19V
Datamatikker
Erhvervsakademi

John Grandt Markvard Høeg

DAOS Obligatorisk opgave

Indhold

Opgave 1

```
package opgave01;

import java.util.Random;

public class Faelles {

		private int taeller = 0;

		public Faelles () {
		}
		/**
		 * Metode der får tråden til at tage tid så der kan forekomme fejl
		 * @param max int værdi der bruges til at udregne max
		 */
		public void tagerRandomTid(int max) {
				Random r = new Random();
				int tal = 0;
				int nymax = Math.abs(r.nextInt())% max +1;
				for (int i = 0; i < nymax;i++) {
						for (int j = 0; j < nymax;j++) {
								tal += 1;
						}
				}
		}
		/*
		 * Metode der henter taeller
		 */
		public int getTaeller() {
				return taeller;
		}
		/*
		 * Metode der skal lave den kritiske section
		 */
		public void kritiskSection() {
				int temp;
				temp = taeller;
				tagerRandomTid(100);
				taeller = temp + 1;
		}
}
```

```
package opgave01;

public class MainApp {

	public static void main(String[] args) {
		//En ny faelles klasse
		Faelles f1 = new Faelles();
		//Der skabes 2 tråde (navn af tråd, faelles klasse der
bliver brugt)
		Thread t1 = new Traadeklassen("Monster Tråd 1", f1);
		Thread t2 = new Traadeklassen("Monster Tråd 2", f1);
		//Thread start
		t1.start();
		t2.start();
	}
}


package opgave01;

public class Traadeklassen extends Thread{

	private Faelles faelles;
	private String navn;
	/**
	 * Constructor for trådeklassen
	 * @param navn, String som bruges til at navngive klassen
	 * @param faelles, Faelles, tager en faelles klasse i brug
	 */
	public Traadeklassen (String navn, Faelles faelles) {
		super();
		this.faelles = faelles;
		this.navn = navn;
	}
	//Run metode for tråden.
	public void run() {
		for (int j = 0; j < 100; j++) {
			this.faelles.kritiskSection();
			this.faelles.tagerRandomTid(500);
			System.out.println(this.navn + " " + "Taeller
er: " + this.faelles.getTaeller());
		}
		System.out.println("Taeller er: " +
this.faelles.getTaeller());
	}

}
```

Opgave 2

```
package opgave02;

import java.util.Random;

public class Faelles {

	private boolean flag[];
	private int turn;
```

```
	private int taeller = 0;

	public Faelles () {
		flag = new boolean[2];
		flag[0] = false;
		flag[1] = false;
	}
	/**
	 * Metode der får tråden til at tage tid så der kan forekomme fejl
	 * @param max int værdi der bruges til at udregne max
	 */
	public void tagerRandomTid(int max) {
		Random r = new Random();
		int tal = 0;
		int nymax = Math.abs(r.nextInt())% max +1;
		for (int i = 0; i < nymax;i++) {
			for (int j = 0; j < nymax;j++) {
				tal += 1;
			}
		}
	}
	/*
	 * Metode der henter taeller
	 */
	public int getTaeller() {
		return taeller;
	}
	/*
	 * Metode der skal lave den kritiske section
	 */
	public void kritiskSection() {
		int temp;
		temp = taeller;
		tagerRandomTid(100);
		taeller = temp + 1;
	}
	/**
	 * Metode der sætter flag[id] til falsk eller sandt
	 * @param flag, boolean array bruges til at spørge om falsk eller
sandt
	 * @param id, int bruges til at identificer specific flag.
	 */
	public void setFlag(boolean flag, int id) {
		this.flag[id] = flag;
	}
	/**
	 * Simpel get flag metode
	 * @param id, int bruges til at identificer specific flag.
	 * @return sandt eller falsk på bestemt id
	 */
	public boolean getFlag(int id) {
		return flag[id];
	}
	/**
	 * Metode der sætter turn til 1 eller 0
	 * @param turn, int der bruges til hvems tur det er.
	 */
	public void setTurn(int turn) {
		this.turn = turn;
```

```
		}
		/**
		 * Simpel get turn metode
		 * @return turn.
		 */
		public int getTurn() {
			return turn;
		}
}
```

```
package opgave02;

public class MainApp {

		public static void main(String[] args) {
			//En ny faelles klasse
			Faelles f1 = new Faelles();
			//Id skal enten være 0 eller 1
			//Der skabes 2 tråde (navn af tråd, faelles klasse der
bliver brugt)
			Thread t1 = new Traadeklassen("Monster Tråd 1", f1, 0);
			Thread t2 = new Traadeklassen("Monster Tråd 2", f1, 1);
			//Thread start
			t1.start();
			t2.start();
		}
}
```

```
package opgave02;

public class Traadeklassen extends Thread {

		private Faelles faelles;
		private String navn;
		private int thisId;
		private int concurrentId;
		/**
		 * Constructor for trådeklassen
		 * @param navn, String som bruges til at navngive klassen
		 * @param faelles, Faelles, tager en faelles klasse i brug
		 * @param thisId, int der bruges for at adskille tråene i petersens
algoritme
		 */
		public Traadeklassen(String navn, Faelles faelles, int thisId) {
			super();
			this.faelles = faelles;
			this.navn = navn;
			this.thisId = thisId;
		}
		//Run metode for tråden.
		public void run() {
			concurrentId = (thisId + 1) % 2;
			for (int j = 0; j < 100; j++) {
				faelles.setFlag(true, thisId);
				faelles.setTurn(concurrentId);
				while (faelles.getFlag(concurrentId)
```

```
                                                && faelles.getTurn()
== concurrentId);
                                    this.faelles.kritiskSection();
                                    faelles.setFlag(false, thisId);
                                    this.faelles.tagerRandomTid(100);
                                    System.out.println(this.navn + " " + "Taeller
er: " + this.faelles.getTaeller());
                          }
                          System.out.println("Taeller er: " +
this.faelles.getTaeller());
              }
}
```

Opgave 3

```
package opgave03;
import java.util.Random;

public class Faelles {

              private int taeller = 0;

              public Faelles () {
              }
              /**
               * Metode der får tråden til at tage tid så der kan forekomme fejl
               * som er synchroniseret.
               * @param max int værdi der bruges til at udregne max
               */
              public synchronized void tagerRandomTid(int max) {
                            Random r = new Random();
                            int tal = 0;
                            int nymax = Math.abs(r.nextInt())% max +1;
                            for (int i = 0; i < nymax;i++) {
                                          for (int j = 0; j < nymax;j++) {
                                                        tal += 1;
                                          }
                            }
              }
              /*
               * Metode der henter taeller
               * som er synchroniseret.
               */
              public synchronized int getTaeller() {
                            return taeller;
              }
              /*
               * Metode der skal lave den kritiske section
               * som er synchroniseret.
               */
              public synchronized void kritiskSection() {
                            int temp;
                            temp = taeller;
                            tagerRandomTid(100);
                            taeller = temp + 1;
              }
}
```

```
package opgave03;

public class MainApp {

	public static void main(String[] args) {
			//En ny faelles klasse
			Faelles f1 = new Faelles();
			//Der skabes 2 tråde (navn af tråd, faelles klasse der
bliver brugt)
			Thread t1 = new Traadeklassen("Monster Tråd 1", f1);
			Thread t2 = new Traadeklassen("Monster Tråd 2", f1);
			//Thread start
			t1.start();
			t2.start();
	}
}


package opgave03;

public class Traadeklassen extends Thread{

	private Faelles faelles;
	private String navn;
	/**
	 * Constructor for trådeklassen
	 * @param navn, String som bruges til at navngive klassen
	 * @param faelles, Faelles, tager en faelles klasse i brug
	 */
	public Traadeklassen (String navn, Faelles faelles) {
			super();
			this.faelles = faelles;
			this.navn = navn;
	}
	//Run metode for tråden.
	public void run() {
			for (int j = 0; j < 100; j++) {
				this.faelles.kritiskSection();
				this.faelles.tagerRandomTid(100);
				System.out.println(this.navn + " " + "Taeller
er: " + this.faelles.getTaeller());
			}
			System.out.println("Taeller er: " +
this.faelles.getTaeller());
	}

}
```