Simple Thread Example in Delphi

```
type
  TMyThread = class(TThread)
  private
    FActive: Bool;
    FSomeList: TStringList;
    procedure SYNC_Something;
    procedure SYNC_Started;
    procedure SYNC_Stopped;
    procedure SetActive(const Value: Bool);
  protected
    procedure Execute; override;
  public
    constructor Create;
    destructor Destroy; override;
    property Active: Bool read FActive write SetActive;
  end;


implementation


constructor TMyThread.Create;
begin
  inherited Create(True); //You practically always need to pass 'True' because 'False' is just meaningless...
  FSomeList:= TStringList.Create;
  Resume; //Now that everything is created, we officially start the thread.
end;

destructor TMyThread.Destroy;
begin
  FActive:= False; //Make sure our process is stopped...
  WaitFor; //Wait for the process to stop before we continue freeing things...
  FSomeList.Free;
  inherited;
end;

procedure TMyThread.SetActive(const Value: Bool);
begin
  //Not necessary to be a procedure setter, but you might want to do some preparation at this point...
  FActive:= Value;
end;

procedure TMyThread.Execute;
var
  X: Integer;
begin
  //Everything done inside here, and any procedures called from inside here, are done from within this thread...
  while not Terminated do begin //Keep looping at all times until thread is terminated...
    if FActive then begin //Should this thread be doing something?
      Synchronize(SYNC_Started);

      //START threaded code
      FSomeList.Clear;
      for X := 1 to 1000 do begin
        FSomeList.Append('Some Random Text');
        Synchronize(SYNC_Something);
      end;
      //END threaded code

      Synchronize(SYNC_Stopped);
    end else begin
      Sleep(1); //This keeps the thread from going crazy and maxing the processor...
    end;
  end;
end;

procedure TMyThread.SYNC_Something;
```

```
begin
  //Now we can do something which accesses the outside of the thread
  //This would presumably trigger an event notifying the main thread that something has been done or needs to
be done
end;

procedure TMyThread.SYNC_Started;
begin
  //Now we can do something which accesses the outside of the thread
  //This would presumably trigger an event notifying the main thread that this thread has begun its work
end;

procedure TMyThread.SYNC_Stopped;
begin
  //Now we can do something which accesses the outside of the thread
  //This would presumably trigger an event notifying the main thread that this thread has finished its work
end;
```

Following is the example of running thread:

```
procedure TMainForm.Button1Click(Sender: TObject);
begin
  TMyThread.Create(false);
end;
```