Introduction to Android Concurrence task

CS 436 Software Development on Mobile

Dr.Paween Khoenkaw

Department of Computer Sciece
Maejo University

Why do we need concurrence task ?

Concurrence task

Case study: find prime at x

```
private long findPrime(long x){
  long i=3,c=0;
     while(true)
              {
              if(isPrime(i)==true)
              {
              c++;
                 if(c==x)return i;
              }
              i++;}}
```

```
private boolean isPrime(long x){
  for(long i=2;i<x;i++)
   {
      if((x%i)==0)
       {
         return false;
       }
    }
     return true;
}
```

```
public void onClick(View v) {
textview1.setText(String.format("%d",
findPrime(Long.valueOf(edittext1.getText().toString())));
            }
```

Project: prime_nothread

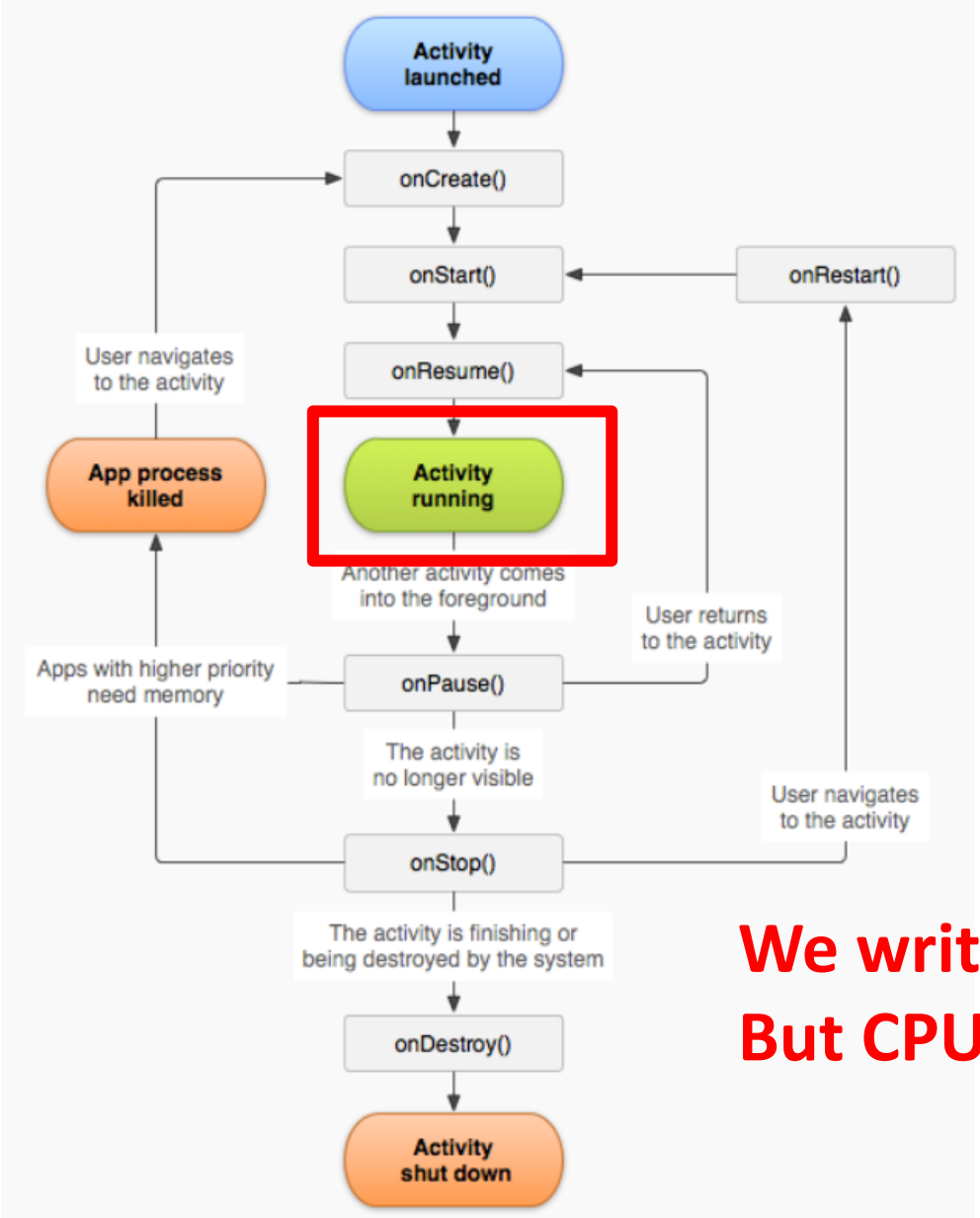
Concurrence task

This is block mode operation

ANR in com.example.prime_nothread (com.example.prime_nothread/.MainActivity)
Reason: keyDispatchingTimedOut
Load: 0.87 / 0.32 / 0.18
CPU usage from 48234ms to 0ms ago:
99% 577/com.example.prime_nothread: 99% user + 0% kernel

100% TOTAL: 94% user + 5.2% kernel

Concurrence task



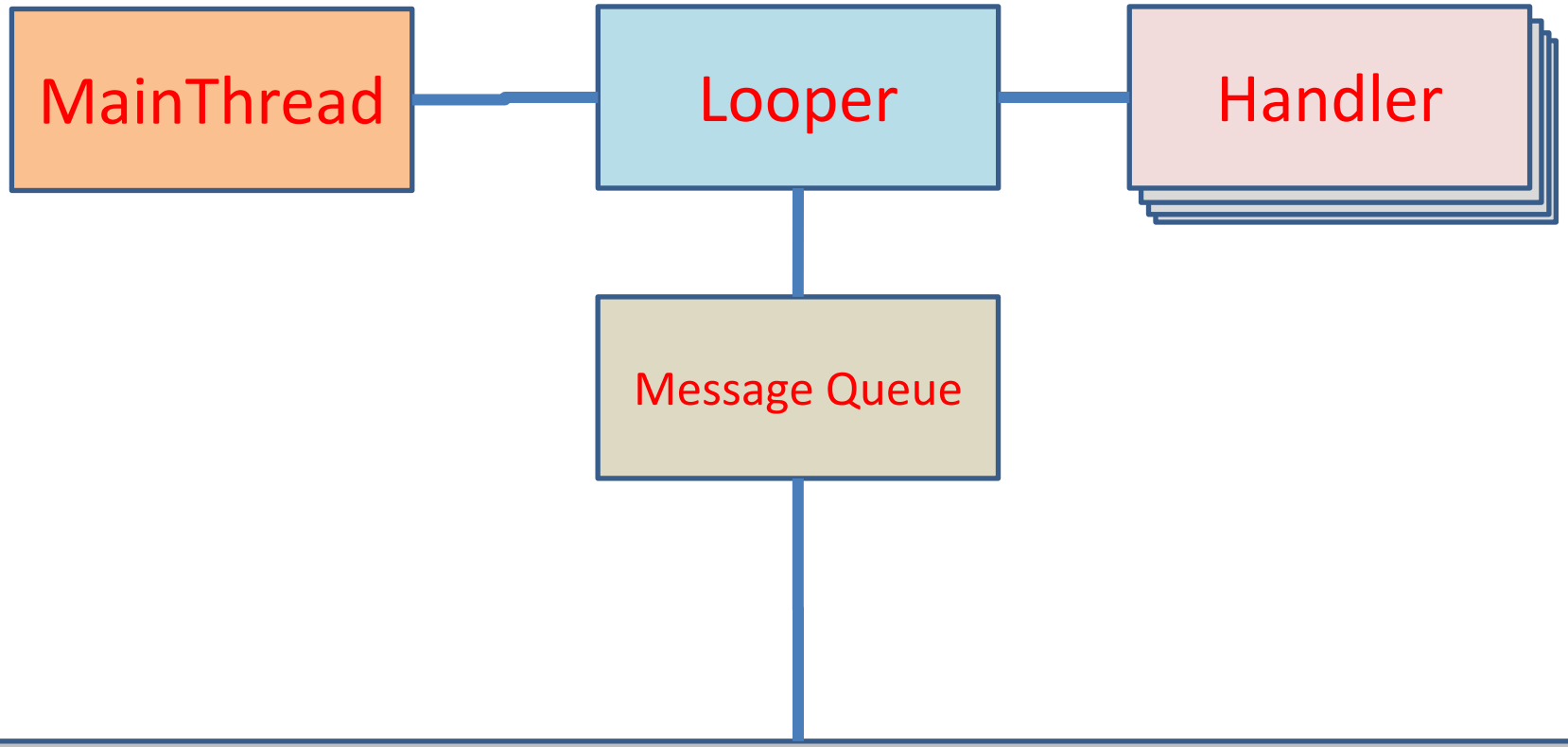
Activity lifecycle

We written in event driven model

But CPU is TuringMachine

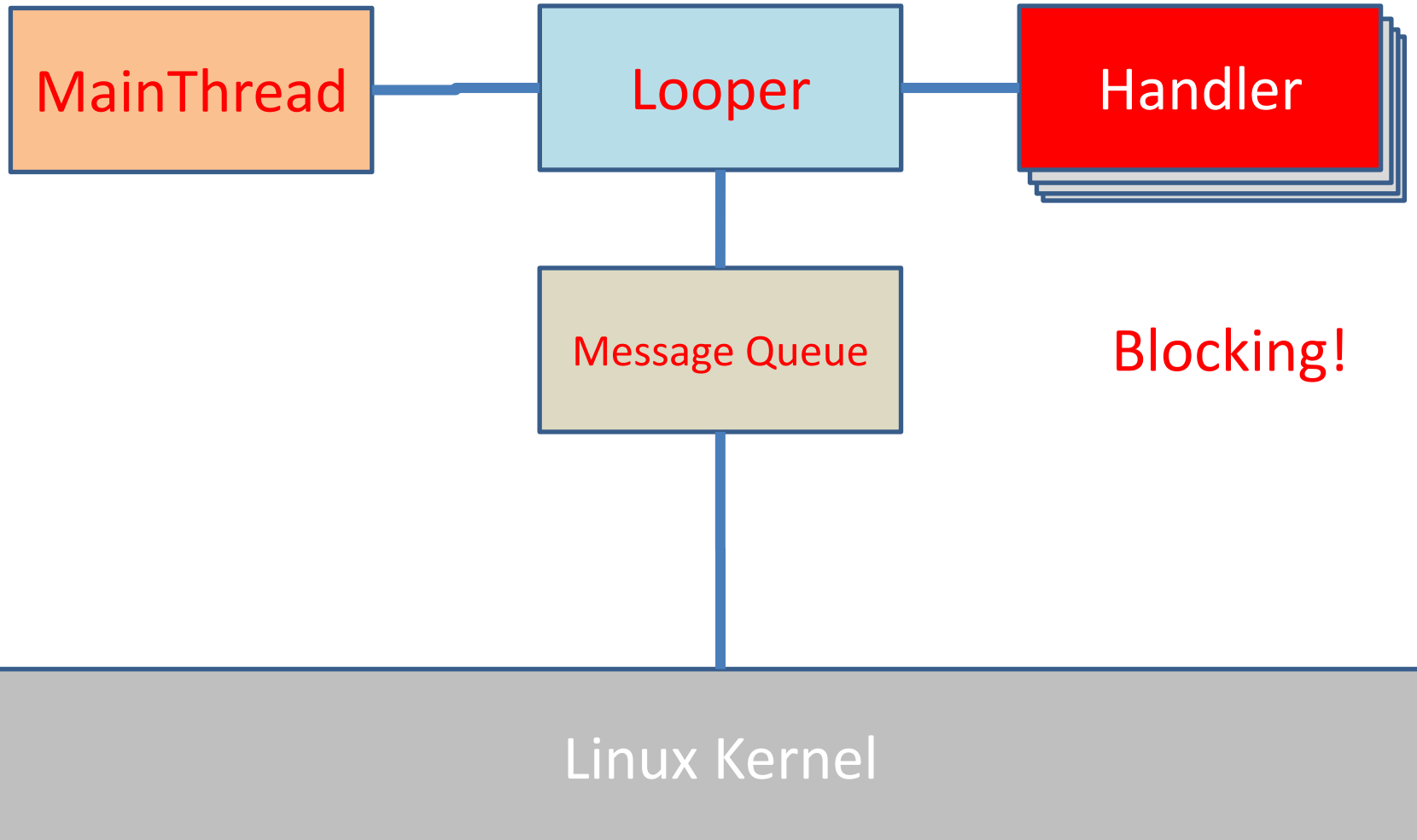
Concurrence task

Android Event Loop Model



Concurrence task

Android Event Loop Model



Concurrence task

1) Java read buffer in block mode
2) You cannot block the GUI thread for too long in Android
3) Android strict network operation in Gui thread since Honeycomb

Concurrence task

Java Thread / Asynctask / Intent Service

Concurrence task

Java Thread

Concurrence task

Thread VS Runnable

Thread

- Class
- No multiple inheritance
- Destroyed after complete
- Large overhead
- Ready to run
- More control
- Advance task

Runnable

- Interface
- Multiple implement
- Reusable
- Small footprint
- Need thread class to run
- No control Just run
- Simple task

Concurrence task:Thread

Java Runnable interface

```
public class primeFinderThread implements Runnable{
@Override
public void run() {
updater.x=String.format("%d",
findPrime(Long.valueOf(edittext1.getText().toString())));
}
```

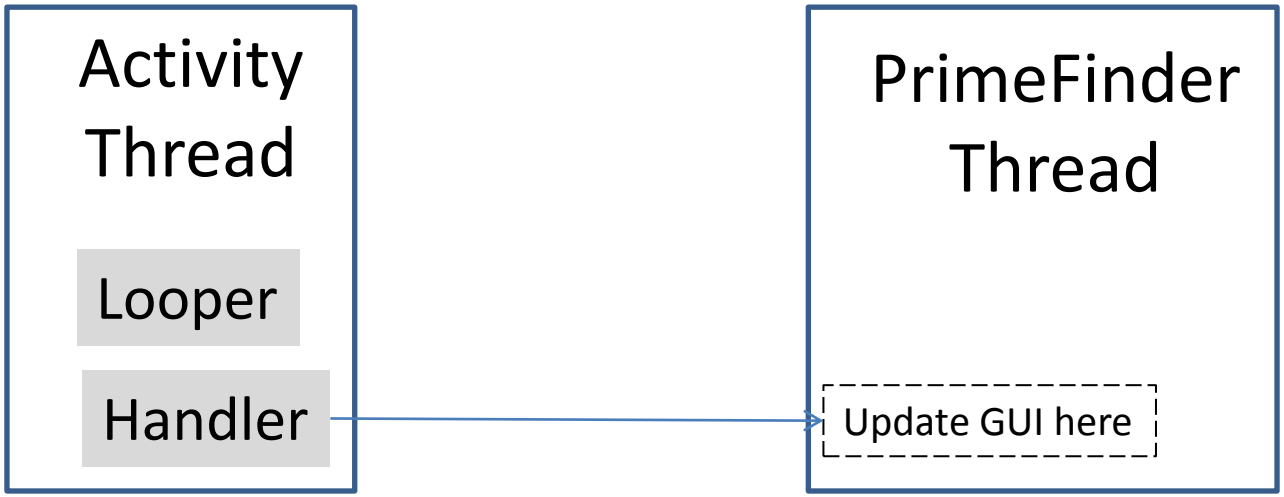
```
final primeFinderThread worker=new primeFinderThread();
button1.setOnClickListener(new View.OnClickListener() {
                                @Override
                                public void onClick(View v) {
                                        new Thread(worker).start();
                                }
                        });
```

Project: Prime_thread

Concurrence task:Thread

Java Runnable interface

Update GUI thread



Concurrence task:Thread

Java Runnable interface

```
// OnCreate
Handler h;
h=new Handler();
```

```
private class UpdateText implements Runnable{
public String x;
@Override
public void run() {
textview1.setText(x);
                                    }}
```

```
public class primeFinderThread implements Runnable{
UpdateText updater=new UpdateText();
@Override
public void run() {
updater.x=String.format("%d",
findPrime(Long.valueOf(edittext1.getText().toString())));
h.post(updater);
}
```

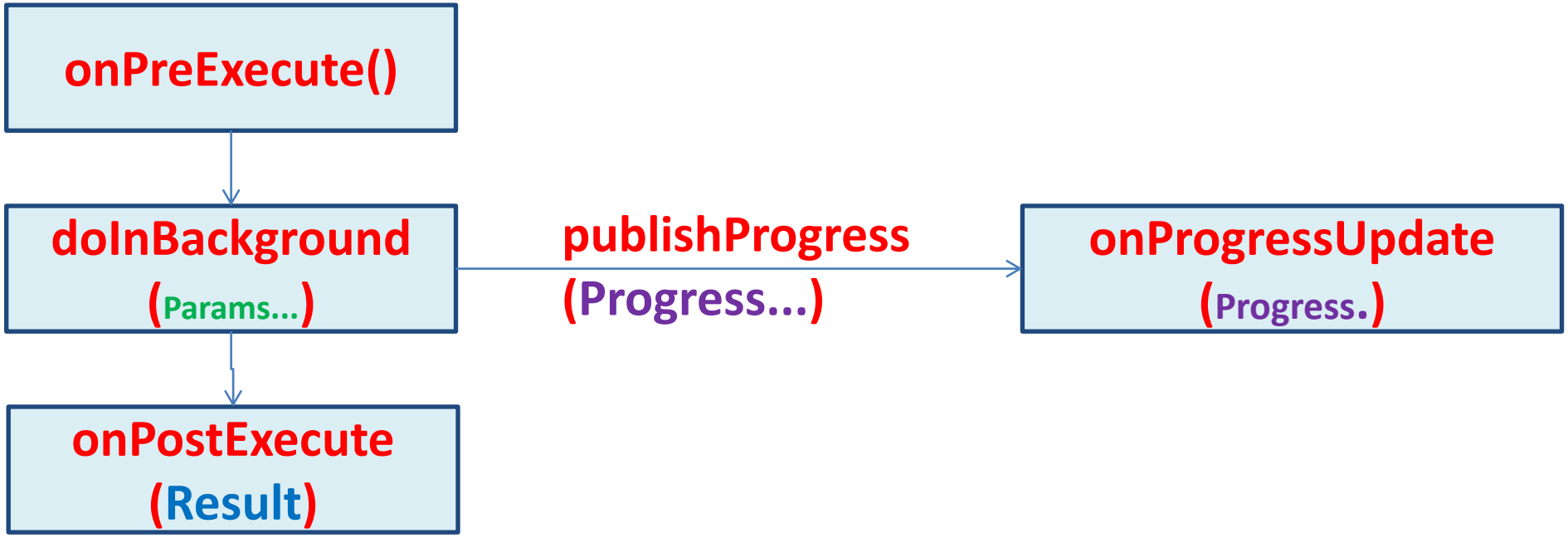
Concurrence task

Asynctask

AsyncTask is designed to be a helper class around Thread and Handler and does not constitute a generic threading framework.

Concurrence task

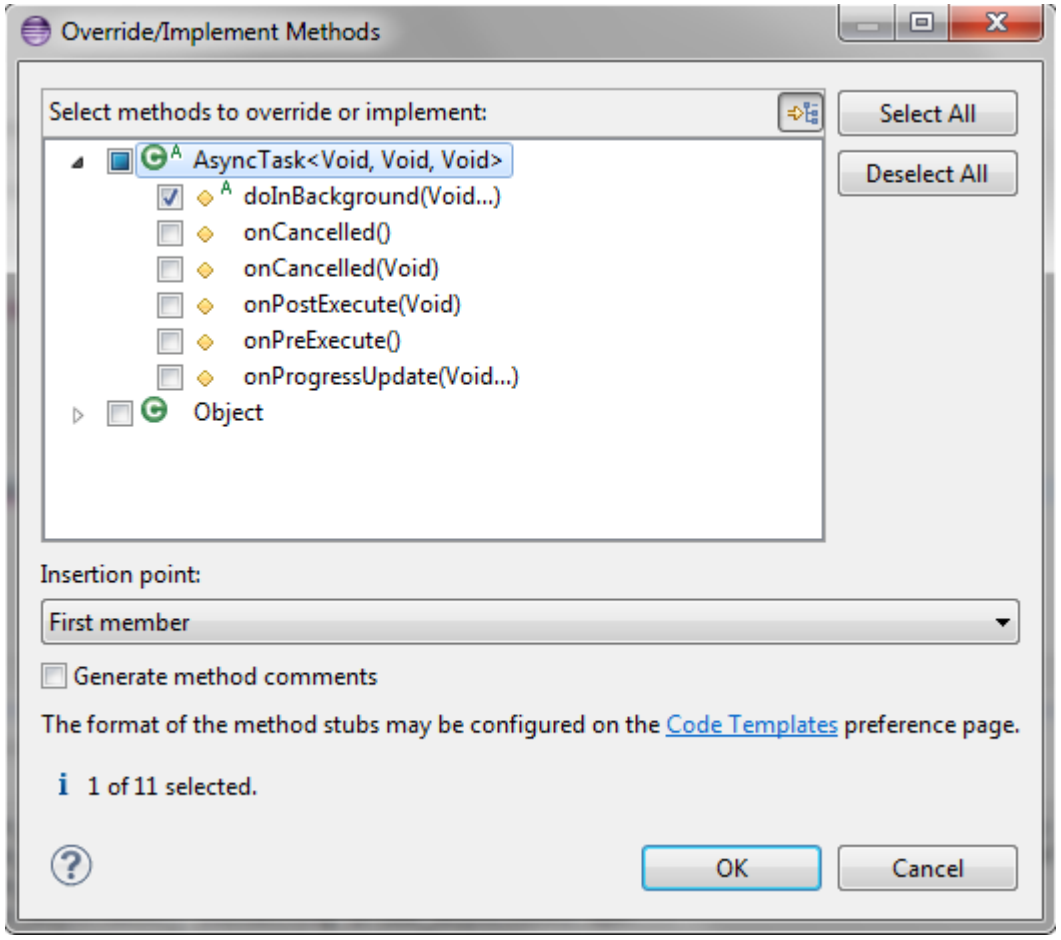
Asynctask



Concurrence task

Asynctask

private class MyTask extends AsyncTask<Void, Void, Void> { ... }



Concurrence task

Asynctask

```
public class primeFinder extends AsyncTask<Long,Long,Long>{
@Override
protected void onProgressUpdate(Long... values) {
super.onProgressUpdate(values);
textview1.setText(String.format("%d primes was found! ",values[0] ));
}
@Override
protected void onPostExecute(Long result) {
super.onPostExecute(result);
textview1.setText(String.format("%d",result ));
}
@Override
protected Long doInBackground(Long... params) {
return findPrime(params[0]);
}
                    private long findPrime(long x){....}
                    private boolean isPrime(long x){...}
}
```

Project: Prime_asynctask

Concurrence task

Asynctask

```
@Override
public void onClick(View v) {
primeFinder worker=new primeFinder();
worker.execute(Long.valueOf(edittext1.getText().toString()));
                                    }
```

Project: Prime_asynctask

Concurrence task

Intent Service

IntentService is a base class for Services that handle asynchronous requests (expressed as Intents) on demand. Clients send requests through startService(Intent) calls

Concurrence task

Intent Service

1) Create service class by extends IntentService
2) Register service in AndroidManifest
3) Start service using intent
4) Communicate with main thread using broadcast receiver

Concurrence task

Intent Service

1) Create service class by extends IntentService

```
public class primeFinderService extends IntentService {
        public primeFinderService(String name) {
                super(name);
        }
        public primeFinderService() {
                super("primeFinderService");
        }
        @Override
        protected void onHandleIntent(Intent intent) {
                        long x=findPrime(intent.getLongExtra("data",5));
                        Intent broadcastIntent = new Intent();
                        broadcastIntent.setAction("primefinder is done the job");
                        broadcastIntent.addCategory(Intent.CATEGORY_DEFAULT);
                        broadcastIntent.putExtra("result", x);
                        sendBroadcast(broadcastIntent);
                }
                private long findPrime(long x){...}
                private boolean isPrime(long x){...}
  }
```

Project: Prime_intentservice

Concurrence task

Intent Service

1) Create service class by extends IntentService

```
private long findPrime(long x){
long i=3,c=0;
while(true)
 {
            if(isPrime(i)==true)
            {
           c++;
            if(c==x)return i;
            Intent broadcastIntent = new Intent();
broadcastIntent.setAction("primefinder update progress");
broadcastIntent.addCategory(Intent.CATEGORY_DEFAULT);
broadcastIntent.putExtra("result", c);
sendBroadcast(broadcastIntent);
            }
            i++;
}
}
```

Project: Prime_intentservice

Concurrence task

Intent Service

2) Register service in AndroidManifest

```
<service android:name="primeFinderService"></service>
```

Concurrence task

Intent Service

3) Start service using intent

```
primefinder = new Intent(this,primeFinderService.class);
primefinder.putExtra("data",Long.valueOf(edittext1.getText().toString()) );
startService(primefinder);
```

Concurrence task

Intent Service

4) Communicate with main thread using broadcast receiver

```
IntentFilter filter = new IntentFilter("primefinder is done the job");
    filter.addCategory(Intent.CATEGORY_DEFAULT);
    ResponseReceiver receiver = new ResponseReceiver();
    registerReceiver(receiver, filter);

 IntentFilter filter2 = new IntentFilter("primefinder update progress");
   filter2.addCategory(Intent.CATEGORY_DEFAULT);
   ProgressReceiver receiver2 = new ProgressReceiver();
   registerReceiver(receiver2, filter2);

public class ResponseReceiver extends BroadcastReceiver {
@Override
public void onReceive(Context arg0, Intent arg1) {
textview1.setText(String.format("%d", arg1.getLongExtra("result", 0)));
                              }}
public class ProgressReceiver extends BroadcastReceiver {\
@Override
public void onReceive(Context arg0, Intent arg1) {
textview1.setText(String.format("%d primes was found", arg1.getLongExtra("result", 0)));
                              }}
```

Concurrence task

Conclusion

-Java Thread

- -Long run background process
- -Use Thread class if advance process control is required
- -Use Runnable interface for simple task
- **-TCP/IP Socket**

-Asynctask

- -Background task with GUI update
- -Short operation process
- **-HTTP Post**

-Service

- -Long run background process (with thread)
- -May block main thread
- -Use IntentService if auto start is required
- **-TCP/IP Server**