Mobile Systems Engineering

Joana Iljazi
Ringaile Valiaugaite
Stanislav Kimov

Android Power Manager

Introduction

Android device functions are not optimal and consume a lot of energy. The battery of a device could be used more efficiently. To optimize the battery consumption, we need to monitor the behavior of a device by gathering different metrics and developing several energy consumption modes that alter the device behavior according to these modes to use energy efficiently.

Examples of gathering metrics

We divided the project in three parts. The first part we gathered the measurements about the device, while in the second part we showed the list of all the currently running processes and the selected process' measurement. The third part is to set some modes which gives a warning if a process is overloading the device.

1st part.

For the first part, we were able to gather these metrics:

- Cpu Frequency Measurement (read from the file "/sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq");
- Cpu Usage Measurement (read from the file "/proc/stat");
- Network Measurement (read from the file "/proc/net/dev");
- Gps Status;
- Memory Measurement;
- Mobile Status;
- Screen Status;
- Wifi Status;
- Battery measurement;
- Bluetooth Status;

Most of the measurements are collected from the Android Context Class which is an interface that provides the information about the application environment. [1]

[1] More info: http://developer.android.com/reference/android/content/Context.html

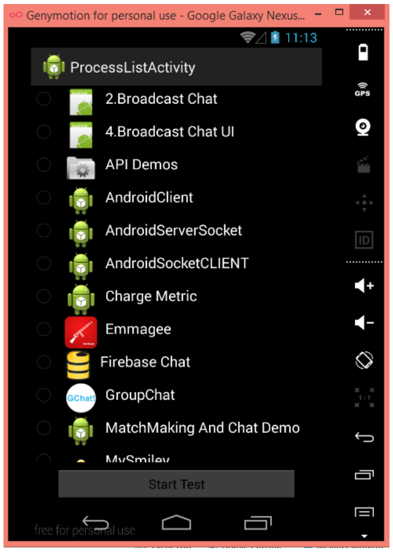
Problems

To test our application in a real environment, we started to use Genymotion emulator[2] rather than an already build-in emulator in Android Studio because of its speed and it is easy-to-use. Although while running the application on the emulator, we were able to get only some measurements, while CPU frequency and usage measurements were not read , therefore we decided to test on a real device (Google Nexus 5), but we were still not able to get these metrics unless the device is rooted.

2nd Part.

To continue with our application, the second part was to list all the currently running processes and after the selected process to show the single process measurements.
We gathered these measurements:

- CPU Info (read from file “sys/devices/system/cpu/”);
- Memory Info (read from file “/proc/meminfo”);
- Battery Info (read from files “/sys/class/power_supply/battery/current_now”, “/sys/class/power_supply/battery/batt_current”, “/sys/class/power_supply/battery/batt_current_adc” and “/sys/class/power_supply/battery/current_avg”);
- Traffic Info (read from files “/proc/uid_stat/" + uid + "/tcp_rcv” and “/proc/uid_stat/" + uid + "/tcp_snd”).



ProcessListActivity

○ 2.Broadcast Chat
○ 4.Broadcast Chat UI
○ API Demos
○ AndroidClient
○ AndroidServerSocket
○ AndroidSocketCLIENT
○ Charge Metric
○ Emmagee
○ Firebase Chat
○ GroupChat
○ MatchMaking And Chat Demo
○ MySmiley

Start Test

We can see the list of the currently running applications and choose one of them to see the current load of the application every second. PowerManager shows in the log the current

[2] https://www.genymotion.com/

memory of application and the device memory as well as application CPU usage and System CPU usage and Network Traffic as for example:

```
App/Available Memory:13.76/854.51MB
App/System CPU:2.97%/31.68%
CurrentN/A,Net Traffic(KB)N/A
```

Every time, when an application is chosen from the list, it is also started, so it is possible to see the application load while the application is in use and how it changes during the time.

Problems

To show the list of currently running processes we used
ActivityManager.getRunningAppProcesses()
but instead of listing all the processes, PowerManager just shows the partial list of currently running processes, as on Google Nexus 5 it shows just the applications installed by an user.

3rd Part

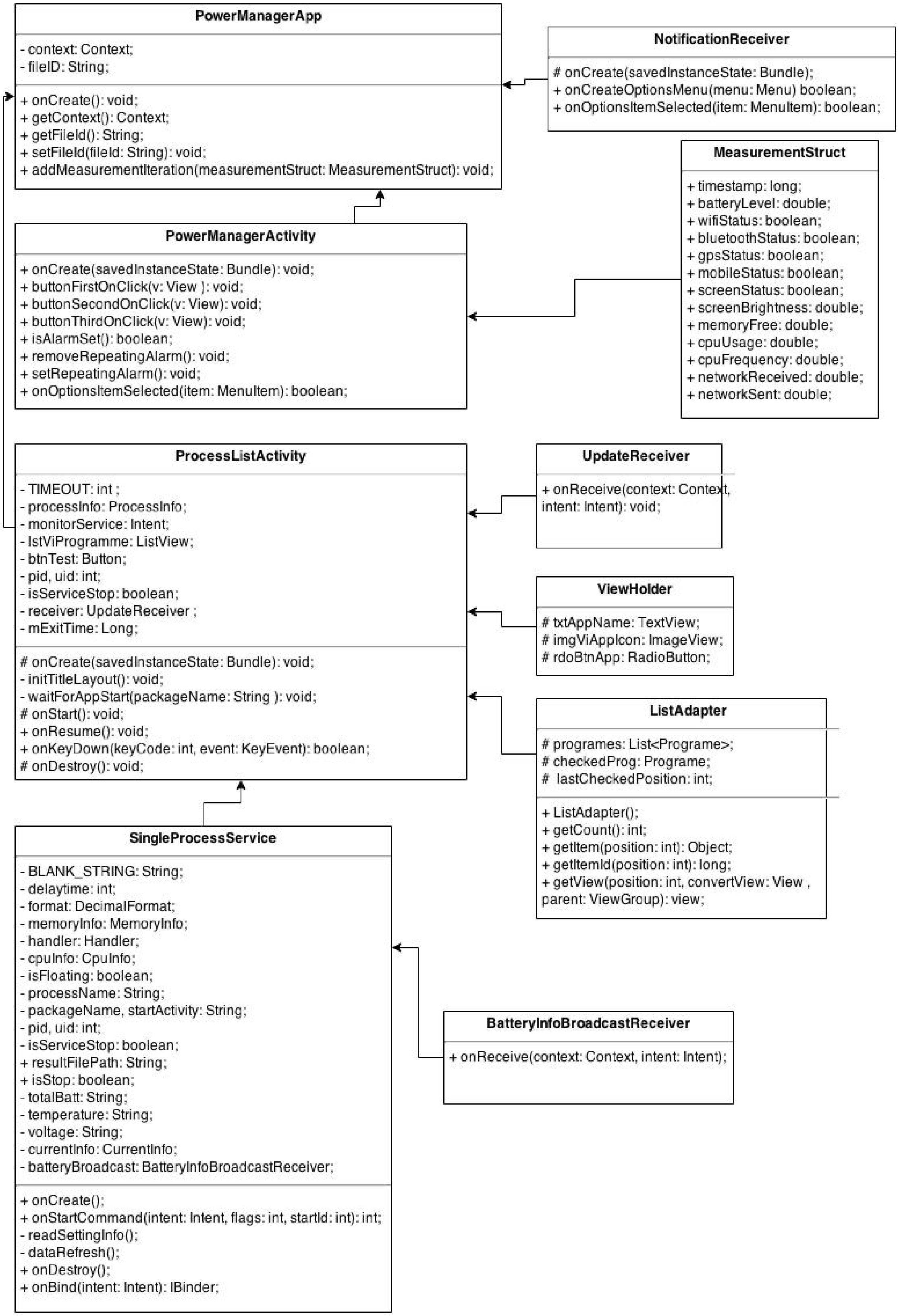
We set three modes on the device: “Saving”, “Normal” and “Performance”.
For the “Saving” mode, we check if the process load is bigger than 50 % or there is less than 75% free memory.
For the “Normal mode”, we check if the process load is more than 75% or there is less than 25% free memory.
If the criteria is met, an user gets a warning about the current state of Processor load and memory free and it turns off wifi and bluetooth and reduces the brightness of the screen.
The “Performance” mode is a default mode, just to monitor the performance of a device.

Problems

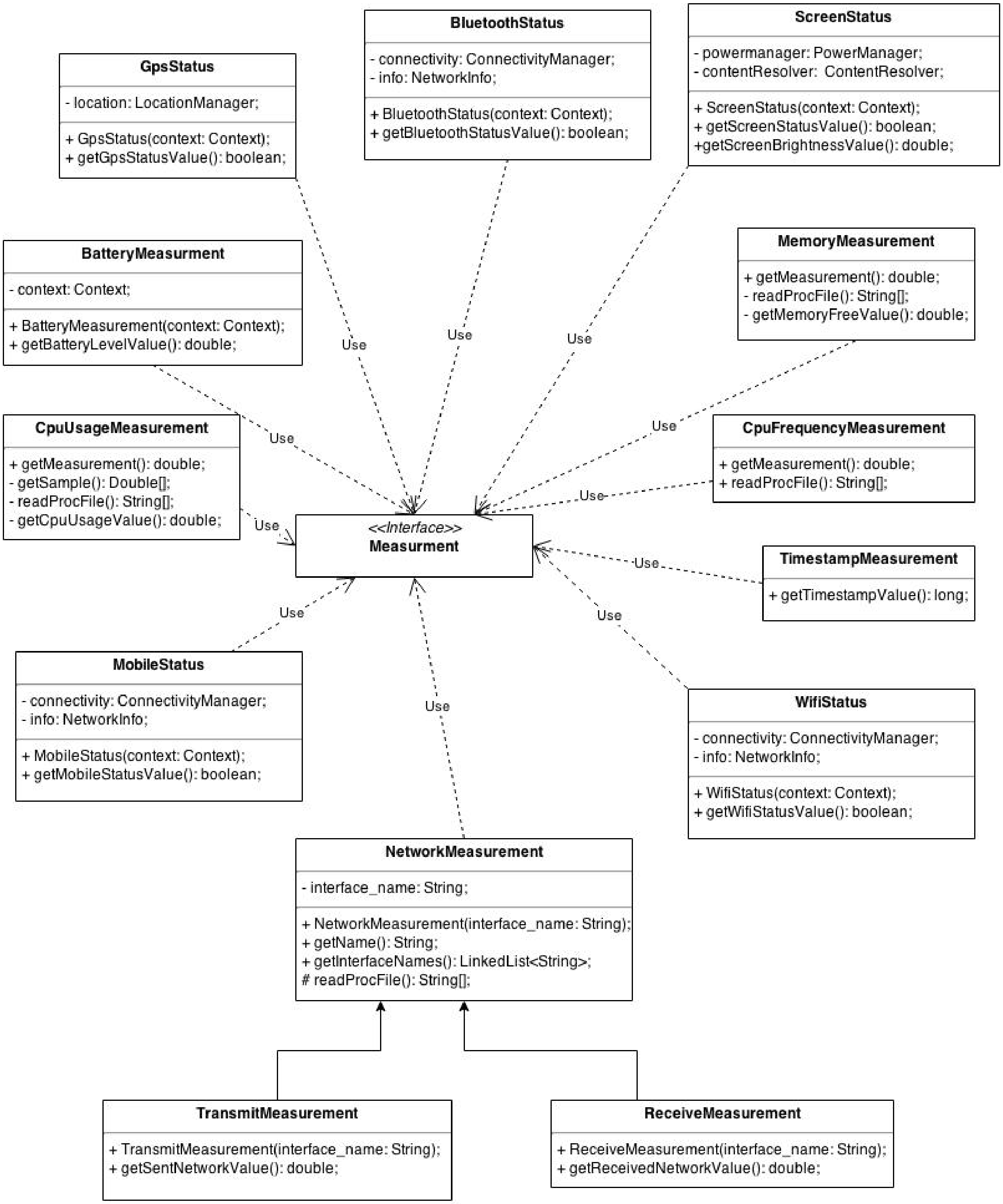
When there are several warnings that need to be shown to a user, they just overlap, therefore we created the message with all the information about the current process load in one warning.

UML Class diagrams

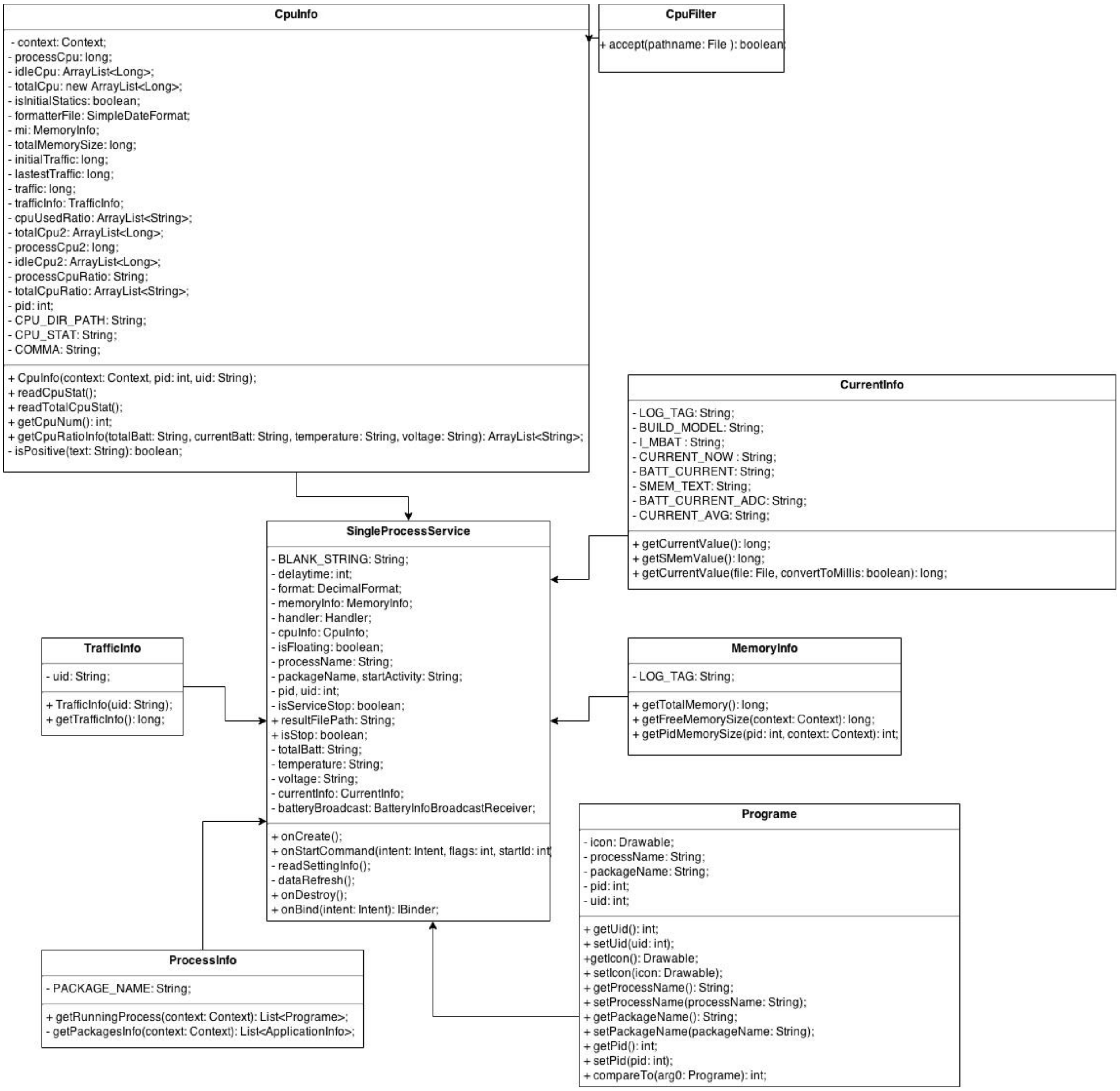
Application Class Diagram


PowerManagerApp
- context: Context;
- fileID: String;
+ onCreate(): void;
+ getContext(): Context;
+ getFileId(): String;
+ setFileId(fileId: String): void;
+ addMeasurementIteration(measurementStruct: MeasurementStruct): void;
NotificationReceiver
# onCreate(savedInstanceState: Bundle);
+ onCreateOptionsMenu(menu: Menu) boolean;
+ onOptionsItemSelected(item: MenuItem): boolean;
MeasurementStruct
+ timestamp: long;
+ batteryLevel: double;
+ wifiStatus: boolean;
+ bluetoothStatus: boolean;
+ gpsStatus: boolean;
+ mobileStatus: boolean;
+ screenStatus: boolean;
+ screenBrightness: double;
+ memoryFree: double;
+ cpuUsage: double;
+ cpuFrequency: double;
+ networkReceived: double;
+ networkSent: double;
PowerManagerActivity
+ onCreate(savedInstanceState: Bundle): void;
+ buttonFirstOnClick(v: View): void;
+ buttonSecondOnClick(v: View): void;
+ buttonThirdOnClick(v: View): void;
+ isAlarmSet(): boolean;
+ removeRepeatingAlarm(): void;
+ setRepeatingAlarm(): void;
+ onOptionsItemSelected(item: MenuItem): boolean;
ProcessListActivity
- TIMEOUT: int ;
- processInfo: ProcessInfo;
- monitorService: Intent;
- lstViProgramme: ListView;
- btnTest: Button;
- pid, uid: int;
- isServiceStop: boolean;
- receiver: UpdateReceiver ;
- mExitTime: Long;
# onCreate(savedInstanceState: Bundle): void;
- initTitleLayout(): void;
- waitForAppStart(packageName: String): void;
# onStart(): void;
+ onResume(): void;
+ onKeyDown(keyCode: int, event: KeyEvent): boolean;
# onDestroy(): void;
UpdateReceiver
+ onReceive(context: Context,
intent: Intent): void;
ViewHolder
# txtAppName: TextView;
# imgViAppIcon: ImageView;
# rdoBtnApp: RadioButton;
ListAdapter
# programes: List<Programe>;
# checkedProg: Programe;
# lastCheckedPosition: int;
+ ListAdapter();
+ getCount(): int;
+ getItem(position: int): Object;
+ getItemId(position: int): long;
+ getView(position: int, convertView: View ,
parent: ViewGroup): view;
SingleProcessService
- BLANK_STRING: String;
- delaytime: int;
- format: DecimalFormat;
- memoryInfo: MemoryInfo;
- handler: Handler;
- cpuInfo: CpuInfo;
- isFloating: boolean;
- processName: String;
- packageName, startActivity: String;
- pid, uid: int;
- isServiceStop: boolean;
+ resultFilePath: String;
+ isStop: boolean;
- totalBatt: String;
- temperature: String;
- voltage: String;
- currentInfo: CurrentInfo;
- batteryBroadcast: BatteryInfoBroadcastReceiver;
+ onCreate();
+ onStartCommand(intent: Intent, flags: int, startId: int): int;
- readSettingInfo();
- dataRefresh();
+ onDestroy();
+ onBind(intent: Intent): IBinder;
BatteryInfoBroadcastReceiver
+ onReceive(context: Context, intent: Intent);

Application Measurements Class Diagram


GpsStatus
- location: LocationManager;
+ GpsStatus(context: Context);
+ getGpsStatusValue(): boolean;
BluetoothStatus
- connectivity: ConnectivityManager;
- info: NetworkInfo;
+ BluetoothStatus(context: Context);
+ getBluetoothStatusValue(): boolean;
ScreenStatus
- powermanager: PowerManager;
- contentResolver: ContentResolver;
+ ScreenStatus(context: Context);
+ getScreenStatusValue(): boolean;
+getScreenBrightnessValue(): double;
BatteryMeasurment
- context: Context;
+ BatteryMeasurement(context: Context);
+ getBatteryLevelValue(): double;
MemoryMeasurement
+ getMeasurement(): double;
- readProcFile(): String[];
- getMemoryFreeValue(): double;
Use
Use
Use
Use
Use
CpuUsageMeasurement
+ getMeasurement(): double;
- getSample(): Double[];
- readProcFile(): String[];
- getCpuUsageValue(): double;
Use
CpuFrequencyMeasurement
+ getMeasurement(): double;
+ readProcFile(): String[];
Use
<<Interface>>
Measurment
Use
TimestampMeasurement
+ getTimestampValue(): long;
Use
Use
MobileStatus
- connectivity: ConnectivityManager;
- info: NetworkInfo;
+ MobileStatus(context: Context);
+ getMobileStatusValue(): boolean;
Use
WifiStatus
- connectivity: ConnectivityManager;
- info: NetworkInfo;
+ WifiStatus(context: Context);
+ getWifiStatusValue(): boolean;
NetworkMeasurement
- interface_name: String;
+ NetworkMeasurement(interface_name: String);
+ getName(): String;
+ getInterfaceNames(): LinkedList<String>;
# readProcFile(): String[];
TransmitMeasurement
+ TransmitMeasurement(interface_name: String);
+ getSentNetworkValue(): double;
ReceiveMeasurement
+ ReceiveMeasurement(interface_name: String);
+ getReceivedNetworkValue(): double;

Single Process Measurements Class Diagram



CpuInfo

```
 - context: Context;
- processCpu: long;
- idleCpu: ArrayList<Long>;
- totalCpu: new ArrayList<Long>;
- isInitialStatics: boolean;
- formatterFile: SimpleDateFormat;
- mi: MemoryInfo;
- totalMemorySize: long;
- initialTraffic: long;
- lastestTraffic: long;
- traffic: long;
- trafficInfo: TrafficInfo;
- cpuUsedRatio: ArrayList<String>;
- totalCpu2: ArrayList<Long>;
- processCpu2: long;
- idleCpu2: ArrayList<Long>;
- processCpuRatio: String;
- totalCpuRatio: ArrayList<String>;
- pid: int;
- CPU_DIR_PATH: String;
- CPU_STAT: String;
- COMMA: String;
```

```
+ CpuInfo(context: Context, pid: int, uid: String);
+ readCpuStat();
+ readTotalCpuStat();
+ getCpuNum(): int;
+ getCpuRatioInfo(totalBatt: String, currentBatt: String, temperature: String, voltage: String): ArrayList<String>;
- isPositive(text: String): boolean;
```

CpuFilter

```
+ accept(pathname: File ): boolean;
```

CurrentInfo

```
- LOG_TAG: String;
- BUILD_MODEL: String;
- I_MBAT : String;
- CURRENT_NOW : String;
- BATT_CURRENT: String;
- SMEM_TEXT: String;
- BATT_CURRENT_ADC: String;
- CURRENT_AVG: String;
```

```
+ getCurrentValue(): long;
+ getSMemValue(): long;
+ getCurrentValue(file: File, convertToMillis: boolean): long;
```

SingleProcessService

```
- BLANK_STRING: String;
- delaytime: int;
- format: DecimalFormat;
- memoryInfo: MemoryInfo;
- handler: Handler;
- cpuInfo: CpuInfo;
- isFloating: boolean;
- processName: String;
- packageName, startActivity: String;
- pid, uid: int;
- isServiceStop: boolean;
+ resultFilePath: String;
+ isStop: boolean;
- totalBatt: String;
- temperature: String;
- voltage: String;
- currentInfo: CurrentInfo;
- batteryBroadcast: BatteryInfoBroadcastReceiver;
```

```
+ onCreate();
+ onStartCommand(intent: Intent, flags: int, startId: int
- readSettingInfo();
- dataRefresh();
+ onDestroy();
+ onBind(intent: Intent): IBinder;
```

TrafficInfo

```
- uid: String;
```

```
+ TrafficInfo(uid: String);
+ getTrafficInfo(): long;
```

MemoryInfo

```
- LOG_TAG: String;
```

```
+ getTotalMemory(): long;
+ getFreeMemorySize(context: Context): long;
+ getPidMemorySize(pid: int, context: Context): int;
```

Programe

```
- icon: Drawable;
- processName: String;
- packageName: String;
- pid: int;
- uid: int;
```

```
+ getUid(): int;
+ setUid(uid: int);
+getIcon(): Drawable;
+ setIcon(icon: Drawable);
+ getProcessName(): String;
+ setProcessName(processName: String);
+ getPackageName(): String;
+ setPackageName(packageName: String);
+ getPid(): int;
+ setPid(pid: int);
+ compareTo(arg0: Programe): int;
```

ProcessInfo

```
- PACKAGE_NAME: String;
```

```
+ getRunningProcess(context: Context): List<Programe>;
- getPackagesInfo(context: Context): List<ApplicationInfo>;
```