Compass APP based on Android

## Purpose/Functionality:

This Android app uses a magnetic field sensor and an accelerometer sensor to determine the orientation of the mobile device in the real world coordinate system (i.e. compass). It displays the different sensor values, and the final orientation in degrees. It also displays a wind rose, rotating according to the orientation of the mobile device.

## Inputs:

* Sensor data of magnetic field sensor for x-, y-, and z-axis.
* Sensor data of accelerometer sensor for x-, y-, and z-axis.

## Outputs:

* Sensor data of magnetic field sensor for x-, y-, and z-axis as TextView
* Sensor data of accelerometer sensor for x-, y-, and z-axis as TextView
* Calculated Orientation of mobile device in degrees as TextView
* Main cardinal directions (i.e. North, East, South, West) as TextView
* Rotating wind rose as ImageView

## Used Sensors:

* Magnetic field sensor
* Accelerometer sensor

## Sensor data processing to achieve compass function:

Every time, one of the sensors is changed, the according x-, y- and z-axis values are updated. The updated values are then converted into a rotation matrix R.

//get rotation matrix

SensorManager.*getRotationMatrix*(R, **null**, accelerometerValues, magneticFieldValues);

This rotation matrix is used to perform a coordinate transformation from the mobile device coordinate system to the world coordinate system.

//get rotation around each axis out of rotation matrix SensorManager.*getOrientation*(R, values);

Knowing the sensor values of the mobile device in the world coordinate system allows calculating the device’s orientation. Finally, the orientation value is converted to degrees.

//convert rotation value to degrees

**float** degree = (**float**) Math.*toDegrees*(values[0]);

## Principal Methods/Listeners:

* **protected** **void** onCreate(Bundle savedInstanceState)  
  This method is invoked at the beginning of the program (i.e. the creation of the app). It invokes the methods initialization() and calculateOrientation() explained bellow.
* **private** **void** initialization()  
  This method initializes all TextViews, the ImageView, the SensorManager and the two Sensors.
* **public** **void** onSensorChanged(SensorEvent event)  
  This method is invoked every time there is a change of one of the two sensors. The stored values of the according sensor are updated. Finally, the method calculateOrientation() is invoked.
* **private** **void** calculateOrientation()  
  This method is used for calculating the current orientation of the mobile device in degrees and updating the rotation of the wind rose according to it. Details about the data processing can be found on the first page.

## Extra Performance:

The app displays a wind rose image. It’s rotation is updated every time the calculateOrientation() method is invoked. An ImageView was created for displaying the wind rose image and initialized the same way as the TextViews. A matrix is created and used in the postRotate() method. The postRotate() method rotates the image according to the current degree value around its center. Finally, the ImageView is updated.

//display and rotate wind rose according to current

//orientation value

Matrix matrix=**new** Matrix();

compass.setScaleType(ScaleType.*MATRIX*); //required

// arguments (rotation, ratation center x, rotation center y)

matrix.postRotate((**float**) -degree, compass.getDrawable().getBounds().width()/2, compass.getDrawable().getBounds().height()/2);

compass.setImageMatrix(matrix);