

# Location Base

# Location system

## LocationManager

android.location

- Предоставляет
  - ▶ Получение текущей позиции
  - ▶ Мониторинг изменений
  - ▶ Запрос на оповещении при достижении месторасположения

### ▶ Получение доступа к LocationManager

- Context.getSystemService (LOCATION\_SERVICE)
- Service : Context.LOCATION\_SERVICE

```
private var locationManager: LocationManager? = null
```

```
locationManager = getSystemService(LOCATION_SERVICE) as LocationManager
```

# Выбор провайдера

## ► Global Positioning System (GPS)

- LocationManager.GPS\_PROVIDER
- Спутниковый
- Высокая точность (до 3 м)
- энергоемкий

## ► Network-based

- LocationManager.NETWORK\_PROVIDER
- Информация идет от комбинации WiFi точек и сотовых вышек
- варьируется от 10 м –100м
- менее энергоемкий чем GPS

# Установление разрешений

## ► Course location providers

- Низкой точности
- запрос ACCESS\_COURSE\_LOCATION разрешения

```
<uses-permission android:name="android.permission.ACCESS_COURSE_LOCATION"
```

## ► Fine location providers

- высокоточный
- запрос ACCESS\_FINE\_LOCATION
- Подразумевает course location permission

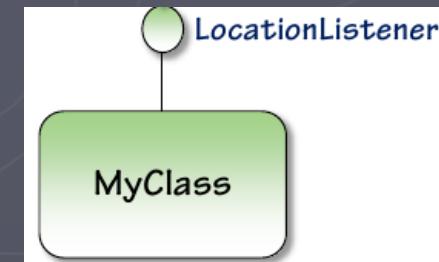
```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

# Интерфейс LocationListener

Содержит методы для информирования о местоположении и изменении поставщика

## ► onLocationChanged

- Вызывается с текущей информацией о местоположении
- Получает экземпляр класса Location



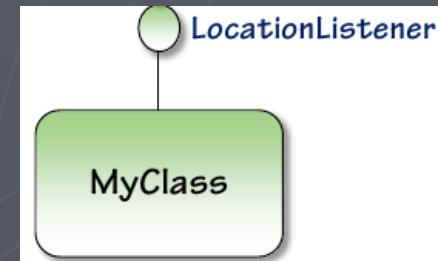
# Интерфейс LocationListener

## ► onProviderEnabled/onProviderDisabled

- Вызывается →enables/disables поставщика

## onStatusChanged

- Существенные изменения в статусе поставщика
- Зависит от поставщика



# Location класс

- ▶ Обеспечивает общую информацию о местоположении
- ▶ Поля
  - `getLatitude`, `getLongitude`, `getAccuracy`
  - широта / долгота и точность в метрах
- ▶ Время
  - `getTime`: ( UTC )
    - ▶ Может варьироваться от разных поставщиков
  - `getElapsedRealtimeNanos`: время наносекундах с момента загрузки устройства
    - ▶ Android 4.2 (API 17) и постоянна для провайдеров

## ► Другую информацию

- `getSpeed`, `getBearing`, `getAltitude`
- `has...` (метод) проверка на наличие данных

# Получение обновлений

```
locationManager = getSystemService(LOCATION_SERVICE) as LocationManager  
provider = locationManager.getProvider(LocationManager.GPS_PROVIDER)!!  
demoListener = DemoLocation()  
  
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,  
    500, 10f, demoListener)
```

- ▶ 1) получение ссылки на location service
- ▶ 2) Создать экземпляр реализации LocationListener
- ▶ 3) вызов метода requestLocationUpdates
  - Передать имя нужного поставщика
  - Передать минимальное время (в миллисекундах), для получения уведомлений или 0
  - Передать минимальное расстояние (в метрах), для получения уведомления или 0
  - Передать ссылку на реализацию LocationListener
- ▶ 4) обновления выполняются до вызова removeUpdates

# Пример

```
class DemoLocation : LocationListener {  
  
    private val TAG: String = "Demo Location"  
    override fun onLocationChanged(location: Location) {  
        val provider = location.provider  
        val lat = location.latitude  
        val lng = location.longitude  
        val accuracy = location.getAccuracy();  
        val time = location.getTime()  
        Log.d(TAG, provider+ " " + lat +"  
              + lng+ " "+ accuracy+ " "+ time);  
    }  
}
```

```
class MainActivity : AppCompatActivity() {    lm.removeUpdates(demoListener);  
    lateinit var locationManager: LocationManager  
    lateinit var provider: LocationProvider  
    lateinit var demoListener: LocationListener  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        locationManager = getSystemService(LOCATION_SERVICE) as LocationManager  
        demoListener = DemoLocation();  
        if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_COARSE_LOCATION) !=  
            PackageManager.PERMISSION_GRANTED &&  
            ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_COARSE_LOCATION) !=  
            PackageManager.PERMISSION_GRANTED) {  
            return;  
        }  
  
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,  
            500, 10f, demoListener)
```

```
2020-12-20 19:07:11.083 16574-16574/by.bstu.patsei.locationbase D/Demo Location: gps 37.421998333333335  
-122.0840000000002 20.0 1608480431000  
2020-12-20 19:07:12.075 16574-16574/by.bstu.patsei.locationbase D/Demo Location: gps 37.421998333333335  
-122.0840000000002 20.0 1608480432000  
2020-12-20 19:07:13.079 16574-16574/by.bstu.patsei.locationbase D/Demo Location: gps 37.421998333333335  
-122.0840000000002 20.0 1608480433000  
2020-12-20 19:07:14.079 16574-16574/by.bstu.patsei.locationbase D/Demo Location: gps 37.421998333333335  
-122.0840000000002 20.0 1608480434000  
2020-12-20 19:07:15.082 16574-16574/by.bstu.patsei.locationbase D/Demo Location: gps 37.421998333333335  
-122.0840000000002 20.0 1608480435000  
2020-12-20 19:07:16.085 16574-16574/by.bstu.patsei.locationbase D/Demo Location: gps 37.421998333333335  
-122.0840000000002 20.0 1608480436000
```

# Быстрое (не точное) получение одиночного местоположения

```
val locManager = getSystemService(LOCATION_SERVICE)  
                  as LocationManager  
  
val recentLocation = locManager  
    .getLastKnownLocation(LocationManager.GPS_PROVIDER)  
    as Location
```

- ▶ Сматрит в буфере последнее значение (м.б. устаревшее, null, час назад....)
- ▶ Быстро (немедленный ответ)

# Текущее получение одиночного местоположения

```
locationManager = getSystemService(LOCATION_SERVICE) as LocationManager  
  
demoListener = DemoLocation();  
locationManager.requestSingleUpdate(LocationManager.GPS_PROVIDER,  
        demoListener, null);
```

- ▶ Использует метод обратного вызова (requestLocationUpdates)
- ▶ Запускается в основном потоке

# Индивидуальные особенности провайдера

- представлены классом LocationProvider

```
val lm = getSystemService(LOCATION_SERVICE) as LocationManager
val gpsProvider = lm.getProvider(LocationManager.GPS_PROVIDER);
```

- Уровень требований к мощности
  - getPowerRequirements
- Точность
  - getAccuracy

- ▶ Что провайдер требует, чтобы определить местоположение
  - requiresSatellite, requiresNetwork, requiresCell
- ▶ Какая информация предоставляется
  - supportsAltitude, supportsBearing, supportsSpeed

# Выбор провайдера по требованиям

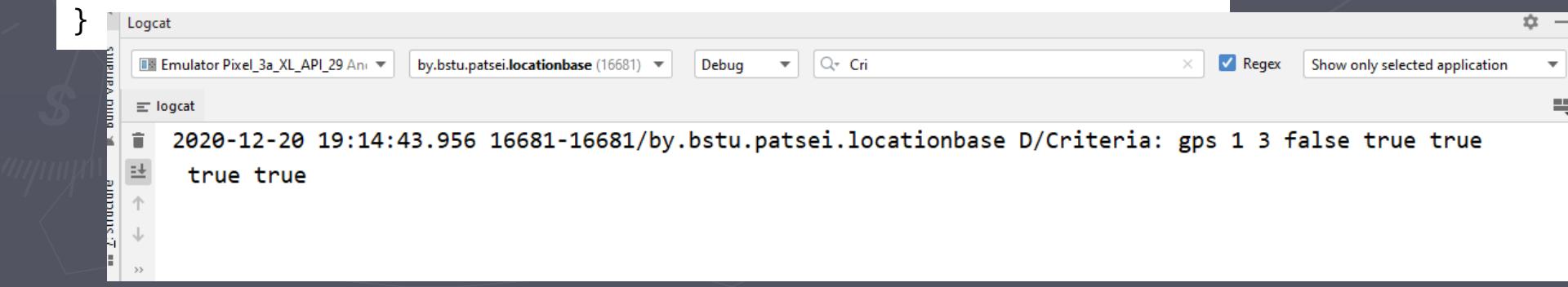
- ▶ 1) Указание желаемого поведения поставщика Criteria class
  - определить качество информации
  - необходимая информация
    - ▶ setAltitudeRequired, setBearingRequired, setSpeedRequired
  - требования к точности (широта / долгота)
    - ▶ setAccuracy setSpeedAccuracy, setVerticalAccuracy
  - использование ресурсов
    - ▶ setPowerRequirement, setCostAllowed

## ► 2) В классе критериев сопоставить поведение поставщикам

- если поставщик отвечает критериям
  - LocationProvider.meetsCriteria
- получить список поставщиков услуг, которые удовлетворяют критериям
  - LocationManager.getProviders
  - все или включенные
  - ограничен ли разрешениями приложение
- Обновления запроса местоположения с помощью провайдера, который отвечает критериям
  - LocationManager.requestLocationUpdates / requestSingleUpdate

```
val criteria = Criteria()
criteria.setAccuracy(Criteria.ACcuracy_FINE)
criteria.setAltitudeRequired(true)
val lm = getSystemService(LOCATION_SERVICE) as LocationManager
val matchingProviderNames =
lm.getProviders(criteria, false) as List<String>

for(providerName in matchingProviderNames) {
    val provider = lm.getProvider(providerName) as LocationProvider
    val name = provider.getName()
    val horizontalAccuracy = provider.getAccuracy()
    val powerRequirements = provider.getPowerRequirement()
    val hasMonetaryCost = provider.hasMonetaryCost()
    val requiresCell = provider.requiresCell()
    val requiresNetwork = provider.requiresNetwork()
    val requiresSatellite = provider.requiresSatellite()
    val supportsAltitude = provider.supportsAltitude()
    val supportsBearing = provider.supportsBearing()
    val supportsSpeed = provider.supportsSpeed()
    Log.d("Criteria", name + " " + horizontalAccuracy + " "
        + powerRequirements + "+ " + requiresCell + " "
        + requiresNetwork + "+ " + requiresSatellite + " "
        + supportsAltitude + "+ " + supportsBearing);
}
```



# Проверка доступности поставщика

есть ли возможность работать поставщиком

## ► LocationListener

- onEnabled / onDisabled – зависит от пользователя

## ► Проверка перед использованием

- LocationManager.isProviderEnabled

```
val isAvailable =  
    lm.isProviderEnabled(LocationManager.NETWORK_PROVIDER)  
if(!isAvailable) {  
    val dialog =  
        AlertUserDialog("Enable Location Services",  
                        Settings.ACTION_LOCATION_SOURCE_SETTINGS)  
    dialog.show(getFragmentManager(), null)  
}
```

# Факторы влияющие на поставщика

## ► Wi-Fi выключен

- сеть провайдера ограничена вышками сотовой связи
- Снижает точность
- ConnectivityManager - определить, Wi-Fi активен

## ► Airplane mode

- Отключает Wi-Fi и сотовую связь
- провайдер по-прежнему сообщает, что он включен
- Settings.Global класс - AIRPLANE\_MODE\_ON для определения состояния

# Включить необходимые функции

► автоматически отображать нужный экран

- `startActivity` с соот. Intent

- Location providers:

`Settings.ACTION_LOCATION_SOURCE_SETTINGS`

- настройка Wi-Fi:

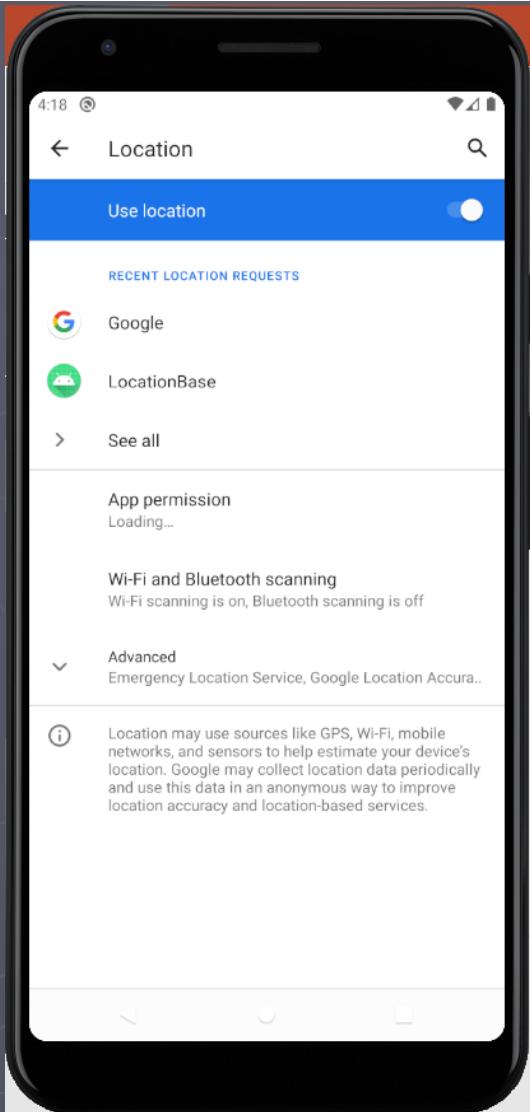
`Settings.ACTION_WIFI_SETTINGS`

- Настройки режима:

`Settings.ACTION_AIRPLANE_MODE_SETTINGS`



```
startActivity(Intent(  
        Settings.ACTION_LOCATION_SOURCE_SETTINGS))  
}
```



# Пассивный провайдер

- ▶ не инициирует никакого явного мониторинга местоположения
  - LocationManager.PASSIVE\_PROVIDER
  - информация о местоположении идет, когда другой источник запрашивает его
  - провайдер идентифицируется методом Location.getProvider

```
locationManager = getSystemService(LOCATION_SERVICE) as LocationManager

demoListener = DemoLocation()

if (ActivityCompat.checkSelfPermission
    (this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
    ActivityCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
    return;
}

locationManager.requestLocationUpdates(LocationManager.PASSIVE_PROVIDER,
    500, 10f, demoListener)
```

## ► Основной поток приложения выполняет:

- реализации методов обратного вызова
- обработку сообщений, вычисление, форматирование и т.д.



Удаление обработки определения местоположения из главного потока

- Основной поток отвечает за обработку пользовательского интерфейса, общее управление, intents, services и т.д.
- Другой поток
  - Сохранение информации о местоположении в файловой системе или базе данных
  - сетевые вызовы передачи данных

# Фоновое отслеживание местоположения

- ▶ Мониторинг местоположения в фоновом режиме → Service
  - Если выполняется в activity, то останавливается после того, как пользователь выходит из приложения
    - ▶ Очистка activity и приложения, если не фокус пользователя
  - Сервис позволит выполнять в фоне без необходимости процесса activity

# Проблема установления разрешений

Начиная с версии 6.0  
FINE\_LOCATION рассматривается как  
опасное разрешение

```
"gps" location provider requires ACCESS_FINE_LOCATION permission
```

<https://developer.android.com/training/permissions/requesting.html>

# Установление разрешений до API 23

## ► Course location providers

- Низкой точности
- запрос ACCESS\_COURSE\_LOCATION разрешения

```
<uses-permission android:name="android.permission.ACCESS_COURSE_LOCATION">
```

## ► Fine location providers

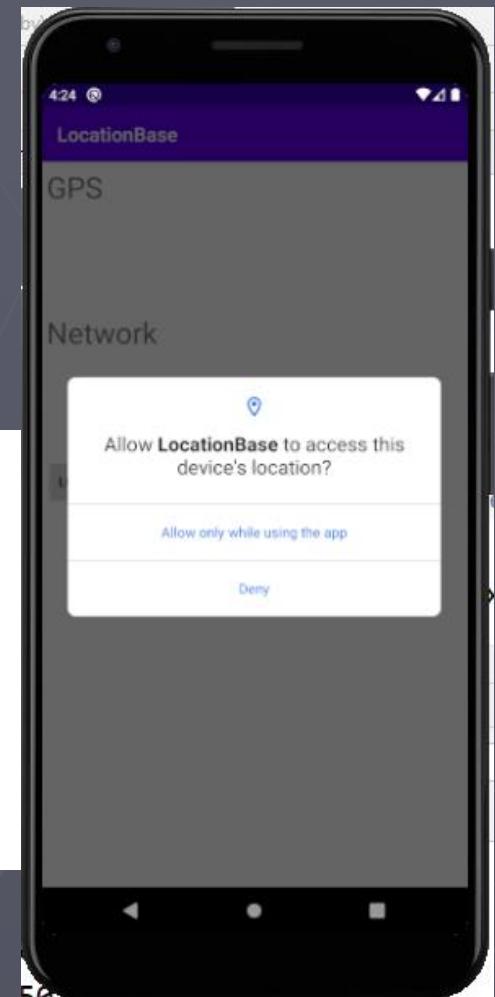
- высокоточный
- запрос ACCESS\_FINE\_LOCATION
- Подразумевает course location permission

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

# Установление разрешений после API 23

- 1) При инсталляции – диалог на разрешение(запоминается – в дальнейшем не выводится)

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
  
    ActivityCompat.requestPermissions(this,  
        arrayOf(Manifest.permission.ACCESS_FINE_LOCATION),  
        1);
```



# Формы представления location

- ▶ latitude и longitude
- ▶ Компоненты: city, state/province, street,
- ▶ Address class
  - getThouroughfare: улица
  - getLocality: город
  - getAdminArea: область
  - getMaxAddressLineIndex
  - getAddressLine
  - getLatitiude/getLongitude

# Geocoder класс

## ► Заполняет экземпляры Address

- `getFromLocation`

- Возвращает один или несколько экземпляров адреса для LAT / LNG координат

- `getFromLocationName`

- Возвращает один или несколько адресов экземпляров для указанного местоположения

- Оба метода возвращают `List <Address>`

- Количество адресов можно определить

```

class MainActivity : AppCompatActivity() {

    lateinit var locationManager: LocationManager
    lateinit var provider: LocationProvider
    lateinit var demoListener: LocationListener

    var tvEnabledGPS: TextView? = null
    var tvStatusGPS: TextView? = null
    var tvLocationGPS: TextView? = null
    var tvEnabledNet: TextView? = null
    var tvStatusNet: TextView? = null
    var tvLocationNet: TextView? = null
    var tvAddress: TextView? = null
    var context: Context = this

    var sbGPS = StringBuilder()
    var sbNet = StringBuilder()

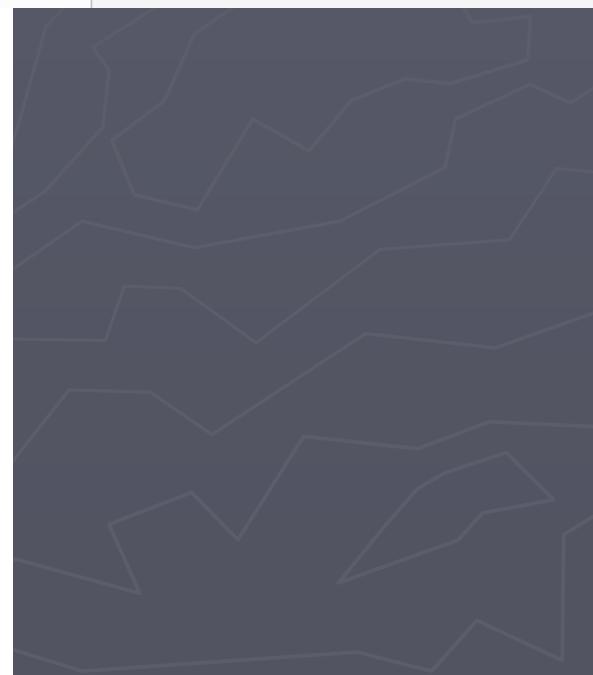
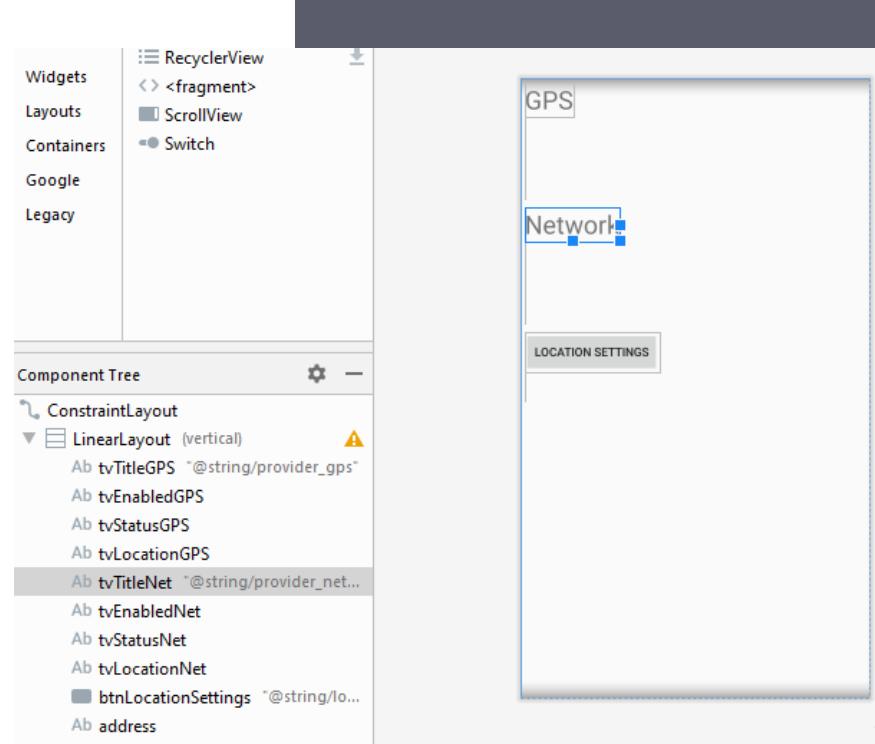
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        tvEnabledGPS = findViewById<View>(R.id.tvEnabledGPS) as TextView
        tvStatusGPS = findViewById<View>(R.id.tvStatusGPS) as TextView
        tvLocationGPS = findViewById<View>(R.id.tvLocationGPS) as TextView
        tvEnabledNet = findViewById<View>(R.id.tvEnabledNet) as TextView
        tvStatusNet = findViewById<View>(R.id.tvStatusNet) as TextView
        tvLocationNet = findViewById<View>(R.id.tvLocationNet) as TextView
        tvAddress = findViewById<View>(R.id.address) as TextView

        locationManager = getSystemService(LOCATION_SERVICE) as LocationManager

        if (ActivityCompat.checkSelfPermission(
                this, Manifest.permission.ACCESS_COARSE_LOCATION) !=
            PackageManager.PERMISSION_GRANTED &&
            ActivityCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_COARSE_LOCATION) !=
            PackageManager.PERMISSION_GRANTED) {
            return;
        }
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
            500, 10f, locationListener )
    }
}

```



```
private fun showLocation(location: Location?) {
    if (location == null) return
    if (location.provider == LocationManager.GPS_PROVIDER) {
        tvLocationGPS!!.text = formatLocation(location)
    } else if (location.provider ==
        LocationManager.NETWORK_PROVIDER) {
        tvLocationNet!!.text = formatLocation(location)
    }
}
private fun formatLocation(location: Location?): String? {
    return if (location == null) "" else String.format("Coordinates: lat = %1$.4f, lon = %2$.4f, time = %3$tF %3$tT",
        location.latitude, location.longitude, Date(location.time))
}
private fun checkEnabled() {
    tvEnabledGPS!!.text = "Enabled: " + locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)
    tvEnabledNet!!.text = "Enabled: " + locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER)
}

fun onClickLocationSettings(view: View?) {
    startActivity(Intent(
        Settings.ACTION_LOCATION_SOURCE_SETTINGS))
}
private val locationListener: LocationListener = object : LocationListener {
    override fun onLocationChanged(location: Location) {
        showLocation(location)
    }
    override fun onProviderDisabled(provider: String) {
        checkEnabled()
    }
    override fun onProviderEnabled(provider: String) {
        checkEnabled()

        if (ActivityCompat.checkSelfPermission(context, Manifest.permission.ACCESS_FINE_LOCATION)
            != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(context,
Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
            return
        }
        showLocation(locationManager.getLastKnownLocation(provider))
    }

    override fun onStatusChanged(provider: String, status: Int, extras: Bundle) {
        if (provider == LocationManager.GPS_PROVIDER) {
            tvStatusGPS!!.text = "Status:$status"
        } else if (provider == LocationManager.NETWORK_PROVIDER) {
            tvStatusNet!!.text = "Status: $status"
        }
    }
}
```

```

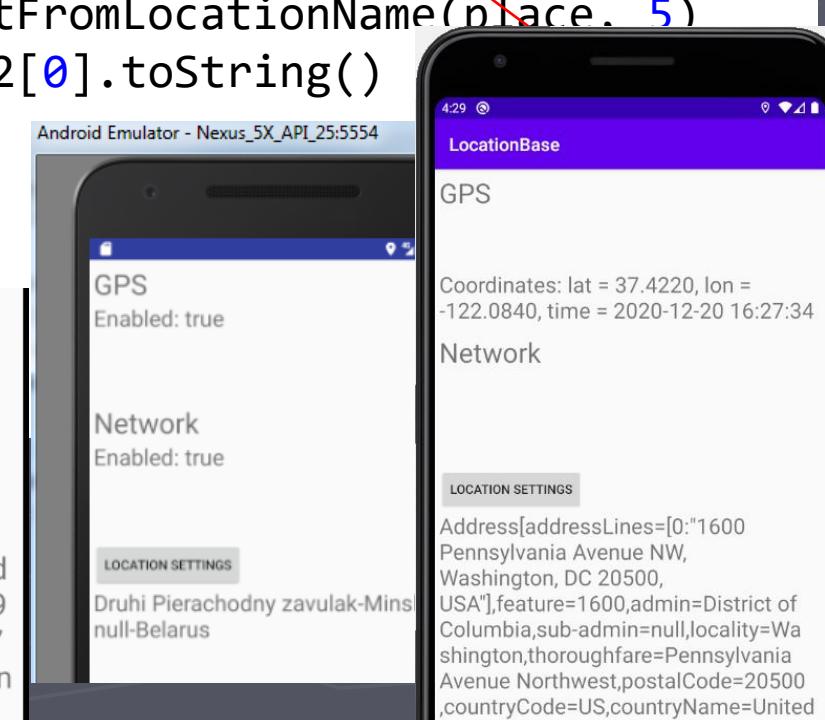
val geocoder = Geocoder(context)
try {
    val addrList1 = geocoder.getFromLocation(53.8872, 27.6524, 5)
    if (addrList1!!.size > 0 && addrList1 != null) {
        val result = java.lang.StringBuilder()

        tvAddress!!.text = addrList1[0].featureName + " - "
            + addrList1[0].locality + " - "
            + addrList1[0].adminArea + " - "
            + addrList1[0].countryName
        val place = "1600 Pennsylvania NW Ave, Washington, DC"
        val addrList2 = geocoder.getFromLocationName(place, 5)
        tvAddress!!.text = addrList2[0].toString()
    }
} catch (e: IOException) {
    e.printStackTrace()
}
}

```

Address[addressLines=[0:"1600 Pennsylvania Avenue NW, Washington, DC 20500, USA"],feature=1600,admin=District of Columbia,sub-admin=null,locality=Washington,thoroughfare=Pennsylvania Avenue Northwest,postalCode=20500,countryCode=US,countryName=United States,hasLatitude=true,latitude=38.8976633,hasLongitude=true,longitude=-77.0365739,phone=null,url=null,extras=null]

КОЛИЧЕСТВО



# Geocoder - зависит от Google библиотек и сервисов

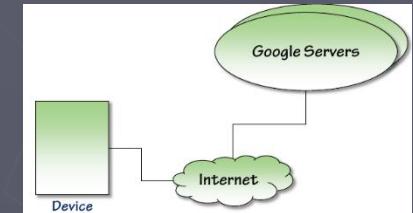
- Может быть недоступен на некоторых устройствах
- Основан на веб-сервис
- `getFromLocation` и `getFromLocationName` обращается к удаленным серверам

Google библиотеки доступны?

```
if(Geocoder.isPresent()) {  
    try {  
        val geocoder = Geocoder(context)  
        val addrList = geocoder.getFromLocationName(...)  
        if(addrList != null && addrList.size() > 0)  
            // ok  
    }  
    catch(IOException e) {  
        // error  
    }  
}
```

адреса не соответствуют запросу  
Ошибка взаимодействия

библиотеки недоступны  
или ошибки при взаимодействии  
сервера



# Google Play Services

► набор стандартных сервисных функций, устанавливаемых вместе с приложением Google Play

- сервис позиционирования Fused Location Provider
- Maps API

Google APIs - необходимо установить или обновить

# История

## ► Maps V1

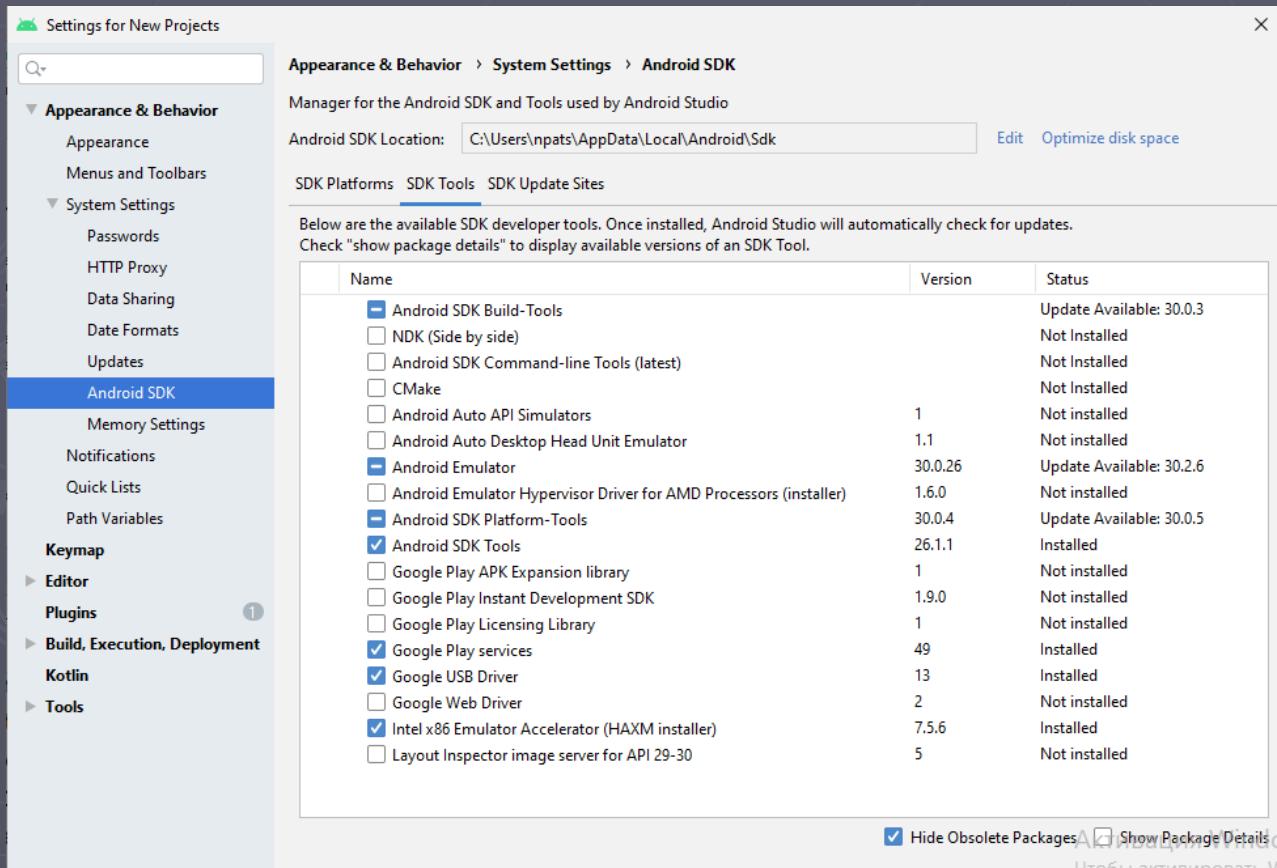
- На заре ->Maps SDK встроенный (до 2009)
- Позволял загружать картографические библиотеки в приложение
- MapView виджет

## ► Maps V2

- Android 3.0 (2011)
- Добавили фрагменты для управления

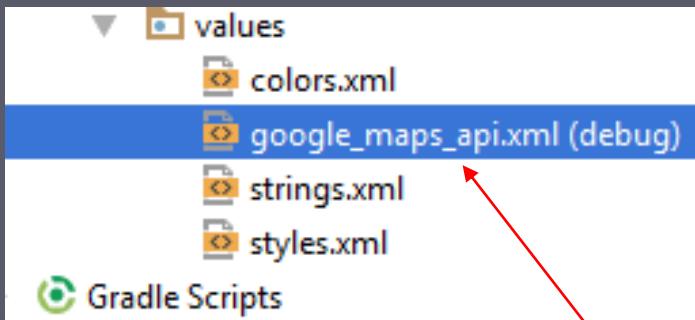
# Maps и Location

- 1) устройство с установленным приложением Play Store (ассоциирован с Google Services Framework и OpenGL ES 2.0+)
- 2) эмулятор с Google APIs.

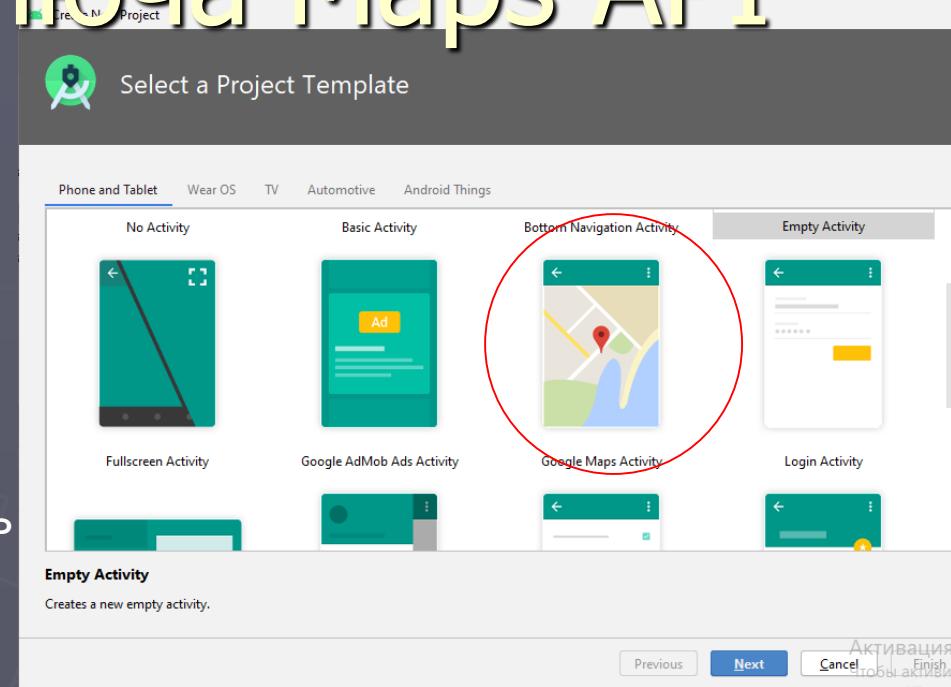


# Получение ключа Maps API

Новое приложение



позволит приложению использовать картографический сервис Google.



<resources>

<!--

*TODO: Before you run your application, you need a Google Maps API key.*

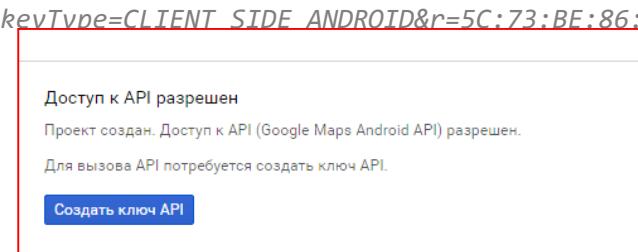
To get one, follow this link, follow the directions and press "Create" at the end:

[https://console.developers.google.com/flows/enableapi?apiid=maps\\_android\\_backend&keyType=CLIENT\\_SIDE\\_ANDROID&r=5C:73:BE:86:EC:39:7A:FE:60:8B:17:BA:06:B0:02:28:18:C1:BB%3Bby.bstu.patsei.mapapp](https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=5C:73:BE:86:EC:39:7A:FE:60:8B:17:BA:06:B0:02:28:18:C1:BB%3Bby.bstu.patsei.mapapp)

You can also add your credentials to an existing key, using these values:

Package name:  
by.bstu.patsei.mapapp

SHA-1 certificate fingerprint:  
5C:73:BE:86:4F:EC:39:7A:FE:60:8B:17:BA:06:B0:02:28:18:C1:BB



## ► Получение ключа

## ► Отладочный ключ

\\$USER\.android\debug.keystore

## ► Java SDK → keytool

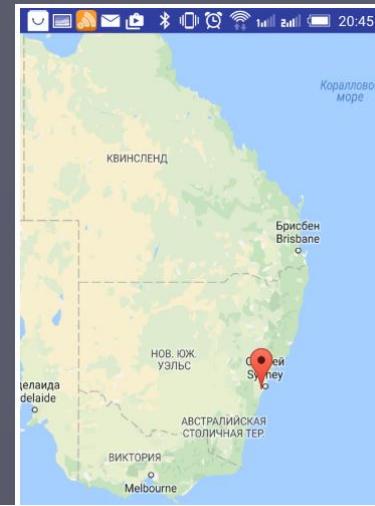
```
keytool -list -v -keystore ... -alias androiddebugkey -storepass android -keypass android
```

Certificate fingerprints:

MD5: 98:84:0E:36:F0:B3:48:9C:CD:13:EB:C6:D8:7F:F3:B1

SHA1: E6:C5:B1:EB:8A:F4:35:B0:04:84:3E:6E:C3:88:BD:B2:66:52:E7:09

Signature algorithm name: SHA1withRSA



```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".yMapsActivit" />
```

Специального вида фрагмент на основе  
com.google.android.gms.maps.SupportMapFragment

```
class MapsActivity : AppCompatActivity(), OnMapReadyCallback {  
  
    private lateinit var mMap: GoogleMap  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_maps)  
        // Obtain the SupportMapFragment and get notified when the map is ready to be used  
        val mapFragment = supportFragmentManager  
            .findFragmentById(R.id.map) as SupportMapFragment  
        mapFragment.getMapAsync(this)  
    }  
}
```

Загружает карты асинхронно

```
override fun onMapReady(googleMap: GoogleMap) {  
    mMap = googleMap
```

Доставляет в дальнейшем  
GoogleMap объекты

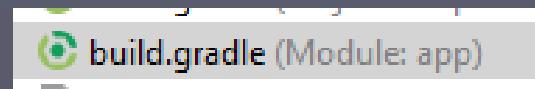
```
// Add a marker in Sydney and move the camera  
val sydney = LatLng(-34.0, 151.0)  
mMap.addMarker(MarkerOptions().position(sydney).title("Marker in Sydney"))  
mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney))  
}
```

Инкапсулирует координаты

камера - ссылка на позицию  
виртуальных глаз пользователя по  
отношению к поверхности Земли

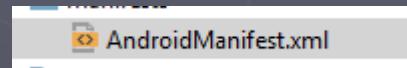
# ► Добавление к существующему приложению

- Добавить зависимости



```
dependencies {  
    implementation fileTree(dir: "libs", include: ["*.jar"])  
    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"  
    implementation 'androidx.core:core-ktx:1.3.2'  
    implementation 'androidx.appcompat:appcompat:1.2.0'  
    implementation 'com.google.android.gms:play-services-maps:17.0.0'  
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
```

- Разрешения



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

- КЛЮЧ

```
<meta-data  
    android:name="com.google.android.geo.API_KEY"  
    android:value="
```

```
//      mMap.addMarker(MarkerOptions().position(sydney).title("Marker in Sydney"))
//      mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney))

    val minsk = LatLng(0.0, 0.0)
    mMap.addMarker( MarkerOptions().position(minsk).title("Zero"))
    mMap.moveCamera(CameraUpdateFactory.newLatLng(minsk))
    mMap.moveCamera(CameraUpdateFactory.)
```

Update newLatLng (LatLng latLng)

Update that moves the center of the screen to a latitude and longitude defined by a LatLng object. This centers the camera on the LatLng

object containing the desired latitude and longitude.

containing the transformation.

google.android.gms:play-services-maps:17.0.0@aar

(ng) on localhost ↗

|   |                       |
|---|-----------------------|
| newLatLng(LatLng!)                            | CameraUpdate!         |
| newCameraPosition(CameraPosition!)            | CameraUpdate!         |
| newLatLngBounds(LatLngBounds!, Int)           | CameraUpdate!         |
| newLatLngBounds(LatLngBounds!, Int, Int, ...) | CameraUpdate!         |
| newLatLngZoom(LatLng!, Float)                 | CameraUpdate!         |
| scrollBy(Float, Float)                        | CameraUpdate!         |
| zoomBy(Float)                                 | CameraUpdate!         |
| zoomBy(Float, Point!)                         | CameraUpdate!         |
| zoomIn()                                      | CameraUpdate!         |
| zoomOut()                                     | CameraUpdate!         |
| zoomTo(Float)                                 | CameraUpdate!         |
| zza(ICameraUpdateFactoryDelegate!)            | Unit                  |
| val   | val name = expression |
| arrayOf                                       | arrayOf(expr)         |
| listOf  | listOf(expr)          |

mMap.moveCamera(CameraUpdateFactory.zoomTo(12));

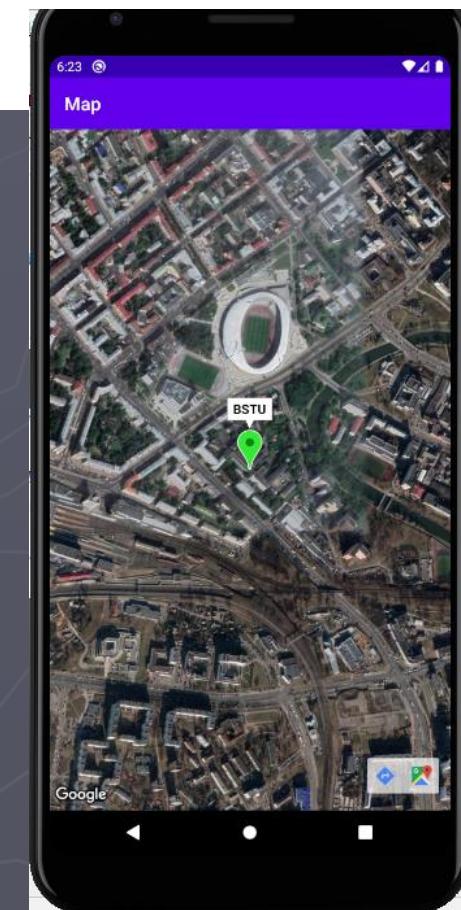
Число от 2 до 21 - определяет коэффициент масштабирования

```
override fun onMapReady(googleMap: GoogleMap) {  
    mMap = googleMap  
  
    mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);  
    mMap.addMarker(MarkerOptions().position(LatLng(53.891442, 27.559891))  
        .title("BSTU").icon(BitmapDescriptorFactory  
        .defaultMarker(BitmapDescriptorFactory.HUE_GREEN)));  
  
    mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(LatLng(53.891442,  
27.559891), 20F))  
}
```

ТИП

Добавить маркер, заголовок, цвет

плавно анимирует карту от своего  
первоначального состояния в новое  
состояние



# Добавление информационных окон

- ▶ 1) создать разметку для всплывающего окна

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout <LinearLayout xmlns:android="http://schemas.android.com/apk
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="horizontal"> >
<ImageView <ImageView
    android:id="@+id/icon"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_vertical"
    android:padding="2dip"
    android:src="@drawable/ic_launcher"
    android:contentDescription="@string/icon"/> />
<LinearLayout <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"> >
    <TextView <TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="25sp"
        android:textStyle="bold"/> />
    <TextView <TextView
        android:id="@+id/snippet"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```



## ► 2) InfoWindowAdapter реализаци

```
class PopupAdapter: GoogleMap.InfoWindowAdapter {  
    lateinit var popup: View  
    lateinit var inflater: LayoutInflater  
  
    constructor(inflater: LayoutInflater) {  
        this.inflater = inflater  
    }  
    override fun getInfoWindow(p0: Marker?): View {  
        TODO("Not yet implemented")  
    }  
  
    override fun getInfoContents(p0: Marker?): View {  
        if (popup == null) {  
            popup = inflater.inflate(R.layout.popup, null)  
        }  
        var tv = popup.findViewById<TextView>(R.id.title)  
        tv.text = p0?.title;  
        tv = popup.findViewById(R.id.snippet)  
        tv.setText(p0?.snippet)  
        return(popup)  
    }  
}
```

- 1) Сначала ОС вызывает `getInfoWindow()` передавая маркер – если возврат `View` - он будет использоваться для информационного окна, если `null` - Android будет вызывать `getInfoContents()`. `getInfoContents()` – если возвращать `View`, Android будет использовать его как "тело" информационного окна, если `null` - окно отобразится по умолчанию
- 2) Можно - заменить окно, замените содержимое, или примите значение по умолчанию
- 3) Можно - заменить окно, замените содержимое, или примите значение по умолчанию

- ▶ 3) сказать Google Map использовать InfoWindowAdapter
- ▶ Можно использовать setOnInfoWindowClickListener () на GoogleMap если необходимо обрабатывать нажатия на инфо окне.

```
mMap.setOnInfoWindowClickListener(this);  
  
}  
  
override fun onInfoWindowClick(p0: Marker?) {  
    Toast.makeText(this, p0.getTitle(), Toast.LENGTH_LONG).show()  
}  
}
```

# Обработка taps на маркере

- ▶ 1) создать реализацию OnMarkerClickListener интерфейса и добавить его к GoogleMap с помощью setOnMarkerClickListener ().
- ▶ 2) вызывается - onMarkerClick (), когда пользователь нажимает на маркер, и нужно передать объект маркера

```
mMap.setOnMarkerClickListener(this);
```

```
}
```

```
override fun onMarkerClick(p0: Marker?): Boolean {  
    Toast.makeText(this, p0?.getTitle(), Toast.LENGTH_LONG).show()  
    return (false)  
}
```

# Поддержка drag-and-drop

- ▶ 1) отключена по умолчанию
- ▶ 2) для подключения автоматической поддержки :
  - вызов draggable(true) на MarkerOptions при создании

```
val lat = 53.891442
val lon = 27.559891
mMap.addMarker(MarkerOptions().position(LatLng(lat, lon))
    .title(getString(title))
    .snippet(getString(snippet))
    .draggable(true));
```

- или setDraggable(true) на марке позже

```
mMap.setOnMarkerDragListener(this);
```

```
mapFrag.setRetainInstance(true);
```

сохранение в  
экземпляре фрагмента  
новые позиции - не  
сбрасывать их в  
исходное положение.

```
@Override
public void onMarkerDragStart(Marker marker) {
    LatLng position=marker.getPosition();
    Log.d(getClass().getSimpleName(), String.format("Drag from %f:%f",
        position.latitude,
        position.longitude));
}

@Override
public void onMarkerDrag(Marker marker) {
    LatLng position=marker.getPosition();
    Log.d(getClass().getSimpleName(),
        String.format("Dragging to %f:%f", position.latitude,
        position.longitude));
}

@Override
public void onMarkerDragEnd(Marker marker) {
    LatLng position=marker.getPosition();
    Log.d(getClass().getSimpleName(), String.format("Dragged to %f:%f",
        position.latitude,
        position.longitude));
}
```

# Возможности

- ▶ 1) создать свой класс маркера и добавить его в модели
- ▶ 2) рисовать линии и области
  - polyline соединяющую все маркеры
  - polygon с указанными маркерами
- ▶ 3) Менять управление на карте
  - уровень масштабирования либо кнопки + и -
  - изменить центр карты
  - изменять угол наклона камеры вместо традиционной точки зрения сверху вниз
  - изменить ориентацию карты
- ▶ 4) Выполнять tracking изменений камеры
  - setOnCameraChangeListener () на GoogleMap,
  - Реализацию OnCameraChangeListener, который будет называться с onCameraChange ()
- 5) Анимация
- 6) Helper Libraries и сторонних карт

# Пример Java

```
public class MapsActivity extends FragmentActivity implements LocationListener {  
  
    private GoogleMap mMap; // Might be null if Google Play services APK is not available.  
  
    LocationManager locationManager;  
    String provider;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_maps);  
        setUpMapIfNeeded();  
  
        locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);  
  
        provider = locationManager.getBestProvider(new Criteria(), false);  
  
        Location location = locationManager.getLastKnownLocation(provider);  
  
        if (location != null) {  
  
            onLocationChanged(location);  
  
        }  
  
    }  
}
```

```
@Override
public void onLocationChanged(Location location) {

    Double lat = location.getLatitude();
    Double lng = location.getLongitude();

    mMap.clear();
    mMap.addMarker(new MarkerOptions().position(new LatLng(lat, lng)).title("Your location"));
    mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(new LatLng(lat, lng), 12));

    Geocoder geocoder = new Geocoder(getApplicationContext(), Locale.getDefault());

    try {
        List<Address> listAddresses = geocoder.getFromLocation(lat, lng, 1);
        if (listAddresses != null && listAddresses.size() > 0) {
            Log.i("PlaceInfo", listAddresses.get(0).toString());
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

@Override
protected void onPause() {
    super.onPause();
    locationManager.removeUpdates(this);
}
```

```
@Override
protected void onResume() {
    super.onResume();
    setUpMapIfNeeded();

    locationManager.requestLocationUpdates(provider, 400, 1, this);
}

private void setUpMapIfNeeded() {
    // Do a null check to confirm that we have not already instantiated the map.
    if (mMap == null) {
        // Try to obtain the map from the SupportMapFragment.
        mMap = ((SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map))
            .getMap();
        // Check if we were successful in obtaining the map.
        if (mMap != null) {
            setUpMap();
        }
    }
}
```

► <https://developers.google.com/maps/documentation/geolocation/intro?hl=ru>

The screenshot shows a web page from the Google Developers site. On the left, there's a sidebar with navigation links for 'Best Practices', 'Policies and Terms' (including Usage and Billing, Policies, and Terms of Service), and 'Other APIs' (Directions API, Distance Matrix API, Elevation API, Geocoding API, Places API, Roads API, and Time Zone API). The 'Places API' link is highlighted with a light gray background. The main content area has a title 'Geolocation requests' and a sub-section 'Geolocation requests are sent using POST to the following URL:' followed by a code snippet: `https://www.googleapis.com/geolocation/v1/geolocate?key=YOUR_API_KEY`. Below this, there's a note about specifying a key in the request. The bottom section is titled 'Request body' with a note about the request body being JSON and supported fields, including 'homeMobileCountryCode'. The footer of the page is visible at the very bottom.

Best Practices

Policies and Terms

Usage and Billing

Policies

Terms of Service

Other APIs

Directions API

Distance Matrix API

Elevation API

Geocoding API

Places API

Roads API

Time Zone API

## Geolocation requests

Geolocation requests are sent using POST to the following URL:

```
https://www.googleapis.com/geolocation/v1/geolocate?key=YOUR_API_KEY
```

You must specify a key in your request, included as the value of a `key` parameter. This key identifies your application for purposes of quota management. L

### Request body

The request body must be formatted as JSON. The following fields are supported,

- `homeMobileCountryCode` : The mobile country code (MCC) for the device's