

Distributed Systems HS 2016

Assignment 1

Markus Hauptner, Johannes Beck, Linus Fessler

October 7, 2016

4 Mini-Test

1. (Sensor Framework)

A) a) List available sensors on a device

```
1 @Override
2 protected void onCreate(Bundle savedInstanceState) {
3     super.onCreate(savedInstanceState);
4     setContentView(R.layout.activity_main);
5
6     SensorManager sensorManager = (SensorManager)
7         getSystemService(Context.SENSOR_SERVICE);
8     List<Sensor> sensors =
9         sensorManager.getSensorList(Sensor.TYPE_ALL);
10
11     ListView listView = (ListView) findViewById(R.id.listView1);
12     listView.setAdapter(new ArrayAdapter<Sensor>(this,
13         android.R.layout.list_item, sensor));
14 }
```

b) Retrieve the value range of a specific sensor

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

c) Retrieve the value range of a specific sensor

```
1 SensorManager sensorManager = (SensorManager)
2     getSystemService(Context.SENSOR_SERVICE);
3 Sensor sensor =
4     sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
5 // to register
6 sensorManager.registerListener(this, sensor,
7     SensorManager.SENSOR_DELAY_FASTEST);
8 // and to unregister
9 sensorManager.unregisterListener(this);
```

B) The mistake is in line 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 its values as they might be used again by other applications. Depending on what the log function does (i.e. if it changes the values) and other applications use the values, they will receive wrong values and this will become a problem. To avoid this, line 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 `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 appropriate 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:

```
1 <uses-permission android:name="android.permission.SEND_SMS" />
2 <uses-permission
    android:name="android.permission.ACCESS_FINE_LOCATION" />
```