PRAKTIKUM

THREAD METHOD

A. TUJUAN PEMBELAJARAN

1. Mampu mengimplementasikan beberapa method dalam thread
2. Memahami kegunaan beberapa method dalam thread

B. DASAR TEORI

Pada praktikum ini akan dilakukan percobaan untuk implementasi beberapa method yang ada dalam Thread.

C. TUGAS PENDAHULUAN

1. Jelaskan kegunaan method sleep() dan join() dalam Thread!
2. Jelaskan kegunaan method getName() dan setName() dalam Thread!
3. Jelaskan kegunaan setPriority() dalam Thread!
4. Apa yang dimaksud dengan Daemon dalam Thread?
5. Apa kelebihan atau manfaat dari penggunaan Thread pool?
6. Apa perbedaan Thread pool dan ThreadGroup?

D. PERCOBAAN

Percobaan 1 : Method sleep() dalam Thread

Jalankn class ThreadSleep berikut ini, amati dan catat hasil pengamatan anda. Hapuslah baris program yang diberi tanda merah, kemudian jalankan lagi program anda. Amati perbedaan yang terjadi setelah anda menghilangkan baris code tersebut!

```
class TreadSleep extends Thread{
 public void run(){
  for(int i=1;i<100;i++){
   try{Thread.sleep(500);}catch(InterruptedException e){System.out.println(e);}
   System.out.println(i);
  }
 }
 public static void main(String args[]){
  TreadSleep t1=new TreadSleep();
  TreadSleep t2=new TreadSleep();

  t1.start();
  t2.start();
 }
}
```

Percobaan 2 : Method join() dalam Thread

Jalankan program ThreadJoin berikut. Amati hasil outputnya dan buatlah analisa mengenai penggunaan method join() dalam program tersebut!

```
class ThreadJoin extends Thread {
    public void run() {
        for (int i = 1; i <= 5; i++) {
            try {
                Thread.sleep(500);
            } catch (Exception e) {
                System.out.println(e);
            }
            System.out.println(i);
        }
    }

    public static void main(String args[]) {
        ThreadJoin t1 = new ThreadJoin();
        ThreadJoin t2 = new ThreadJoin();
        ThreadJoin t3 = new ThreadJoin();
        t1.start();
        try {
            t1.join();
        } catch (Exception e) {
            System.out.println(e);
        }

        t2.start();
        t3.start();
    }
}
```

Percobaan 3 : Method getName() dan setName()

Program berikut adalah contoh penggunaan method getName() dan setName() dalam Thread!

```java
public class NamingThread extends Thread {

    public void run() {
        System.out.println("running...");
    }

    public static void main(String args[]) {
        NamingThread t1 = new NamingThread();
        NamingThread t2 = new NamingThread();
        System.out.println("Name of t1:" + t1.getName());
        System.out.println("Name of t2:" + t2.getName());

        t1.start();
        t2.start();

        t1.setName("Sonoo Jaiswal");
        System.out.println("After changing name of t1:" + t1.getName());
    }
}
```

Percobaan 4 : Method setPriority() dalam Thread

Dalam program berikut terdapat dua buah thread yaitu m1 dan m2 yang merupakan intance dari ThreadPriority class. Thread m1 di set priority MAX, sedangkan m2 priority MIN. Thread m1 distart sebelum m2 namun thread manakah yang dijalankan terlebih dahulu oleh thread scheduler? Mengapa demikian?

```java
public class ThreadPriority extends Thread {
    public void run() {
        System.out.println("running thread name is:" + Thread.currentThread().getName());
        System.out.println("running thread priority is:" + Thread.currentThread().getPriority());

    }
    public static void main(String args[]) {
        ThreadPriority m1 = new ThreadPriority();
        ThreadPriority m2 = new ThreadPriority();
        m1.setPriority(Thread.MIN_PRIORITY);
        m2.setPriority(Thread.MAX_PRIORITY);
        m1.start();
        m2.start();

    }
}
```

Percobaan 5 : Implementasi Daemon Thread

Berikut adalah contoh implementasi daemon thread dalam java.

```java
public class DaemonThread extends Thread {

    public void run() {
        if (Thread.currentThread().isDaemon()) {//checking for daemon thread
            System.out.println("daemon thread work");
        } else {
            System.out.println("user thread work");
        }
    }

    public static void main(String[] args) {
        DaemonThread t1 = new DaemonThread();//creating thread
        DaemonThread t2 = new DaemonThread();
        DaemonThread t3 = new DaemonThread();

        t1.setDaemon(true);//now t1 is daemon thread

        t1.start();//starting threads
        t2.start();
        t3.start();
    }
}
```

Percobaan 6 : Thread Pool

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class ThreadPool {
    public static void main(String[] args) {
        ExecutorService executor = Executors.newFixedThreadPool(5);//creating a pool of 5 threads
        for (int i = 0; i < 10; i++) {
            Runnable worker = new WorkerThread("" + i);
            executor.execute(worker);//calling execute method of ExecutorService
        }
        executor.shutdown();
        while (!executor.isTerminated()) {
        }
        System.out.println("Finished all threads");
    }
}

class WorkerThread implements Runnable {
    private String message;
    public WorkerThread(String s) {
        this.message = s;
    }
    public void run() {
        System.out.println(Thread.currentThread().getName() + " (Start) message = " + message);
        processmessage();//call processmessage method that sleeps the thread for 2 seconds
        System.out.println(Thread.currentThread().getName() + " (End)");//prints thread name
    }
    private void processmessage() {
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Percobaan 7 : Thread Group

```
public class ThreadGroupTest implements Runnable {

    public void run() {
        System.out.println(Thread.currentThread().getName());
    }

    public static void main(String[] args) {
        ThreadGroupTest runnable = new ThreadGroupTest();
        ThreadGroup tg1 = new ThreadGroup("Parent ThreadGroup");

        Thread t1 = new Thread(tg1, runnable, "one");
        t1.start();
        Thread t2 = new Thread(tg1, runnable, "two");
        t2.start();
        Thread t3 = new Thread(tg1, runnable, "three");
        t3.start();

        System.out.println("Thread Group Name: " + tg1.getName());
        tg1.list();

    }
}
```

E. LATIHAN

Latihan 1 : Aplikasi Jam

Buatlah aplikasi Jam menggunakan Thread dan method sleep()!

Latihan 2 : Aplikasi Timer

Buatlah aplikasi Timer yang memunculkan alert pada waktu yang telah ditentukan oleh user!

F. TUGAS

1. Buatlah sebuah aplikasi Jam Digital dengan fitur Multi Alarm! Jumlah alarm bisa ditambah dan dikurangi oleh user.

G. LAPORAN RESMI

Buatlah laporan untuk hasil percobaan aplikasi Jam Digital dengan fitur Multi Alarm. Tambahkan analisa dari hasil percobaan tersebut.