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`:

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

```
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;
  }
}
```