L03. Labwerk MultiThreading

Inleiding

Stapsgewijze handleiding voor Multi-Threading in Java:

Maak een nieuw Java-project

- Open uw favoriete IDE en maak een nieuw Java-project.
- Noem de project: "**LabwerkMultiThreading**"

Maak een nieuwe Java-klasse

- Klik met de rechtermuisknop op het project en selecteer "Nieuw" > "Klasse".
- Geef de klasse de naam "ThreadVoorbeeld" en klik op "Voltooien".

```
public class ThreadVoorbeeld {
    // Code voor deze klasse komt later
}
```

Maak een nieuwe Thread-klasse

- In de ThreadVoorbeeld-klasse maakt u een nieuwe innerlijke klasse die de Thread-klasse uitbreidt.
- Overschrijf de run() methode in deze klasse.

```
public class ThreadVoorbeeld {
    // Maak een nieuwe innerlijke klasse die de Thread-klasse uitbreidt
    private class InnerThread extends Thread {
        // Overschrijf de run() methode
        @Override
        public void run() {
            // Code voor deze methode komt later
        }
    }
}
```

Definieer de code die moet worden uitgevoerd in de thread

- Voeg binnen de run() methode de code toe die u wilt uitvoeren in de thread.

```
public class ThreadVoorbeeld {
    // Maak een nieuwe innerlijke klasse die de Thread-klasse uitbreidt
    private class InnerThread extends Thread {
        // Overschrijf de run() methode
        @Override
        public void run() {
            // Definieer de code die moet worden uitgevoerd in de thread
            System.out.println("Hallo vanaf thread " + this.getId());
        }
```

```
    }
}
```

Maak een instantie van de Thread-klasse

- Maak in de ThreadVoorbeeld-klasse een instantie van de Thread-klasse.
- Geef de zojuist gemaakte innerlijke klasse als argument door aan de constructor.

```
public class ThreadVoorbeeld {
    public static void main(String[] args) {
        // Maak een instantie van de Thread-klasse
        InnerThread thread = new InnerThread();
    }

    // Maak een nieuwe innerlijke klasse die de Thread-klasse uitbreidt
    private static class InnerThread extends Thread {
        // Overschrijf de run() methode
        @Override
        public void run() {
            // Definieer de code die moet worden uitgevoerd in de thread
            System.out.println("Hallo vanaf thread " + this.getId());
        }
    }
}
```

Start de thread

- Roep de start() methode aan op de instantie van de Thread-klasse die u zojuist heeft gemaakt.

```
public class ThreadVoorbeeld {
    public static void main(String[] args) {
        // Maak een instantie van de Thread-klasse
        InnerThread thread = new InnerThread();

        // Start de thread
        thread.start();
    }

    // Maak een nieuwe innerlijke klasse die de Thread-klasse uitbreidt
    private static class InnerThread extends Thread {
        // Overschrijf de run() methode
        @Override
        public void run() {
            // Definieer de code die moet worden uitgevoerd in de thread
            System.out.println("Hallo vanaf thread " + this.getId());
        }
    }
}
```

Test uw code

- Voer uw Java-programma uit en verifieer dat de code binnen de thread wordt uitgevoerd.

```
public class ThreadVoorbeeld {

    public static void main(String[] args) {
        // Maak een instantie van de Thread-klasse
        InnerThread thread = new InnerThread();

        // Start de thread
        thread.start();

        // Verifieer dat de code binnen de thread wordt uitgevoerd
        System.out.println("Hallo vanaf de hoofdthread");
    }

    // Maak een nieuwe innerlijke klasse die de Thread-klasse uitbreidt
    private static class InnerThread extends Thread {
        // Overschrijf de run() methode
        @Override
        public void run() {
            // Definieer de code die moet worden uitgevoerd in de thread
            System.out.println("Hallo vanaf thread " + this.getId());
        }
    }

}
```

Gebruik de Thread.sleep() methode

- Voeg binnen de run() methode van de innerlijke klasse de Thread.sleep() methode toe om een vertraging in de uitvoering van de thread te simuleren.

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

        // Maak meerdere instanties van de Thread-klasse
        InnerThread thread1 = new InnerThread();
        InnerThread thread2 = new InnerThread();

        // Start de threads
        thread1.start();
        thread2.start();

        // Verifieer dat de code binnen de threads wordt uitgevoerd
        System.out.println("Hallo vanaf de hoofdthread");
    }

    // Maak een nieuwe innerlijke klasse die de Thread-klasse uitbreidt
    private static class InnerThread extends Thread {
        // Overschrijf de run() methode
        @Override
        public void run() {
```

```
            // Definieer de code die moet worden uitgevoerd in de thread
            System.out.println("Hallo vanaf thread " + this.getId());
        }
    }

}
```

Gebruik de join() methode

- Maak in de ThreadVoorbeeld-klasse nog een instantie van de Thread-klasse.
- Roep de join() methode aan op de eerste instantie van de thread om te wachten tot deze is voltooid voordat de tweede instantie van de thread wordt gestart.
- Roep de join() methode aan op de tweede instantie van de thread om te wachten tot deze is voltooid voordat de derde instantie van de thread wordt gestart.
-

```
// ...
```

Gebruik het synchronized-woord

- Voeg binnen de run() methode van de innerlijke klasse het synchronized-woord toe om gedeelde bronnen te beschermen tegen gelijktijdige toegang door meerdere threads.

En dat is het! U weet nu hoe u multi-threaded toepassingen in Java kunt maken en uitvoeren.

```
public class ThreadVoorbeeld {
    private int count = 0;

    public static void main(String[] args) {
        ThreadVoorbeeld voorbeeld = new ThreadVoorbeeld();

       // Maak meerdere instanties van de Thread-klasse


        InnerThread thread1 = new InnerThread(voorbeeld);


        InnerThread thread2 = new InnerThread(voorbeeld);

        InnerThread thread3 = new InnerThread(voorbeeld);

       // Start de threads
              thread1.start();


              thread2.start();

              thread2.start();

              // Verifieer dat de code binnen de threads wordt uitgevoerd
```

```
        System.out.println("Hallo vanaf de hoofdthread");
    }

    // Maak een nieuwe innerlijke klasse die de Thread-klasse uitbrei
dt
    private static class InnerThread extends Thread {
        private ThreadVoorbeeld voorbeeld;

        public InnerThread(ThreadVoorbeeld voorbeeld) {
            this.voorbeeld = voorbeeld;
        }

        // Overschrijf de run() methode
        @Override
        public void run() {
            // Roep de synchronized-methode aan om race-voorwaarden t
e voorkomen
            voorbeeld.incrementCount();
            System.out.println("Hallo vanaf thread " + this.getId() +
", count is nu " + voorbeeld.getCount());
        }
    }

    // Gebruik synchronized om race-voorwaarden te voorkomen
    public synchronized void incrementCount() {
        count++;
    }

    public int getCount() {
        return count;
    }
}
```

Ik hoop dat dit u heeft geholpen om multi-threading in Java te begrijpen en te implementeren!