GRUPPE 1

Velkommen til
IN1010 – Uke 9

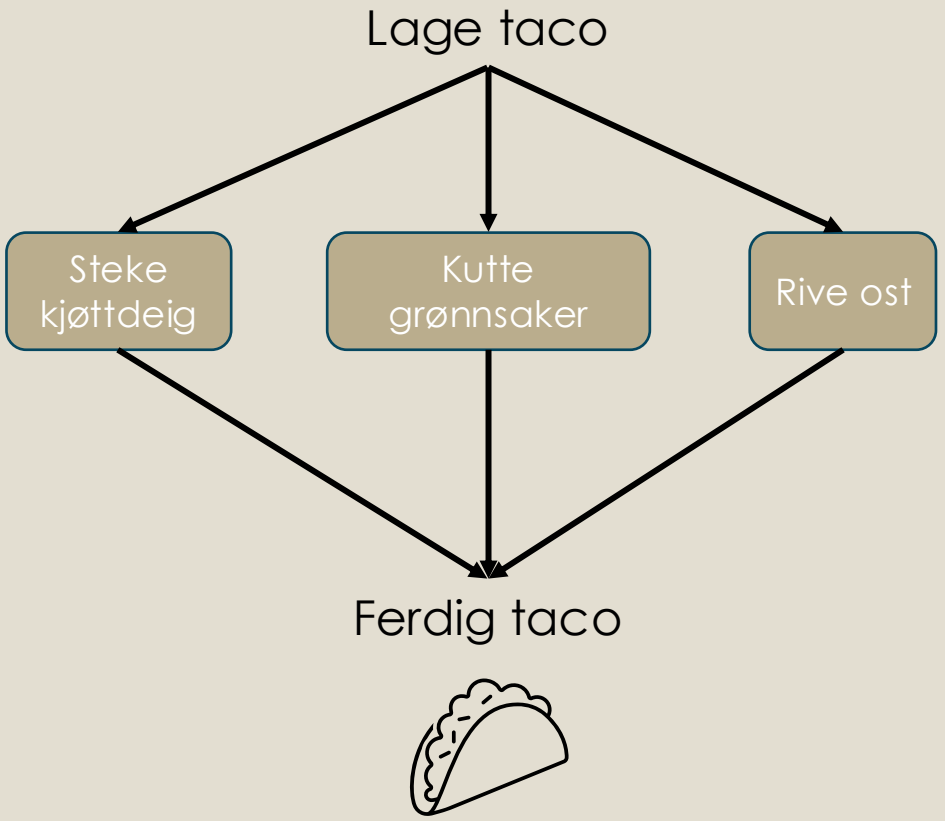
Plan for dagen

- Parallellitet
- Tråder
- Trådsikkerhet
  - Lock
  - Condition
  - Monitor

Parallellitet

- Så langt er vi vant med sekvensielle programmer
  - Kode blir kjørt fra topp til bunn, linje for linje
  - Kun en ting skjer av gangen

- Parallellitet lar oss gjøre flere oppgaver samtidig
  - F.eks. når vi lager taco



Hvordan kan vi oppnå dette i java?

- Tråder
  - En tråd er en sekvens med instruksjoner i et program
  - Vi kan starte flere tråder som kan eksekveres likt
  - Tråder kan vente, sove og avbrytes

- For å oppnå dette i Java må vi:
  - Lage en klasse som implementerer Runnable-interfacet
  - Implementere run() metoden
  - Starte tråden

Kode eksempel

```java
public class Tråd implements Runnable {

    @Override
    public void run() {
        System.out.println("Tråd kjører");
    }
}
```

Oppretter klasse som implementerer Runnable

```java
public static void main(String[] args) {
    Thread tråd = new Thread(new Tråd());
    tråd.start();
}
```

Oppretter en instans av klassen Thread med klassen vår som argument

Utfordringer med tråder

```java
public static void main(String[] args) {
    Thread tråd = new Thread(new Tråd());
    tråd.start();
    System.out.println("Main tråd kjører");
}
```

```
Tråd kjører
Main tråd kjører
```

```
Main tråd kjører
Tråd kjører
```

Utfordringer med tråder

```java
public class Teller implements Runnable {
    static int teller = 0;

    @Override
    public void run() {
        teller++;
        System.out.println(teller);
    }

    Run | Debug
    public static void main(String[] args) {
        Thread teller1 = new Thread(new Teller());
        Thread teller2 = new Thread(new Teller());

        teller1.start();
        teller2.start();
        System.out.println(Teller.teller);
    }
}
```

```
1
2
2
```

```
2
2
2
```

Hvordan håndtere dette?

```java
public static void main(String[] args) {
    Thread teller1 = new Thread(new Teller());
    Thread teller2 = new Thread(new Teller());

    try {
        teller1.start();
        teller1.join();

        teller2.start();
        teller2.join();
    } catch (InterruptedException e) {
        System.err.println(e);
    }

    System.out.println(Teller.teller);
}
```

```
1
2
2
```

Join venter til tråd er ferdig med å eksekvere før resten av programmet får kjørt

Synkronisering med låser

Låsen sørger for at kun en tråd kan aksessere variablene av gangen

```
1
2
```

```java
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

public class Teller implements Runnable {
    Lock lock = new ReentrantLock();
    int teller = 0;

    @Override
    public void run() {
        lock.lock();
        try {
            teller++;
            System.out.println(teller);
        } finally {
            lock.unlock();
        }
    }

    Run | Debug
    public static void main(String[] args) {
        Teller teller = new Teller();
        Thread teller1 = new Thread(teller);
        Thread teller2 = new Thread(teller);

        teller1.start();
        teller2.start();

    }
}
```

Monitor

- En klasse som håndterer synkroniseringen av trådene
- Tar kompleksiteten ut fra trådene
- Lagrer delt informasjon utenfor trådene

```java
class Bank {
    int pengebelop = 100;
    Lock laas = new ReentrantLock();
    Condition pengerIBanken = laas.newCondition();

    public void ta(int b) throws InterruptedException {
        laas.lock();
        try {
            while (pengebelop < b) {
                pengerIBanken.await();
            }
            pengebelop -= b;
        } finally {
            laas.unlock();
        }
    }

    public void gi(int b) {
        laas.lock();
        try {
            pengebelop += b;
            pengerIBanken.signalAll();
        } finally {
            laas.unlock();
        }
    }

    public int saldo() {
        laas.lock();
        try {
            return pengebelop;
        } finally {
            laas.unlock();
        }
    }
}
```

```java
class GiOgTa implements Runnable {
    int belop;
    Bank bank;

    public GiOgTa(int be, Bank ba) {
        belop = be;
        bank = ba;
    }

    public void run() {
        for (int i = 0; i < 10; i++) {
            try {
                bank.ta(belop);
            } catch (InterruptedException e) {
                System.err.println(("Avbrudd ved uttak av: " + belop));
            }
            bank.gi(belop);
            System.out.println(bank.saldo());
        }
    }
}
```

```java
class BankHoved {
    Run | Debug
    public static void main(String[] args) {
        Bank bank = new Bank();

        for (int i = 1; i <= 25; i++) {
            GiOgTa giOgTa = new GiOgTa(i, bank);
            Thread giOgTaThread = new Thread(giOgTa);
            giOgTaThread.start();
        }
    }
}
```