1 Android Manifest

- Описание проекта, в том числе настройки и конфигурации, например версии.
- Прописаны permission и составляющие проекта.

2 Activity

- Первой в программе вызывается MainActivity(? не помню).
- Цикл жизни.
- Activity надо прописывать в манифесте.

3 BroadCastReceiver

??На один и тот же интент, на один и тот же тип интента может быть очень много. И они вызываются последовательно. Если там в какой-нибудь цепочке первый взял на час захватил процессор и работает упорно, все другие будут ждать оповещения об этом.??

BroadCastReceiver - класс-обработчик intent-ов, то есть класс-обработчик широковещательных сообщений. Он может быть подписан на несколько разных интентов. Если к нему приходит несколько интентов за раз, то они выстраиваются в цепочку и обрабатываются по-одному, поэтому есть определённые ограничения на работу BroadCastReceiver. Если вы попробуете обрабатывать интент больше 5 секунд, обработчик принудительно прикончат, чтобы другие интенты не ждали, пока их обработают. Из-за этой особенности количество действий, которое вы можете сделать с помощью BroadCastReceiver, довольно ограничено. Как правило, всё, что делают в BroadCastReceiver - это посылают другой интент, чтобы запустить Activity, Service(это такой класс, позже мы зачем он нужен и что это такое) или что-нибудь подобное.

Другое назначение BroadCastReceiver это получать системные оповещения. Есть некоторое количество системных оповещений, например о том, что андроид загрузился, или, например, у вас маленький заряд батарейки. Можно подписаться и на них и как-то среагировать.

Как и Activity, BroadCastReceiver должен быть прописан в андроид манифесте(на самом деле его можно зарегистрировать и программно, и иногда так и делают, но мы этого касаться не будем):

```
1 <receiver android:name=".LocationChangedReceiver" />
```

Теперь, надо как-то сказать приложению, на какие intent мы подписываем наш BroadCastReceiver. Вообще-то говоря, можно это сделать прямо в манифесте. Так обычно и делают, если вы подписываетесь на системные события. А можно сделать это програмно. Мы будем делать это программно:

```
1 mLocationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
2 Intent intent = new Intent(this, LocationChangedReceiver.class);
3 mLocationChangedIntent =
4 PendingIntent.getBroadcast(this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);
```

Здесь я, когда создаётся Activity, получаю доступ к некоторому LocationManager, который вы уже видели раньше. Затем я создаю Intent ручками, где явно прописываю класс, куда интент должен быть отправлен. Это то, что называют explicit Intent. Вообще, интенты бывают implicit, а бывают explicit. Explicit интенты - это интенты, для которых мы явно прописываем, к какому классу его отправить, implicit - когда мы некоторым образом описываем, куда мы хотим передать сообщение, и потом уже сам андроид разбирается, кого надо запустить по этому описанию. В данном случае, мы при создании интента явно указываем класс приёмника. Соответсвенно, Intent ему и будет доставлен.

На самом деле, здесь отправляется не совсем интент, а то, что называется PendingIntent - наследник класса интент. Главное его отличие от обычного интента в том, что он учитывает права приложения.

Как мы уже говорили, вы можете запускать Activity(приложение) из другого приложения, но у этого Activity(приложения) может не быть прав на работу с какими-то там ресурсами. Если эти права нужны, вы можете послать PendingIntent и передать тем самым Activity(приложению), которое вы запускаете, свои права на исполнение. Вам не всегда это нужно, но некоторые API просто по умолчанию используют PendingIntent. Понятно, что в нашем случае BroadCastReceiver это часть нашего же приложения, и у него есть все те же самые права, поэтому PendingIntent как бы и не нужен. Но дело а том, что посылаем мы его с помощью LocationManager, а интерфейс LocationManager требует PendingIntent, поэтому отправляется здесь именно PendingIntent.

Когда приходит интент, который предназначен для данного BroadCastReceiver, вызывается метод onReceive(). Одним из его аргументов является экземпляр класса Context - базовый класс для частей приложения. Он предоставляет доступ ко всем андроидовским ресурсам. Через контест вы можете обращаться к ресурсам системы. К примеру, Activity - наследник класса Context. Поэтому, мы можем вызвать, к примеру, метод getSystemService(), определённый в классе Context - для доступа к какому-то сервису. Activity является наследником класса Context, поэтому этот метод вызывается без всякого префикса. А вот BroadCastReceiver не является наследником класса Context, но доступ ко всяким менеджерам вполне себе может понадобиться, поэтому-то контекст и передаётся методу onReceive().

В нашем Reciever мы не будем делать ничего такого, мы просто напечатаем в Log некое сообщение(что такое Log мы поподробнее ещё обсудим).

```
1  public class LocationChangedReceiver extends BroadcastReceiver {
2
3      private static final String TAG = LocationChangedReceiver.class.getSimpleName()
           ;
4
5      public LocationChangedReceiver() { }
6
7      @Override
8      public void onReceive(Context context, Intent intent) {
9          final String locationKey = LocationManager.KEY_LOCATION_CHANGED;
10
11         if (intent.hasExtra(locationKey)) {
12             Location location = (Location) intent.getExtras().get(locationKey);
13
14             Intent startService = new Intent(context, ForecastUpdateService.class);
15             startService.putExtra(LocationManager.KEY_LOCATION_CHANGED, location);
16             context.startService(startService);
17         }
18     }
19 }
```

Если обобщить:

- Один из стандартных классов андроид, завязан на несколько(может один) intent.
- Получает интент, к которому он привязан и быстро реагирует(с технической точки зрения при получении intent вызывается onReceive(), который должен быстро выполняться).
- Если один BroadCastReceiver соотетствует нескольким intent, то они построятся в цепочку и будут выполняться последовательно.
- Поэтому если по какому-то intent метод выполняется слишком долго, то его прикончат, чтобы другие intent не ждали.
- Так что обычно, его используют только чтобы послать какой-то другой intent или получить системное оповещение.
- Receiver должен быть прописан в Android манифесте.

```
1 <receiver android:name=".LocationChangedReceiver" />
```

- Подписку на intent можно тоже записать в Manifest.
- А можно это сделать и програмно. Explicit intent - явно указываем класс, к которому intent отправить. Бывает ещё implicit intent. На самом деле здесь посылается не intent а наследник класса Intent - PendingIntent, в котором учитываются права приложения, из которого нас запустили.
- Intent - это такой способ общаться между частями приложения. В него можно записать дополнительную информацию.

```
1 startService.putExtra(LocationManager.KEY_LOCATION_CHANGED, location);
```

- Доставать информацию из Intent можно так:

```
1 Location location = (Location) intent.getExtras().get(locationKey);
```

- Класс Context - это базовый класс для частей приложения. К примеру, Activity - наследник класса Context. Он предоставляет доступ ко всем андроидовским ресурсам. BroadCastReceiver не является наследником класса Context, поэтому он туда передаётся.

4 Log

Мы открываем наш DeviceMonitor. LogCat.

- Записи в log выводятся на экран Logcat.
- Логирование происходит так:

```
1     ...
2 private static final String TAG =
3             SMSReceiver.class.getSimpleName();
4     ...
5 Log.d(TAG, "SMS received");
```

5 Permissions

- Чтобы работать с сетью надо подключить соотетсвующие permission.

```
1 <uses-permission android:name="android.permission.INTERNET" />
2 <uses-permission  android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

6 Service

- Service также надо прописывать в манифесете:

```
1 <service
2     android:name=".DatabaseService"
3     android:exported="false" >
4 </service>
```

- Для того, чтобы делать продолжительные действия BroadCastReceiver не подходит. Для этого нужно использовать Service.

- Ключик exported задаёт, позволено ли другим приложениям запускать наш сервис, Activity и тд.
- Если к нему обращаются много intent, они также становятся в очередь.
- Запускают intent к service через команду startService(...);
- При обработке intent запускается метод onHandleIntent(Intent intent);

7 Connection

- Самый простой способ работать с сетью - это HttpURLConnection.
- Возвращается методом .openConnection():

```
1 URL url = new URL(uri.toString());
2 connection =
3         (HttpURLConnection) url.openConnection();
```

- Протокол http может обрабатывать различные запросы. Например, запрос на получение информации. Если мы хотим получить информацию - пишем следующее:

```
1 connection.setRequestMethod("GET");
2 connection.connect();
```

Пример считывания информации:

```
1  InputStream inputStream = connection.getInputStream();
2  StringBuilder buffer = new StringBuilder();
3  if (inputStream == null) {
4      return;
5  }
6  reader = new BufferedReader(new InputStreamReader(inputStream));
7  String line;
8  while ((line = reader.readLine()) != null) {
9      buffer.append(line);
10     buffer.append("\n");
11 }
12
13 if (buffer.length() == 0) {
14     return;
15 }
```