Ekstragruppe: Tråder

...

- Hva er en tråd?
- Hvordan bruker vi en tråd ?
- Synkronisering
  - Locks
  - Monitors

Hva er en tråd?

- En sekvens av instruksjoner
- Hvis vi ønsker at to oppgaver skal utføres samtidig bruker vi tråder
  - Utføres da i parallel
- Trådene kan dele på minne, og snakke sammen med hverandre, men....

Tråder i java

- To måter å lage en tråd på
  - implements Runnable (Det er denne måten vi vil bruke i 1010)
    - Runnable er et interface
  - extends Thread
    - Thread er en klasse
- Vi bruker Runnable fordi:
  - gjenbruke samme tråd på forskjellige oppgaver
  - samme oppgave utfører flere tråder
- Thread = worker
- Runnable = task
- https://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html
  - Dokumentasjonen på Runnable grensesnittet
- du starter en tråd med metoden start()

Oppgave

Lag en klasse som implementer grensesnitter Runnable.

Klassen skal ta inn en id (et heltalltall) i konstruktøren

Klassen sin jobb er er å skrive ut alle tallen fra 0 til og med 10 ganget med id.

Utskriften skal være “<id> * <tall mellom 0 og 10> = <resultat>”

Skrive et hovedprogram som starter opp trådene (start 10 tråder)

Løsningsforslag

```java
class Hovedprogram{
  public static void main(String[] args) {

    for(int i = 0; i < 10; i++){
      new Thread(new Gange(i)).start();
    }
  }
}

class Gange implements Runnable {

  private int id;

  public Gange(int id){
    this.id = id;
  }

  public void run(){
    for(int i = 0; i <= 10; i++){
      System.out.println(id + " * " + i + " = " + id*i);
    }
  }
}
```

Vise kjøring i terminalen!

Eksempel på kode:

```java
public class Main {
    public static void main(String[] args) {
        Runnable task = new MyTask();
        Thread worker1 = new Thread(task);
        Thread worker2 = new Thread(task);
        worker1.start();
        worker2.start();
    }
}

class MyTask implements Runnable {
    private final int MAX_COUNT = 10000;
    private int sharedCounter = 0;

    public void run() {
        System.out.println("Starting! Shared counter = " + sharedCounter);
        for (int i = 0; i < MAX_COUNT; i++) {
            sharedCounter = sharedCounter + 1;
        }
        System.out.println("Done! Shared counter = " + sharedCounter);
    }
```

“Hva tror du kommer til å bli skrevet ut ?”

```
marlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$ javac *.jav
a
marlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$ java Main
Starting! Shared counter = 0
Starting! Shared counter = 0
Done! Shared counter = 9605
Done! Shared counter = 14206
marlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$ java Main
Starting! Shared counter = 0
Starting! Shared counter = 0
Done! Shared counter = 10910
Done! Shared counter = 12304
marlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$ java Main
Starting! Shared counter = 0
Starting! Shared counter = 0
Done! Shared counter = 9724
Done! Shared counter = 13038
marlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$ java Main
Starting! Shared counter = 0
Starting! Shared counter = 0
Done! Shared counter = 10232
Done! Shared counter = 12396
marlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$
```

Synkronisering

- Hvis to tråder har tilgang på samme data må vi passe på.
- Huskeregel: kun en tråd kan endre en gitt data om gangen
- Men hvordan kan man løse dette ??
  - Det finnes flere måter å løse dette på !

Løsning 1: Locks

- For å bruke en Lock må man importer
  - java.util.concurrent.locks.Lock (Et grensesnitt, bl.a. to metoder lock() og unlock())
  - java.util.concurrent.locks.ReentrantLock (Lager de faktiske låseobjekten)
- Når man bruker Lock med lock() og unlock() er det viktig at man bruker en try-catch blokk. Nå må man også huske finally.
  - Finally er en kode snutt som vil skje uansett. Når det kommer til Locks må man passe på å alltid unlock() i en finally blokk slik at hvis det skulle skje noe med en tråd kan de andre trådene få fortsette

```
46    public void metode(){
47      lock.lock()
48      try{
49        <kode her>
50      }catch(Exception e){
51        <hvis det er en exception og catche>
52      }finally{
53        lock.unlock() //vil alltid skje
54      }
```

Hvordan kan du løse problemet i koden ved å bruke locks?

```java
public class Main {
    public static void main(String[] args) {
        Runnable task = new MyTask();
        Thread worker1 = new Thread(task);
        Thread worker2 = new Thread(task);
        worker1.start();
        worker2.start();
    }
}

class MyTask implements Runnable {
    private final int MAX_COUNT = 10000;
    private int sharedCounter = 0;

    public void run() {
        System.out.println("Starting! Shared counter = " + sharedCounter);
        for (int i = 0; i < MAX_COUNT; i++) {
            sharedCounter = sharedCounter + 1;
        }
        System.out.println("Done! Shared counter = " + sharedCounter);
    }
```

Løsningsforslag

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

class Main {
    public static void main(String[] args) {
        Runnable task = new MyTask();
        Thread worker1 = new Thread(task);
        Thread worker2 = new Thread(task);
        worker1.start();
        worker2.start();
    }
}

class MyTask implements Runnable {
    private final Lock lock = new ReentrantLock();
    private final int MAX_COUNT = 10000;
    private int sharedCounter = 0;

    public void run() {
        System.out.println("Starting! Shared counter = " + sharedCounter);
        for (int i = 0; i < MAX_COUNT; i++) {
            lock.lock();
            try{
              sharedCounter = sharedCounter + 1;
            }finally{
              lock.unlock();
            }
        }
        System.out.println("Done! Shared counter = " + sharedCounter);
    }
}
```

```
narlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$ java Main
Starting! Shared counter = 0
Starting! Shared counter = 0
Done! Shared counter = 18512
Done! Shared counter = 20000
narlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$ java Main
Starting! Shared counter = 0
Starting! Shared counter = 0
Done! Shared counter = 17779
Done! Shared counter = 20000
narlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$ java Main
Starting! Shared counter = 0
Starting! Shared counter = 0
Done! Shared counter = 18197
Done! Shared counter = 20000
narlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$ java Main
Starting! Shared counter = 0
Starting! Shared counter = 0
Done! Shared counter = 19651
Done! Shared counter = 20000
narlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$
```

Løsning 2: Monitor

- Et objekt som innkapsler den delta dataen
- Monitor objektet har metoder som gjør at de andre klassen f.eks. kan endre data og hente ut data

PS: Den metoden blir brukt i mye i 1010 fordi det er en god objektorientert måte å gjøre synkronisering på

Hvordan kan du løse problemet i koden ved å bruke Monitor?

```java
public class Main {
    public static void main(String[] args) {
        Runnable task = new MyTask();
        Thread worker1 = new Thread(task);
        Thread worker2 = new Thread(task);
        worker1.start();
        worker2.start();
    }
}

class MyTask implements Runnable {
    private final int MAX_COUNT = 10000;
    private int sharedCounter = 0;

    public void run() {
        System.out.println("Starting! Shared counter = " + sharedCounter);
        for (int i = 0; i < MAX_COUNT; i++) {
            sharedCounter = sharedCounter + 1;
        }
        System.out.println("Done! Shared counter = " + sharedCounter);
    }
```

Løsningsforslag

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

public class Main {
    public static void main(String[] args) {
        CountMonitor monitor = new CountMonitor();
        Runnable task = new MyTask(monitor);
        Thread worker1 = new Thread(task);
        Thread worker2 = new Thread(task);
        worker1.start();
        worker2.start();
    }
}
class CountMonitor {
    private final Lock lock = new ReentrantLock();
    private int sharedCounter = 0; // The protected data.

    public void increment() {
        lock.lock();
        try {
            sharedCounter = sharedCounter + 1;
        } finally {
            lock.unlock();
        }
    }
    public int getCounter() { return sharedCounter; }
}
```

```java
class MyTask implements Runnable {
    private final int MAX_COUNT = 10000;
    private final CountMonitor monitor;

    public MyTask(CountMonitor monitor) { this.monitor = monitor; }

    public void run() {
        for (int i = 0; i < MAX_COUNT; i++) {
            monitor.increment();
        }
        System.out.println("Done! Shared counter = " + monitor.getCounter());
    }
}
```

```
Done! Shared counter = 20000
marlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$ java Main
Done! Shared counter = 17280
Done! Shared counter = 20000
marlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$ java Main
Done! Shared counter = 20000
Done! Shared counter = 20000
marlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$ java Main
Done! Shared counter = 19080
Done! Shared counter = 20000
marlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$ java Main
Done! Shared counter = 20000
Done! Shared counter = 19698
marlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$ java Main
Done! Shared counter = 16745
Done! Shared counter = 20000
marlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$ java Main
Done! Shared counter = 20000
Done! Shared counter = 20000
marlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$ java Main
Done! Shared counter = 17089
Done! Shared counter = 20000
marlen@happyskappy:~/Documents/Gruppelærer/IN1010-2020/ekstra/uke10$
```

SPØRSMÅL??