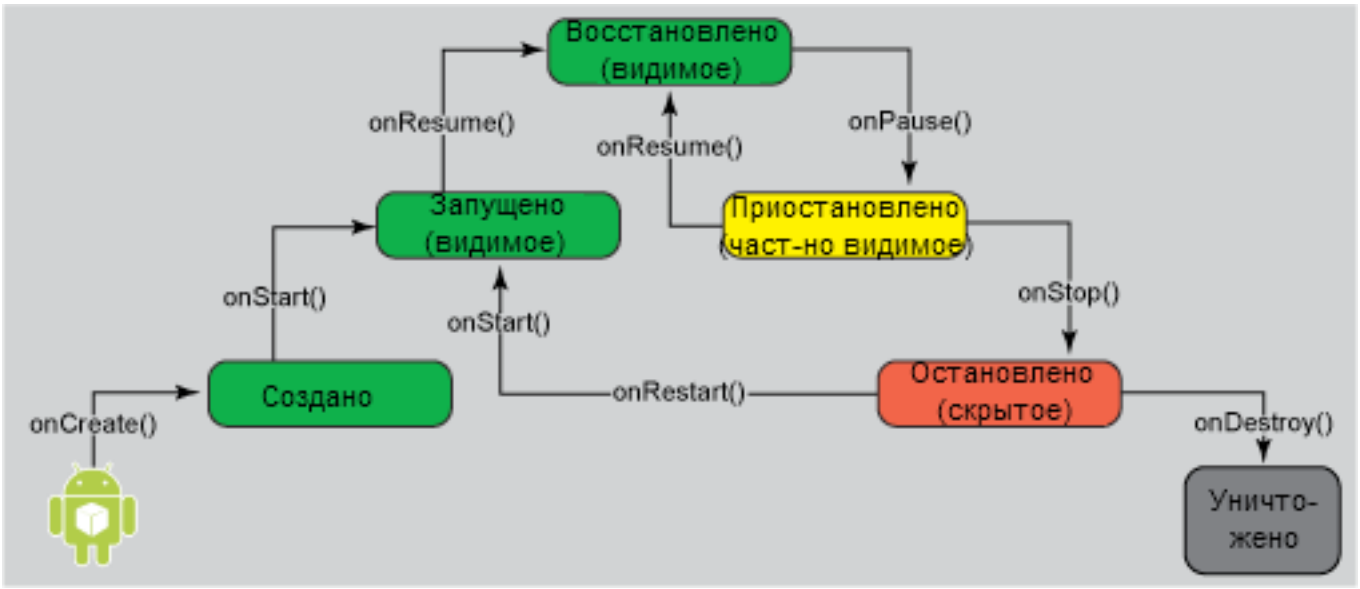
1 Android Manifest

- Описание проекта, в том числе настройки и конфигурация.
- Прописаны permission и составляющие проекта, такие как:
  1. Activity
  2. BroadCastReceiver
  3. Service

2 Activity

- Экран приложения
- Первой в программе вызывается Activity, к которому приходит интент MAIN. Это также прописывается в Android Manifest.
- Цикл жизни:



3 BroadCastReceiver

BroadCastReceiver - класс-обработчик широковещательных сообщений(intent). 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 интент не обрабатывается дольше 5 секунд, чтобы другие не ждали. Из-за этого количество действий, которое вы можете сделать с помощью BroadCastReceiver, довольно ограничено(обычное использование - послать другой интент, чтобы запустить Activity, Service или что-нибудь подобное).

Другое назначение BroadCastReceiver это получать системные оповещения, такие как оповещения о маленьком заряде батарейки или о том, что Android загрузился.

BroadCastReceiver регистрируется в Android Manifest'е так:

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

В принципе, можно регистрировать и программно.

Указать, на какие intent'ы подписан BroadCastReceiver, можно двумя способами:

- В Android Manifest (обычно при подписке на системные события)
- 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:

- Получаем доступ к LocationManager
- Создаем Intent и прописываем класс, куда интент должен быть направлен.
- Создаем PendingIntent от нашего Intent.

Класс 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);
```

PendingIntent - наследник класса Intent. Главное его отличие от обычного интента в том, что он учитывает права приложения. Можно запускать Activity(приложение) из другого приложения, но у этого Activity(приложения) может не быть прав на работу с некоторыми ресурсами. Если эти права нужны, вы можете послать PendingIntent и передать тем самым Activity(приложению), которое вы запускаете, свои права на исполнение. Также некоторые API просто по умолчанию используют PendingIntent(например, так делает LocationManager).

Интенты бывают:

- explicit - для них явно прописывается, к какому классу его отправить
- implicit - неявно описывается, куда нужно передать сообщение. Дальше в этом разбирается Android.

В коде выше интент - explicit.

Когда приходит интент, который предназначен для данного BroadCastReceiver, вызывается метод onReceive(). Одним из его аргументов является экземпляр класса Context - базовый класс для частей приложения. Через него можно 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. А вот BroadCastReceiver не является наследником класса Context, но доступ к различным менеджерам может понадобиться, поэтому контекст и передаётся методу onReceive().

В нашем Receiver 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 Android, завязан на несколько(может один) intent.
- Получает intent, к которому он привязан и быстро реагирует(с технической точки зрения при получении intent вызывается onReceive(), который должен быстро выполняться).
- Если один BroadCastReceiver соответствует нескольким intent, то они построятся в цепочку и будут выполняться последовательно.
- Поэтому если по какому-то intent метод выполняется слишком долго, то его выполнение оборвется, 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 Manifest.

```
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.
- Класс Context - это базовый класс для частей приложения. К примеру, Activity - наследник класса Context. Он предоставляет доступ ко всем андроидовским ресурсам и поэтому является параметром метода onReceive у BroadcastReceiver.

4 Log

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

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

5 Service

- Service также надо прописывать в манифесете:

```
<service
    android:name=".DatabaseService"
    android:exported="false" >
</service>
```

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

6 Permissions

- Чтобы работать с сетью надо подключить соотетсвующие permission.

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

7 Connection

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

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

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

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

Пример считывания информации:

```
InputStream inputStream = connection.getInputStream();
StringBuilder buffer = new StringBuilder();
if (inputStream == null) {
    return;
}
reader = new BufferedReader(new InputStreamReader(inputStream));
String line;
while ((line = reader.readLine()) != null) {
    buffer.append(line);
    buffer.append("\n");
}

if (buffer.length() == 0) {
    return;
}
```