Løsningsforslag – Uke 11

1.

a)

Når du ønsker å vite at x antall oppgaver/ gjerne av x antall tråder er utført er det en fordel å benytte CountDownLatch. Denne teller ned til 0 og åpner da barrieren. CountDownLatchen *kan ikke resettes*.

CyclicBarrier er nyttig når du vil at det skal skje noe for hver n-te gang noe annet har blitt utført, uavhengig av hvilke tråder som gjorde oppgaven. CyclicBarrier teller ned til 0, og begynner så *på nytt igjen.*

b)

CyclicBarrier – fordi vi vil gjøre det hver femte gang(det blir altså noe som gjentar seg).

2.

CountDownEksempel.java

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

class CountDownEksempel{
  private static final int ANT_TRAADER = 10;
  public static void main(String[] args) {
    //Lag en barriere - sett til å vente for ANT_TRAADER kall til countDown()
    CountDownLatch altFerdig = new CountDownLatch(ANT_TRAADER);

    //Lag og start tråder
    LagreStorsteVerdi monitor = new LagreStorsteVerdi();
    for(int i = 0 ; i < ANT_TRAADER; i++){
      new Thread(new Deltager(monitor, altFerdig)).start();
    }

    //Vent på at alle deltagerene har lagt inn et tall.
    try{
      altFerdig.await();
    } catch(InterruptedException e){}
    System.out.println("Største: " + monitor.hentStorste());
  }
}
```

LagreStorsteVerdi.java

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

class LagreStorsteVerdi{
  private final Lock laas = new ReentrantLock();
  private int storste;

  public int hentStorste(){
    return storste;
  }

  public void giTall(int tall){
    laas.lock();
    try{
      storste = Math.max(storste, tall);
```

```
    }
    finally{
      laas.unlock();
    }
  }
}
```

Deltager.java

```
import java.util.concurrent.CountDownLatch;
import java.util.Random;

class Deltager implements Runnable{
  private LagreStorsteVerdi monitor;
  private CountDownLatch altFerdig;
  private int id;
  private static int antallArbeidereSomGjorDenneJobben = 0;

  public Deltager(LagreStorsteVerdi monitor, CountDownLatch altFerdig){
    this.id = antallArbeidereSomGjorDenneJobben++;
    this.monitor = monitor;
    this.altFerdig = altFerdig;
  }

  public void run(){
    //Generer og gi et tilfeldig tall
    Random random = new Random();
    int tall = random.nextInt(100);
    System.out.printf("Tråd #%d genererte tallet: %d \n", id, tall);
    monitor.giTall(tall);

    //Si i fra at vi er ferdig - vent på resten
    altFerdig.countDown();
    try{
      altFerdig.await();
    }
    catch(InterruptedException e){}

    //Hvis denne deltageren gav det høyeste tallet vant vedkommende!
    if(tall == monitor.hentStorste()){
      System.out.printf("Tråd #%d vant!\n", id);
    }
  }

}
```