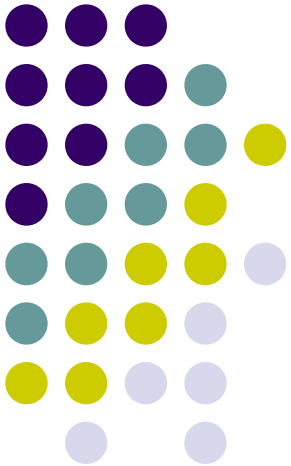
Android Programming

05 – Dealing with Threads



Limitations

- The classic threads used in JavaSe still can be used
- The problem is a runnable object is unable to affect the UI thread
- In order to create runnable tasks to affect UI, handlers or Observer Pattern must be used
- From UI thread it is not possible to create HTTP requests, these requests can only be created in threads.

Using Handlers

- **Handler** subclass used
- A single handler per activity is usually enough
- Handlers are able to send and receive Message objects
  - In order to send a message a new message object should be obtained from the queue of handler
  - messages can contain objects
  - sendMessage() is used to inform the handler
  - Handler should be subclassed and handleMessage() must be overridden

Coding Handlers

Creating the Handler

```
Handler messageHandler = new Handler(){
        @Override
        public void handleMessage(Message msg) {
            txt.setText(String.valueOf(msg.what));
        }

    };
```

Sending message from a thread

```
 Thread t = new Thread(new Runnable() {
     @Override
     public void run() {
          while(isRunning){
               try {
               Thread.sleep(1000);
               } catch (InterruptedException e) {
               e.printStackTrace();
               }
               count++;
               messageHandler.sendEmptyMessage(count);
               }
          }
});
t.start();
```

Using Observer Pattern - Listeners

```
public class CounterComponent {

    CountListener listener;
    CountThread t;



    private int count = 0;
    public void startCount(ExecutorService srv,
CountListener listener){
        this.listener = listener;

        t = new CountThread();
        srv.execute(t);


    }


   class CountThread implements Runnable{

       @Override
       public void run() {
           count =0;
            while (count<100){

                count++;
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                listener.countUpdate(count);

            }
       }

   }

}
```

```
public interface CountListener{
        public void countUpdate(int value);
    }
```

in activity:

```
btnStart.setOnClickListener(v->{

          comp = new CounterComponent();
          comp.startCount(((SampleApplication)
getApplication()).getExecutorService(),new
CounterComponent.CountListener() {
              @Override
              public void countUpdate(int value) {

txtCount.setText(String.valueOf(value));
              }
          });

       });
```