calibration needed");
                }
            }
        }

}
```

See also

- Category:Location API (JSR-179)

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.