



## Contents

- [1 Overview](#)
- [2 Source code: CompassMIDlet.java](#)
- [3 Source code: LocationCanvas.java](#)
- [4 Screenshot](#)
- [5 See also](#)

## Overview

The Location API (JSR-179) has been part of [Java](#) implementations in [S60](#) devices since S60 3rd Edition and in [Series 40](#) devices since Series 40 6th Edition. Basically it provides classes for getting coordinates from GPS and also for accessing landmark store (for viewing and editing landmarks).

Here is a simple way of getting coordinates:

```
try {
    LocationProvider locationProvider = LocationProvider.getInstance(null);
    if (lp != null) {
        = lp.getLocation(300);
    if (l.isValid()) {
        = l.getQualityAndCoordinates();
    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 for example:

- altitude
- speed and course
- additional textual address information about a location by using AddressInfo class

There is also Orientation class defined in Location API for getting orientation of the device. The

## How\_to\_get\_compass\_directions\_in\_Java\_ME

Orientation.getCompassAzimuth() method can be used for getting the device's horizontal compass azimuth in degrees. Currently (January 2009) there is no implementation of this class in Nokia devices. The example MIDlet below shows, how to get course (and speed) from GPS without using Orientation class and how to draw a compass-like course indicator on the screen. The course is based on the movement, it 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.Alert;
import javax.microedition.lcdui.AlertType;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;

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) {
    }

    protected void showError(String title, String text) {
        Alert alert = new Alert(title, text, null, AlertType.ERROR);
        alert.setTimeout(Alert.FOREVER);
        alert.getType().playSound(Display.getDisplay(this));
        Displayable current = Display.getDisplay(this).getCurrent();
        if (current instanceof Alert) {}
        else Display.getDisplay(this).setCurrent(alert);
    }
}
```

### 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.LocationProvider;
import javax.microedition.location.LocationException;

public class LocationCanvas extends Canvas implements CommandListener, Runnable {
    private CompassMIDlet midlet;
    private Command exitCommand;
    private String latitudeString = "Latitude: ";
```

## How\_to\_get\_compass\_directions\_in\_Java\_ME

```
private String longitudeString = "Longitude: ";
private String courseString = "Course: ";
private String speedString = "Speed: ";
private Thread thread = null;
private Font font;
private float course = 0;
private float speed = 0;

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);
    thread = new Thread(this);
    thread.start();
}

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) {
        thread = null;
        midlet.notifyDestroyed();
    }
}

public void run() {
```

## How\_to\_get\_compass\_directions\_in\_Java\_ME

```
while (true) {
    try {
        Thread.sleep(1000);
        askLocation();
        repaint();
    }
    catch (InterruptedException ie) {}
}

/**
 * Gets the coordinates, speed and course from the GPS.
 */
private void askLocation() {
    try {
        Criteria criteria = new Criteria();
        criteria.setSpeedAndCourseRequired(true);
        LocationProvider lp = LocationProvider.getInstance(criteria);
        if (lp != null) {
            Location l = lp.getLocation(300);
            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 {
                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());
    }
}
}
```

## Screenshot



Latitude: 61.485517806882  
Longitude: 23.762218287066002  
Course: 345.18°  
Speed: 3.96km/h



## See also

- [Category:Location API \(JSR-179\)](#)