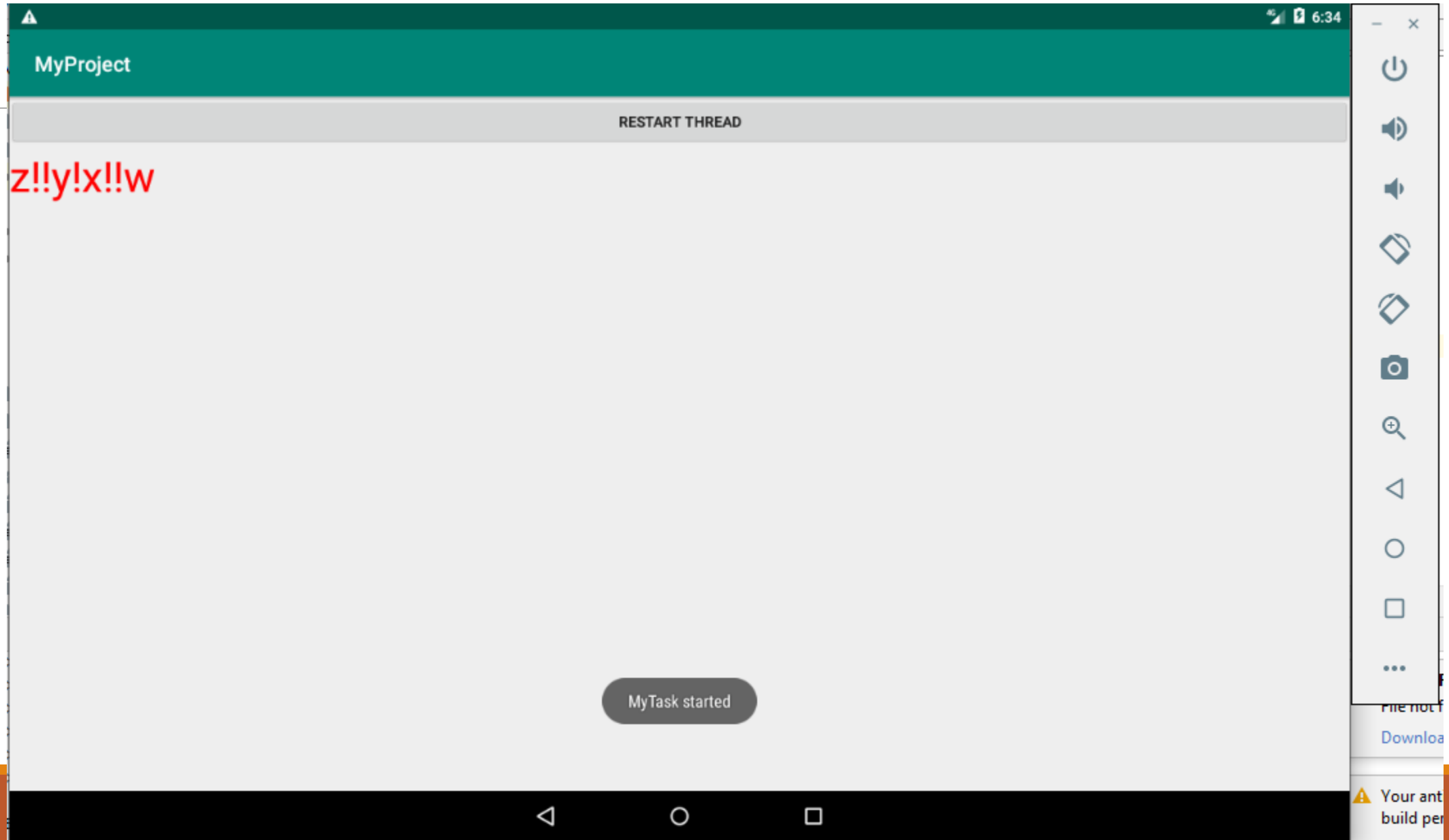


Многопоточность в Android-приложениях

ЛОГИНОВА Ю.В.

Демонстрация всплывающих сообщений



Метод onCreate() вызывается при создании или перезапуска активности

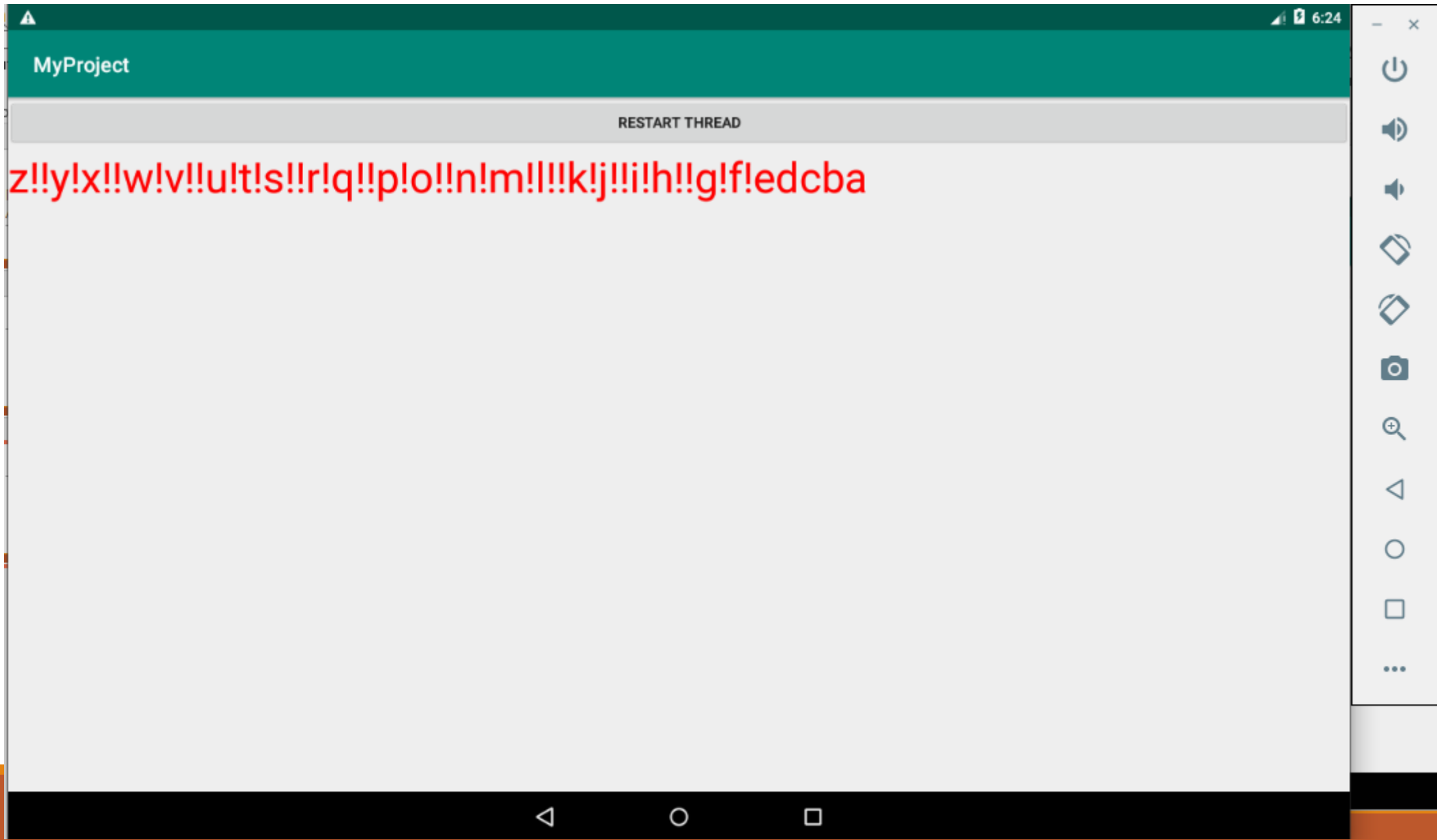
```
44
45
46
47
48
49
52
53
54
55
57
58
59
60

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Button btn = (Button) findViewById(R.id.button1);
    View.OnClickListener oclBtn = (v) -> {
        TextView msgBox = (TextView) findViewById(R.id.msg);
        msgBox.setText(msgBox.getText() + "\n");
        restartThread();
    };
    btn.setOnClickListener(oclBtn);
    restartThread();
}
```

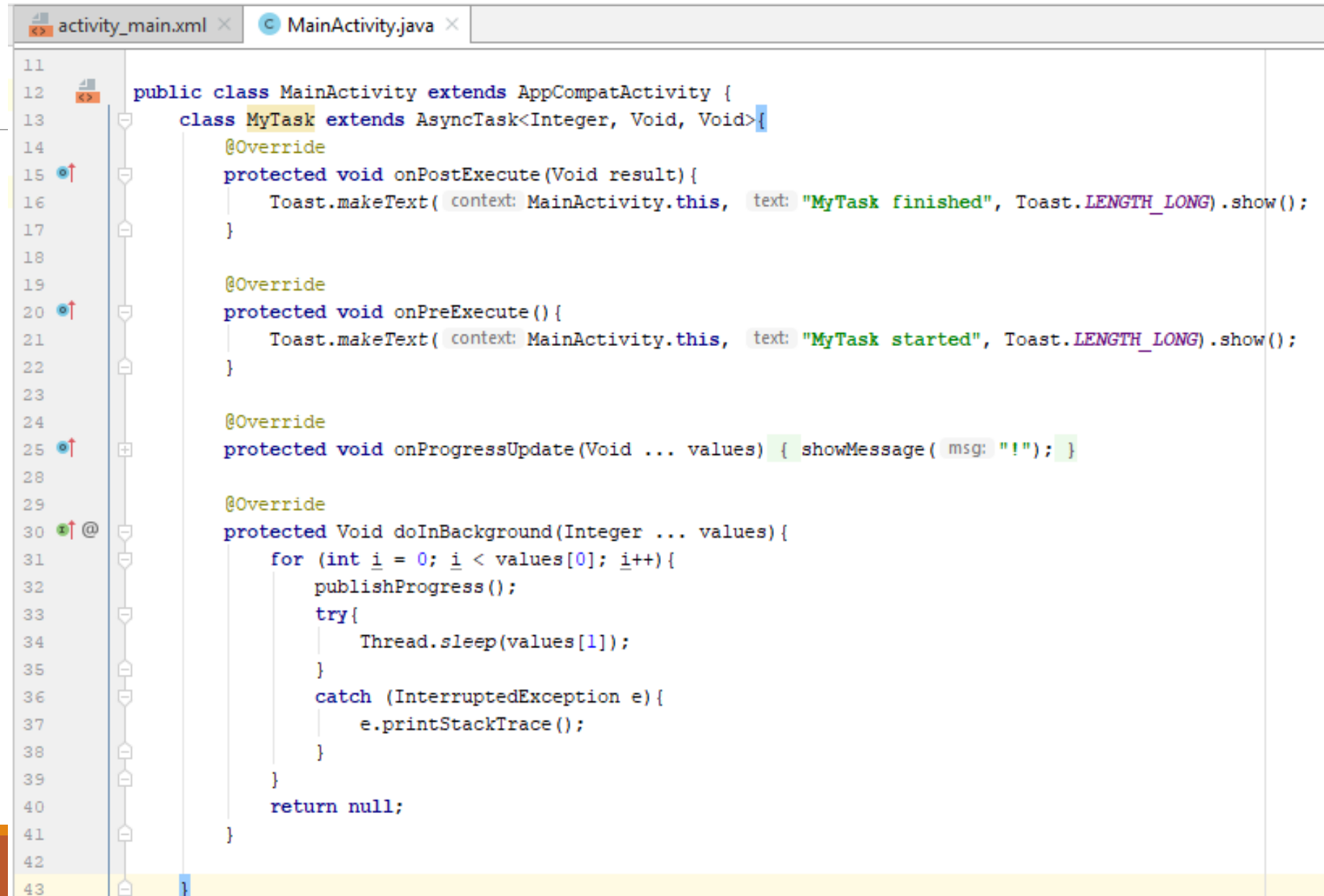
Функции перезапуска потоков и вывода символов

```
61 void restartThread() {  
62     new MyTask().execute(30, 700);  
63     new Thread((Runnable) () → { work(); }).start();  
69 }  
70  
71 void showMessage(final String msg) {  
72     TextView msgBox = (TextView) findViewById(R.id.msg);  
73     msgBox.setText(msgBox.getText() + msg);  
74 }
```

Демонстрация работы потоков



Класс MainActivity



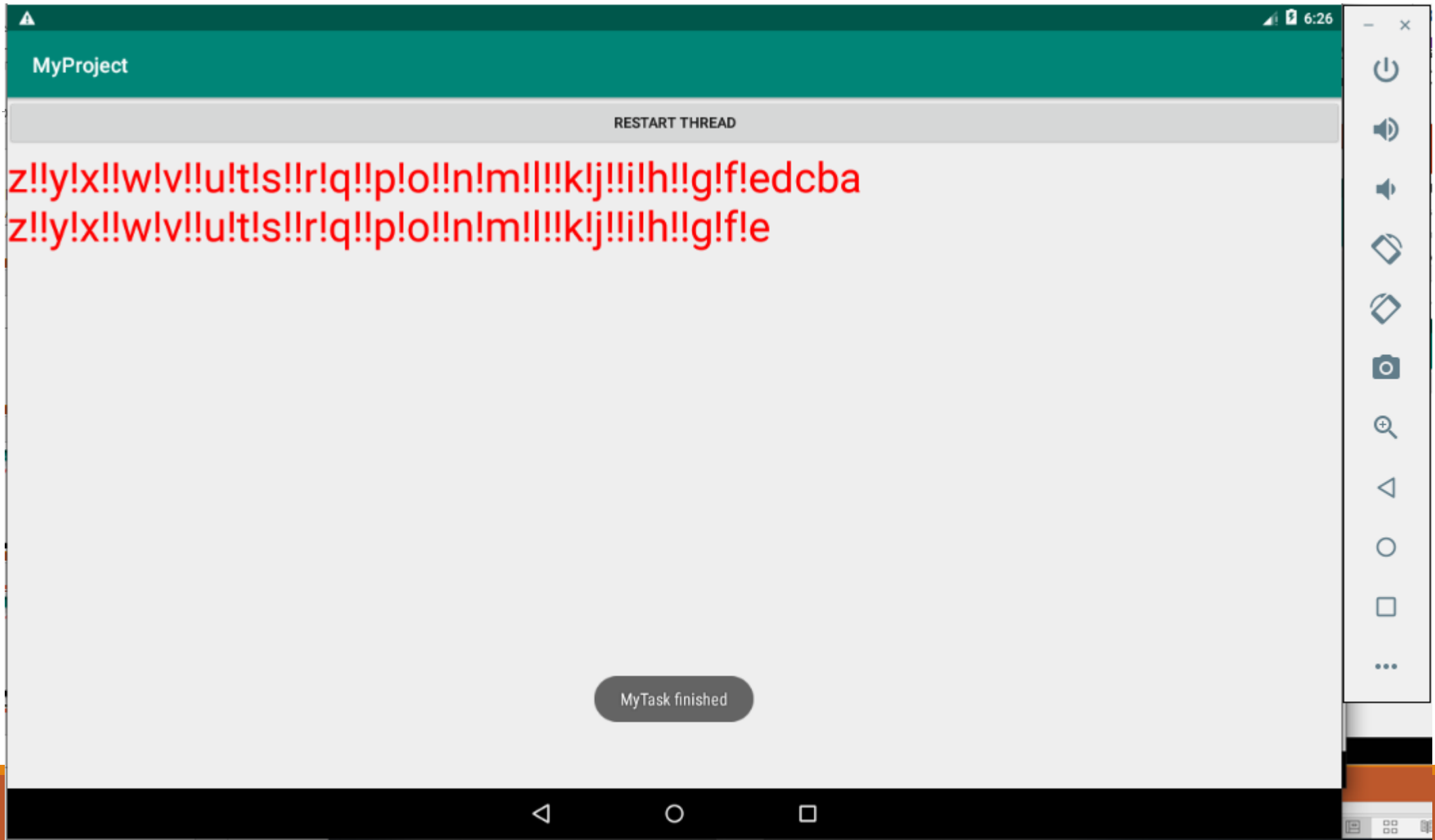
```
11
12 public class MainActivity extends AppCompatActivity {
13     class MyTask extends AsyncTask<Integer, Void, Void>{
14         @Override
15         protected void onPostExecute(Void result){
16             Toast.makeText(context: MainActivity.this, text: "MyTask finished", Toast.LENGTH_LONG).show();
17         }
18
19         @Override
20         protected void onPreExecute() {
21             Toast.makeText(context: MainActivity.this, text: "MyTask started", Toast.LENGTH_LONG).show();
22         }
23
24         @Override
25         protected void onProgressUpdate(Void ... values) { showMessage(msg: "!"); }
26
27
28
29         @Override
30         protected Void doInBackground(Integer ... values){
31             for (int i = 0; i < values[0]; i++){
32                 publishProgress();
33                 try{
34                     Thread.sleep(values[1]);
35                 }
36                 catch (InterruptedException e){
37                     e.printStackTrace();
38                 }
39             }
40             return null;
41         }
42     }
43 }
```

Класс **AsyncTask** позволяет правильно и просто использовать главный поток приложения, обслуживающий интерфейс пользователя

AsyncTask содержит четыре метода, которые можно переопределить:

- Метод **doInBackground()** выполняется в фоновом потоке, должен возвращать определенный результат
- Метод **onPreExecute()** вызывается из главного потока перед запуском метода **doInBackground()**
- Метод **onPostExecute()** выполняется из главного потока после завершения работы метода **doInBackground()**
- Метод **onProgressUpdate()** позволяет сигнализировать пользователю о выполнении фонового потока

Демонстрация перезапуска потока



Вывод символов в главном потоке приложения

```
76 public void work() {  
77     for (int i=122; i>= 97; i--){  
78         try{  
79             final char k = (char)i;  
80             runOnUiThread(() -> { showMessage( msg: k + ""); });  
86             Thread.sleep( millis: 1000 );  
87         }  
88         catch (InterruptedException e){  
89             e.printStackTrace();  
90         }  
91     }  
92 }
```