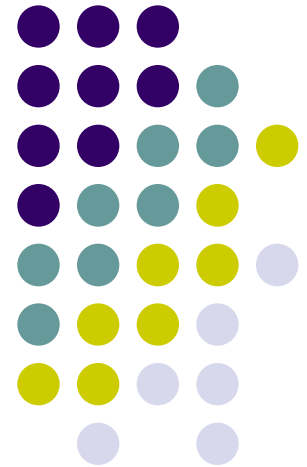


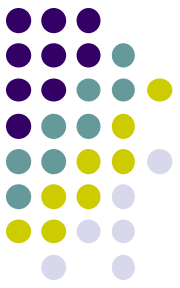
Mobile Application Development

Background Tasks in Android Handler

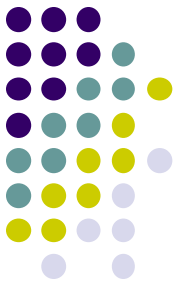
**MOBILE APPLICATION
DEVELOPMENT**



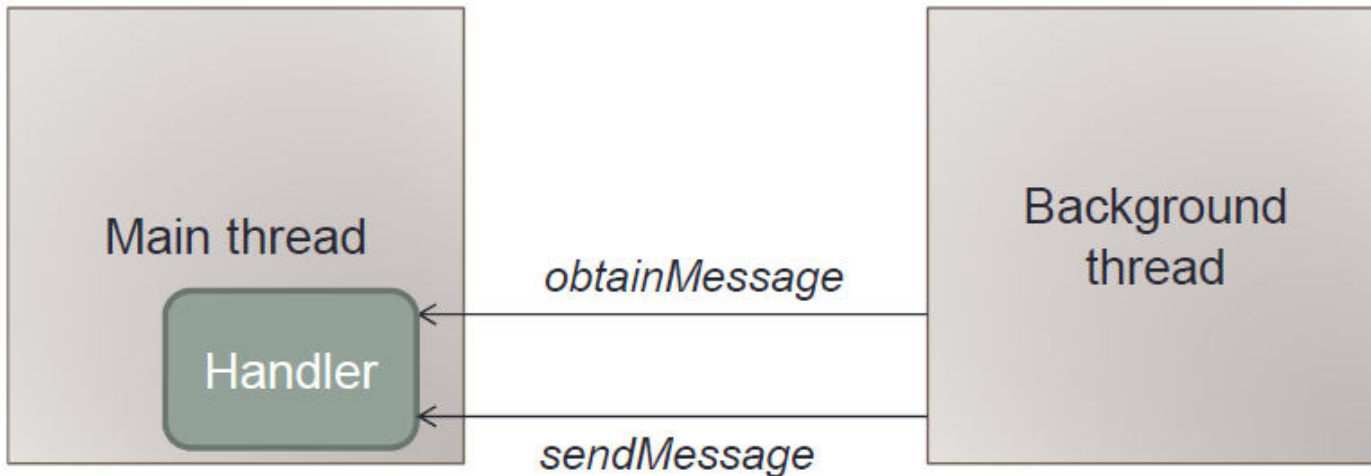
Android's Approach to Slow Activities



- **By default**, **app** runs in the **main thread**.
 - Every statement is executed in sequence.
 - If an **app** perform a long lasting operation, the **app blocks** until the corresponding operation has finished.
- **To provide a good user experience**
 - all slow running operations should run asynchronously.
 - This can be archived via **concurrency processing**.
- **Example:**
 - **Potentially slow operations are** network, file and database access and complex calculations.
- **Android enforces a worst case reaction time of applications.**
 - If an **activity** does not react within 5 seconds to user input, the Android system displays an **Application not responding (ANR) dialog**. From this dialog the user can choose to stop the application.

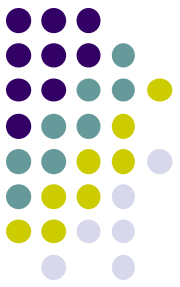


Interacting with the UI

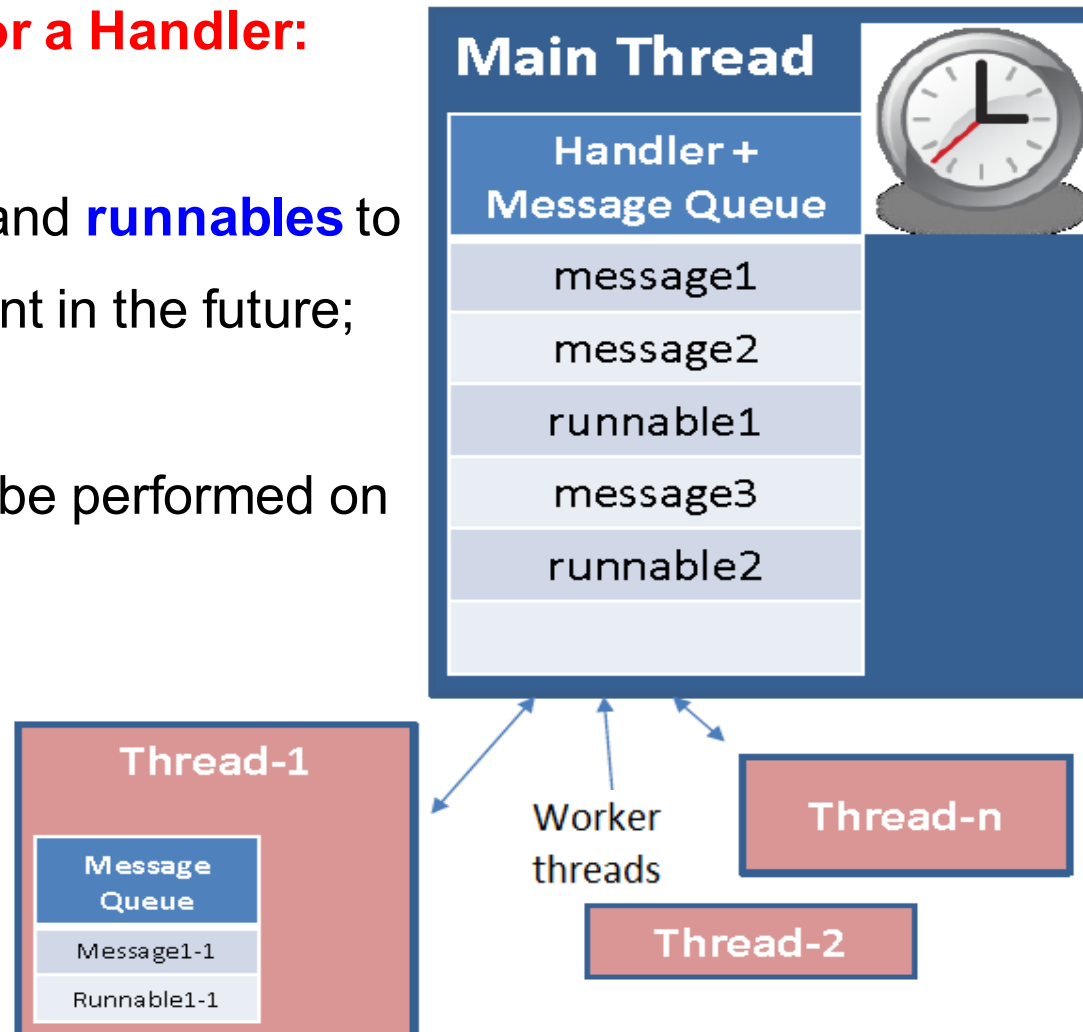


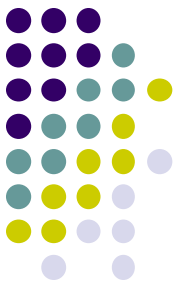
- The **UI thread** creates a **Handler object** internal to itself
- The **working thread** uses this object to **obtain** an **empty message** and **send** a **message** to the **UI thread**

Message based mechanism using Handler class



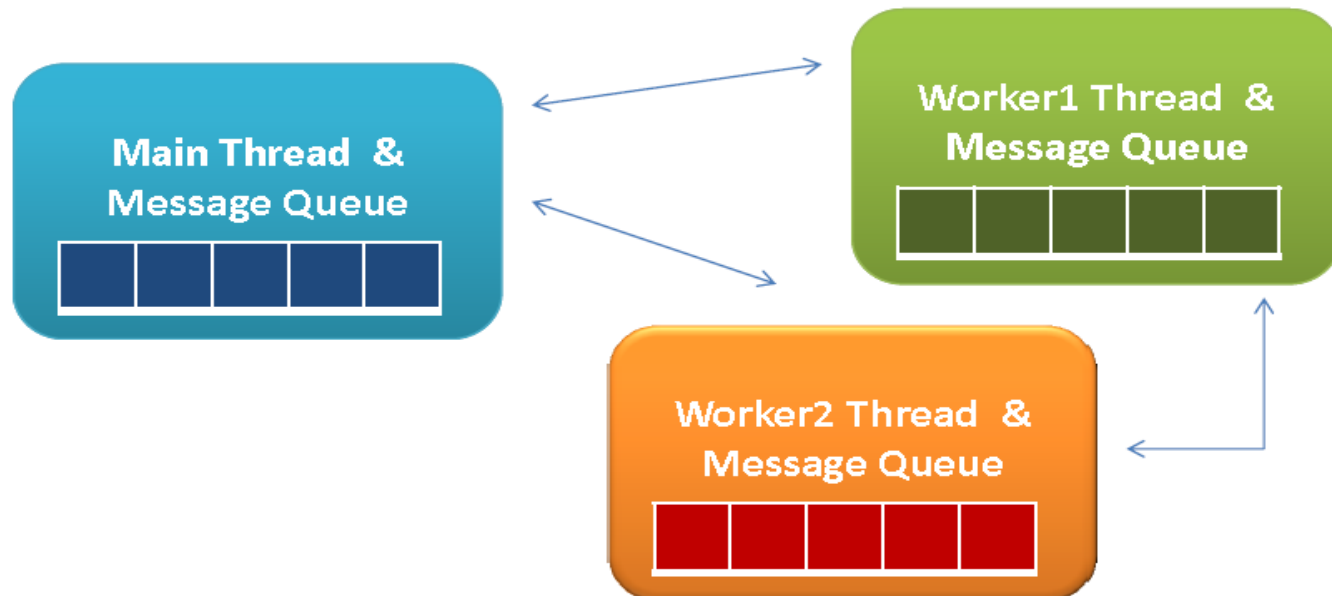
- **There are two main uses for a Handler:**
 - to schedule **messages** and **runnables** to be executed as some point in the future;
 - to **enqueue an action** to be performed on another thread



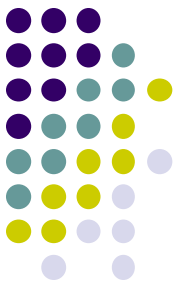


Inter-Thread Communications.

- **Typically** the **main UI thread** sets a handler to get **messages** from its worker threads; however *each worker thread could also define its own handler*.
- A handler in the worker thread creates a **local message-queue** which could be used to receive messages from other threads (**including main**).



Handler. Using Messages



Main Thread

```
...
Handler myHandler= new Handler() {

    @Override
    public void handleMessage(Message msg) {
        // do something with the message...
        // update GUI if needed!
        ...
    }//handleMessage

};//myHandler
...
```

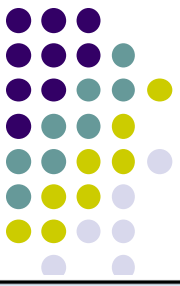
Background Thread

```
...
Thread backgJob = new Thread (new Runnable (){
    @Override
    public void run() {

        // do some busy work here
        // ...
        // get a token to be added to
        // the main's message queue
        Message msg= myHandler.obtainMessage();
        ...
        // deliver message to the
        // main's message-queue
        myHandler.sendMessage(msg);
    }//run

});//Thread

// this call executes the parallel thread
backgroundJob.start();
...
```



Handler. Using Runnables

Main Thread

```
...
Handler myHandler = new Handler();
@Override
public void onCreate(Bundle
    savedInstanceState){
    ...
    Thread myThread1 = new Thread(
        backgroundTask,
        "backAlias1");
    myThread1.start();
} // onCreate
```

```
...
// this is the foreground runnable
private Runnable foregroundTask
    = new Runnable() {
    @Override
    public void run() {
        // work on the UI if needed
    }
}
...
```

Background Thread

```
// this is the "Runnable" object
// representing the background thread

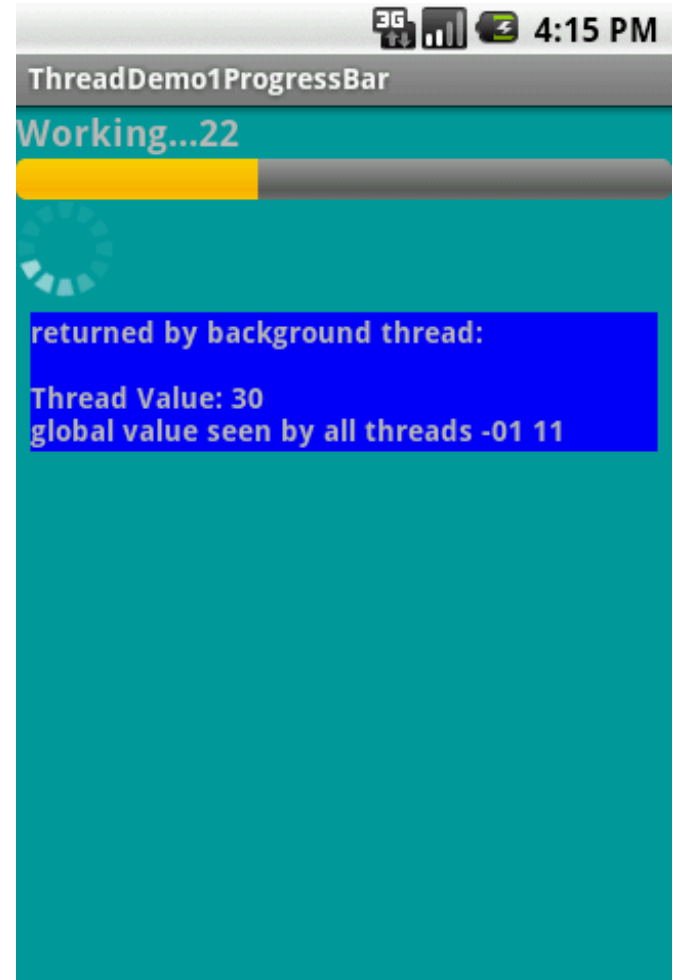
private Runnable backgroundTask
    = new Runnable () {
    @Override
    public void run() {
        // Do some background work here
        myHandler.post(foregroundTask);
    }
} // run
}; // backgroundTask
```

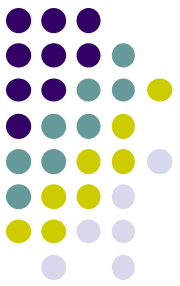
Example 1.

Using Message Passing



- The **main thread** displays a **horizontal** and a **circular progress bar** widget showing the progress of a slow background operation.
- Some random data is periodically sent from the background thread and the messages are displayed in the **main view**.





public class MainActivity extends Activity {

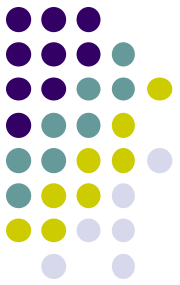
```
ProgressBar    bar1;
ProgressBar    bar2;
TextView       msgWorking;
TextView       msgReturned;
boolean        isRunning= false;
final int      MAX_SEC= 60; // (seconds) lifetime for background thread
String         strTest= "global value seen by all threads ";
int            intTest= 0;
```

```
Handler handler = new Handler() {
```

```
@Override
```

```
public void handleMessage(Message msg) {
```

```
    String returnedValue= (String)msg.obj;
    //do something with the value sent by the background thread here ...
    msgReturned.setText("returned by background thread: \n\n" + returnedValue);
    bar1.incrementProgressBy(2);
    if(bar1.getProgress() == MAX_SEC){ //testing thread's termination
        msgReturned.setText("Done \n back thread has been stopped");
        isRunning= false;
    }
```



```
if(bar1.getProgress() == bar1.getMax()){
    msgWorking.setText("Done");
    bar1.setVisibility(View.INVISIBLE);
    bar2.setVisibility(View.INVISIBLE);
}
else{
    msgWorking.setText("Working..." + bar1.getProgress() );
}
} // handleMessage method
}; //handler
```

@Override

```
public void onCreate(Bundle icicle) {
```

```
    super.onCreate(icicle);
```

```
    setContentView(R.layout.activity_main);
```

```
    bar1= (ProgressBar) findViewById(R.id.progress);
```

```
    bar2= (ProgressBar) findViewById(R.id.progress2);
```

```
    bar1.setMax(MAX_SEC);
```

```
    bar1.setProgress(0);
```

```
    msgWorking= (TextView) findViewById(R.id.TextView01);
```

```
    msgReturned= (TextView) findViewById(R.id.TextView02);
```

```
    strTest+= "-01"; // slightly change the global string
```

```
    intTest= 1;
```

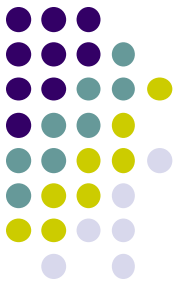
```
} //onCreate
```

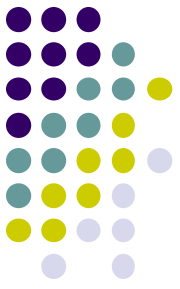
```
public void onStop() {
```

```
    super.onStop();
```

```
    isRunning= false;
```

```
}
```





```
public void onStart() {
```

```
    super.onStart();
```

```
    Thread background = new Thread(new Runnable() {
```

```
        public void run() {
```

```
            try{
```

```
                for(int i = 0; i < MAX_SEC && isRunning; i++) { // Toast method will not work!
```

```
                    Thread.sleep(1000); //one second at a time
```

```
                    Random rnd= new Random();
```

```
                    String data = "Thread Value: "+ (int) rnd.nextInt(101);
```

```
                    data += "\n"+ strTest+ " "+ intTest; //we can see and change (global) class vars
```

```
                    intTest++; //request a message token and put some data in it
```

```
                    Message msg= handler.obtainMessage(1, (String)data);
```

```
                    if(isRunning)
```

```
                        { handler.sendMessage(msg); // if thread is still alive send msg }
```

```
                    }
```

```
                } catch(Throwable t) {}
```

```
            } //run
```

```
        }); //background
```

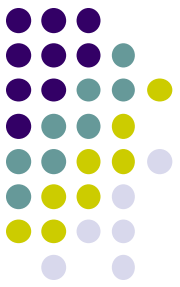
```
        isRunning= true;
```

```
        background.start();
```

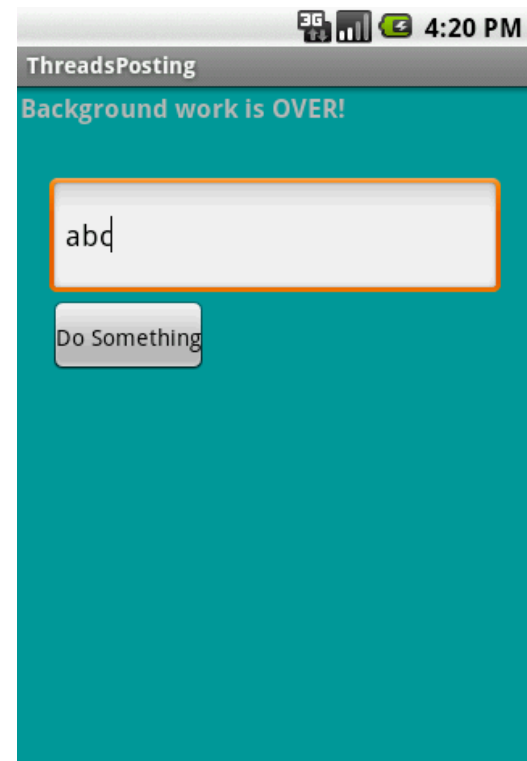
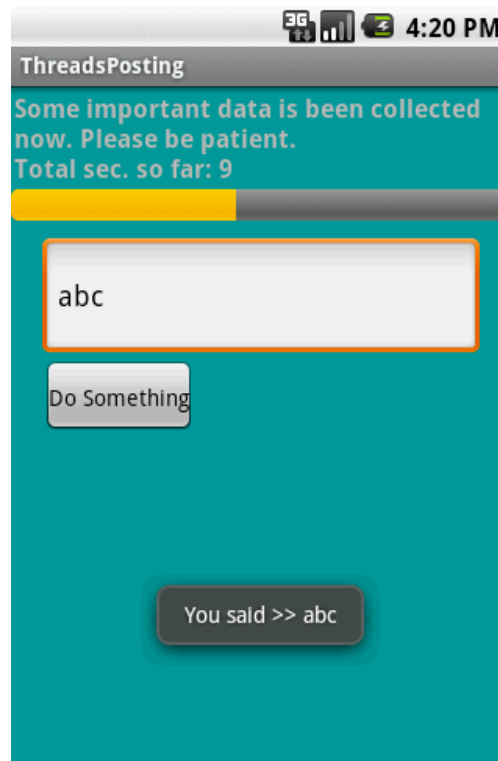
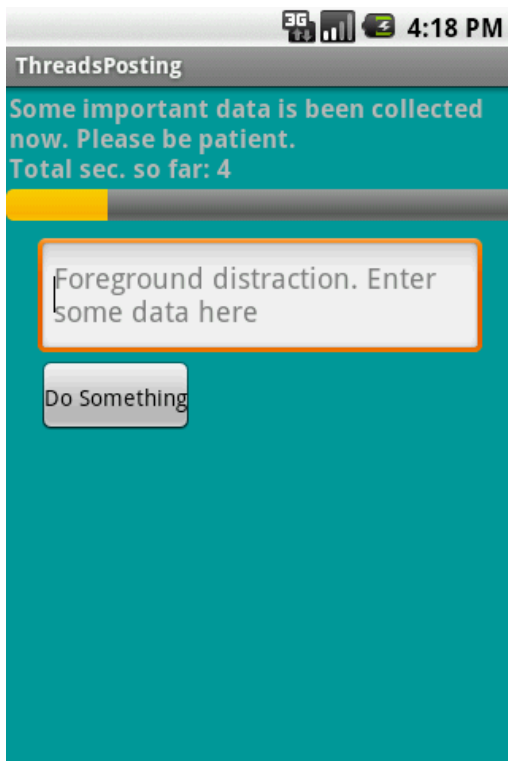
```
    } //onStart
```

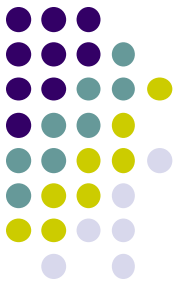
```
}
```

Example 2. Using Runnable



- We will try the same problem presented earlier (a slow background task and a responsive foreground UI) this time using the **posting mechanism** to execute foreground *runnables*.

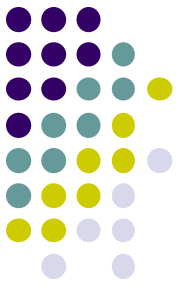




```
public class MainActivity extends Activity {
```

```
    ProgressBar    myBar;  
    TextView       lblTopCaption;  
    EditText       txtBox1;  
    Button         btnDoSomething;  
    int            accum= 0;  
    long           startingMills= System.currentTimeMillis();  
    String         PATIENCE= "Some important data is been collected now. "+ "\nPlease be  
    patient.";
```

```
    Handler myHandler= new Handler();
```



@Override

```
public void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.activity_main);
```

```
    lblTopCaption= (TextView) findViewById(R.id.lblTopCaption);
```

```
    myBar= (ProgressBar) findViewById(R.id.myBar);
```

```
    myBar.setMax(100);
```

```
    txtBox1= (EditText) findViewById(R.id.txtBox1);
```

```
    txtBox1.setHint("Foreground distraction. Enter some data here");
```

```
    btnDoSomething= (Button) findViewById(R.id.btnDoSomething);
```

```
    btnDoSomething.setOnClickListener(new OnClickListener() {
```

```
        @Override
```

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

```
                Editable txt = txtBox1.getText();
```

```
                Toast.makeText(getApplicationContext(), "You said >> "+ txt, LENGTH_LONG).show();
```

```
            } //onClick
```

```
        }); //setOnClickListener
```

```
    } //onCreate
```

@Override

protected void onStart() {

super.onStart(); // create background thread where the busy work will be done

Thread myThread1 = new Thread (backgroundTask, "backAlias1");

myThread1.start();

myBar.incrementProgressBy(0);

}

private Runnable foregroundTask= new Runnable() {

@Override // foreground "Runnable" object responsible for GUI updates

public void run() {

try{

int progressStep= 5;

lblTopCaption.setText (PATIENCE+ "\nTotalsec. so far: "+

(System.currentTimeMillis() -startingMills) / 1000);

myBar.incrementProgressBy(progressStep);

accum+= progressStep;

if(accum>= myBar.getMax()){

lblTopCaption.setText("Background work is OVER!");

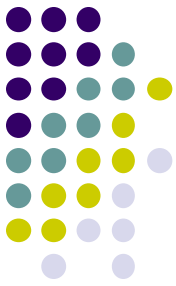
myBar.setVisibility(View.INVISIBLE);

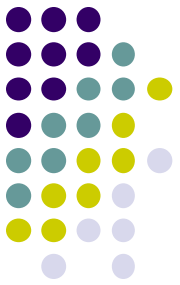
}

} catch(Exception e) { }

} //run

}; //foregroundTask





//this is the "Runnable" object that executes the background thread

```
private Runnable backgroundTask= new Runnable() {
```

```
    @Override
```

```
        public void run() {
```

```
            //busy work goes here...
```

```
            try{
```

```
                for(int n=0; n<20; n++) {
```

```
                    Thread.sleep(1000);
```

```
                    myHandler.post (foregroundTask);
```

```
                }
```

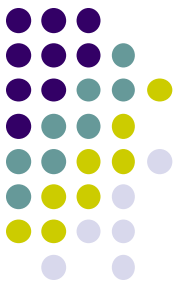
```
            } catch(InterruptedException e) { }
```

```
        } //run
```

```
    }; //backgroundTask
```

```
} //MainActivity
```

Passing data through the message

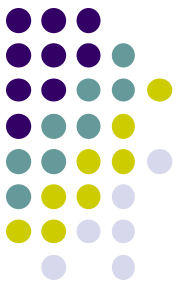


Three steps to send data in (Background thread)

- Create a **bundle**
- Put data in
- **Set** the data field of the **message**

```
Bundle b = new Bundle();  
b.putInt("int", 10);  
Message msg = handler.obtainMessage();  
msg.setData(b);  
handler.sendMessage(msg);
```

Passing data through the message (Cont.)

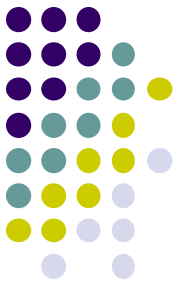


Two steps to receive data in (UI thread)

- Get **bundle** from the message
- Get the data

```
Bundle b = msg.getData();  
int i = b.getInt("int");  
tv.setText(tv.getText()+ "." + i);
```

References



- **Background tasks in Android**
- **Threads in Java**
- **Android – Threads**