

Java on Mobile Devices Feb 26

16:00 – ??: Threading and
J2ME Low Level User Interface



Uploading Examples

You may share your applications with others

<http://positionserver.itu.dk:8000>

LaCoMoCo
Mobility Lab



IT University of Copenhagen

Projects

Logged on:
Jørgen,
Staunstrup
Logout

Projects:

Id	Title	Created	Uploaded
33	Falling Coconuts	2004-02-25	✓
32	BattleField - A multiplayer mobile game	2004-02-19	✓
31	Software environment for outdoor positioning	2004-02-13	
30	Symbian 6.1 - GSM positioning	2003-11-04	
29	3D Visualization of ITU with WLAN localization	2003-10-14	
27	Dynamic audio in real life location-based games	2003-10-10	
26	Implementation of Location and Time based information exchange system	2003-02-01	

Mobility Lab
Projects

Bluetooth network
Tracked BtClients



How to Upload



Projects

Logged on:
Jørgen,
Staunstrup
Logout

Mobility Lab
Projects

Bluetooth network

Tracked BtClients

GSM Location

WEB API (TDC)

MAP API

GSM SDK

WLAN Location

WEB API

Java API

Clients

Maps

Example apps

Connected clients

Tracked clients

Title of project:

Falling Coconuts

Supervisor:

Jørgen Staunstrup

Supervisor email:

jst@itu.dk

Short abstract of project:

Very simple game with nice graphics. It is a variation of Falling Numbers

Project description - including type (thesis, 4 or 16-week etc.), members, links etc.):

Final project for the course Java on Mobile Devices Spring 2003.

Upload application (eg. zip or jar file)

C:\JSt\activ\JPMA\apps\FallingNumbers\bin\Fa

Gennemse...

(leave this field empty, if you don't wish to upload your project.)



Mobility Lab

LaCoMoCo
Mobility Lab



IT University of Copenhagen

Clients

Logged on:
Jørgen,
Staunstrup
Logout

Mobility Lab
Projects

Bluetooth network
Tracked BtClients

GSM Location
WEB API (TDC)
MAP API
GSM SDK

WLAN Location
WEB API
Java API
Clients
Maps
Example apps
Connected clients
Tracked clients

Profile box
My profile
My projects

The Mobility Lab is providing students and researchers different resources to create mobile-, ubiquitous- or context-dependent services. The resources are listed below.

Equipment available for student projects:

- Sony Ericsson P900
- Nokia 6600
- Nokia 9210i
- Nokia 7650
- Nokia N-gage
- Nokia 3410
- GSM Modem
- Garmin GPS reciever
(Contact your supervisor for further information about lending devices.)

Technolgy and infrastruicture:

• Bluetooth network

A bluetooth accesspoint is installed in room 3.10/3.11 to help students deploying midlets and applications to the phone. Click on the link to you phone and you are able to tranfer the application file to the phone.

Later on, a java api will be available for students to develop their own application using the bluetooth access points for mobile applications. Examples of applications could be location based applications (rssi) , payment applications, community based applications etc.

• GSM Location

LaCoMoCo helps you to create GSM location in two different ways; via cell-id (WEP API) and via triangulation.

The GSM WEB API is an API to request coordinates according to the position of your mobile phone in the TDC-GSM network. Click on the WEB API under GSM in the menu. Contact your supervisor for further information about creating triangulation projects.

• WLAN Location



Animation



nextframe



Threading

Multiple points of execution within a program (application)

Concurrency, multitasking, parallelism, multiprocessing, multiprogramming, coroutines, ...



A thread is a single sequential flow of control within a program

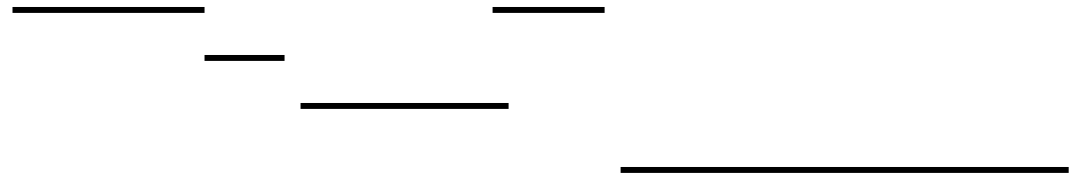


Multiple Threads

Abstraction



Reality



Threads in Java

```
class proc extends Thread {  
    public void run() {  
        ...  
        while ... { ... nextframe(); }  
    }  
}
```

```
Proc t= new proc(); t.start();
```



Animation Example

```
public void run() {  
    while (true) {  
        try { Thread.sleep(100); }  
        catch (InterruptedException ie) {}  
        s.nextFrame();  
        mLayerManager.paint(g, 35, 30);  
        flushGraphics();  
    }  
}
```



Example

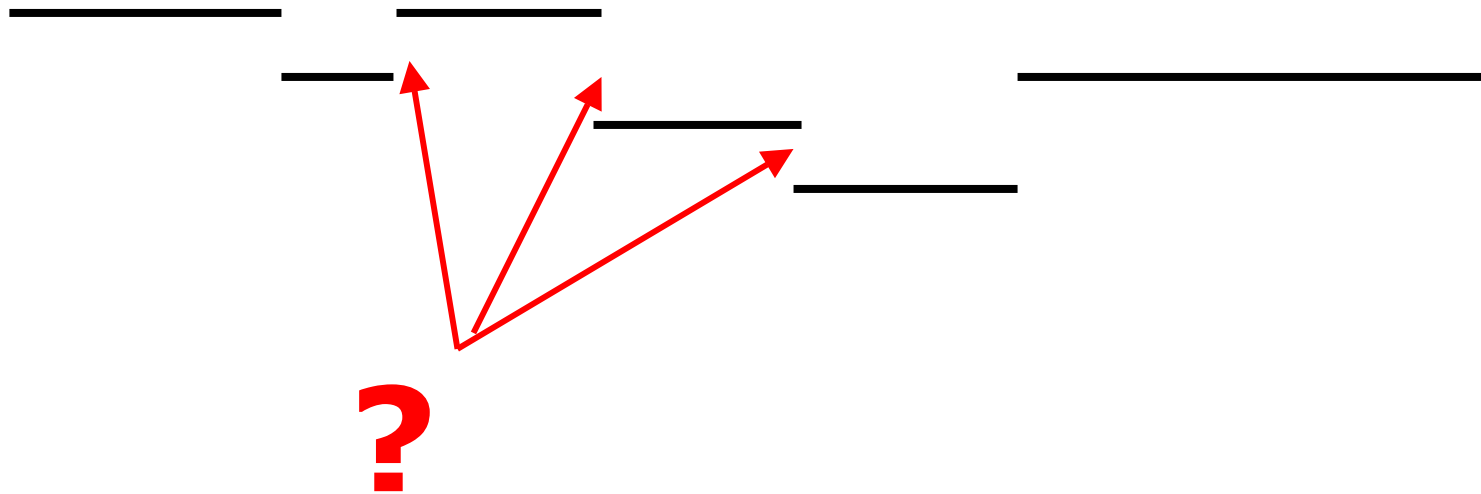
Find the earliest meeting time acceptable to every member of a group of people

t	8	9	10	11	12	13	14	15	16	17	18
F(t)	9	9	10	13	13	13	16	16	16	17	20

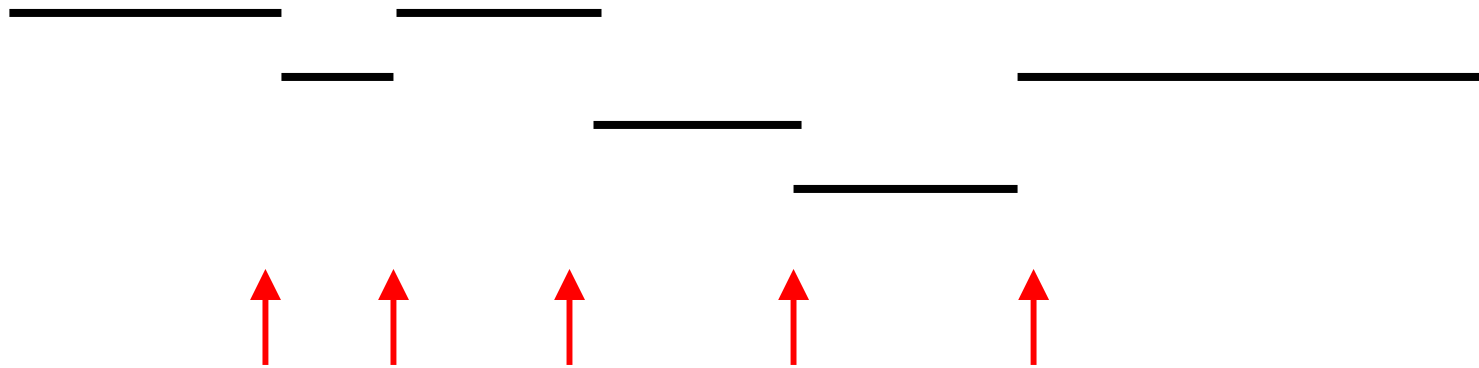
```
class person extends Thread {  
    public void run() {  
        do { if (m < F(m)) m = F(m) } while ( ... )  
    }  
}
```



Scheduling



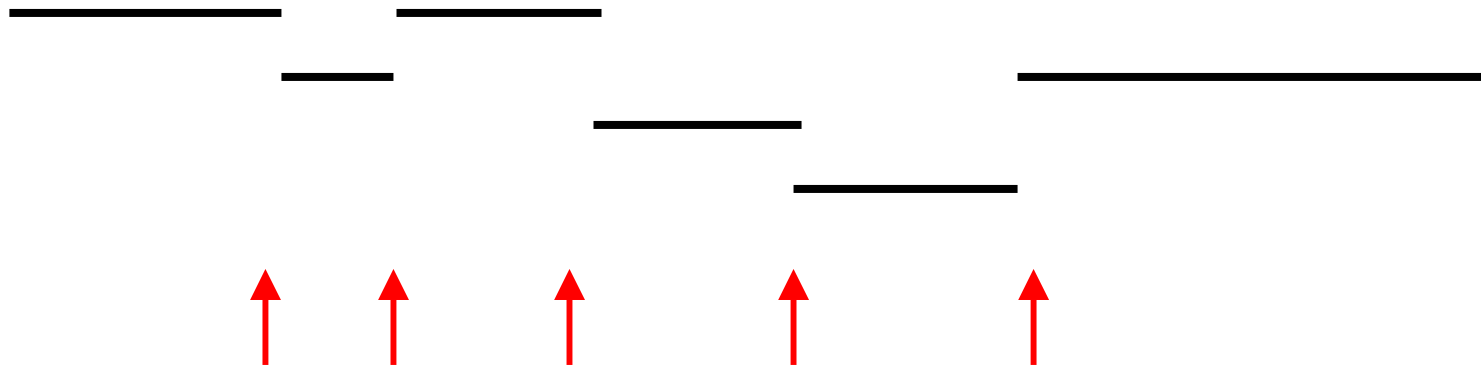
Preemptive Scheduling



**Enforced by underlying software
(an operating system)**



Cooperative Scheduling



Performed by the application code
In Java: "yield" or "sleep"



Threads in Java (1)

```
class person extends Thread {  
    public void run() {  
        do { yield();  
            if (m < F(m)) m = F(m)  
        } while ( ... )  
    }  
}
```

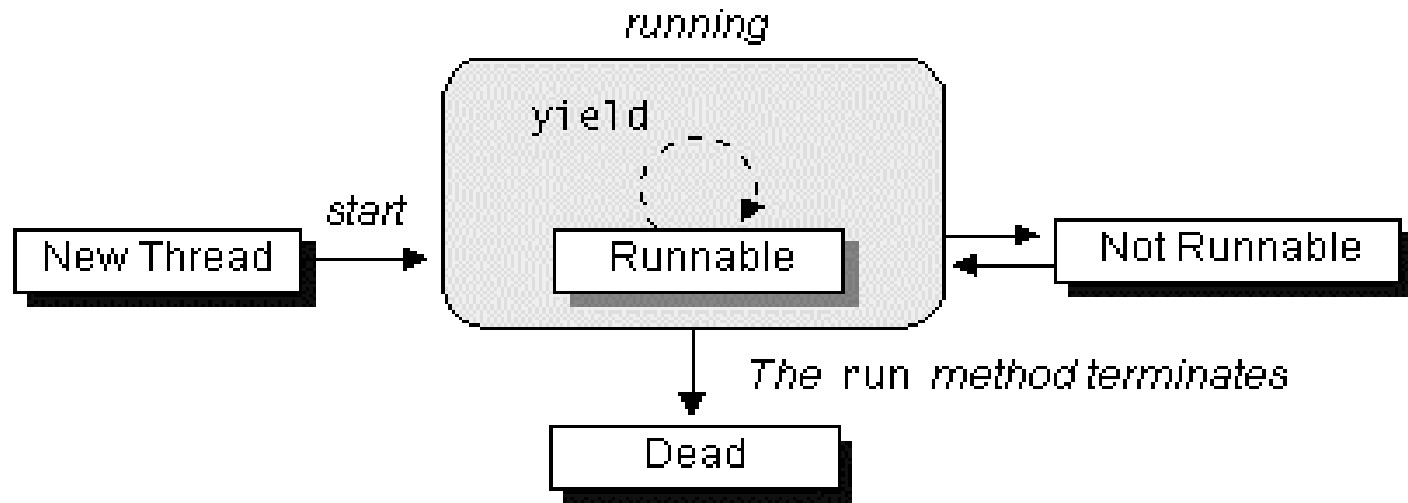


Threads in Java (2)

```
class person extends ... implements Runnable {  
    public void run() {  
        do { yield();  
            if (m < F(m)) m = F(m)  
        } while ( ... )  
    }  
}
```



The Life Cycle of a Thread



A thread becomes "Not Runnable" when one of these events occurs:

- Its sleep method is invoked.
- The thread calls the wait method.
- The thread is blocking on I/O.



Java Interpreters



- **Java Byte Code (.class files)**
- **Java interpreter in web browsers (Java VM)**
- **CVM (small footprint supporting Java 2)**
- **KVM (small footprint supporting CLDC)**



KVM

KVM is the *minimal* VM for the J2ME platform

- **Written from scratch in C.**
- **Portable**
 - Cooperative scheduling
 - Garbage collection
- **Omitted.**
 - Floating point support
 - JNI
 - Finalization
 - Some exception classes
 - a few more.



Break



Timers

Timer: A thread that is done at a specific time or periodically

```
class RemindTask extends TimerTask {  
    public void run() {  
        ...}  
}
```

```
public class Reminder {  
    Timer timer;  
    public Reminder(int seconds) {  
        timer = new Timer();  
        timer.schedule(new RemindTask(), seconds*1000);  
    }  
}
```



Architecture of Falling Coconuts



Synchronization

```
threadA: {  
    acc1= acc1-amount;  
    acc2= acc2+amount;  
}
```

 scheduling

```
threadB: {  
    sum= acc1+acc2;  
}
```



Java Memory Model

When a thread uses the value of a variable, the value it obtains is in fact a value stored into the variable by some thread

In the absence of explicit synchronization, a Java implementation is free to update the main memory in an order **that may be surprising**. Therefore the programmer who prefers **to avoid surprises should use explicit synchronization**



Surprise

```
class Simple {  
    int a = 1, b = 2;  
    void to() {  
        a = 3; b = 4;  
    }  
    void fro() {  
        System.out.println("a= "+a+" b= "+b);  
    }  
}
```

a= 1 b= 4 !!!!!!!



Monitor

*Class where each method is **atomic***

(indivisible, critical region, ...)

```
class account {  
    int acc1, acc2;  
    synchronized void transfer(int amount) {  
        acc1 = acc1 - amount;  
        acc2 = acc2 + amount;  
    }  
    synchronized int sum() {  
        return acc1 + acc2;  
    }  
}
```



Busy Waiting

```
class account {  
    ...  
    synchronized int sum() { return acc1 + acc2; }  
}  
  
public void run() {  
    ...  
    while (sum() < limit) { // nothing  
        ... yield();  
    }  
}
```

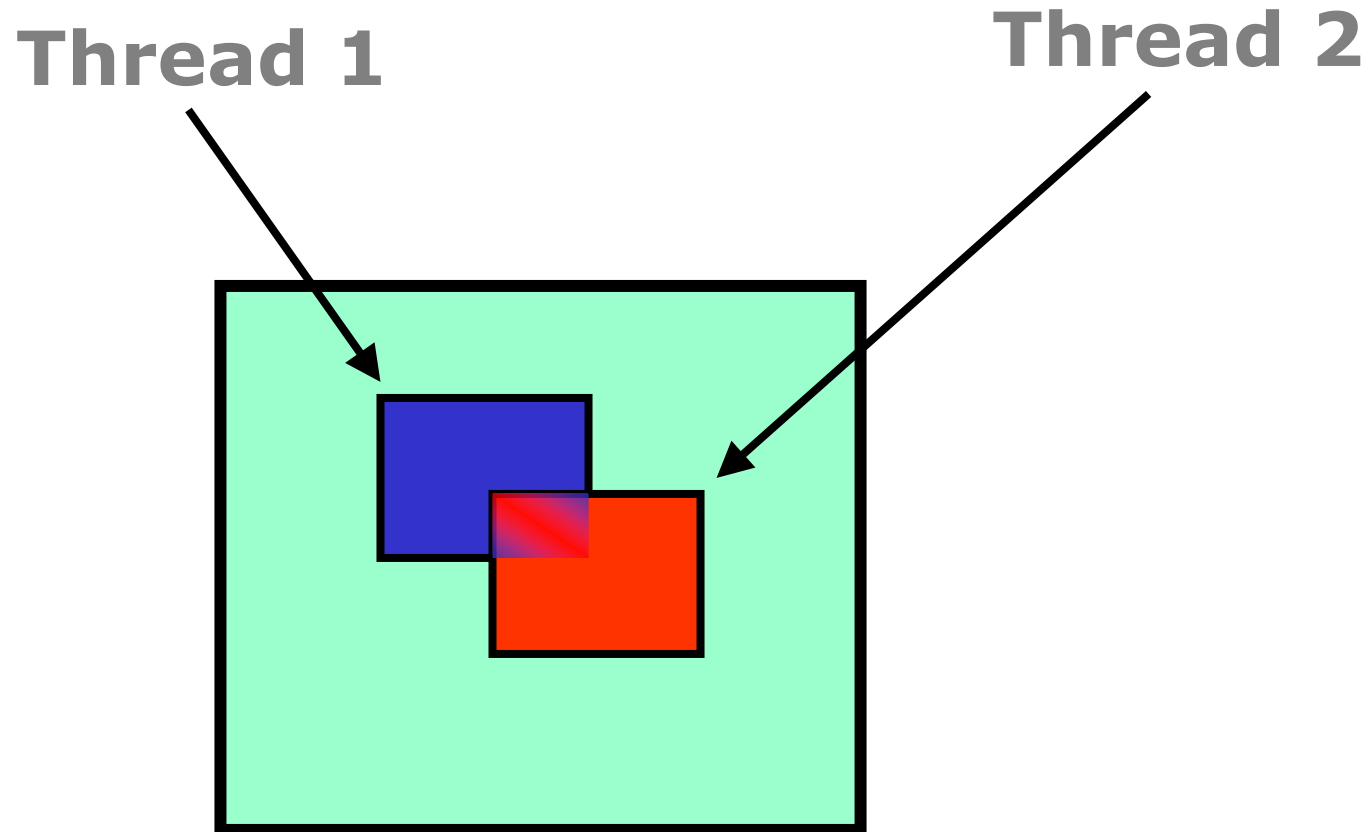


Waiting and Notification

```
class account {  
    ...  
    synchronized int sum() { return acc1 + acc2; }  
    synchronized int overLimit() {  
        if ((acc1+acc2)<cctest.limit) { this.wait(); }  
        return acc1 + acc2;  
    }  
  
    synchronized void add1(int amount) {  
        acc1= acc1 + amount;  
        if ((acc1+acc2)>=cctest.limit) { this.notify(); }  
    }  
}
```



Display is also Shared Data



J2ME Event Handling

- High-level API commands
- Low-level events (single key presses and releases)
- Calls to the `paint()` method of a Canvas class

are **atomic**

Implementation will never call an callback before a prior call to *any* other callback has returned.

Documentation: Package `javax.microedition.lcdui`

