

# MicroQ

En simulering av en mikrokö.

# Introduktion - Bakgrund



# Introduktion - Bakgrund

- Att stå i kö är tråkigt.
- Att vänta på andra är tråkigt.
- Att äta är kul.
- Minimera tiden vi står i mikrokön!



# Introduktion - Bakgrund

- Öka antalet mikrovågsugnar
- Begränsa tiden en student har att värma sin mat
- Minska väntetid.



# Introduktion – MicroQ

- En simulering.
- Ta in parametrar.
- Mäta tider.
- Ge ut ett resultat.

# Introduktion - Demonstration

- Ge oss 5 sekunder så startar vi programmet.

# Programmeringsspråk

Erlang och Java

# Programmeringsspråk -Erlang

- Arbeta med två språk
- Nytt för oss
- "Lightweight"
- Skicka meddelanden mellan processer
- Hanterar tillstånden





# Programmeringsspråk

## -Java

- Imperativt objektorienterad högnivåspråk
  - Inbyggt synkroniseringsstöd
  - Färdiga bibliotek
  - Automatiserad testning och dokumentation
  - Gränssnitt Java ↔ Erlang
- 
- Hanterar kön, mikrovågsugnar & tidmätning



# Systemarkitektur

Erlang

# Systemarkitektur -Erlang

Erlangmetod

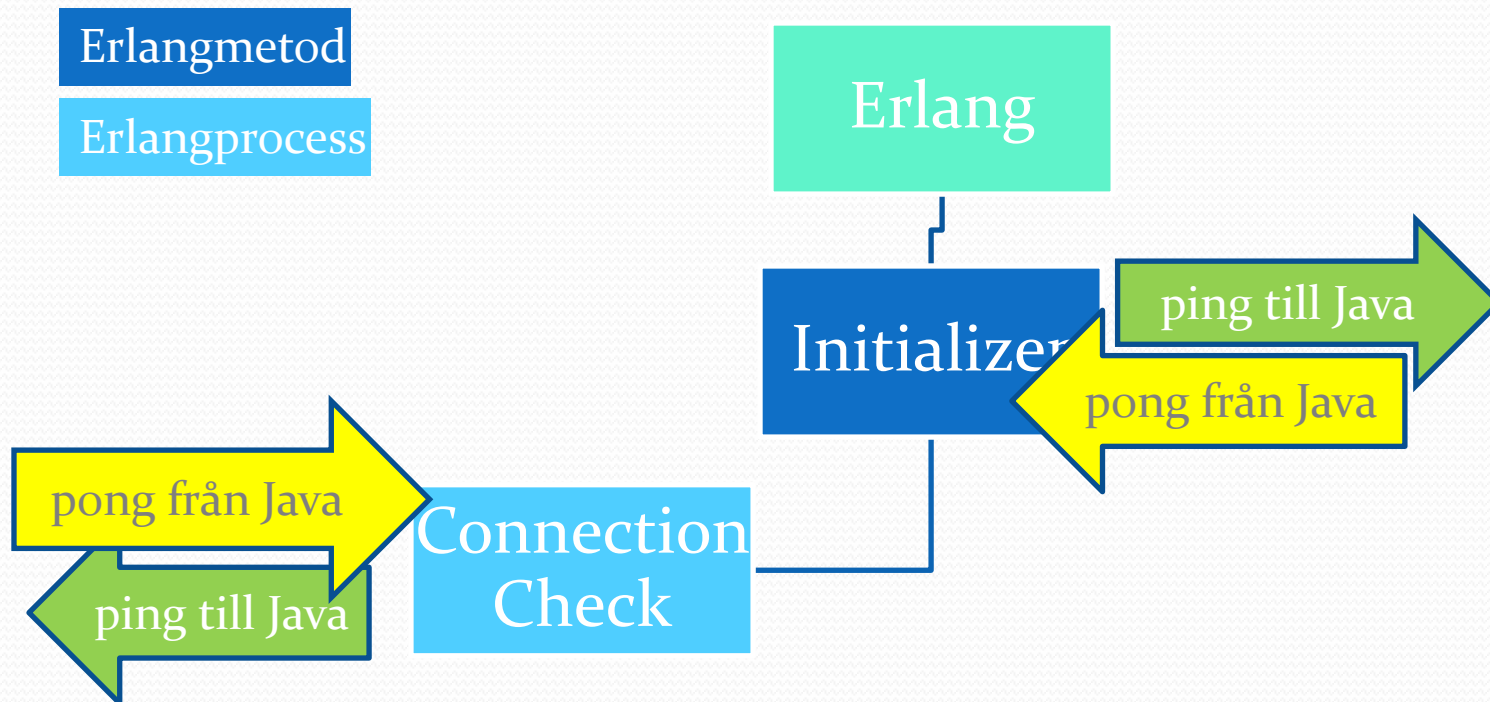
Erlangprocess

Erlang

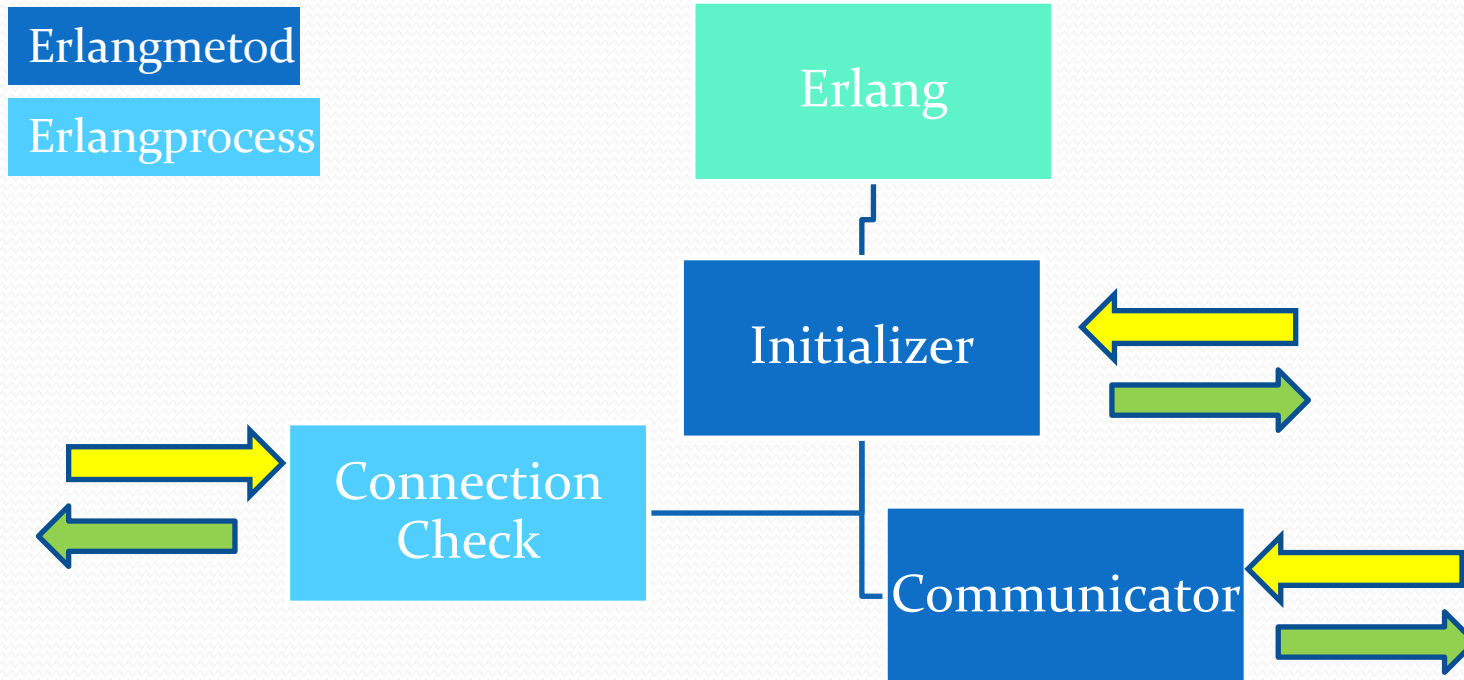
Initializer



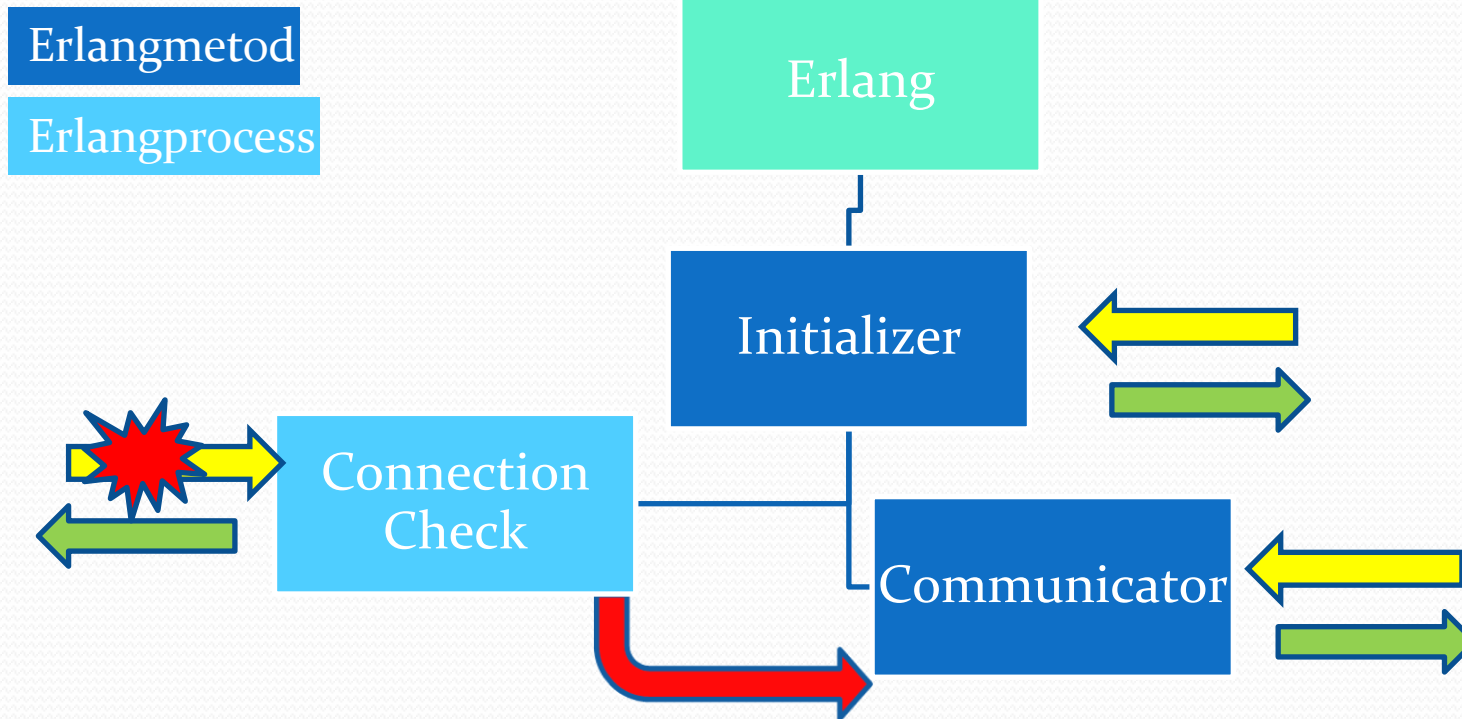
# Systemarkitektur -Erlang



# Systemarkitektur

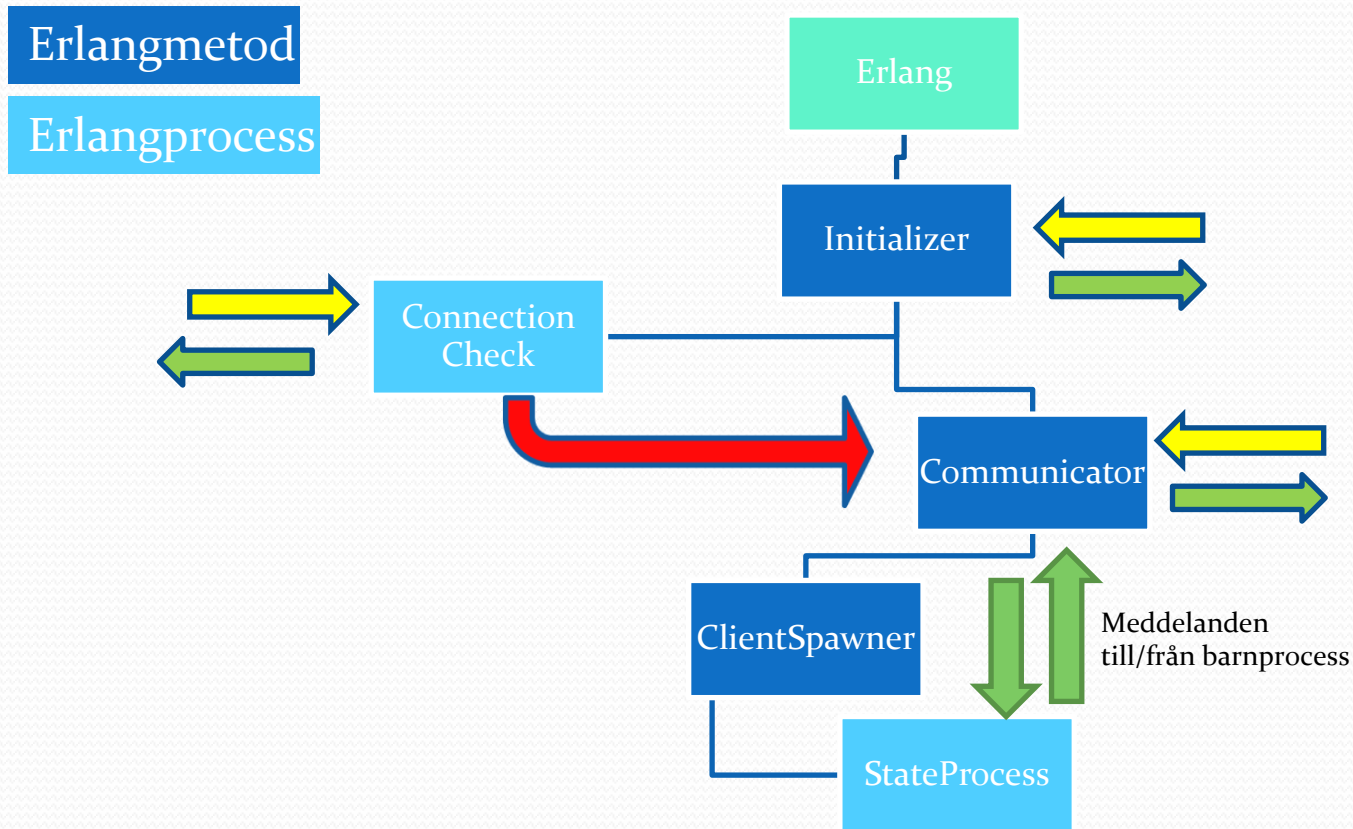


# Systemarkitektur -Erlang



Om connection check inte får pong tillbaka från Java så skickas ett kill-meddelande till communicator

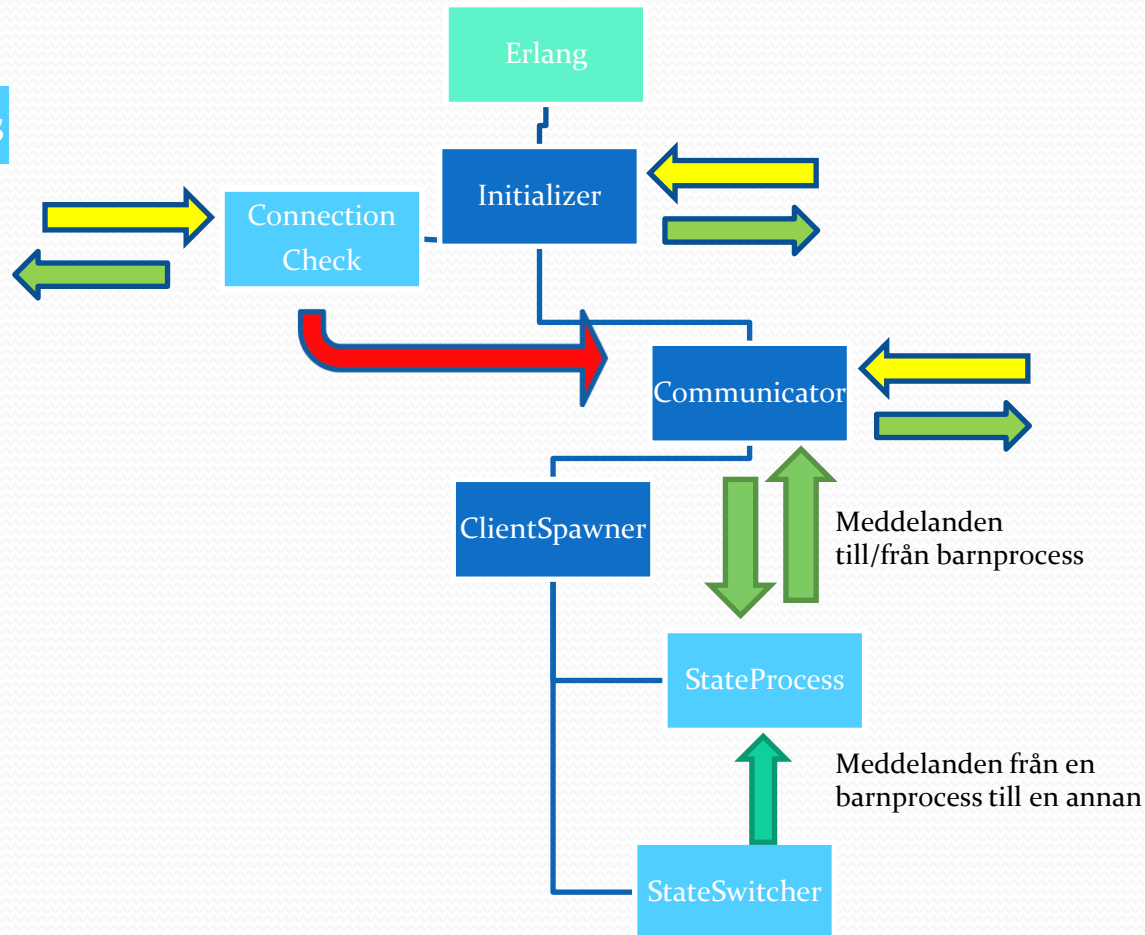
# Systemarkitektur -Erlang



# Systemarkitektur -Erlang

Erlangmetod

Erlangprocess





# Systemarkitektur

Java

# Systemarkitektur -Java

Javametod/-objekt

Javatråd

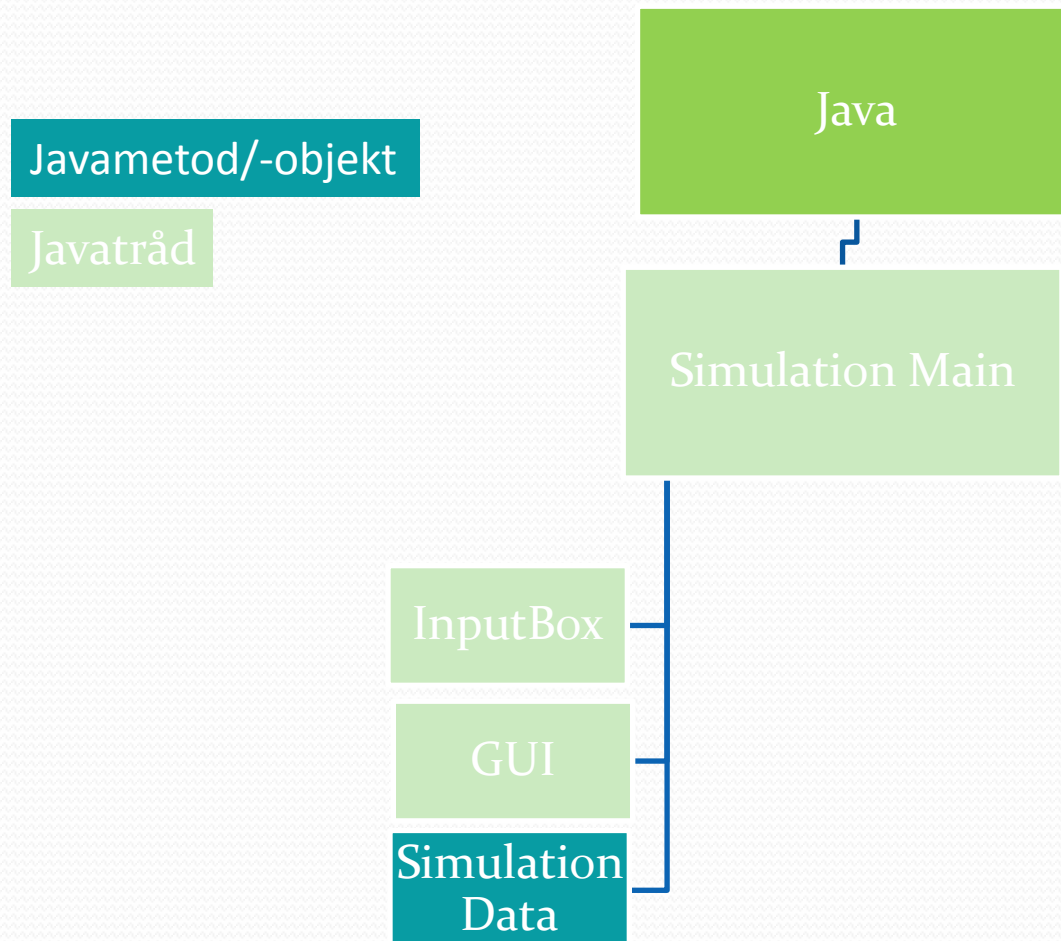
Java

Simulation Main

```
graph TD; Java[Java] --- SimulationMain[Simulation Main];
```

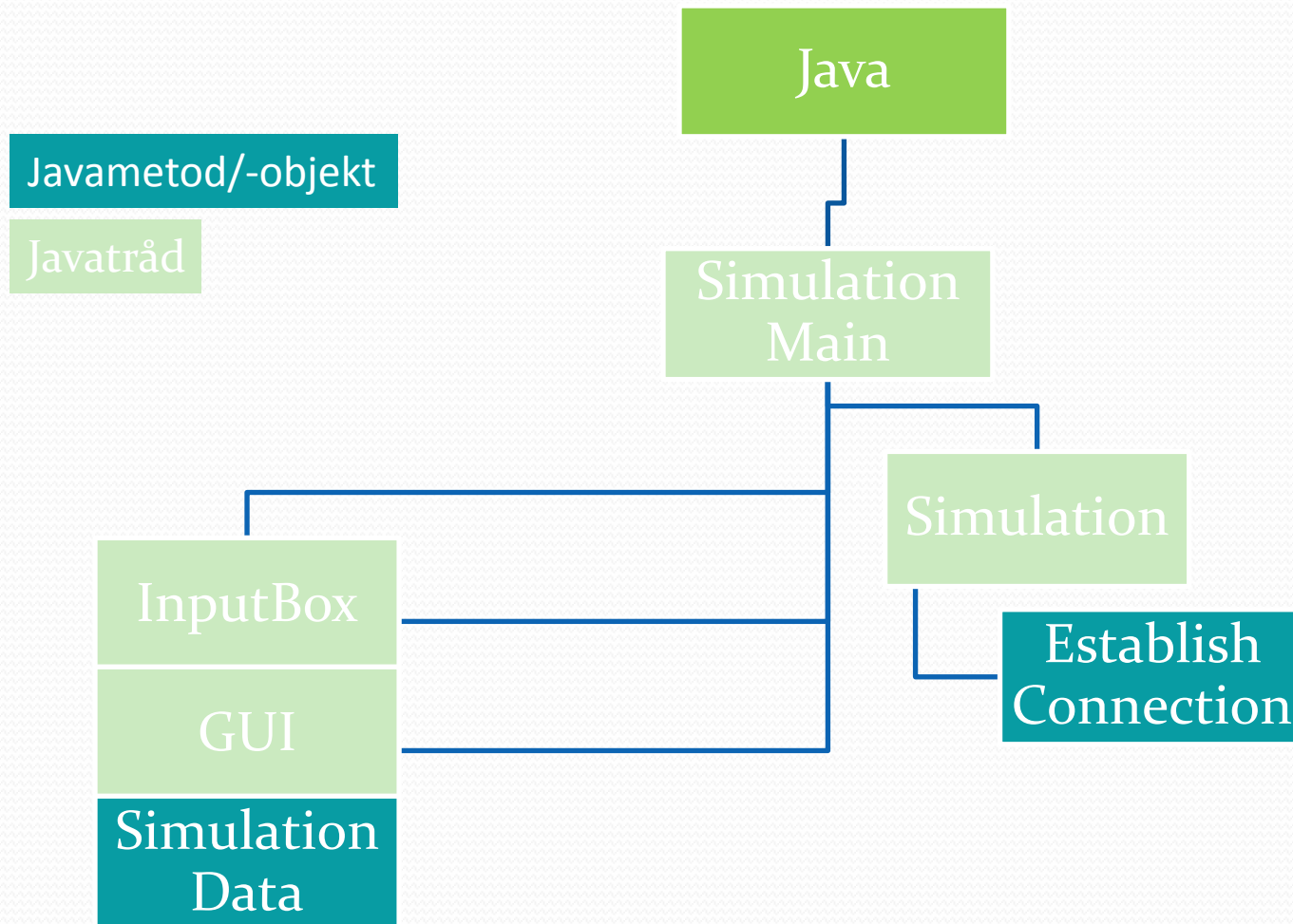
# Systemarkitektur

## -Java

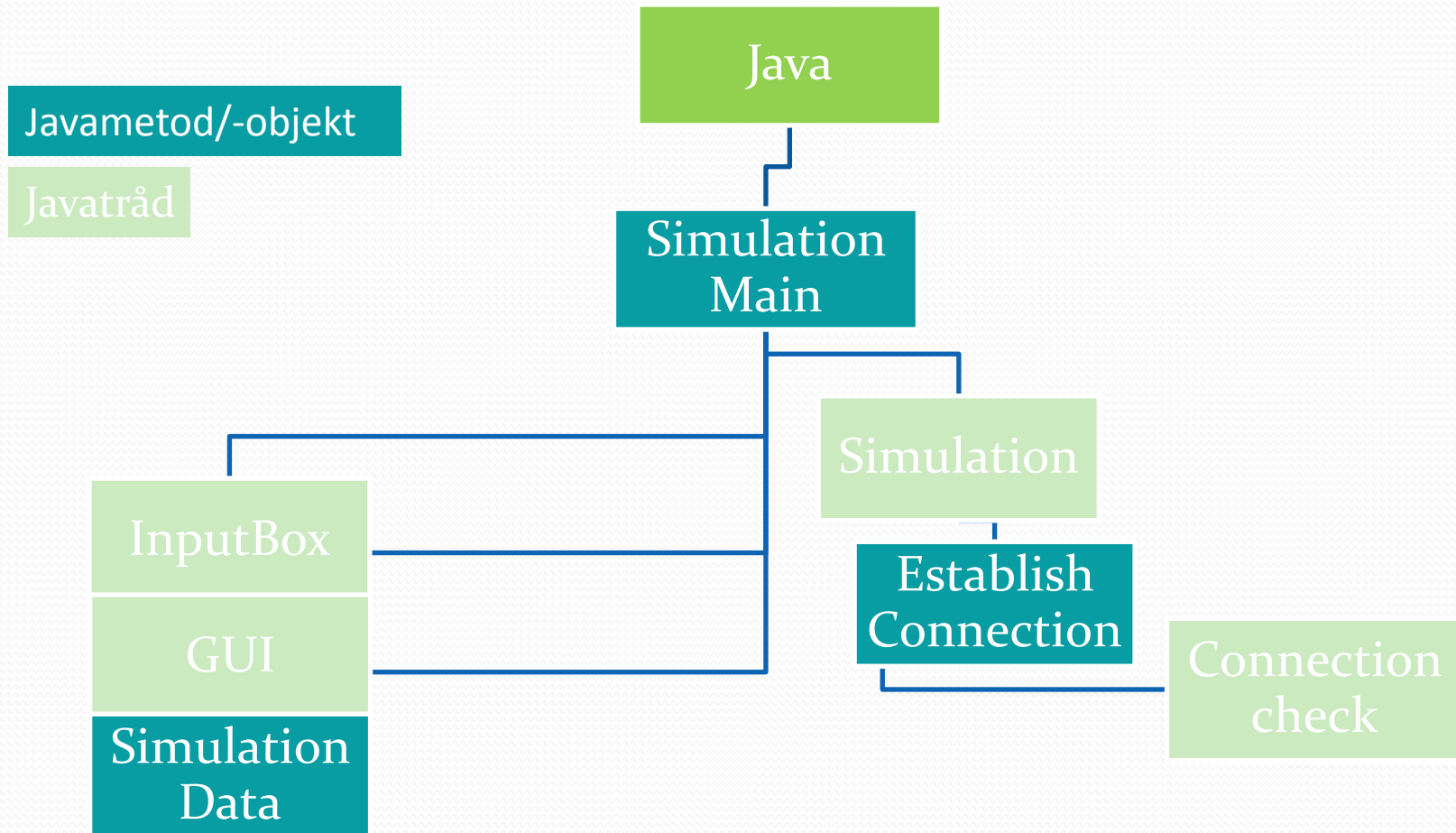


# Systemarkitektur

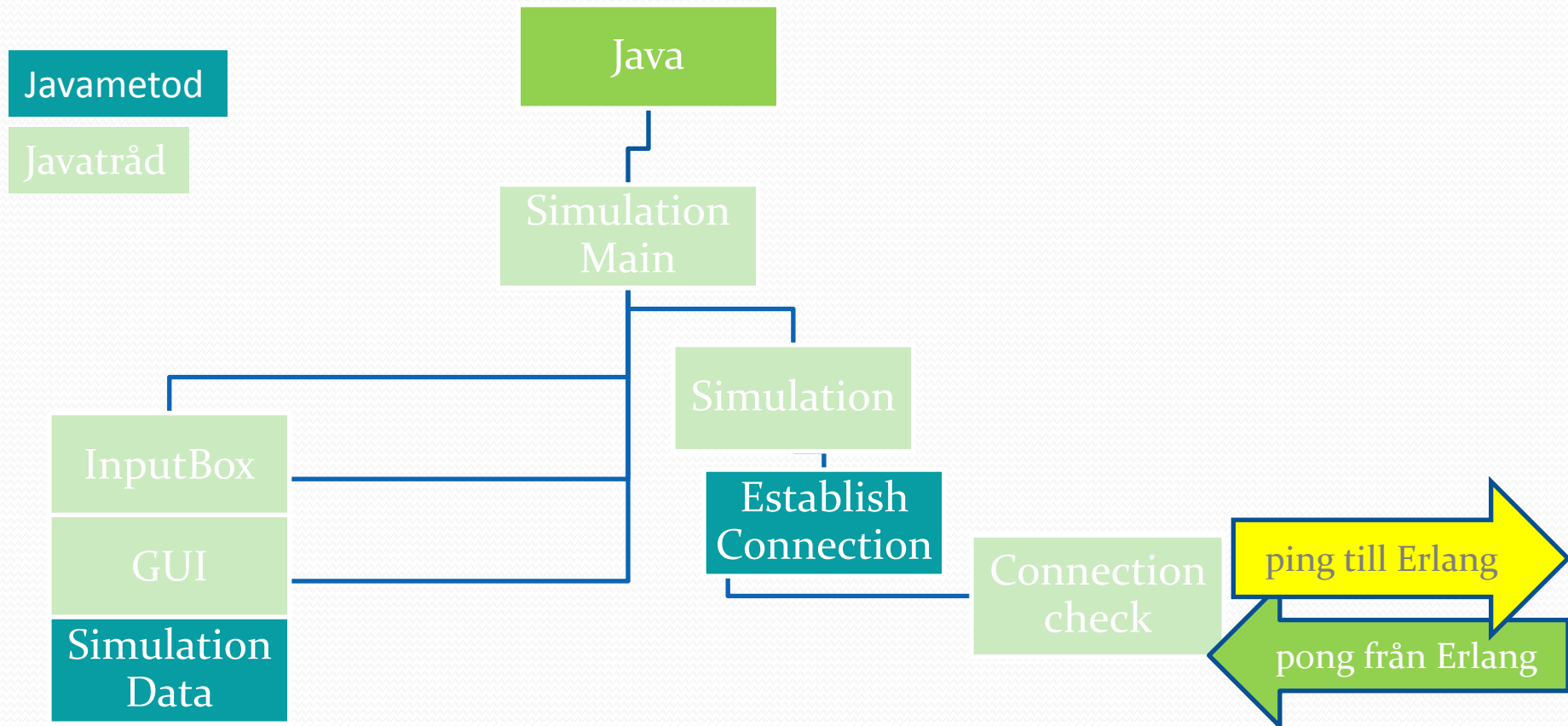
## -Java



# Systemarkitektur -Java

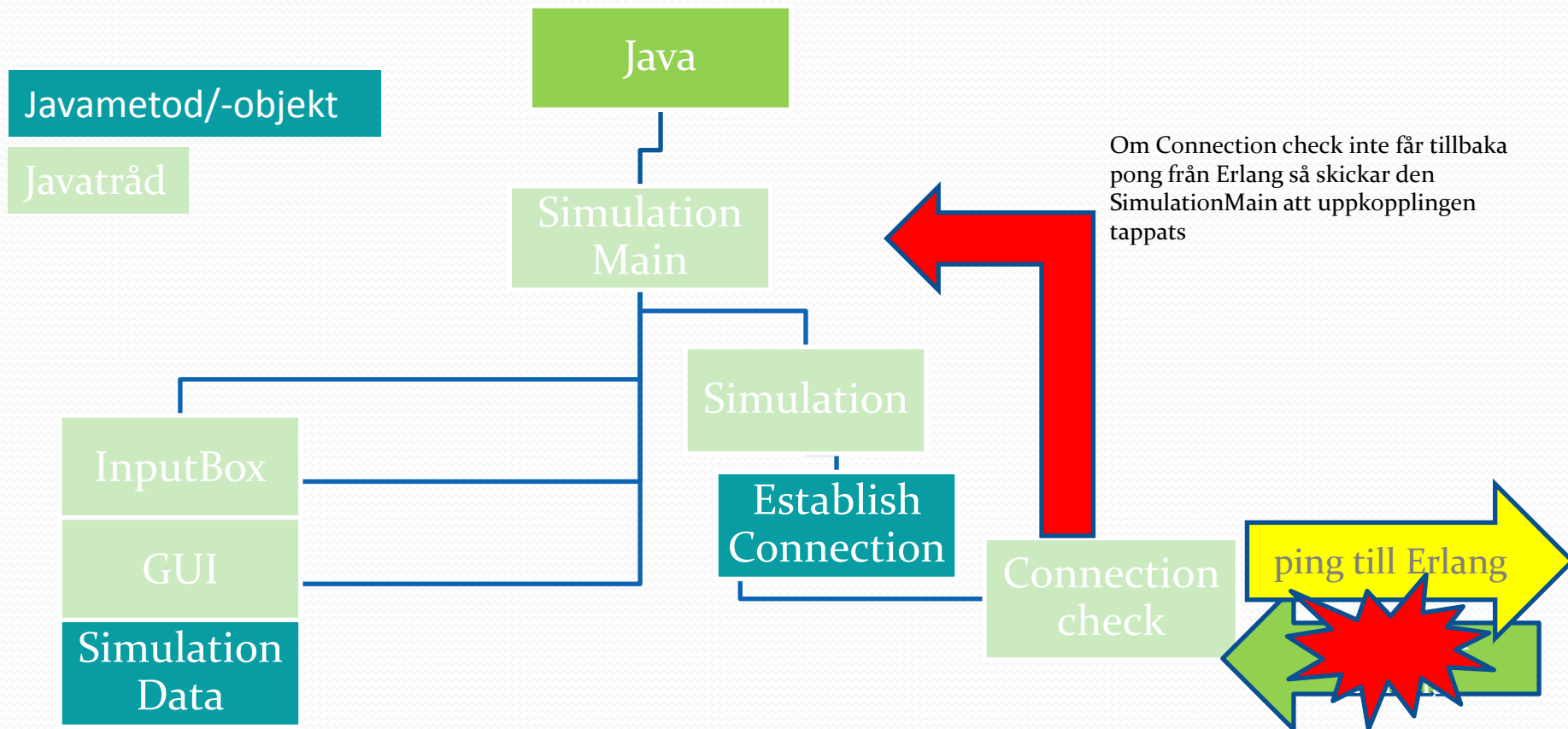


# Systemarkitektur -Java

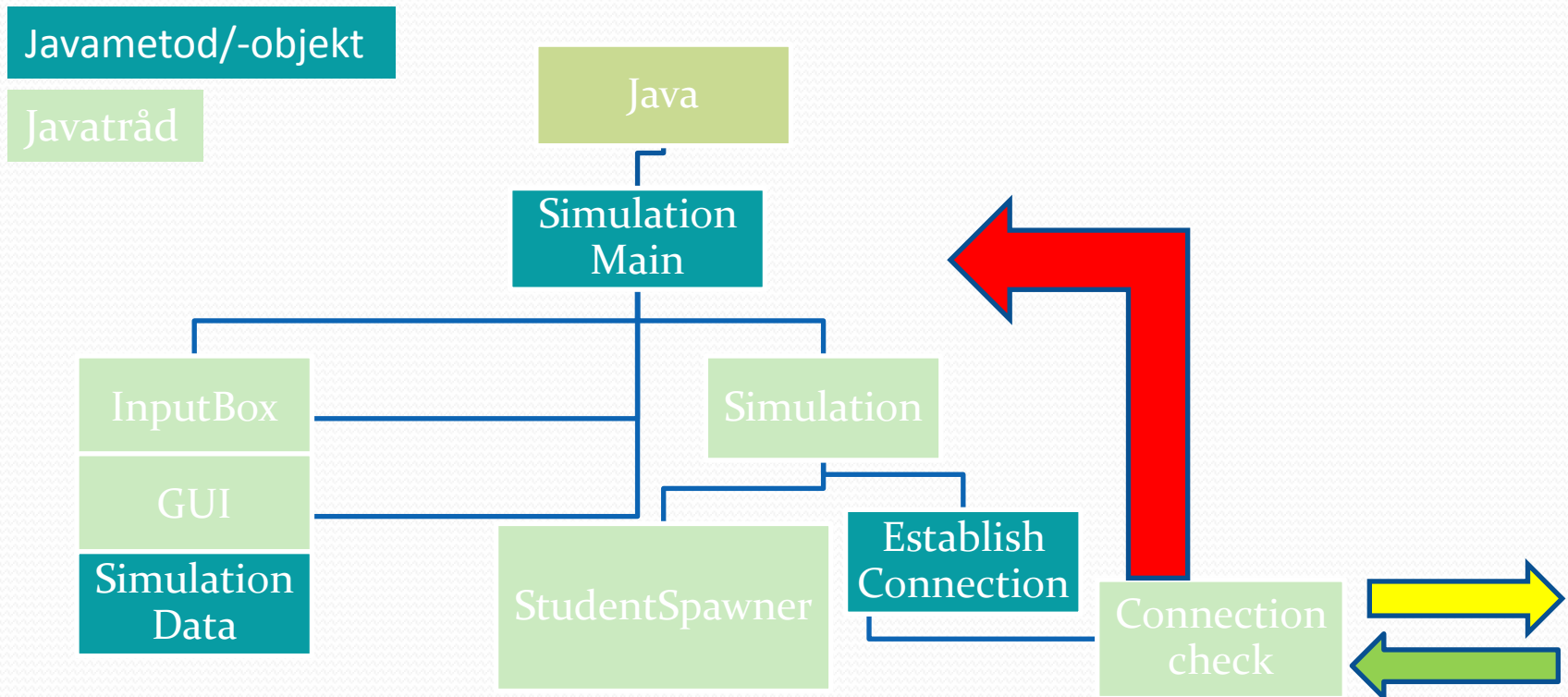


# Systemarkitektur

## -Java



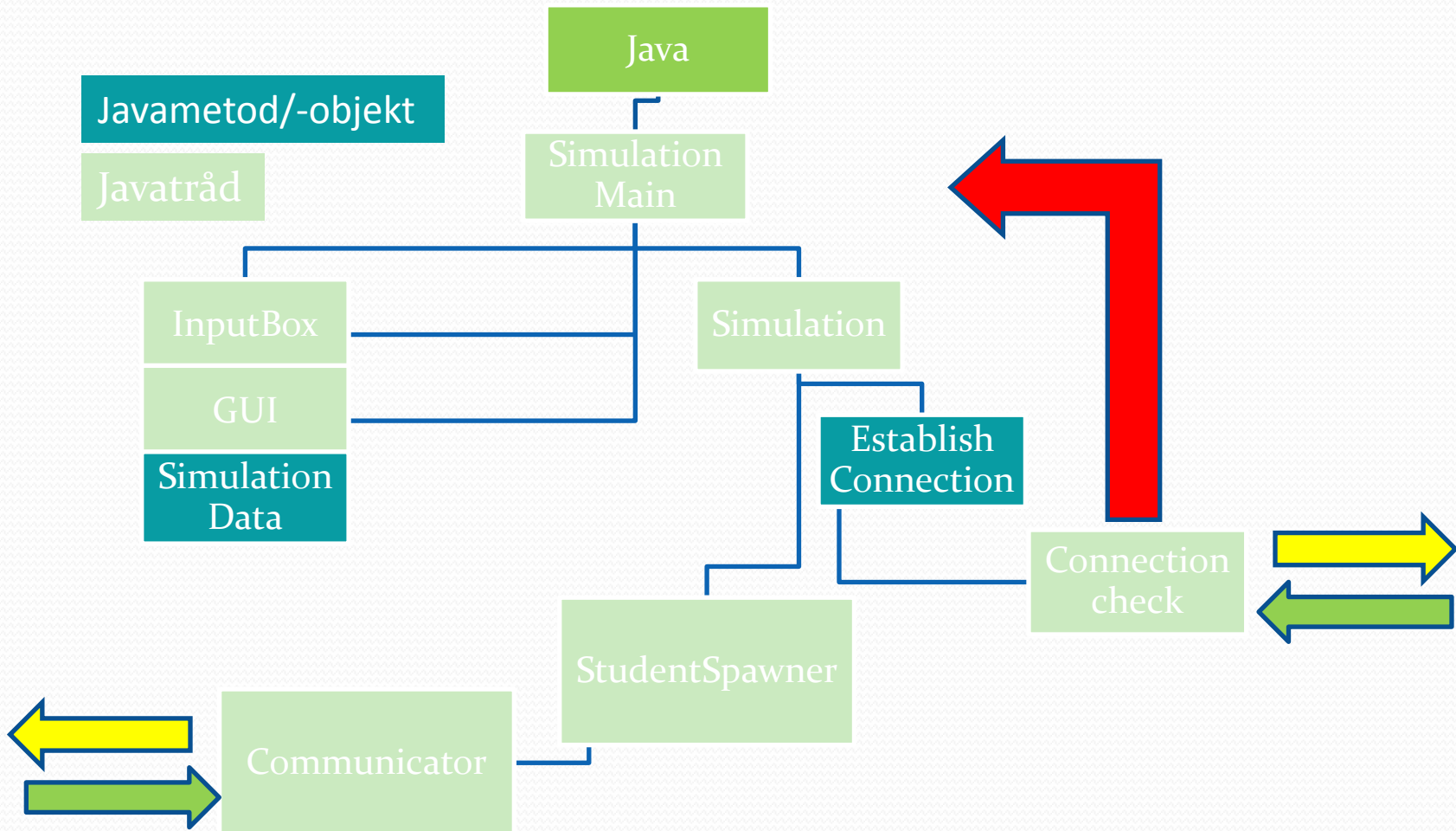
# Systemarkitektur -Java



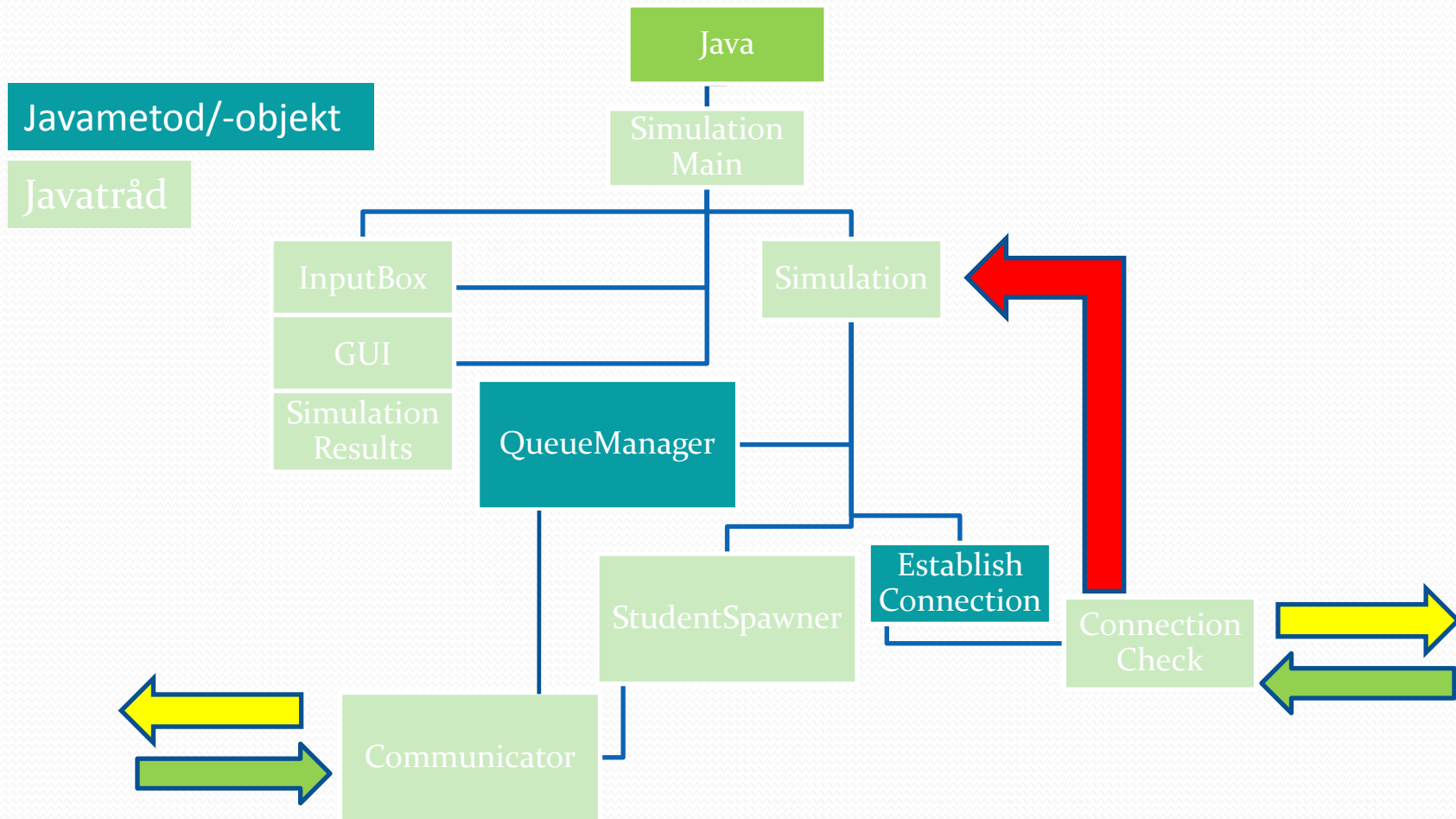


# Systemarkitektur

## -Java



# Systemarkitektur -Java



# Planering och organisering

# Planering och organisering

- Sittit tillsammans i grupp
  - Snabb feedback på arbetet
  - Kunna diskutera
  - Planera



# Planering och organisering

## ● Java

- Björn och Marcus
  - GUI
  - Resultatdata/mätningar
  - Synkronisering och hantering av mikrokön
- Simon
  - Kommunikation med Erlang
  - Skapande och hantering av studentprocesser

# Planering och organisering

## ● Erlang

- Simon
  - Kommunikation med Java
- Nanna
  - Hantering av tillståndsprocesser

# Planering och organisering

- Dokumentation, tester, repository
  - Simon
    - Javadoc, EDoc
    - JUnit, (Eunit)
    - GitHub
  - Nanna
    - Edoc
    - Rapport
    - GitHub

# Diskussion

Concurrency och deadlocks



# Diskussion

## -concurrency, Java

- Utnyttjar Javatrådar
- GUI
- Simuleringen
- ConnectionCheck
- Studentspawnare
- Kommunikationstrådar
- Till viss del även message passing

# Diskussion

## -concurrency, Erlang

- Tillståndsprocesser
- Tillståndsbytare
- Processerna i Erlang använder message passing
- Kommunikationen med Java

# Diskussion

## -concurrency, synkronisering

- Finns det någon ledig mikro?
- Finns det någon student i kön?
- Räknaren som håller koll på antalet spawnade studenter.

# Diskussion

## -concurrency, fördelar

- Erlangdelarna blir väldigt små och överskådliga
- Behöver inte synkronisera någon data i Erlang
- GUI:t körs samtidigt som beräkningar utförs i andra trådar och Erlangprocesser

# Diskussion

## -concurrency, nackdelar

- Prestandaförlust vid kommunikationen mellan Erlang och Java.
- Komplex datorarkitektur

# Diskussion

## -concurrency, alternativ

- Skriva allt i ett språk
- Sköta simuleringen enbart i Erlang. Endast GUI i Java.

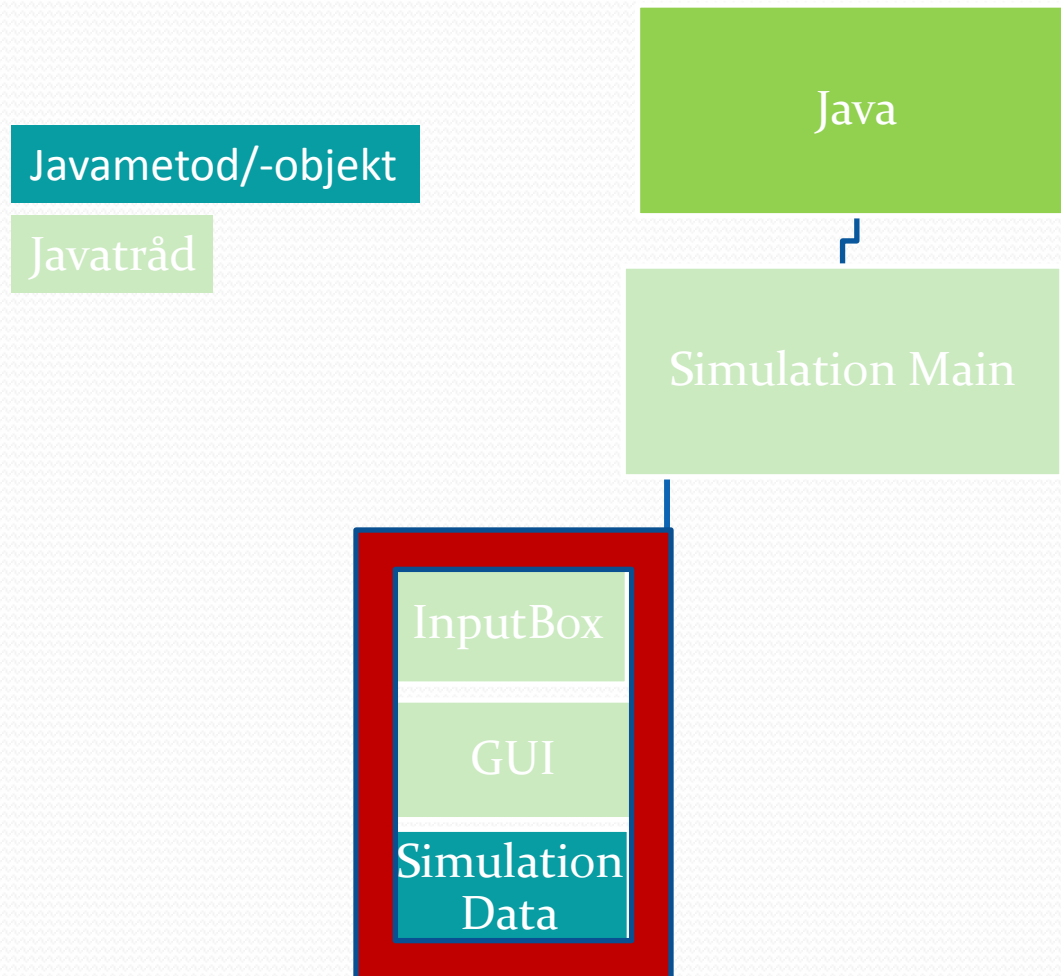
# Diskussion -deadlocks

- Om en Communicator-tråd oväntat dör
- Om InputBox tråden skulle dö innan den kört klart.

# Teknisk demonstration

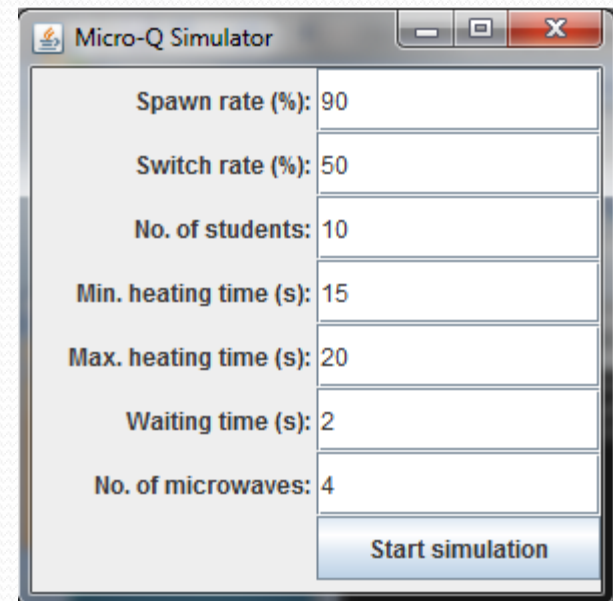


# Teknisk Demonstration



# Teknisk Demonstration

- Startas från SimulationMain
- Läser in de inmatade värdena
- Startar simuleringen med de inlästa värdena
- Släpper en semafor så att GUI:n börjar ritas



The screenshot shows a window titled "Micro-Q Simulator" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains a list of parameters for a simulation, each with a label and a text input field. The parameters and their values are:

Parameter	Value
Spawn rate (%)	90
Switch rate (%)	50
No. of students	10
Min. heating time (s)	15
Max. heating time (s)	20
Waiting time (s)	2
No. of microwaves	4

At the bottom of the window, there is a blue button labeled "Start simulation".

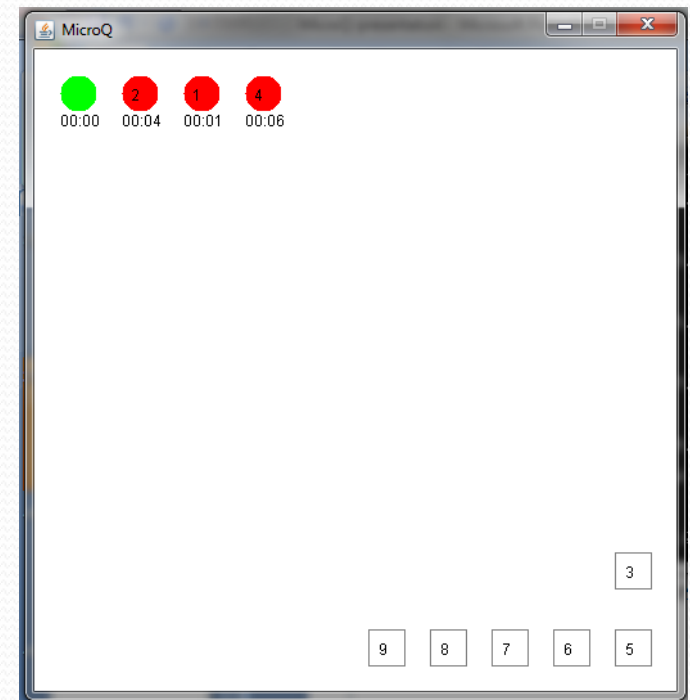
# Teknisk Demonstration

```
21 public class InputBox extends JFrame implements Runnable {
125     public void run() {
126         // TODO Auto-generated method stub
127         button.addActionListener(new ActionListener(){
128             public void actionPerformed(ActionEvent e){
129                 int sR = Integer.parseInt(srfield.getText());
130                 int swR = Integer.parseInt(swfield.getText());
131                 int nOS = Integer.parseInt(stfield.getText());
132                 int hTMax = Integer.parseInt(htmaxfield.getText()*1000;
133                 int hTMin = Integer.parseInt(htminfield.getText()*1000;
134                 int wT = Integer.parseInt(wtfield.getText()*1000;
135                 int mW = Integer.parseInt(mcfld.getText());
136                 SimulationMain.microwaves = new Microwave[mW];
137                 for(int i = 0; i < mW; i++) {
138                     SimulationMain.microwaves[i] = new Microwave(i);
139                     SimulationMain.availableMicros.add(i);
140                 }
141                 InputBox.this.setVisible(false);
142                 SimulationMain.simulation = new Thread(new Simulation(sR, nOS, hTMax, hTMin, wT, mW, swR));
143
144                 SimulationMain.setup_done.release();
145             }
146         });
147     }
148 }
```

- Skapar en JFrame "InputBox" i en ny tråd
- Inmatning av parametrar
- Knapp som startar simuleringen med parametrarna angivna i textfält
- En array med Microwave-objekt
- Semaforen släpps och SimulationMain fortsätter

# Teknisk Demonstration

- Ritar ut en representation av mikrovågsugnar
- En huvudkö
- En kö med de studenter som står på tur att värma mat



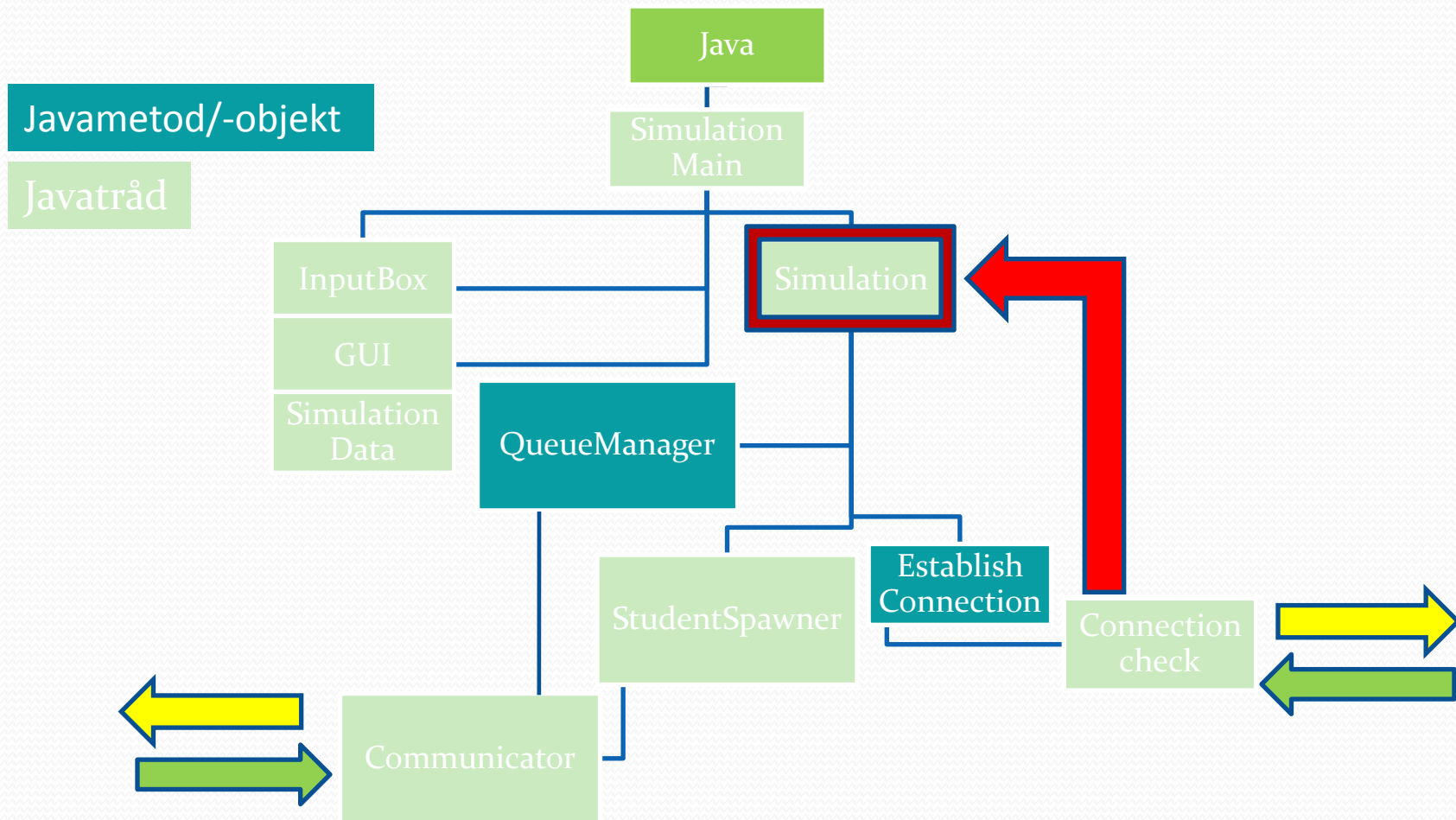
# Teknisk Demonstration

```
22 public class SimulationMain extends JFrame

211- void draw()
212 {
213     Graphics g = getGraphics();
214     Graphics bbg = backBuffer.getGraphics();
215
216     bbg.setColor(Color.WHITE);
217     bbg.fillRect(0, 0, windowWidth, windowHeight);
218
219     for(int i = 0; i < microwaves.length; i++) {
220         microwaves[i].draw(bbg);
221     }
222
223     queueRep.draw(bbg);
224
225     g.drawImage(backBuffer, insets.left, insets.top, this);
226 }
```

- SimulationMain ritar bland annat upp simulerings-GUI:t
- Backbuffer, förhindrar flimmer
- Ritar ut alla Microwave-objekt, en huvudkö samt en sekundärkö

# Teknisk Demonstration



```

private static void runSimulation() {
    while(studentsCompleted < numberOfStudents)
    {
        try {
            queue_sem.acquire();
            System.out.println("Students completed in simulation: " + studentsCompleted);
            if(studentsCompleted >= numberOfStudents)
            {
                break;
            }
            microwave_sem.acquire();
            System.out.println("micro is ready");
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        queue.readyCheck();
    }

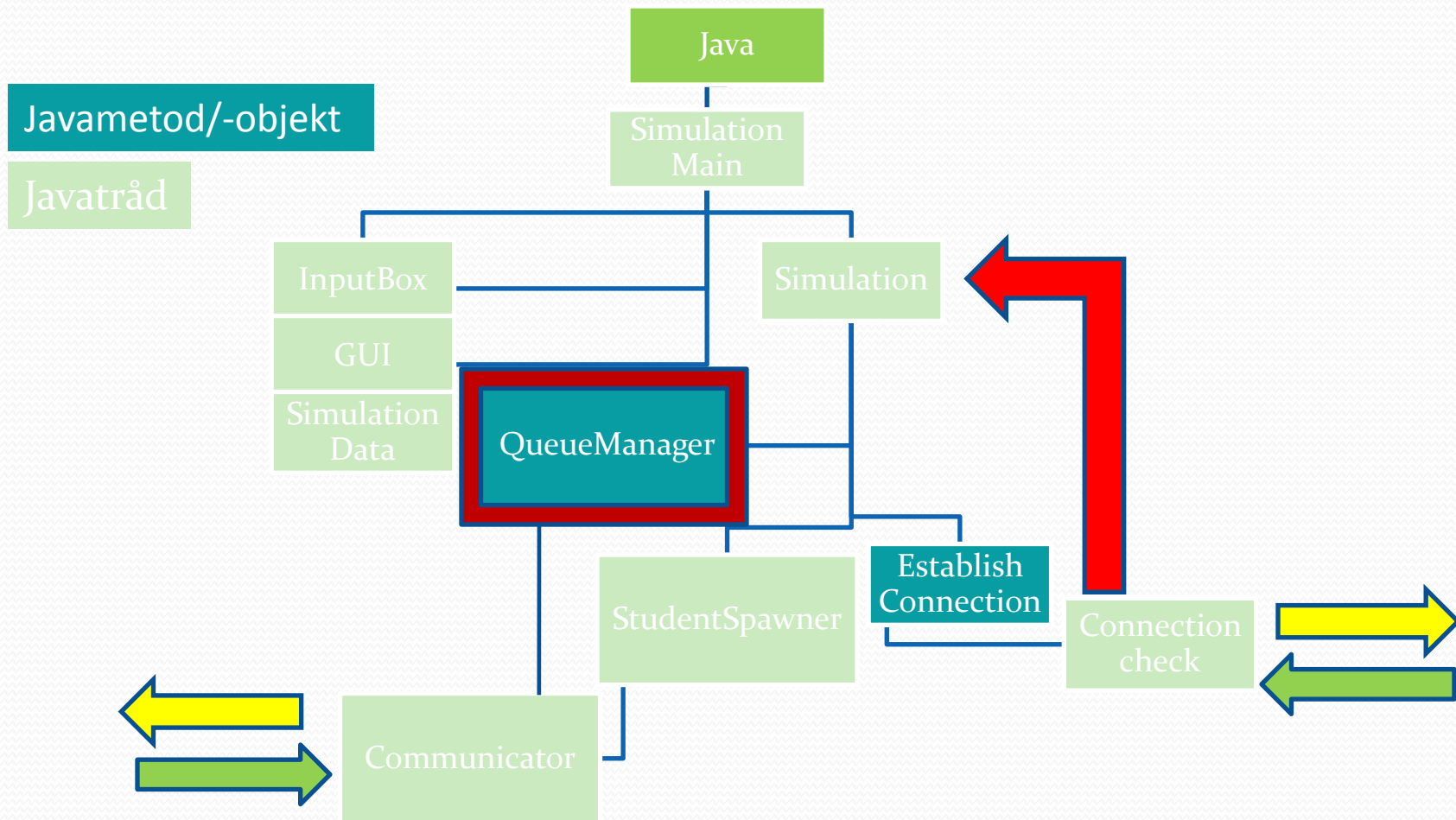
    System.out.println("Outside Simulation loop...");

    OtpErlangObject[] message = new OtpErlangObject[2];
    message[0] = new OtpErlangAtom("mbox");
    message[1] = new OtpErlangAtom("terminate");
    mbox.send(Simulation.erlNodePID, new OtpErlangTuple(message));
    try {
        mbox.receive();
    } catch (OtpErlangExit e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (OtpErlangDecodeException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    SimulationMain.isRunning = false;
}

```

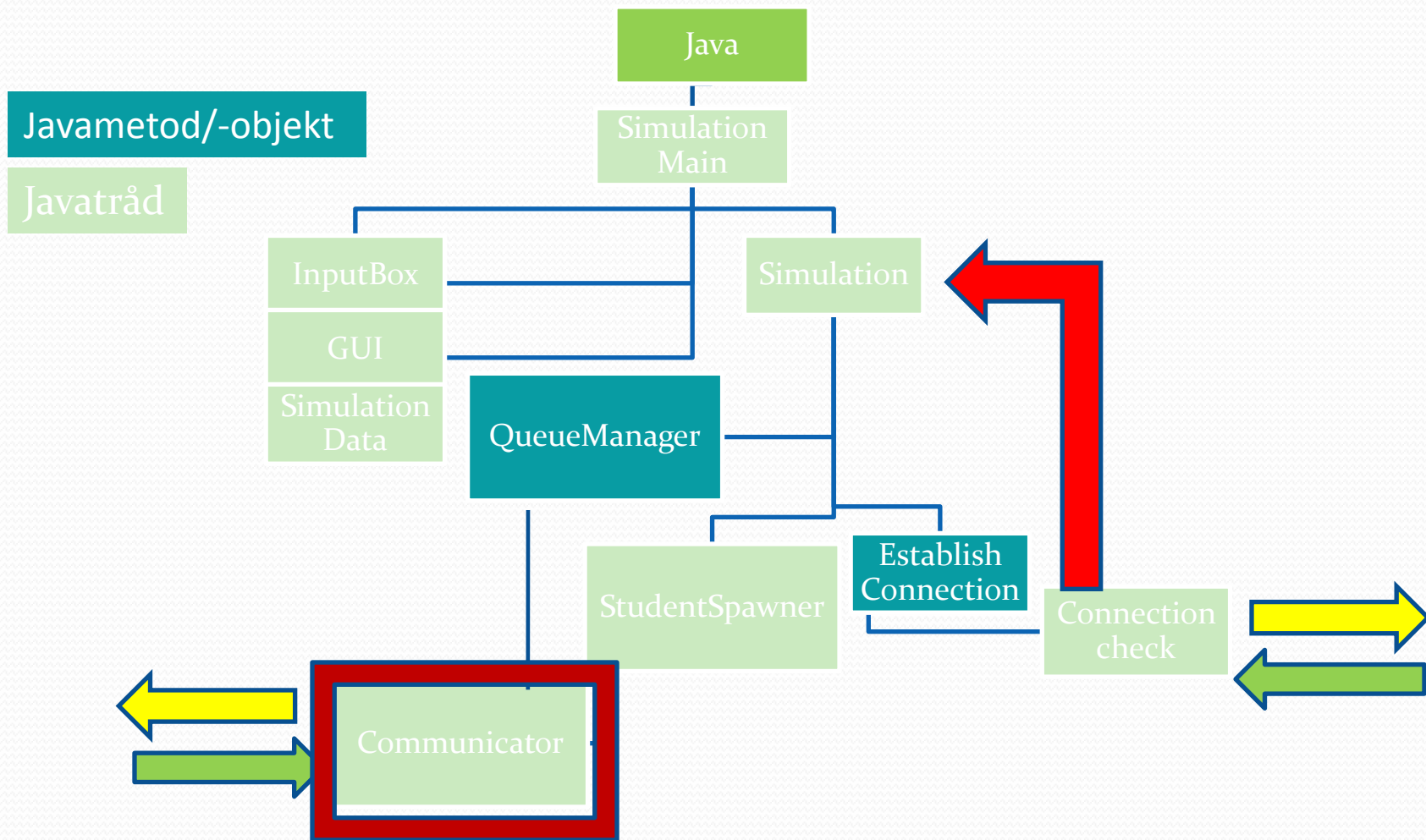
- Plockar kö-semafor
- Kontrollerar antalet färdiga studenter
- Plockar mikro-semafor
- Poppa studenten via readyCheck
- Skickar terminate och väntar på svar
- Sätter isRunning till false

# Teknisk Demonstration





# Teknisk Demonstration



# Teknisk Demonstration

```
128- /**
129-  * Sends an Erlang tuple to the communicator-process running on the Erlang node containing this Communicator's
130-  * mailbox name, the message atom and an Erlang Object.
131-  *
132-  * @param tid      The mailbox name for this instance.
133-  * @param atom     A message to distinguish the different requests to Erlang
134-  * @param obj      An Erlang Object
135-  */
136- public void sendMessage(OtpErlangAtom tid, OtpErlangAtom atom, OtpErlangObject obj) {
137-     OtpErlangObject[] message = new OtpErlangObject[3];
138-     message[0] = tid;
139-     message[1] = atom;
140-     message[2] = obj;
141-     mail.send(Simulation.erlNodePID, new OtpErlangTuple(message));
142- }
```

- Skapar en Erlang-tupel som meddelande
- Skickar meddelandet

# Teknisk Demonstration

```
163 public void run() {
164     OtpErlangTuple tuple = null;
165     tID = ""+Thread.currentThread().getId();
166     mail = Simulation.javaNode.createMbox(tID);
167     if(newClient)
168     {
169         sendMessage(new OtpErlangAtom(tID), atom, sRate);
170         System.out.println("new_client message sent...");
171         try {
172             tuple = (OtpErlangTuple) mail.receive();
173             System.out.println("new_client created with PID " + tuple.elementAt(1).toString());
174         }
175     }
176     ↓
177     int i;
178     synchronized(StudentSpawner.iterSync) {
179         i = StudentSpawner.getIter();
180         s = new Student((OtpErlangPid)tuple.elementAt(1), i);
181         StudentSpawner.iter++;
182     }
183     StudentSpawner.spawn_sem.release();
184     QueueManager.addStudent(s);
185 }
186 else
```

- Skapar en ny Mailbox
- Skickar meddelandet till Erlang med sendMessage
- Väntar på svar
- Skapar ett nytt Student-objekt
- Lägger till i kön

# Teknisk Demonstration

```
191     else
192     {
193         sendMessage(new OtpErlangAtom(tID), atom, pid);
194         System.out.println("Ready-request sent to PID: " + s.getPID());
195         long currTime = System.currentTimeMillis();
196         try {
197             tuple = (OtpErlangTuple) mail.receive(s.getWaitingTime());
198         }
199
200         ↓
201         double idleTime = ((double)(System.currentTimeMillis() - currTime))/1000;
202         SimulationMain.simData.increaseIdleTime(idleTime);
203
204         if(tuple == null) {
205             System.out.println("mail receive timeout...");
206             Simulation.microwave_sem.release();
207             QueueRepresentation.popWaitingQueue(s.getStudentNumber());
208             QueueManager.addStudent(s);
209         }
210         else {
211
```

- Skickar meddelandet till Erlang med sendMessage
- Försöker ta emot ett svar från Erlang
- Släpper semafor
- Tar bort från väntkö
- Lägger tillbaka i huvudkö

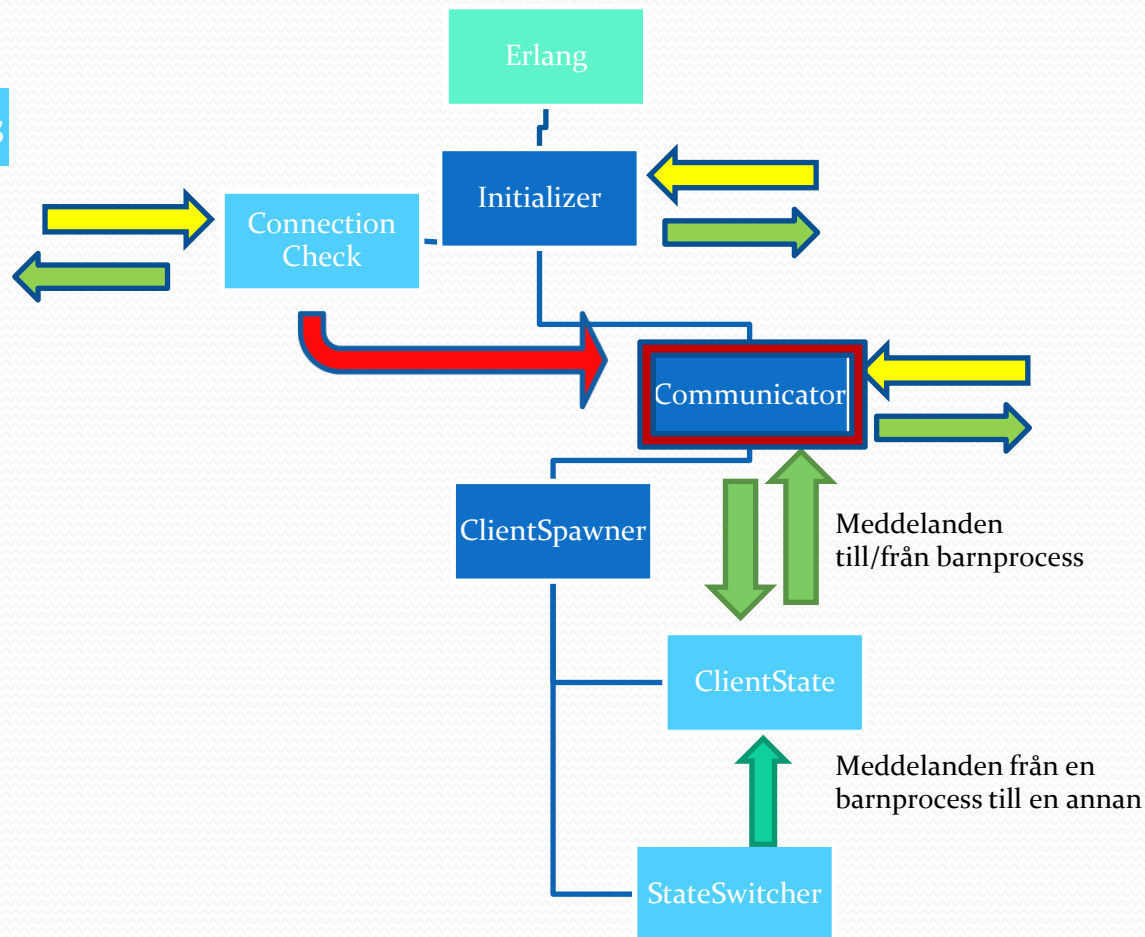
# Teknisk Demonstration

```
216         else {
217             QueueRepresentation.popWaitingQueue(s.getStudentNumber());
218             int i = SimulationMain.availableMicrowaves.remove().intValue();
219             SimulationMain.microwaves[i].setTime((int) (s.getHeatingTime()/1000));
220             SimulationMain.microwaves[i].setCurrentStudentNum(s.getStudentNumber());
221             SimulationMain.microwaves[i].getTimer().start();
222             try {
223                 Thread.sleep(s.getHeatingTime());
224             }
225
226             sendMessage(new OtpErlangAtom(tID), new OtpErlangAtom("kill"), pid);
227             while(SimulationMain.microwaves[i].getTime() > 0) {
228                 try {
229                     Thread.sleep(500);
230                 }
231             }
232
233             SimulationMain.availableMicrowaves.add(i);
234             Simulation.studentsCompleted++;
235             Simulation.microwave_sem.release();
236             if(Simulation.studentsCompleted >= Simulation.numberOfStudents)
237             {
238                 Simulation.queue_sem.release();
239             }
240         }
241     }
242 }
```

- Tar bort från väntekön
- Startar mikrotimer
- ZzzZz...
- Ber Erlang att ta bort tillståndprocess
- Släpp semaforer

# Teknisk Demonstration

Erlangmetod  
Erlangprocess



# Teknisk Demonstration

```
1 %% @doc The communication function which sends and recieves message from its child processes and Java
2 communicator(Pid_list, Java_node) ->
3     receive
4         {Mailbox, new_client, Switch_rate} ->
5             PID = spawner(Switch_rate),
6             {Mailbox, Java_node} ! {new_client, PID},
7             Updated_pid_list = [PID| Pid_list],
8             communicator(Updated_pid_list, Java_node);
9         {Mailbox, ready, PID} ->
10             PID ! {Mailbox, ready},
11             communicator(Pid_list, Java_node);
12         {Mailbox, client_ready, PID} ->
13             {Mailbox, Java_node} ! {client_ready, PID},
14             communicator(Pid_list, Java_node);
15         {Mailbox, terminate} ->
16             terminate(Pid_list),
17             {Mailbox, Java_node} ! good_bye,
18             rpc:call(list_to_atom(lists:append("erlcom@", net_adm:localhost()))), init, stop, []];
19         {_, kill, PID} ->
20             Updated_pid_list = lists:delete(PID, Pid_list),
21             PID ! kill,
22             communicator(Updated_pid_list, Java_node)
23     end.
```

# Guldkorn och Ruttna ägg



# Guldorn

- Kommunikationen mellan Erlang och Java
- Sparsam Erlangkod
- Grafiken
- Synkronisering i simuleringen

# Ruttna ägg

- För många statiska variabler och anrop
- Testning i Erlang
- Tillfälliga lösningar som vi inte hann fixa
  - startup.bat
  - Layout
  - Allmän uppsnyggning av koden
- Kompatibilitet