ТЕХНОТРЕК

Занятие №3

Разработка приложений на Android

Кирилл Филимонов
Юрий Береза

Напоминание

А ты отметился о присутствии на занятии?

Agenda

1. Потоки и процессы
2. Инструменты
3. Android Permissions

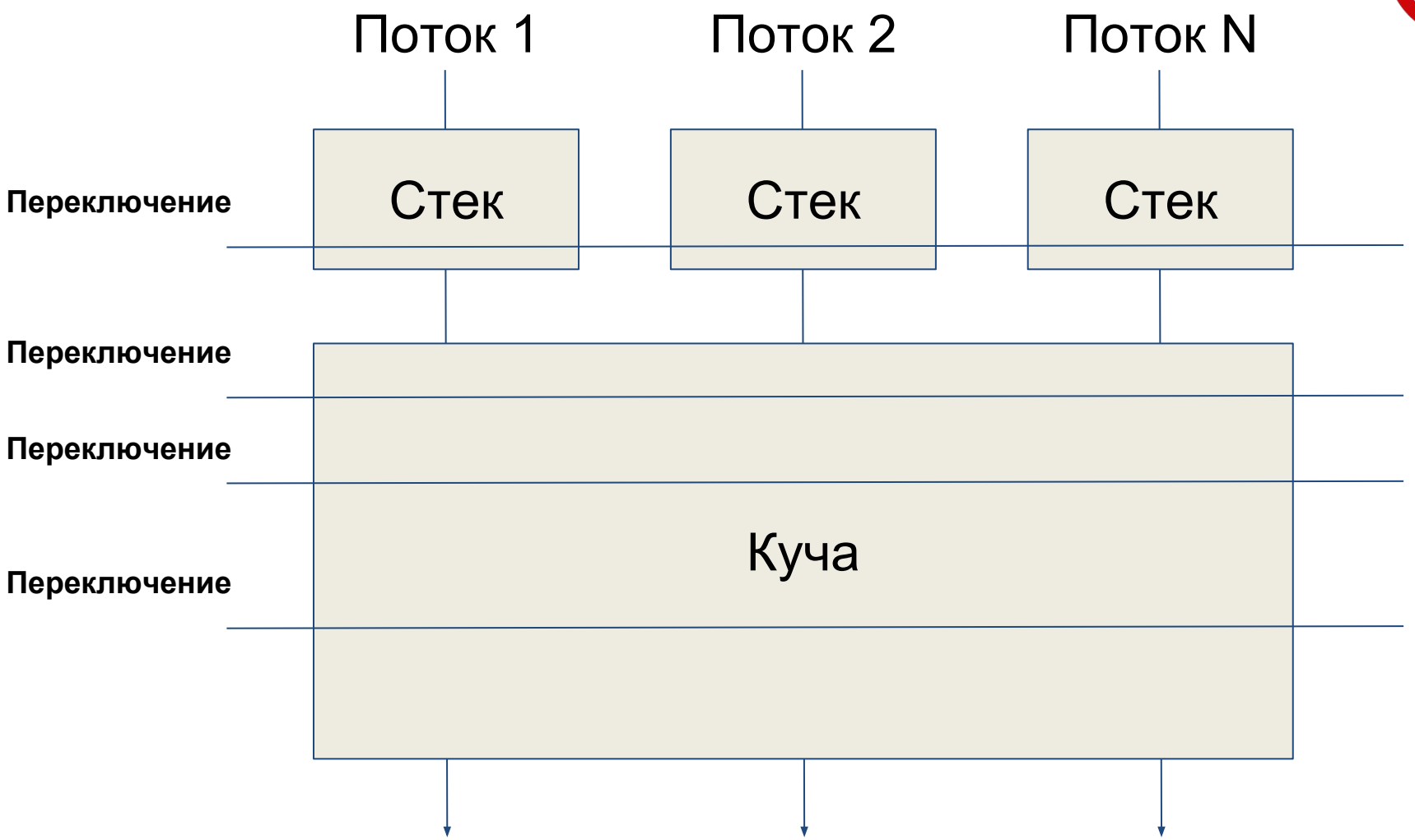
“Нет лучше насилия, чем насилие над самим собой.

Гомер Симпсон

Потоки

- Почему собственно слово «поток»
- Процесс и потоки
- UI поток
- Почему потоки это хорошо
- Почему это иногда плохо
- Как они выглядят с точки зрения программы
- Что такое race condition
- Что такое dead lock

Потоки


Поток 1
Поток 2
Поток N
Стек
Стек
Стек
Переключение
Переключение
Переключение
Переключение
Куча

Java потоки

- Итак, потоки в Java. Основа их – класс java.lang.Thread. Этот класс позволяет создать поток и запустить его на выполнение.
- Существует два пути создания потока.
- Наследование от класса java.lang.Thread и переопределение его метода run.
- Реализация интерфейса java.lang.Runnable и создание потока на основе этой реализации. В принципе это методы эквивалентны, разница в деталях.

```
public class Program {
    public static void main(String[] args) {
        //Создание потока
        Thread myThready = new Thread(new Runnable() {
            public void run() {
                //Этот метод будет выполняться в побочном потоке
                System.out.println("Привет из побочного потока!");
            }
        });
        myThready.start(); //Запуск потока
        System.out.println("Главный поток завершён...");
    }
}
```

```
class AffableThread extends Thread {
    @Override
    public void run() {
        System.out.println("Привет из побочного потока!");
    }
}

public class Program {
    static AffableThread mSecondThread;
    public static void main(String[] args) {
        mSecondThread = new AffableThread();
        mSecondThread.start();
        System.out.println("Главный поток завершён...");
    }
}
```

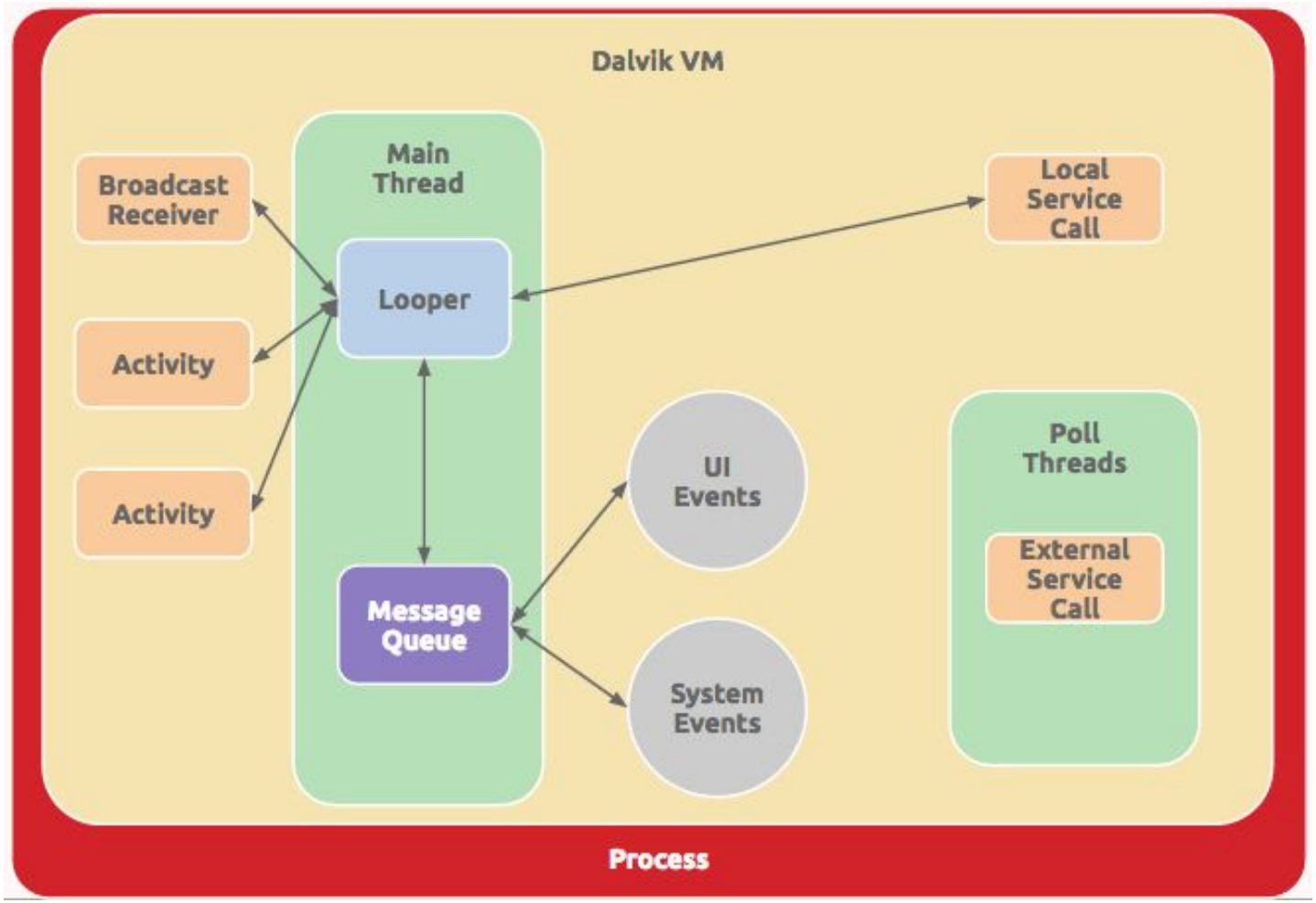
Способы синхронизации в Java

- wait, notify, notifyAll
- synchronized
  - Объекты
  - Методы
- синхронизация через функцию join
- инструменты из пакета java.util.concurrent

Wait, Notify

```
public class DataManager {
  private static final Object monitor = new Object();
  public void sendData() {
    synchronized (monitor) {
      System.out.println("Waiting for data...");
      try {
        monitor.wait();
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
      // continue execution and sending data
      System.out.println("Sending data...");
    }
  }
  public void prepareData() {
    synchronized (monitor) {
      System.out.println("Data prepared");
      monitor.notifyAll();
    }
  }
}
```

UI поток


Dalvik VM
Main Thread
Broadcast Receiver
Looper
Activity
Activity
Local Service Call
UI Events
Message Queue
System Events
Poll Threads
External Service Call
Process

UI и другие потоки

- View.post (Runnable action)
- Activity.runOnUiThread(Runnable action).
- Handler
- AsyncTask

View.Post

```
public class MainActivity extends Activity {

    TextView textView;

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

        textView = (TextView)findViewById(R.id.hello);

        WorkingClass workingClass = new WorkingClass();
        Thread thread = new Thread(workingClass);
        thread.start();
    }

    class WorkingClass implements Runnable{
        @Override
        public void run() {
            //Фоновая работа

            //Отправить в UI поток новый Runnable
            textView.post(new Runnable() {
                @Override
                public void run() {
                    textView.setText("The job is done!");
                }
            });
        }
    }
}
```

Activity.runOnUiThread

```
public class MainActivity extends Activity {

    TextView textView;

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

        textView = (TextView)findViewById(R.id.hello);

        WorkingClass workingClass = new WorkingClass();
        Thread thread = new Thread(workingClass);
        thread.start();
    }

    class WorkingClass implements Runnable{
        @Override
        public void run() {
            //Фоновая работа

            //Отправить в UI поток новый Runnable
            MainActivity.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    textView.setText("The job is done!");
                }
            });
        }
    }
}
```

Handler

```
public class MainActivity extends Activity {
  TextView textView;
  @Override
  protected void onCreate(…) {
    …
    WorkingClass workingClass = new
WorkingClass(true);
    Thread thread = new Thread(workingClass);
    thread.start();
  }

  class WorkingClass implements Runnable{

    public static final int SUCCESS = 1;
    public static final int FAIL = 2;
    private boolean dummyResult;

    public WorkingClass(boolean dummyResult){
      this.dummyResult = dummyResult;
    }
```

```
  @Override
  public void run() {
      //Фоновая работа

      //Отправить в хэндлеру сообщение
      if (dummyResult){
      //Можно отправить пустое сообщение со статусом
          uiHandler.sendEmptyMessage(SUCCESS);
      } else {
      //Или передать в месте с сообщением данные
          Message msg = Message.obtain();
          msg.what = FAIL;
          msg.obj = "An error occurred";
          uiHandler.sendMessage(msg);
      }
  }
  Handler uiHandler = new Handler(new Handler.Callback({
        @Override
        public boolean handleMessage(Message msg) {
            switch (msg.what) {
            case WorkingClass.SUCCESS:
                textView.setText("Success");
                return true;
            case WorkingClass.FAIL:
                textView.setText((String)msg.obj);
                return true;
            }
            return false;
        }
    });
}
```

AsyncTask

- Чтобы работать с AsyncTask, надо создать класс-наследник и в нем прописать свою реализацию необходимых нам методов.
- doInBackground – будет выполнен в новом потоке, здесь решаем все свои тяжелые задачи. Т.к. поток не основной - не имеет доступа к UI.
- onPreExecute – выполняется перед doInBackground, имеет доступ к UI
- onPostExecute – выполняется после doInBackground (не срабатывает в случае, если AsyncTask был отменен), имеет доступ к UI
- onProgressUpdate — вызывается в потоке пользовательского интерфейса. Этот метод используется для отображения любых форм прогресса в пользовательском интерфейсе, пока идут вычисления в фоновом режиме.

```
private class ShowDialogAsyncTask extends AsyncTask<Void, Integer, Void> {

    int progress_status;

    @Override
    protected void onPreExecute() {
    }

    @Override
    protected Void doInBackground(Void... params) {
    }

    @Override
    protected void onProgressUpdate(Integer... values) {
    }

    @Override
    protected void onPostExecute(Void result) {
    }
}
```

Loader

- Простой API, позволяющий Acitivity и Fragment осуществлять асинхронную загрузку
- Один экземпляр LoaderManager для Activity или Fragment

- initLoader(int id, Bundle args, LoaderCallbacks callbacks)
- restartLoader(int id, Bundle args, LoaderCallbacks callbacks)
- destroyLoader(int id)
- getLoader(int id)

Permissions

Безопасность Android приложений:

- kernel level
- изоляция процессов (sandbox)
- безопасный IPC
- пользовательские разрешения (permissions)

Добавление permission

- по-умолчанию приложение не обладает разрешениями

AndroidManifest.xml:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.app.myapp" >
    <uses-permission android:name="android.permission.RECEIVE_SMS"
/>
    ...
</manifest>
```

Типы

normal - разрешения, не представляющие большого риска для приватных данных пользователя или работы устройства.

- разрешение на изменение часового пояса
- разрешение на доступ к режиму вибрации
- ...

dangerous - разрешения, представляющие риск для пользовательских данных или устройства.

- разрешение на доступ к контактам
- разрешение на чтение SMS
- ...

Предоставление разрешений

targetSdkVersion < 23:

- все разрешения запрашиваются при установке приложения
- нет возможности частичной выдачи разрешений

либо принимаем все и устанавливаем приложение, либо отказываемся от установки

Предоставление разрешений

targetSdkVersion >= 23:

- разрешения категории normal выдаются автоматически
- разрешения категории dangerous требуют runtime проверки
- `SecurityException` при попытке воспользоваться api без разрешения

у пользователя появилась возможность динамически выдавать и забираться dangerous разрешения у приложений.

Группы разрешений

- разрешения разделены на группы
- если приложению требуется разрешение и ни одного разрешения из группы еще не было выдано, необходимо выполнять запрос
- если приложению требуется разрешение и какое-либо разрешение из группы уже выдано, то разрешение будет выдано автоматически
- normal разрешения не влияют на группы dangerous разрешений
- на targetSdkVersion < 23 при установке запрашиваются разрешения на группу

Группы разрешений

Группа:

```
CONTACTS
```

Разрешения:

```
READ_CONTACTS
WRITE_CONTACTS
GET_ACCOUNTS
```

Группы разрешений

- CALENDAR
- CAMERA
- CONTACTS
- LOCATION
- MICROPHONE
- PHONE
- SENSORS
- SMS
- STORAGE

Пользовательские разрешения

возможно объявлять собственные разрешения

AndroidManifest.xml:

```
<manifest
   xmlns:android="http://schemas.android.com/apk/res/android"
       package="com.example.myapp" >
       <permission
   android:name="com.example.myapp.permission.DEADLY_ACTIVITY"
           android:label="@string/permlab_deadlyActivity"
           android:description="@string/permdesc_deadlyActivity"

   android:permissionGroup="android.permission-group.COST_MONEY"
           android:protectionLevel="dangerous" />
       ...
   </manifest>
```

Защищенные компоненты

- `Activity`
- `Service`
- `BroadcastReceiver`
- `ContentProvider`

В AndroidManifest.xml компоненту необходимо добавить атрибут `android:permission.`

ТЕХНОТРЕК

Спасибо за внимание!

Кирилл Филимонов – Kirill.Filimonov@gmail.com

Юрий Береза – ybereza@gmail.com