La Clase `Thread`.

CONCEPTOS.
Un hilo es una unidad concurrente de ejecución. Posee su propia pila de llamadas a los métodos que se invoca, sus argumentos y variables locales. Cada aplicación tiene al menos un hilo de ejecución cuando se inicia, que es el hilo principal, en el `ThreadGroup` principal. El tiempo de ejecución mantiene sus propios hilos en el grupo de hilos del sistema.

Existen dos formas de ejecutar código en un nuevo hilo:

- Se puede heredar de la clase `Thread` y sobrescribir su método `run`().
- Construir un nuevo hilo `Thread` y pasar la interface `Runnable` al constructor.

En cualquier caso, se debe invocar al método `start`() para realmente ejecutar el nuevo hilo `Thread`.

Cada hilo posee una prioridad de tipo entero que afecta la forma en que el hilo se administra en el sistema operativo. Un nuevo hilo hereda la prioridad de su clase padre. La prioridad de un hilo se puede ajustar mediante el método `setPriority`(`int`).

Jerarquía de clases:

```
public class Thread  -> extends Object implements Runnable
java.lang.Object      -> java.lang.Thread
```

Subclasses directas:
`ForkJoinWorkerThread, HandlerThread`

Clases anidadas:

`enum`	`Thread.State`	Una representación del estado del hilo.
`interface`	`Thread.UncaughtExceptionHandler`	Implantada por objetos que se desea manejar en donde un hilo se termina por una excepción no capturada.

Constantes:

`int`	`MAX_PRIORITY`	El valor máximo de prioridad permitida para un hilo.
`int`	`MIN_PRIORITY`	El valor mínimo de prioridad permitida para un hilo.
`int`	`NORM_PRIORITY`	El valor normal (predeterminado) de prioridad asignado al hilo principal.

Constructores públicos:

`Thread()`	Construye un nuevo `Thread` sin objeto `Runnable` y un nuevo nombre.
`Thread(Runnable runnable)`	Construye un nuevo `Thread` con un objeto `Runnable` y un nuevo nombre.
`Thread(Runnable runnable, String threadName)`	Construye un nuevo `Thread` con un objeto `Runnable`, incluyendo un nombre.
`Thread(String threadName)`	Construye un nuevo `Thread` sin un objeto `Runnable`, incluyendo un nombre.
`Thread(ThreadGroup group, Runnable runnable)`	Construye un nuevo `Thread` con un objeto `Runnable`, un nuevo nombre y poseído por un `ThreadGroup` pasado como parámetro.
`Thread(ThreadGroup group, Runnable runnable, String threadName)`	Construye un nuevo `Thread` con un objeto `Runnable`, incluyendo un nombre y poseído por un `ThreadGroup` pasado como parámetro.
`Thread(ThreadGroup group, String threadName)`	Construye un nuevo `Thread` sin un objeto `Runnable` incluyendo un nombre y poseído por un `ThreadGroup` pasado como parámetro.
`Thread(ThreadGroup group, Runnable runnable, String threadName, long stackSize)`	Construye un nuevo `Thread` con un objeto `Runnable` incluyendo un nombre y poseído por un `ThreadGroup` pasado como parámetro y tamaño de pila.

Dr. Alejandro S. Cifuentes A.

Métodos públicos:

`Static Thread currentThread()`	Regresa el hilo actual al invocador.
`long getId()`	Regresa el identificador del hilo.
`final String getName()`	Regresa el nombre del hilo.
`final int getPriority()`	Regresa la prioridad del hilo.
`Thread.State getState()`	Regresa el estado actual del hilo.
`final boolean isAlive()`	Regresa true si el receptor está iniciado y en ejecución. (Todavía no ha muerto).
`void run()`	Invoca al método `run`() del objeto `Runnable` que posea el receptor.
`final void setName(String s)`	Asigna el nombre del hilo.
`final void setPriority(int p)`	Asigna la prioridad del hilo.
`static void sleep(long t)`	Duerme al hilo que envió el mensaje durante un tiempo, en milisegundos.
`void start()`	Inicia la ejecución del nuevo hilo.

Los siguientes son otros métodos públicos. Para su uso se recomienda consultar el manual.
`activeCount(), checkAccess(), countStackFrames(), destroy(), dumpStack(), enumerate(), getAllStackTraces(), getContextClassLoader(), getDefaultUncaughtExceptionHandler(), getStackTrace(), getThreadGroup(), getUncaughtExceptionHandler(), holdsLock(), interrupt(), interrupted(), isDaemon(), join(), run(), setContextClassLoader(), setDaemon(), setDefaultUncaughtExceptionHandler(), setUncaughtExceptionHandler(), start(), toString(), yield().`

Cuando se inicia una aplicación, se crea un hilo de ejecución o hilo principal. Este hilo es importante, porque administra y responde a los eventos; además, de dibujar la interfaz e interactuar con el usuario. No conviene bloquearlo con tareas muy pesadas, por ejemplo una consulta en una base de datos grande, un acceso a un sitio web o una larga operación matemática.

DESARROLLO

EJEMPLO 1.

El siguiente ejercicio bloquea la pantalla del dispositivo durante algunos segundos, utilizando un hilo.

Paso 1. Crear un nuevo proyecto Hilos. En la actividad principal MainActivity.java, capturar el siguiente código:

```
import android.app.*;
import android.os.*;
import android.view.View;
import android.view.View.*;
import android.widget.*;
public class MainActivity extends Activity implements OnClickListener {
    private EditText jet1;
    private Button   jbn1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        jet1 = (EditText) findViewById(R.id.xet1);
        jbn1 = (Button) findViewById(R.id.xbn1);
        jbn1.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        try{
            int num = Integer.parseInt(jet1.getText().toString());
            Thread.sleep(num*1000);
        }catch (NumberFormatException e){
            Toast.makeText(this,"Ingresar los segundos.", Toast.LENGTH_SHORT).show();
        } catch (InterruptedException e) { }
    }
}
```

Paso 2. En el archivo activity_main.xml predeterminado, capturar el siguiente código:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <EditText
        android:id="@+id/xet1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="ingresar segundos..."
        android:inputType="number"/>
    <Button
        android:id="@+id/xbn1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Bloquear la pantalla"/>
</LinearLayout>
```

Paso 3. Ejecutar la aplicación. Se muestra la plantilla del proyecto. Ingresar algún número de segundos, por ejemplo 30, para bloquear momentáneamente la pantalla. El usuario **no** puede interactuar con la pantalla.

Si el número de segundos es grande se puede mostrar un mensaje indicando que la actividad no responde, pero ello depende de las capacidades del dispositivo; se sugiere probar con otros valores.

Paso 4. Cambiar el código del método `onClick`() del Paso 1, por el siguiente párrafo indicado en letras negritas. En este caso, se utiliza la clase `Thread` con la interface `Runnable` para crear un nuevo hilo, en el cual se ejecuta la operación, por lo que ya no se tendrá bloqueada la pantalla. Al ejecutar la aplicación se muestra la plantilla del proyecto. Ingresar algún número de segundos, para bloquear momentáneamente la pantalla. El usuario ahora **sí** puede interactuar con la pantalla.

```
import android.app.*;
import  android.os.*;
import android.view.View;
import android.view.View.*;
```

```
import android.widget.*;
public class MainActivity extends Activity implements OnClickListener {
    private EditText jet1;
    private Button   jbn1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        jet1 = (EditText) findViewById(R.id.xet1);
        jbn1 = (Button) findViewById(R.id.xbn1);
        jbn1.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        try{
            final int num = Integer.parseInt(jet1.getText().toString());
            new Thread(new Runnable(){
                public void run() {
                    try {
                        Thread.sleep(num*1000);
                    } catch (InterruptedException ie) {}
                }
            }).start();
        }catch (NumberFormatException e){
            Toast.makeText(this,"ingresar segundos...", Toast.LENGTH_SHORT).show();
        }
    }
}
```

Paso 5. Enseguida, cambiar el contenido del archivo activtiy_main.xml por el siguiente código:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:orientation="vertical">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <EditText
            android:id="@+id/xet1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="ingresar segundos..."
            android:inputType="number"/>
        <Button
            android:id="@+id/xbn1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Bloquear la pantalla"/>
    </LinearLayout>
```

```
    <TextView
        android:id="@+id/xtv2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="\nAl digitar el botón, el hilo\nse bloquea n segundos..." />
</LinearLayout>
```

Paso 6. Ejecutar la aplicación. Ahora, el hilo se bloquea, pero no la pantalla principal.

Paso 7. En el archivo MainActivity.java realizar los siguientes cambios en su posición correspondiente:

a. Agregar la instancia:

```
private TextView jtv2;
```

b. Crear su objeto:

```
jtv2 = (TextView) findViewById(R.id.xtv2);
```

c. Dentro de cláusula try-catch, inmediatamente después del try agregar:

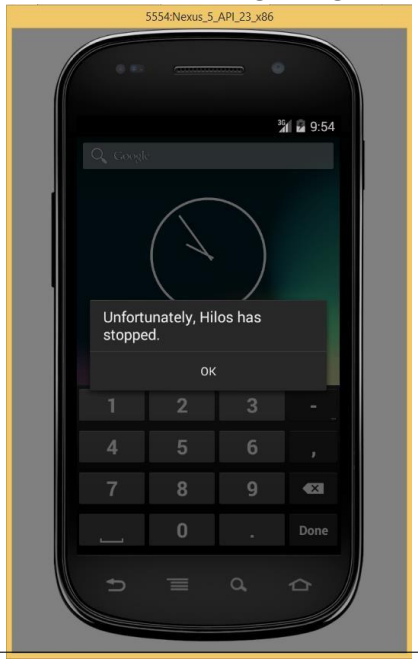
```
jtv2.setText("El hilo se bloqueó " + n + " segundos...");
```

d. El código debe ser similar al siguiente:

```
import android.app.*;
import android.os.*;
import android.view.View;
import android.view.View.*;
import android.widget.*;
public class MainActivity extends Activity implements OnClickListener {
    private EditText jet1;
    private  Button  jbn1;
    private TextView jtv2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        jet1 = (EditText) findViewById(R.id.xet1);
        jbn1 = (Button) findViewById(R.id.xbn1);
        jbn1.setOnClickListener(this);
        jtv2 = (TextView) findViewById(R.id.xtv2);
    }
```

```
    @Override
    public void onClick(View v) {
        try{
            final int n = Integer.parseInt(jet1.getText().toString());
            new Thread(new Runnable(){
                public void run() {
                    try {
                        jtv2.setText("El hilo se bloqueó " + n + " segundos...");
                        Thread.sleep(n*1000);
                    } catch (InterruptedException ie) {}
                }
            }).start();
        }catch (NumberFormatException e){
            Toast.makeText(this,"ingresar segundos...", Toast.LENGTH_SHORT).show();
        }
    }
}
```

Paso 8. Ejecutar la aplicación. Se muestra la plantilla del proyecto solicitando el ingreso de los segundos, por ejemplo 5 segundos. Digitar el botón. Ahora, se muestra un mensaje de detención inesperada, como se indica en la figura siguiente.



Paso 9. En el archivo MainActivity.java se realizan los siguientes cambios en la posición correspondiente. En el método `onClick()` y dentro de la cláusula `try`, localizar la siguiente línea recién cambiada:

```
jtv2.setText("El hilo se bloqueó " + n + " segundos...");
```

Y cambiarla por el siguiente bloque:

```
runOnUiThread(new Runnable() {
   @Override public void run() {
      jtv2.setText("El hilo se bloquea durante "+ n + " segundos");
   }
});
```

Quedando como se indica enseguida:

```
public void onClick(View v) {
    try{
```

```
        final int n = Integer.parseInt(jet1.getText().toString());
        new Thread(new Runnable(){
            public void run() {
                try {
                    runOnUiThread(new Runnable() {
                        @Override public void run() {
                            jtv2.setText("El hilo se bloquea durante "+ n + " segundos");
                        }
                    });
                    Thread.sleep(n*1000);
                } catch (InterruptedException ie) {}
            }
        }).start();
    }catch (NumberFormatException e){
        Toast.makeText(this,"ingresar segundos...", Toast.LENGTH_SHORT).show();
    }
}
```

Paso 10. Por último, al ejecutar la aplicación, el hilo interior ahora sí interactúa con el hilo principal de la interface. Ingresar algún número de segundos y observar el mensaje:

EJEMPLO 2.

El siguiente ejercicio muestra la ejecución de dos hilos y el uso de sus métodos para extraer información acerca de ellos y de su ejecución.

Paso 1. Crear un nuevo proyecto Hilos2. En la actividad principal MainActivity.java, capturar el siguiente código:

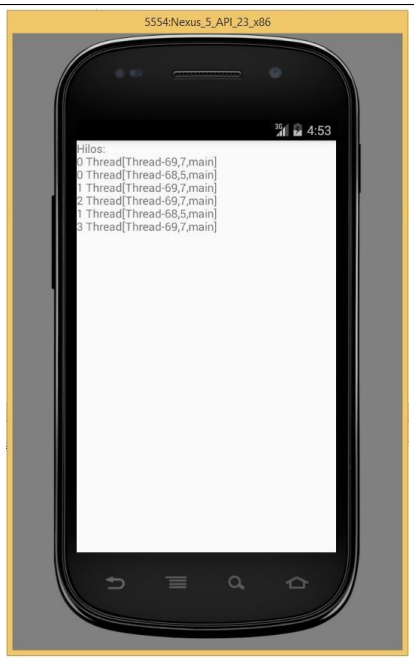
```
import android.app.*;
import android.os.*;
import android.widget.*;
public class MainActivity extends Activity {
    Handler       h = new Control();
    TextView      jtv1;
    @Override
    protected void onCreate(Bundle b) {
        super.onCreate(b);
        setContentView(R.layout.activity_main);
```

```
        jtv1=(TextView) findViewById(R.id.xtv1);
        Hilo h1 = new Hilo(10, 1000);     // 1seg = 1000ms
        Hilo h2 = new Hilo( 5, 500);
        h2.setPriority(7);
        h1.start();
        h2.start();
    }
    class Hilo extends Thread{
        int n, t;
        Message m;
        Bundle b;
        Hilo(int n, int t){
            this.n = n;
            this.t = t;
        }
        public void run(){
            for(int i=0; i<n; i++){
                try{
                    Thread.sleep(t);
                }catch(InterruptedException ie){}
                m = h.obtainMessage();
                b = new Bundle();
                b.putInt("cuenta", i);                              //cuenta=i
                b.putString("hilo", currentThread().toString());//hilo=currentThread()
                m.setData(b);
                h.sendMessage(m);
            }
        }
    }
    class Control extends Handler{
        public void handleMessage(Message m){
            int n;
            String s;
            n = m.getData().getInt("cuenta");                    //cuenta=i
            s = m.getData().getString("hilo");                   //hilo=currentThread()
            jtv1.append("\n" + n + " " + s);
        }
    }
}
```

Paso 2. En el archivo activity_main.xml predeterminado, capturar el siguiente código:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <TextView
        android:id="@+id/xtv1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hilos:" />
</RelativeLayout>
```

Paso 3. Por último, ejecutar la aplicación. Se muestra la plantilla del proyecto con la ejecución de los hilos, línea por línea. Observar los datos mostrados para cada hilo.



EJERCICIO 1.

Realizar los siguientes cambios al Ejemplo 2. En el archivo MainActivity.java:

a. Asignar un nombre a cada hilo antes de iniciarlos, por ejemplo:

```
h1.setName("HILO 1");
h2.setName("HILO 2");
```

b. Localizar y modificar la siguiente línea para mostrar el nombre del hilo:

```
b.putString("hilo", currentThread().getName().toString());
```

c. Las imágenes debe ser similares a las siguientes:

d. Obtener y mostrar el ID de cada hilo.

e. Cambiar la temporización de cada hilo para mostrar la ejecución de cada hilo.

NOTA. Obtener las imágenes de la ejecución de las aplicaciones y generar un reporte. Guardar el archivo con la sintaxis AlumnoTarea21Grupo.pdf. Enviarlo al sitio indicado por el profesor.