Distributed Systems HS 2016
Assignment 1

Markus Hauptner, Johannes Beck, Linus Fessler

October 9, 2016

4 Mini-Test

1. (Sensor Framework)

   A) a) List available sensors on a device

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);

  SensorManager sensorManager = (SensorManager)
     getSystemService(Context.SENSOR_SERVICE);
  List<Sensor> sensors =
     sensorManager.getSensorList(Sensor.TYPE_ALL);

  ListView listView = (ListView) findViewById (R.id.listView1);
  listView.setAdapter(new ArrayAdapter<Sensor>(this,
     android.R.layout.list_item,  sensor));
}
```

   b) Retrieve the value range of a specific sensor

```
SensorManager sensorManager = (SensorManager)
    getSystemService(Context.SENSOR_SERVICE);
Sensor sensor = sensorManager.getDefaultSensor(Sensor.TYPE_*);
    // where * can be replaced by the type of the sensor
float valueRange = sensor.getMaximumRange();
```

   c) Retrieve the value range of a specific sensor

```
SensorManager sensorManager = (SensorManager)
    getSystemService(Context.SENSOR_SERVICE);
Sensor sensor =
    sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
// to register
sensorManager.registerListener(this, sensor,
    SensorManager.SENSOR_DELAY_FASTEST);
// and to unregister
sensorManager.unregisterListener(this);
```

B) The mistake is in lines 15/22 where event.values are not copied. This is a mistake since the application doesn't own the event and thus shouldn't change it's values as they might be used again by other applications. Depending on if the log method or any other method in the class `SensorValuesDetector` changes the values in `accelerometerValues` or `proximityValues` and other applications use the values, they will receive wrong values and this will become a problem. To avoid this, lines 15/22 should clone the values with: `event.values.clone()`.

2. (Activity lifecycle)

   Resumed, Paused and Stopped. The corresponding callback functions are: `void onResume()`, `void onPause()`, `void onStop()`.

3. (Resources)

   Strings should be defined in `res/values/strings.xml`. The advantage of this approach is that a string can be reused many times and when there is the need to change this string, it can be changed once in `strings.xml` and take effect everywhere the string is used. Also, all strings are in one place so it's easy to find a specific string.

4. (Intents)

   Explicit intents explicitly state the class of the component to be run. In the following code snippet, TargetActivity will be run, although not only components of type Activity can be run but also components of type Service and BroadcastReceiver.

```
1 Intent intent = new Intent(this, TargetActivity.class);
2 startActivity(intent);
```

   Explicit intents can also be used to pass data from one component to the target component using method `intent.putExtra()`. That data can be retrieved in the target component using `getIntent().getExtras()`.

   Implicit intents on the other hand do not specify the target component explicitly but instead an appropriate target component is determined based on the intent information supplied in the `AndroidManifest.xml` file (action, type, scheme, categories).

5. (Service lifecycle)

   a) False

   b) True

   c) True

   d) False

6. (AndroidManifest file)

   Inside the `<application>` tag, register the LocationService class:

```
1 <service android:name=".LocationService"></service>
```

   Outside of the `<application>` tag, register permissions to send an sms (only needed if the text message is sent over SMS) and to access fine location and register feature to use GPS:

```
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission
    android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-feature android:name="android.hardware.location.gps" />
```