

Trådar

- En tråd beskriver en separat aktivitet inom ett program.
- Ett program kan ha flera trådar vilka exekverar parallellt (verkligt om man har flera processorer eller pseudoparallellt på en processor).
- Varje tråd har en aktuell exekveringspunkt.
- Varje tråd i ett Javaprogram beskrivs av en instans av klassen `Thread`.

```
Thread t = new Thread(r);    // skapar en ny tråd
```

Där `r` refererar till ett objekt av en klass som implementerar gränssnittet `Runnable`.

Detta gränssnitt har bara en metod:

```
void run();
```

Tråden startas med anropet

```
t.start();
```

Då kommer koden i metoden `run` att exekveras.

Andra grundläggande metoder i klassen `Thread`:

<code>interrupt()</code>	ber tråden att avsluta sin exekvering
<code>interrupted()</code>	ger true om tråden blivit ombedd att sluta
<code>sleep(m)</code>	låter tråden vänta i <code>m</code> ms (och <code>n</code> ns), ger
<code>sleep(m,n)</code>	<code>InterruptedException</code> om tråden blivit ombedd att sluta
<code>t.join()</code>	väntar tills tråden <code>t</code> har avslutats
<code>t.join(m)</code>	väntar tills tråden <code>t</code> har avslutats, väntar högst <code>m</code> ms
<code>t.join(m,n)</code>	väntar tills tråden <code>t</code> har avslutats, väntar högst <code>m</code> ms och <code>n</code> ns
<code>getPriority()</code>	ger prioriteten för tråden <code>t</code>
<code>setPriority(p)</code>	ändrar prioriteten för tråden <code>t</code>

```

public class Skrivare implements Runnable {
    public Thread aktivitet = new Thread(this);
    private String text;
    private long intervall;
    public Skrivare(String txt, long tid) {
        text=txt;
        intervall = tid*1000;
    }
    public void run() {
        while(true) {
            try {
                Thread.sleep(intervall);    // vänta
            }
            catch (InterruptedException e) { }
            System.out.print(text + " "); System.out.flush();
        }
    }
}

```

Om tråden ska kunna avbrytas:

```
public void run() {  
    while(!Thread.interrupted()) {  
        try {  
            Thread.sleep(intervall);  
        }  
        catch (InterruptedException e) {  
            break; // avbryt while-satsen  
        }  
        System.out.print(text + " "); System.out.flush();  
    }  
}
```

- Programkod som är utformad så att flera parallella trådar samtidigt kan använda sig av den utan att det blir fel, kallas *trådsäker* (*thread safe*) kod.
- I Java kan man göra en klass trådsäker genom att ange att vissa metoder skall vara *synkroniserade*.
- Ett objekt blir låst för andra trådar så länge en synkroniserad metod exekveras.

```
class Konto {  
    private double saldo;  
    ...  
    public synchronized void transaktion(double belopp) {  
        if (belopp<0 && saldo+belopp<0  
            System.out.println("Uttag ej möjligt!");  
        else  
            saldo = saldo+belopp;  
    }  
    ...  
}
```

Synkronisering med **wait** och **notify**.

```
import java.util.*;
public class SimpleQueue {
    private List<Object> l = new Vector<>();
    public int size() {
        return l.size();
    }
    public synchronized void put(Object obj) {
        l.add(obj);
        notify();
    }
    public synchronized Object take() {
        while (l.isEmpty())
            try {
                wait();
            }
            catch (InterruptedException e) {
                return null;
            }
        Object obj = l.get(0);
        l.remove(0);
        return obj;
    }
}
```