

LAPORAN PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK

PERTEMUAN KE-14



Disusun Oleh :

NAMA : Raden Isnawan Argi Aryasatya

NIM : 195410257

JURUSAN : Teknik Informatika

JENJANG : S1

KELAS : TI-5

Laboratorium Terpadu
Sekolah Tinggi Manajemen Informatika Komputer
AKAKOM
YOGYAKARTA

2021

PERTEMUAN KE-14 (KONKURENSI)

TUJUAN

Dapat menggunakan kelas Thread untuk menangani konkurensi

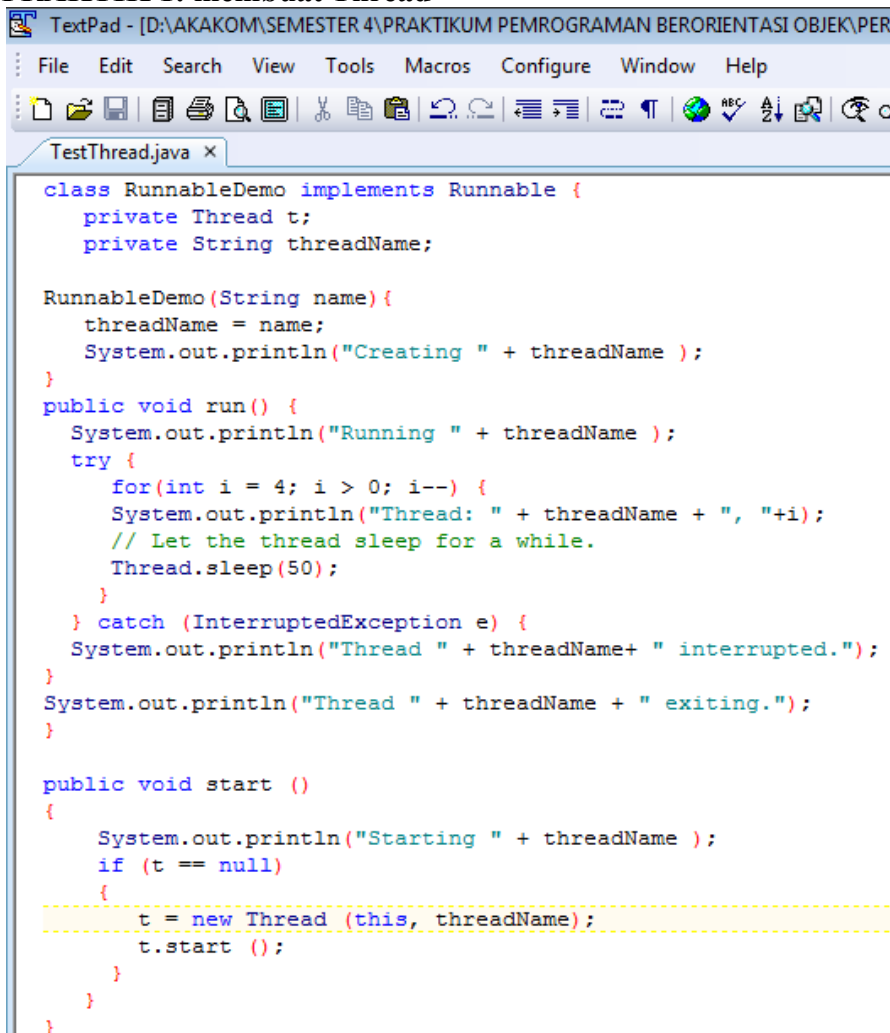
DASAR TEORI

Sebuah thread, digunakan untuk meningkatkan fungsionalitas dan performansi dengan cara melakukan beberapa tugas pada saat yang sama, yaitu bersamaan. Ada dua metode untuk menerapkan thread di dalam Java :

1. penerapan sebuah antarmuka
2. perpanjangan sebuah class

Thread merupakan pemrograman Java tingkat menengah, untuk itu mahasiswa diharapkan sudah mengenal dengan konsep-konsep dasar Object Oriented Paradigm dan mengerti dengan istilah seperti 'extending', 'interface' dan 'class'. Alasan mengapa ada dua cara membuat thread. Hal ini dikarenakan jika sebuah class sudah menjadi sebuah class turunan dari beberapa class selain 'Thread', maka ini tidak dapat memperpanjang 'Thread' karena beberapa turunan tidak diperbolehkan di dalam pemrograman bahasa Java. Jadi, dalam kasus seperti itu kita gunakan antarmuka 'Runnable' sebagai gantinya.

PRAKTIK 1: membuat Thread



```
TextPad - [D:\AKAKOM\SEMESTER 4\PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK\PER
File Edit Search View Tools Macros Configure Window Help
TestThread.java x
class RunnableDemo implements Runnable {
    private Thread t;
    private String threadName;

    RunnableDemo(String name){
        threadName = name;
        System.out.println("Creating " + threadName );
    }
    public void run() {
        System.out.println("Running " + threadName );
        try {
            for(int i = 4; i > 0; i--) {
                System.out.println("Thread: " + threadName + ", " + i);
                // Let the thread sleep for a while.
                Thread.sleep(50);
            }
        } catch (InterruptedException e) {
            System.out.println("Thread " + threadName + " interrupted.");
        }
        System.out.println("Thread " + threadName + " exiting.");
    }

    public void start ()
    {
        System.out.println("Starting " + threadName );
        if (t == null)
        {
            t = new Thread (this, threadName);
            t.start ();
        }
    }
}
```

penjelasan:

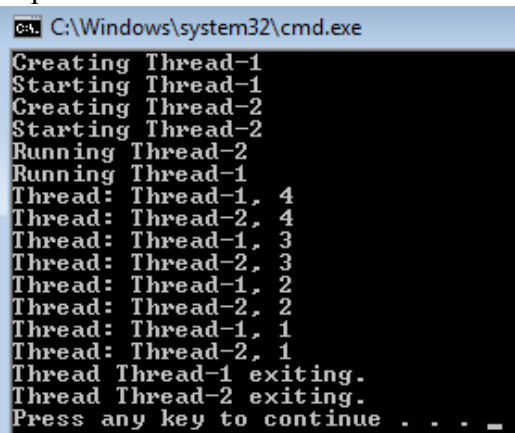
output tidak ada karena main method belum dibuat. Inti dari program ini adalah untuk membuat Thread dan Runnable yang digunakan untuk membuat aplikasi multitasking atau aplikasi yang bisa menjalankan tugas ganda secara bersamaan. Thread merupakan sebuah rangkaian eksekusi yang berjalan ketika program dijalankan atau bisa disebut sebagai kode yang mengatur alur jalannya program. Pada contoh program pertama ini, kita membuat Runnable yang digunakan agar thread bisa menjalankan tugasnya. Cara paling sederhana untuk membuat thread adalah dengan mengimplementasikan interface Runnable pada class. Jadi kita cukup menimplementasikan fungsi dari method run.

Untuk pembuatan programnya, kita mendeklarasikan class RunnableDemo implements Runnable. Class tersebut memiliki dua variabel yaitu t bertipe Thread dan threadName bertipe String. Lalu kita membuat konstruktor untuk mengakses nama dengan threadName = name. Lalu kita cetak kalimat dengan menuliskan **System.out.println("Creating " + threadName);**. selanjutnya kita buat method bernama run() yang di dalamnya ada try dan catch. Try dan catch digunakan untuk menangkap jenis error InterruptedException agar program tidak crash saat error itu terjadi. Lalu kita buat perulangan for berupa **for(int i = 4; i > 0; i--)** yang artinya perhitungan dimulai dari angka 4 dan terus berkurang hingga angka 1. Sementara Thread.sleep(50) artinya kita mengset waktu pending selama 50 millisecond/1detik. Terakhir kita buat method start() yang akan memunculkan mode start setelah program di-create.

PRAKTIK 2: Menggunakan kelas RunnableDemo

```
public class TestThread {  
    public static void main(String args[]) {  
        RunnableDemo R1 = new RunnableDemo( "Thread-1");  
        R1.start();  
        RunnableDemo R2 = new RunnableDemo( "Thread-2");  
        R2.start();  
    }  
}
```

output:



```
C:\Windows\system32\cmd.exe  
Creating Thread-1  
Starting Thread-1  
Creating Thread-2  
Starting Thread-2  
Running Thread-2  
Running Thread-1  
Thread: Thread-1, 4  
Thread: Thread-2, 4  
Thread: Thread-1, 3  
Thread: Thread-2, 3  
Thread: Thread-1, 2  
Thread: Thread-2, 2  
Thread: Thread-1, 1  
Thread: Thread-2, 1  
Thread Thread-1 exiting.  
Thread Thread-2 exiting.  
Press any key to continue . . . _
```

penjelasan:

program di atas merupakan main method dari program di praktik 1 yang kita buat tadi. Di dalam program ini kita membuat instansiasi objek. Sebelum itu, kita awali eksekusi dengan menuliskan **public static void main(String args[])**. Di dalamnya kita mulai instansiasi dengan membuat objek R1 yang akan memunculkan Thread 1 dengan method start. Kedua kita membuat objek R2 untuk memunculkan Thread 2 dengan method start. Kita lakukan itu dengan menuliskan baris-baris kode:

```
RunnableDemo R1 = new RunnableDemo( "Thread-1");  
R1.start();  
RunnableDemo R2 = new RunnableDemo( "Thread-2");  
R2.start();
```

PRAKTIK 3: Membuat Thread menggunakan extend Thread

```
TestThread.java x
class ThreadDemo extends Thread {
    private Thread t;
    private String threadName;

    ThreadDemo( String name){
        threadName = name;
        System.out.println("Creating " + threadName );
    }
    public void run() {
        System.out.println("Running " + threadName );
        try {
            for(int i = 4; i > 0; i--) {
                System.out.println("Thread: "+threadName + ", " + i);
                // Let the thread sleep for a while.
                Thread.sleep(50);
            }
        } catch (InterruptedException e) {
            System.out.println("Thread "+threadName+" interrupted.");
        }
        System.out.println("Thread " + threadName + " exiting.");
    }

    public void start ()
    {
        System.out.println("Starting " + threadName );
        if (t == null)
        {
            t = new Thread (this, threadName);
            t.start ();
        }
    }
}
```

penjelasan:

program ini sebenarnya outputnya sama dengan program tadi, hanya sekarang kita menggunakan Thread, bukan Runnable. Thread merupakan sebuah rangkaian eksekusi yang berjalan ketika program dijalankan atau bisa disebut sebagai kode yang mengatur alur jalannya program.

Program diawali dengan membuat class ThreadDemo dan extends ke Thread. Di dalamnya ada variabel Thread t dan String threadName. Kita membuat konstruktor ThreadDemo untuk mengakses nama dengan threadName = name. Kita cetak kalimat dengan menuliskan System.out.println("Creating " + threadName);. Setelah itu kita buat method bernama run() yang di dalamnya ada try dan catch. Try dan catch digunakan untuk menangkap jenis error InterruptedException agar program tidak crash saat error terjadi. Lalu kita buat perulangan for berupa for(int i = 4; i > 0; i--) yang artinya perhitungan dimulai dari angka 4 dan terus berkurang hingga angka 1. Thread.sleep(50) artinya kita mengset waktu pending selama 50 millisecond/1detik. Terakhir kita buat method start() yang akan memunculkan mode start.

PRAKTIK 4: Menggunakan Thread

```
public class TestThread {
    public static void main(String args[]) {
        ThreadDemo T1 = new ThreadDemo( "Thread-1");
        T1.start();
        ThreadDemo T2 = new ThreadDemo( "Thread-2");
        T2.start();
    }
}
```

output di halaman selanjutnya

```
C:\Windows\system32\cmd.exe
Creating Thread-1
Starting Thread-1
Creating Thread-2
Starting Thread-2
Running Thread-2
Running Thread-1
Thread: Thread-1, 4
Thread: Thread-2, 4
Thread: Thread-2, 3
Thread: Thread-1, 3
Thread: Thread-2, 2
Thread: Thread-1, 2
Thread: Thread-2, 1
Thread: Thread-1, 1
Thread Thread-1 exiting.
Thread Thread-2 exiting.
Press any key to continue . . .
```

penjelasan:

Di dalam program ini kita membuat instansiasi objek. Sebelum itu, kita awali eksekusi dengan menuliskan public static void main(String args[]). Di dalamnya kita mulai instansiasi dengan membuat objek T1 yang akan memunculkan Thread 1 dengan method start. Kedua kita membuat objek T2 untuk memunculkan Thread 2 dengan method start. Kita lakukan itu dengan menuliskan baris-baris kode:

```
ThreadDemo T1 = new ThreadDemo( "Thread-1");
T1.start();
ThreadDemo T2 = new ThreadDemo( "Thread-2");
T2.start();
```

PRAKTIK 5: Dengan menggunakan Anonymous Inner Classes

```
MyClass.java x TestThread.java
public class MyClass{
    public MyClass(){
        new Thread(){
            public void run(){
                System.out.println("Threat dijalankan ");
            }
        }.start();
    }

    public static void main(String[] args){
        new MyClass ();
    }
}
```

output:

```
C:\Windows\system32\cmd.exe
Threat dijalankan
Press any key to continue . . .
```

penjelasan:

Kadang kita tidak ingin menciptakan class untuk Thread dan kondisinya tidak cocok untuk pake interface Runnable. Jika hal tersebut terjadi, maka dapat membuat anonymous inner class. Program yang kita buat ini sangat simple karena hanya memunculkan satu baris kalimat. Kita hanya mendeklarasikan MyClass() dan new Thread(), lalu kita buat method run() yang memunculkan kalimat "Threat dijalankan ", lalu thread akan mulai dengan method start(). Terakhir kita buat main method untuk memanggil MyClass dan menampilkan hasil di output dengan new MyClass().

=====

KESIMPULAN

Pada pertemuan ke-14 sekaligus pertemuan terakhir ini, kita belajar banyak tentang konkurensi. Inti dari pembuatan laporan ini adalah mempraktekkan program Thread dan Runnable, sekaligus ditambah program thread anonymous di praktik terakhir. Dari sini, kita menjadi tahu cara membuat Thread sekaligus tahu struktur siklus hidup thread yaitu:

- New: Sebuah thread baru start siklus hidupnya di negara baru.
- Runnable: Setelah thread baru new start, thread menjadi runnable.
- Waiting: Kadang-kadang, thread transisi ke negara menunggu sementara thread menunggu thread lain untuk melakukan task.
- Timed waiting: Sebuah thread runnable dapat memasuki state menunggu waktunya untuk interval waktu tertentu. Sebuah thread di state bagian ini transisi kembali ke keadaan runnable ketika interval waktu berakhir atau ketika acara itu sedang menunggu terjadi.
- Terminated: Sebuah thread runnable memasuki keadaan dihentikan ketika selesai tugasnya atau berakhir.

Terima Kasih