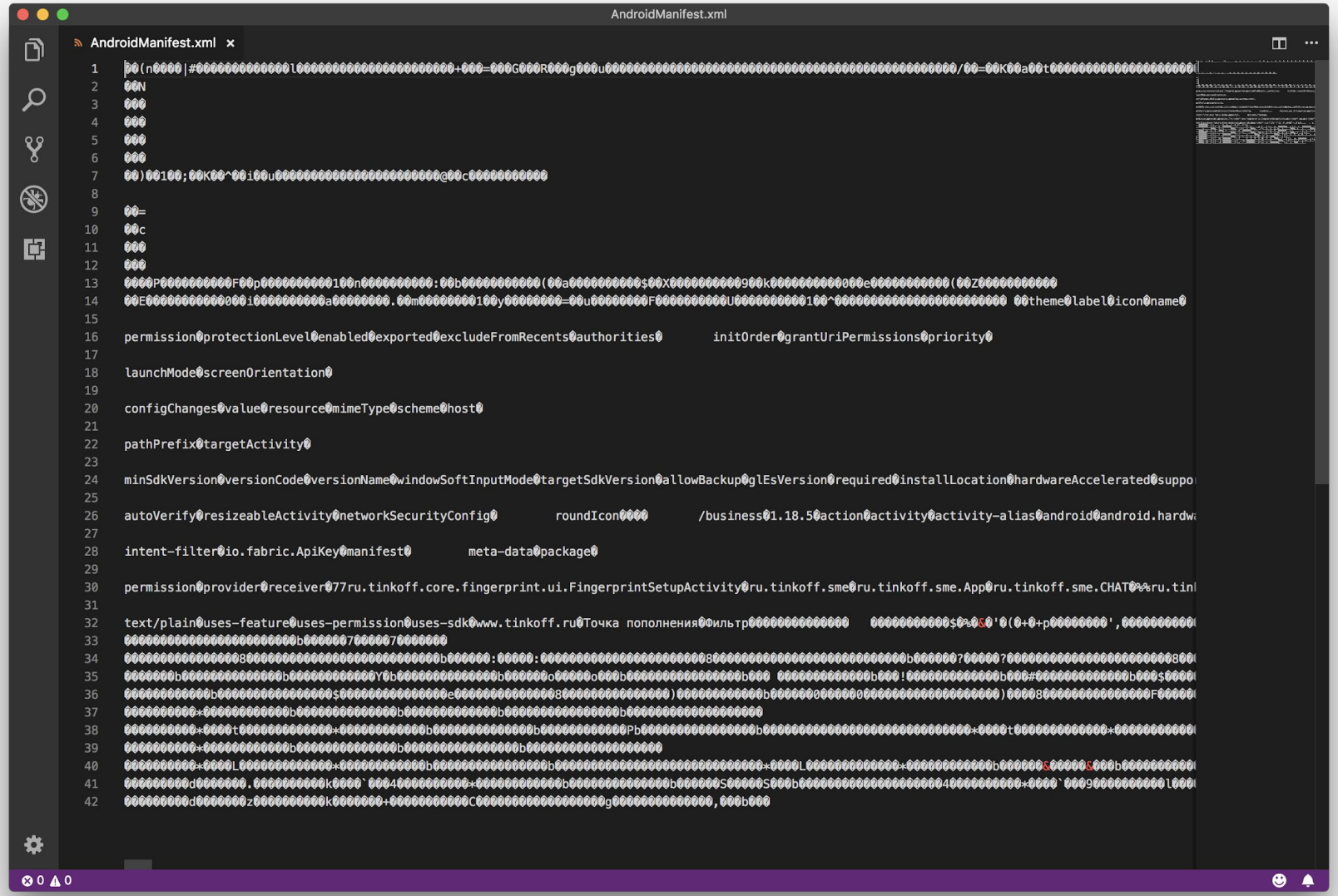
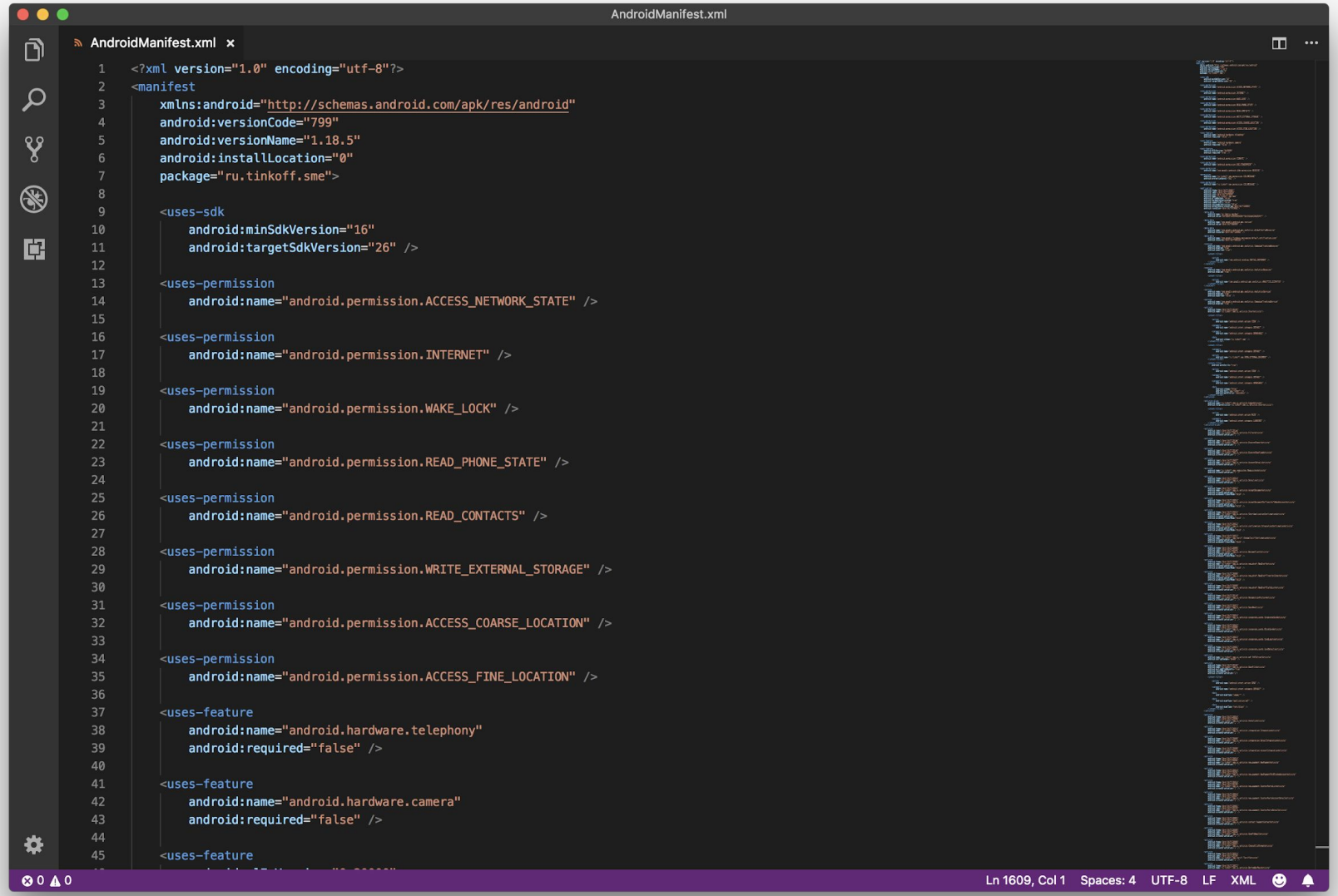
Основные компоненты

План

- Что внутри apk?
- AndroidManifest.xml
- Activity
- BroadcastReceiver
- Service

Что внутри apk?


AndroidManifest.xml
AndroidManifest.xml
permission@protectionLevel@enabled@exported@excludeFromRecents@authorities@ initOrder@grantUriPermissions@priority@
launchMode@screenOrientation@
configChanges@value@resource@mimeType@scheme@host@
pathPrefix@targetActivity@
minSdkVersion@versionCode@versionName@windowSoftInputMode@targetSdkVersion@allowBackup@glEsVersion@required@installLocation@hardwareAccelerated@suppo
autoVerify@resizeableActivity@networkSecurityConfig@ roundIcon@@@@ /business@1.18.5@action@activity@activity-alias@android@android.hardw
intent-filter@io.fabric.ApiKey@manifest@ meta-data@package@
permission@provider@receiver@77ru.tinkoff.core.fingerprint.ui.FingerprintSetupActivity@ru.tinkoff.sme@ru.tinkoff.sme.App@ru.tinkoff.sme.CHAT@%%ru.tin
text/plain@uses-feature@uses-permission@uses-sdk@www.tinkoff.ru@Точка пополнения@@ильтр@@@@@@@@@@@@@@ @@@@@@@@@@@$@%&@'@(@+@+p@@@@@@@@',@@@@@@@@@@
0 0

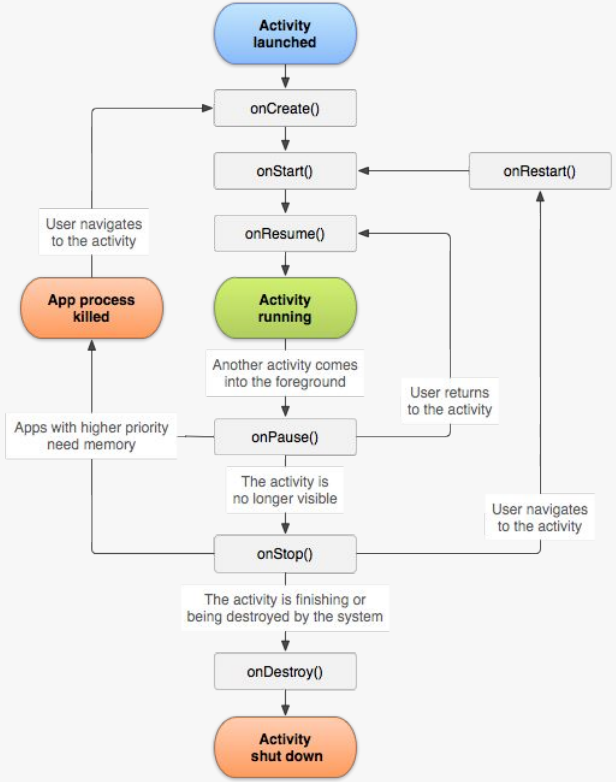

AndroidManifest.xml
AndroidManifest.xml
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
android:versionCode="799"
android:versionName="1.18.5"
android:installLocation="0"
package="ru.tinkoff.sme">
<uses-sdk
android:minSdkVersion="16"
android:targetSdkVersion="26" />
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission
android:name="android.permission.INTERNET" />
<uses-permission
android:name="android.permission.WAKE_LOCK" />
<uses-permission
android:name="android.permission.READ_PHONE_STATE" />
<uses-permission
android:name="android.permission.READ_CONTACTS" />
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-feature
android:name="android.hardware.telephony"
android:required="false" />
<uses-feature
android:name="android.hardware.camera"
android:required="false" />
<uses-feature
0 0
Ln 1609, Col 1
Spaces: 4
UTF-8
LF
XML

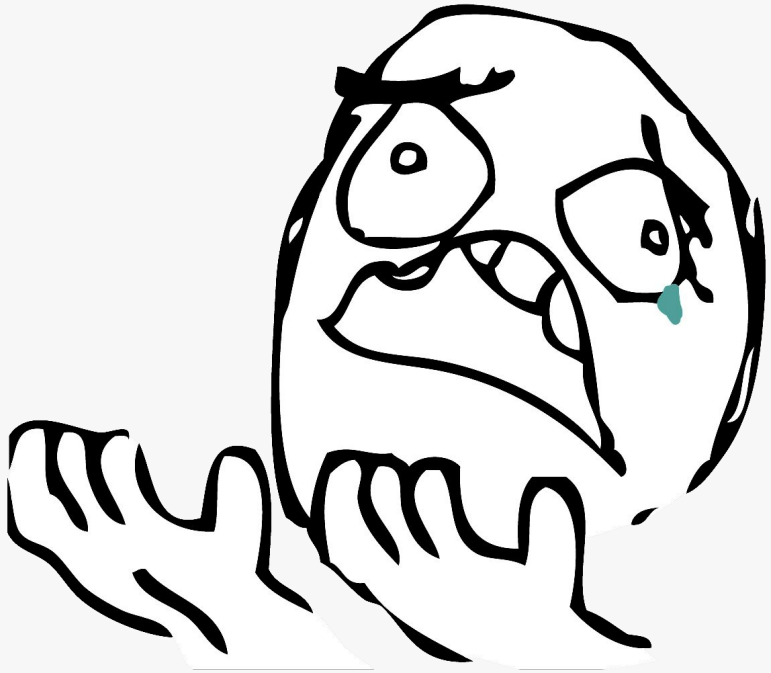
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="string"
          android:versionCode="integer"
          android:versionName="string" >
    . . .
</manifest>
```

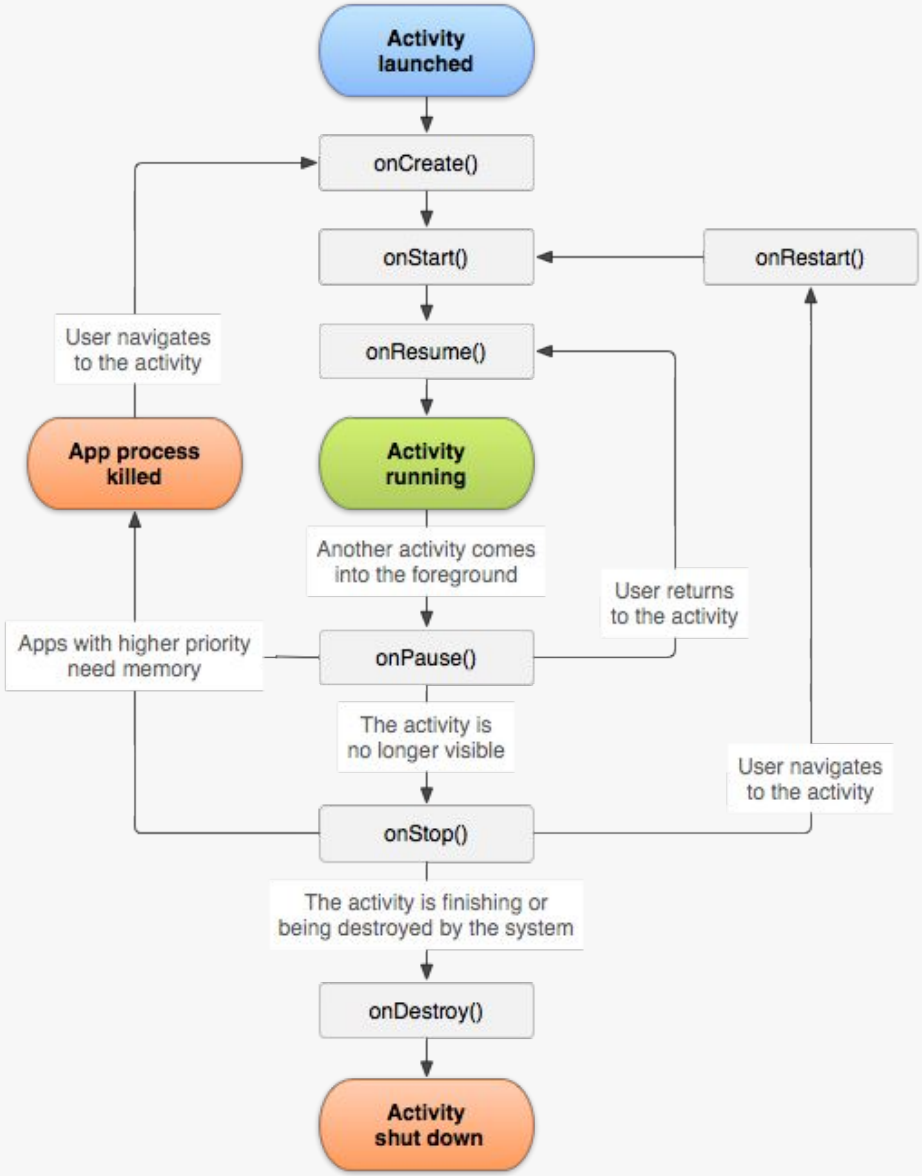
```
<application android:icon="drawable resource"
             android:name="string"
             android:process="string"
             android:theme="resource or theme" >
    . . .
</application>
```

```
class App : Application() {

    override fun onCreate() {
        super.onCreate()
    }

    override fun onLowMemory() {
        super.onLowMemory()
    }
}
```

Activity


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down




Activity launched
onCreate()
onStart()
onRestart()
onResume()
Activity running
User navigates to the activity
App process killed
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Сворачиваем приложение?

Сворачиваем приложение?

- onPause
- onStop
- onDestory?

Разворачиваем приложение?

Разворачиваем приложение?

- onCreate?
- onRestart
- onStart
- onResume

Поворот экрана?

Поворот экрана?

- onPause
- onStop
- onDestroy
- onCreate
- onStart
- onResume

A startActivity B?

A startActivity B?

- A#onPause
- B#onCreate
- B#onStart
- B#onResume
- A#onStop

A startActivity B (transparent)?

A startActivity B (transparent)?

- A#onPause
- B#onCreate
- B#onStart
- B#onResume

A startActivity B (transparent) и поворот?

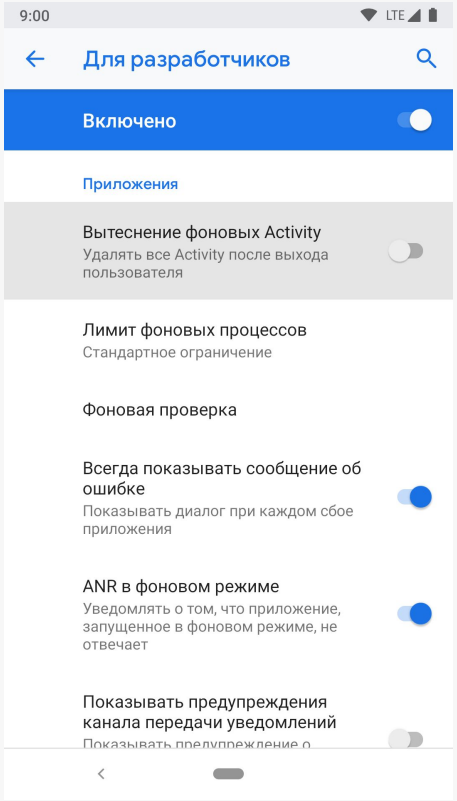
A startActivity B (transparent) и поворот?

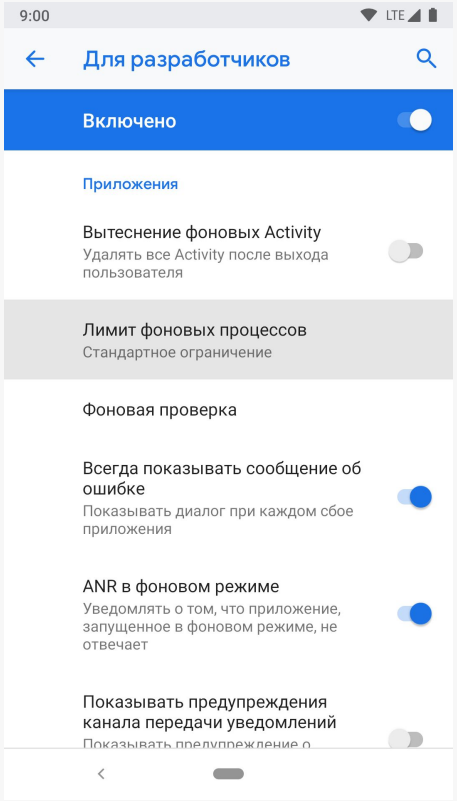
- A#onPause, B#onCreate, B#onStart, B#onResume

- B#onPause, B#onStop, B#onDestroy
- B#onCreate, B#onStart, B#onResume

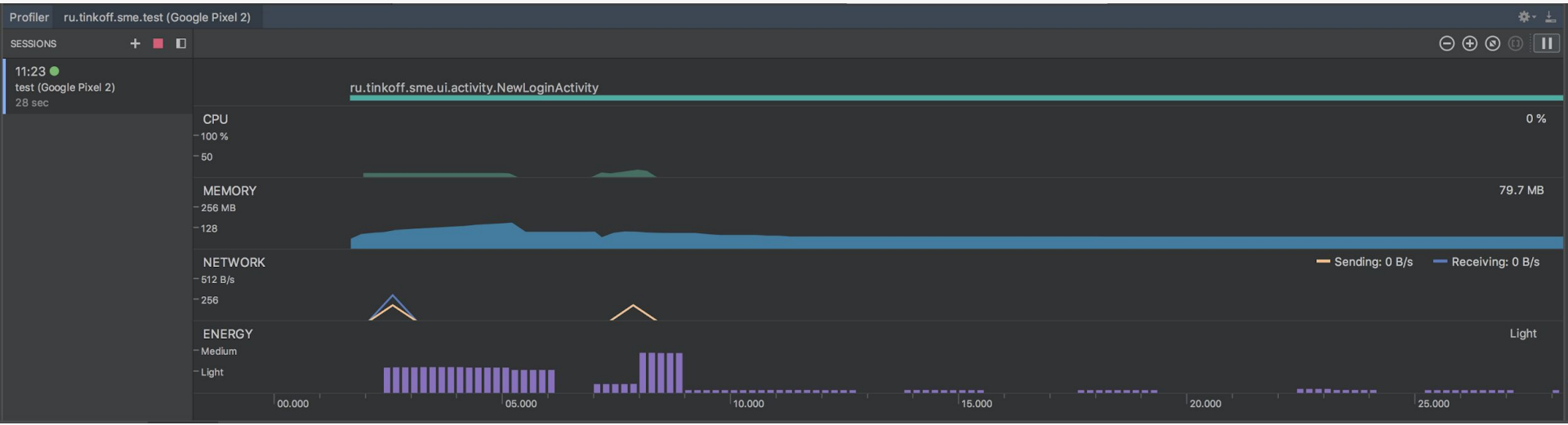
- A#onStop, A#onDestroy
- A#onCreate, A#onStart, A#onResume, A#onPause

Как дебажить?





Как дебажить?



Как сохранять объекты?

Как сохранять объекты?

- `onSaveInstanceState / onCreate / onRestoreInstanceState`

```kotlin
class MainActivity : AppCompatActivity() {

    private var userId: Int = 0

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

        if (savedInstanceState != null) {
            userId = savedInstanceState.getInt(EXTRA_USER_ID)
        }
    }

    override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        outState.putInt(EXTRA_USER_ID, userId)
    }
}
```

Как сохранять объекты?

- `onSaveInstanceState / onCreate / onRestoreInstanceState`
- `onRetainNonConfigurationInstance / getLastNonConfigurationInstance`

Как сохранять объекты?

- onSaveInstanceState / onCreate / onRestoreInstanceState
- onRetainNonConfigurationInstance / getLastNonConfigurationInstance
- onRetainCustomNonConfigurationInstance / getLastCustomNonConfigurationInstance

```java
@Override
public final Object onRetainNonConfigurationInstance() {
    Object custom = onRetainCustomNonConfigurationInstance();

    FragmentManagerNonConfig fragments = mFragments.retainNestedNonConfig();

    if (fragments == null && mViewModelStore == null && custom == null) {
        return null;
    }

    NonConfigurationInstances nci = new NonConfigurationInstances();
    nci.custom = custom;
    nci.viewModelStore = mViewModelStore;
    nci.fragments = fragments;
    return nci;
}
```

```java
public class FragmentManagerNonConfig {
    private final List<Fragment> mFragments;
    private final List<FragmentManagerNonConfig> mChildNonConfigs;
    private final List<ViewModelStore> mViewModelStores;
}

public class ViewModelStore {
    private final HashMap<String, ViewModel> mMap = new HashMap<>();
}

static final class NonConfigurationInstances {
    Object custom;
    ViewModelStore viewModelStore;
    FragmentManagerNonConfig fragments;
}
```

Как еще сохранять объекты?

Как еще сохранять объекты?

- Retain Fragment / ViewModel
- Статическое поле
- Singleton
- Файлы или база данных

`startActivity(Intent intent)`

- Action
- Data

```
val intent = Intent()
intent.action = Intent.ACTION_VIEW
intent.data = Uri.parse("tel:88005557778")
context.startActivity(intent)
```

```
val intent = Intent()
intent.action = Intent.ACTION_VIEW
intent.data = Uri.parse("https://www.tinkoff.ru/")
context.startActivity(intent)
```

`startActivity(Intent intent)`

- Action
- Data
- Category

```
val intent = Intent()
intent.action = Intent.ACTION_MAIN
intent.addCategory(Intent.CATEGORY_HOME)
context.startActivity(intent)
```

`startActivity(Intent intent)`

- Action
- Data
- Category
- Type

`startActivity(Intent intent)`

- Action
- Data
- Category
- Type
- Extras

`startActivity(Intent intent)`

- Action
- Data
- Category
- Type
- Extras
- Component

```
val intent = Intent(context, SecondActivity::class.java)
intent.putExtra(EXTRA_ID, id)
context.startActivity(intent)
```

```
val intent = Intent()
intent.component = ComponentName(context.packageName, SecondActivity::class.java.name)
intent.putExtra(EXTRA_ID, id)
context.startActivity(intent)
```

Как общаться двум Activity?

Как общаться двум Activity?

- `startActivityForResult(...)`

```kotlin
fun startSecondActivity() {
    val intent = Intent(context, SecondActivity::class.java)
    intent.putExtra(EXTRA_ID, id)
    activity.startActivityForResult(intent, MY_REQUEST_CODE)
}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)

    if (requestCode == MY_REQUEST_CODE) {
        // TODO
    }
}
```

```
val data = Intent()
data.putExtra(EXTRA_SOME_INFO, INFO)
setResult(Activity.RESULT_OK, data)
finish()
```

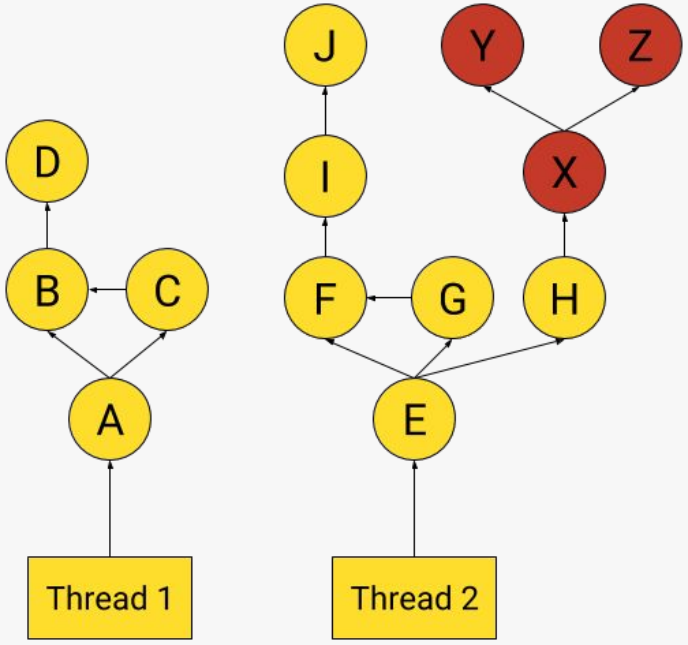
Как общаться двум Activity?

- `startActivityForResult(...)`
- `Статическое поле`
- `Singleton`
- `Файлы или база данных`

Application context vs Activity context

- Время жизни
- Layout Inflation
- startActivity with FLAG_ACTIVITY_NEW_TASK

Application context vs Activity context



Memory leaks

- Передача this куда попало

```kotlin
class MainActivity : AppCompatActivity() {

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

        TheCoolestLibEver.init(this)
    }
}
```

Memory leaks

- Передача this куда попало
- Интерфейсы

```kotlin
class MainActivity : AppCompatActivity(), Runnable {

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

        TheCoolestLibEver.addCallback(this)
    }

    override fun run() {
        // TODO
    }
}
```

Memory leaks

- Передача this куда попало
- Интерфейсы
- Внутренний (inner) классы, анонимные классы, лямбды

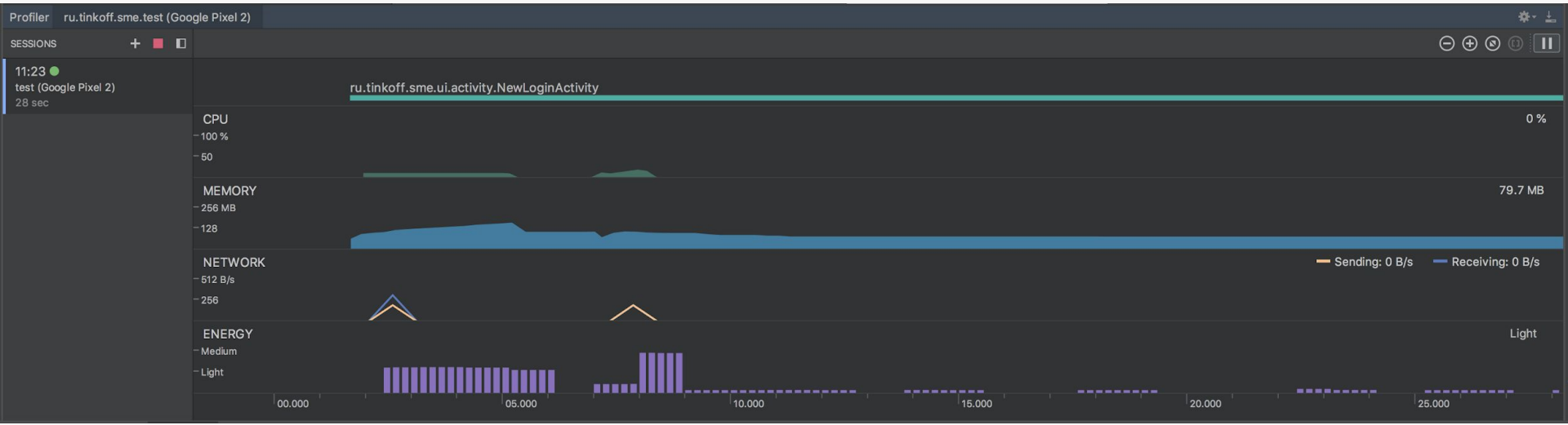
```java
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        TheCoolestLibEver.addCallback(new Runnable() {
            @Override
            public void run() {
                // TODO
            }
        });
    }
}
```

```java
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        TheCoolestLibEver.addCallback(new Callback());
    }

    private class Callback implements Runnable {

        @Override
        public void run() {
            // TODO
        }
    }
}
```

```java
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        TheCoolestLibEver.addCallback(() -> {
            // TODO
        });
    }
}
```

```java
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        TheCoolestLibEver.addCallback(new Callback());
    }

    private static class Callback implements Runnable {

        @Override
        public void run() {
            // TODO
        }
    }
}
```

Как дебажить?



Как дебажить?

```
StrictMode.setVmPolicy(StrictMode.VmPolicy.Builder()
        .detectActivityLeaks()
        .penaltyDeath()
        .build())
```

```
registerActivityLifecycleCallbacks(object : ActivityLifecycleCallbacks {

    override fun onActivityCreated(activity: Activity, savedInstanceState: Bundle?)

    override fun onActivityStarted(activity: Activity)

    override fun onActivityResumed(activity: Activity)

    override fun onActivityPaused(activity: Activity)

    override fun onActivityStopped(activity: Activity)

    override fun onActivitySaveInstanceState(activity: Activity, outState: Bundle)

    override fun onActivityDestroyed(activity: Activity)
})
```

```
<activity android:configChanges=["mcc", "mnc", "locale",
                                 "touchscreen", "keyboard", "keyboardHidden",
                                 "navigation", "screenLayout", "fontScale",
                                 "uiMode", "orientation", "density",
                                 "screenSize", "smallestScreenSize"]
          android:exported=["true" | "false"]
          android:name="string"
          android:screenOrientation=["unspecified" | "behind" |
                                     "landscape" | "portrait" |
                                     "reverseLandscape" | "reversePortrait" |
                                     "sensorLandscape" | "sensorPortrait" |
                                     "userLandscape" | "userPortrait" |
                                     "sensor" | "fullSensor" | "nosensor" |
                                     "user" | "fullUser" | "locked"]
          android:theme="resource or theme" >
```

```kotlin
override fun onConfigurationChanged(newConfig: Configuration) {
    super.onConfigurationChanged(newConfig)
    when (newConfig.orientation) {
        Configuration.ORIENTATION_UNDEFINED -> {
            // TODO
        }

        Configuration.ORIENTATION_LANDSCAPE -> {
            // TODO
        }

        Configuration.ORIENTATION_PORTRAIT -> {
            // TODO
        }
    }
}
```

```
<activity android:configChanges=["mcc", "mnc", "locale",
                                 "touchscreen", "keyboard", "keyboardHidden",
                                 "navigation", "screenLayout", "fontScale",
                                 "uiMode", "orientation", "density",
                                 "screenSize", "smallestScreenSize"]
          android:exported=["true" | "false"]
          android:launchMode=["standard" | "singleTop" |
                              "singleTask" | "singleInstance"]
          android:name="string"
          android:screenOrientation=["unspecified" | "behind" |
                                     "landscape" | "portrait" |
                                     "reverseLandscape" | "reversePortrait" |
                                     "sensorLandscape" | "sensorPortrait" |
                                     "userLandscape" | "userPortrait" |
                                     "sensor" | "fullSensor" | "nosensor" |
                                     "user" | "fullUser" | "locked"]
          android:theme="resource or theme" >
```

BroadcastReceiver

```
<receiver
    android:name=".MyBroadcastReceiver"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <action android:name="android.intent.action.INPUT_METHOD_CHANGED" />
    </intent-filter>
</receiver>
```

```
val receiver = MyBroadcastReceiver()
val filter = IntentFilter(Intent.ACTION_AIRPLANE_MODE_CHANGED)
context.registerReceiver(receiver, filter)
```

```kotlin
class MyBroadcastReceiver : BroadcastReceiver() {

    override fun onReceive(context: Context, intent: Intent) {
        when (intent.action) {
            Intent.ACTION_BOOT_COMPLETED -> {
                // TODO
            }

            Intent.ACTION_INPUT_METHOD_CHANGED -> {
                // TODO
            }
        }
    }
}
```

```
val receiver = MyBroadcastReceiver()
val filter = IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION)
registerReceiver(receiver, filter)
```

```
class MyNetworkCallback : ConnectivityManager.NetworkCallback() {

    override fun onCapabilitiesChanged(network: Network,
                                       networkCapabilities: NetworkCapabilities)

    override fun onLost(network: Network)

    override fun onLinkPropertiesChanged(network: Network,
                                         linkProperties: LinkProperties)

    override fun onUnavailable()

    override fun onLosing(network: Network, maxMsToLive: Int)

    override fun onAvailable(network: Network)
}
```

```
val connectivityManager = context.getSystemService(ConnectivityManager::class.java)

val networkRequest = NetworkRequest.Builder()
        .addCapability(NetworkCapabilities.NET_CAPABILITY_INTERNET)
        .addTransportType(NetworkCapabilities.TRANSPORT_CELLULAR)
        .build()

connectivityManager.registerNetworkCallback(networkRequest, MyNetworkCallback())
```

```
val localBroadcastManager = LocalBroadcastManager.getInstance(context)
localBroadcastManager.registerReceiver(MyBroadcastReceiver(), IntentFilter(CUSTOM_ACTION))


localBroadcastManager.sendBroadcast(Intent().setAction(CUSTOM_ACTION))
```

Service

Виды Service

- Background

Виды Service

- Background
- Foreground

Виды Service

- Background
- Foreground
- Bound

Реализации Service

- Service

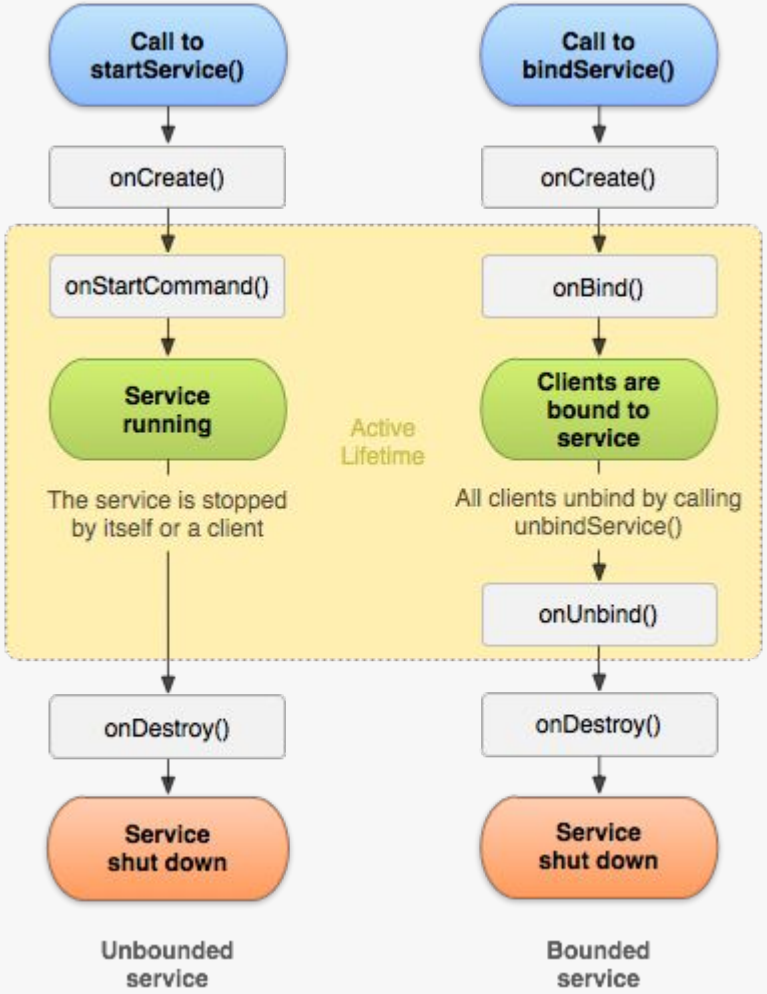
Реализации Service

- Service
- IntentService

Реализации Service

- Service
- IntentService
- JobIntentService

Service Lifecycle



```
val intent = Intent(context, MyService::class.java)
context.startForegroundService(intent)
```

```
class MyService : Service() {

    override fun onCreate() {
        createChannel()
        startForeground(NOTIFICATION_ID, createNotification())
    }

    private fun createChannel() {
        val channel = NotificationChannel(CHANNEL_ID, CHANNEL_NAME, IMPORTANCE_HIGH)
        val notificationManager = getSystemService(NotificationManager::class.java)
        notificationManager.createNotificationChannel(channel)
    }

    private fun createNotification(): Notification {
        return Notification.Builder(this, CHANNEL_ID)
                .setSmallIcon(R.mipmap.ic_launcher_round)
                .setContentTitle("Hello, World!")
                .build()
    }
}
```

```
class MyBoundService : Service() {

    companion object {

        fun fromBinder(binder: IBinder): MyBoundService? {
            if (binder is MyBinder) {
                return binder.service
            }
            return null
        }
    }

    private val binder: IBinder = MyBinder(this)

    override fun onBind(intent: Intent?): IBinder = binder

    private inner class MyBinder(val service: MyBoundService) : Binder()
}
```

```
private val connection = object : ServiceConnection {

    override fun onServiceConnected(name: ComponentName, binder: IBinder) {
        service = MyBoundService.fromBinder(binder)
    }

    override fun onServiceDisconnected(name: ComponentName) {
        service = null
    }
}

private var service: MyBoundService? = null

override fun onStart() {
    super.onStart()
    bindService(Intent(this, MyBoundService::class.java), connection, BIND_AUTO_CREATE)
}

override fun onStop() {
    super.onStop()
    unbindService(connection)
}
```

Планировщик задач

- AlarmManager
- JobScheduler (minSdk = 21+)
- Firebase JobDispatcher (minSdk = 14+, Play Services required)

Планировщик задач

- ~~AlarmManager~~
- ~~JobScheduler (minSdk = 21+)~~
- ~~Firebase JobDispatcher (minSdk = 14+, Play Services required)~~
- WorkManager

Спасибо за внимание