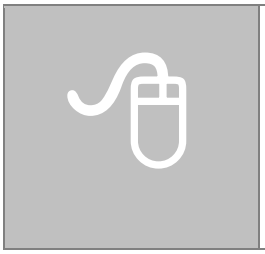
ASYNCTASK – THREAD – HANDLER



✓ **Mục tiêu:**

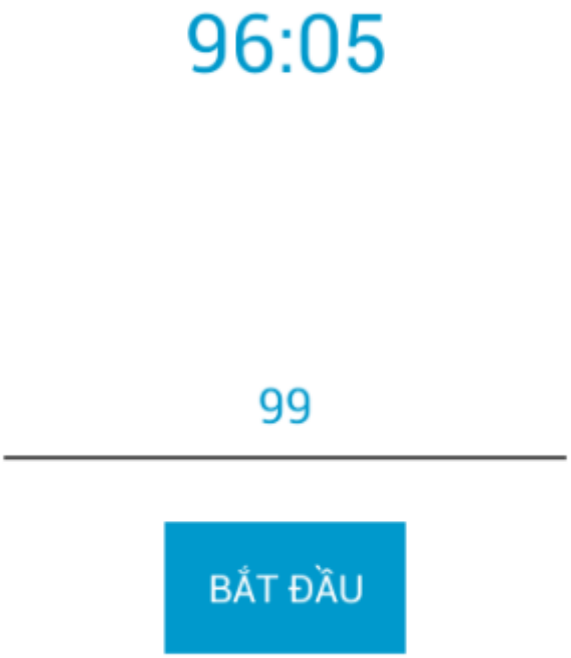
- *Tăng tốc ứng dụng với các luồng xử lý khác nhau.*
- *Xây dựng chế độ làm việc ngầm cho ứng dụng với Asynctask.*

Bài tập 1.1. Cách sử dụng Asynctask

Đề bài:

- Viết chương trình đếm ngược thời gian.
  Chi tiết yêu cầu: Có một EditText để nhập số nguyên dương, một Button khi nhấn vào thì bắt đầu đếm ngược từ số nguyên nhập về 0 và trong quá trình đếm ngược thì hiển thị số thời gian còn lại lên TextView cho người dùng xem (gồm số giây và tích tắc được lấy theo 2 số. Ví dụ: 9:99), khi nhấn vào Button thì Button và Editext đều không được tương tác cho đến khi thời gian được đếm xong (tức là về 0).
  Chú ý: Dùng Asynctask.
  Giao diện:

Nhập số và nhấn nút "BẮT ĐẦU" thì thời gian bắt đầu đếm ngược:



Gợi ý thực hiện:

- Tạo lớp kế thừa "AsyncTask".
- Dùng "setEnabled" để thiết lập cho phép tương tác hoặc không cho đối tượng View

Hướng dẫn chi tiết:

Bước 1: Tạo lớp "Timer" kế thừa AsyncTask. Sau khi kế thừa cần bắt buộc Override lại phương thức "doInBackground". Nhưng để giải quyết bài này cần Override thêm 3 phương thức "onPreExecute", "onProgressUpdate" và "onPostExecute". Sau khi khởi tạo lớp Timer thì phương thức "onPreExecute" sẽ chạy đầu tiên. Sau đó đến "doInBackground" và "onProgressUpdate" chạy song song, từ "doInBackground" có thể gửi dữ liệu qua "onProgressUpdate" thông qua phương thức "publishProgress". Cuối cùng là phương thức "onPostExecute" và kết thúc.

Bước 1. 1: Tạo biến "num" có kiểu giá trị int dùng để lấy số nguyên dương mà người dùng nhập vào.

Bước 1. 2: Lấy dữ liệu nhập vào trong phương thức "onPreExecute":

```
protected void onPreExecute() {
   super.onPreExecute();
   try {
      // Lấy số nhập
```

```
            num = Integer.parseInt
                  (editText.getText().toString()) * 100;

            /*
             * Thiết lập không cho tương tác
             *  với Button và EditText
             */
            button.setEnabled(false);
            editText.setEnabled(false);
        } catch (Exception e) {
        }
    }
```

Bước 1. 3: Viết xử lý trong phương thức "doInBackground":

```
    protected Void doInBackground(Void... params) {
        while (true) {
            try {
                if (num == 0) {
                    return null;
                }
                // Dừng trong 1/100 giây
                Thread.sleep(10);

                num = num - 1;
                /*
                 *  Cập nhật dữ liệu cho
                 *   onProgressUpdate
                 */
                publishProgress(num);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
```

Bước 1. 4: Viết xử lý có "onProgressUpdate" để cập nhật giao diện trong quá trình xử lý dưới "doInBackground":

```
    protected void onProgressUpdate(Integer... values) {
        super.onProgressUpdate(values);
        // Lấy giá trị mà bên doInBackground gửi qua
        // thông qua biến values
        int millis = values[0] % 100;
        textView.setText(values[0] / 100 + ":"
                + ((millis + "").length() != 2 ? "0"
                + millis : millis));
    }
```

Bước 1. 5: Viết xử lý khi kết thúc tiến trình trong phương thức "onPostExecute":

```
    protected void onPostExecute(Void result) {
        super.onPostExecute(result);
        /*
         * Cho phép tương tác với Button
         *  và EditText
         */
        button.setEnabled(true);
        editText.setEnabled(true);
    }
```

Bước 2: Viết gọi lớp vừa tạo cho chạy:

```
public class MainActivity extends Activity {

  TextView textView;
  EditText editText;
  Button button;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      textView = (TextView) findViewById(R.id.textView);
      editText = (EditText) findViewById(R.id.editText);
      button = (Button) findViewById(R.id.button);
  }

  public void onClick(View v) {
      /*
       * Khởi tạo Timer và cho thực hiện
       */
      new Timer().execute();
  }
}
```

Bài tập 1.2. Cách sử dụng Thread.

Đề bài:

- Viết chương trình đếm ngược thời gian.
  Chi tiết yêu cầu: Có một ô nhập số nguyên dương, một Button khi nhấn vào thì bắt đầu đếm ngược từ số nguyên nhập về 0 và trong quá trình đếm ngược thì hiển thị số thời gian còn lại lên TextView cho người dùng xem (gồm số giây và tích tắc

được lấy theo 2 số. Ví dụ: 9:99), khi nhấn vào Button thì Button và Editext đều không được tương tác cho đến khi thời gian được đếm xong (tức là về 0).
Chú ý: Dùng Thread.

Gợi ý thực hiện:

- Khởi tạo Thread và truy xuất đến phương thức "start()".
- Dùng "Handler" để gửi dữ liệu thiết lập hiển thị cho người dùng.
- Dùng "setEnabled" để thiết lập cho phép tương tác hoặc không cho đối tượng View.

Hướng dẫn chi tiết:

Bước 3: Tạo "Handler" để hỗ trợ tương tác với giao diện khi làm việc ở tiến trình.

```
Handler handler = new Handler() {

    @Override
    public void handleMessage(Message msg) {

       if (msg.what == 0) {
          button.setEnabled(false);
          editText.setEnabled(false);
       } else if (msg.what == 1) {
          int millis = msg.arg1 % 100;
          textView.setText(msg.arg1 / 100 + ":"
                + ((millis + "").length() != 2 ? "0"
                + millis : millis));
       } else {
          button.setEnabled(true);
          editText.setEnabled(true);
       }
    }
};
```

Chú ý: Để nhận được dữ liệu thì ta phải thông qua một lớp "Message". Chúng ta muốn gửi dữ liệu hoặc nhận dữ liệu đều phải thông qua lớp này, lớp "Message" cung cấp cho 4 thuộc tính: what (kiểu int), arg1 (kiểu int), arg2 (kiểu int) và obj (kiểu Object).

Bước 4: Tạo biến "numb" kiểu int dùng để lấy số nguyên dương từ người dùng nhập. Sau đó bắt sự kiện khi người dùng ấn Button và viết xử lý:

```
public void onClick(View v) {
    numb = Integer.parseInt
          (editText.getText().toString()) * 100;
    new Thread() {
```

```
        public void run() {
           // 0 là trạng bắt đầu
           handler.obtainMessage(0).sendToTarget();
           while (true) {
              try {
                 if (numb == 0) {
                    // 2 là trạng thái kết thúc
                    handler.obtainMessage(2)
                       .sendToTarget();
                    return;
                 }
                 Thread.sleep(10);
                 numb = numb - 1;
                 // 1 là trạng thái đang chạy
                 handler.obtainMessage(1,
                       numb, 0).sendToTarget();
              } catch (InterruptedException e) {
                 e.printStackTrace();
              }
           }
        }

    // Dùng start() để chạy ngầm, không làm đơ ứng dụng khi
    // đang xử lý tiến trình bên trong
    }.start();
 }
```