IN1010 gruppetime

08.04.2026

Plan for dagen

Monitorer

Tråder

Oppsummering av korrelasjonen mellom Main, Monitor og Tråder

Case:

Kafé 

Håndtere når personer setter seg ved bord og går i fra bord **trådsikkert**.

Monitor

1. Gjør at bare en tråd kan operere om gangen.
(**mutual exclusion**)

2. Styrer når tråder skal vente / utføre run-metoden sin
(condition variables)

Monitor konstruktør

```
public class Kafé {
    private int ledigeSeter;
    private Lock lås = new ReentrantLock();
    private Condition ledigPlass = lås.newCondition();

    public Kafé(int antallSeter) {
        this.ledigeSeter = antallSeter;
    }

    public void settSeg() throws InterruptedException{

    }

    public void reiseSeg() {

    }
}
```

```
public void settSeg() throws InterruptedException{

}
```

```
public void settSeg() throws InterruptedException{
    ledigeSeter--;
}
```

```java
public void settSeg() throws InterruptedException{
    lås.lock();
    try {
        ledigeSeter--;
    } finally {
        lås.unlock();
    }
}
```

```java
public void settSeg() throws InterruptedException{
    lås.lock();
    try {
        while (ledigeSeter == 0) {
            ledigPlass.await();
        }
        ledigeSeter--;
    } finally {
        lås.unlock();
    }
}
```

```
public void reiseSeg() {

}
```

```
public void reiseSeg() {
    ledigeSeter++;
}
```

```
public void reiseSeg() {
    lås.lock();
    try {
        ledigeSeter++;
    } finally {
        lås.unlock();
    }
}
```

```
public void reiseSeg() {
    lås.lock();
    try {
        ledigeSeter++;
        ledigPlass.signalAll();
    } finally {
        lås.unlock();
    }
}
```

Hele kafé-klassen:

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

public class Kafé {
    private int ledigeSeter;
    private Lock lås = new ReentrantLock();
    private Condition ledigPlass = lås.newCondition();

    public Kafé(int antallSeter) {
        this.ledigeSeter = antallSeter;
    }

    public void settSeg() throws InterruptedException{
        lås.lock();
        try {
            while (ledigeSeter == 0) {
                ledigPlass.await();
            }
            ledigeSeter--;
        } finally {
            lås.unlock();
        }
    }

    public void reiseSeg() {
        lås.lock();
        try {
            ledigeSeter++;
            ledigPlass.signalAll();
        } finally {
            lås.unlock();
        }
    }
}
```

Person-klassen

```java
public class Person implements Runnable {
    private String navn;
    private Kafé kafé;

    public Person(String navn, Kafé kafé) {
        this.navn = navn;
        this.kafé = kafé;
    }

    @Override
    public void run() {
        try {
            System.out.println(navn + " prøver å sette seg.");
            kafé.settSeg();
            System.out.println(navn + " sitter nå.");

            // Personen sitter en stund
            Thread.sleep((long) (Math.random() * 3000));

            System.out.println(navn + " reiser seg.");
            kafé.reiseSeg();
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}
```

Main-klassen

```java
public class Main {
    Run | Debug
    public static void main(String[] args) throws InterruptedException {
        Kafé kafé = new Kafé(antallSeter: 3);

        String[] navn = {"Alice", "Bob", "Charlie", "Diana", "Erik", "Fatima"};

        Thread[] tråder = new Thread[navn.length];

        for (int i = 0; i < navn.length; i++) {
            tråder[i] = new Thread(new Person(navn[i], kafé));
            tråder[i].start();
        }

        for (Thread t : tråder) {
            t.join();
        }

        System.out.println(x: "Alle har forlatt kaféen!");
    }
}
```

Main

Oppretter tråder og monitoren

Styrer når tråder starter og slutter

Monitor

Har metoder som trådene kan kalle på

Holder på verdiene trådene potensielt kan f*cke opp

Tråder

Utfører oppgaven definert i oppgavebeskrivbelsen

Takk for meg!!