Concurrency

Thread Feature

Pemrograman Berorientasi Objek

Riza Satria Perdana, S.T., M.T.

Teknik Informatika - STEI ITB

Method Sleep

- Menghentikan thread untuk sementara waktu

```
public class SleepMessages {
    public static void main(String args[]) throws InterruptedException {
        String importantInfo[] = {
            "Mares eat oats",
            "Does eat oats",
            "Little lambs eat ivy",
            "A kid will eat ivy too"
        };

        for (int i = 0; i < importantInfo.length; i++) {
            //Pause for 4 seconds
            Thread.sleep(4000);
            //Print a message
            System.out.println(importantInfo[i]);
        }
    }
}
```

Interrupts

- Thread diminta menghentikan eksekusi untuk mengerjakan hal tertentu
- Mengirim interupsi dengan memanggil method interrupt pada thread objek

Menangani Interrupts

- Catch InterruptExecution objek
- Memanggil Thread.interrupted

```
for (int i = 0; i < inputs.length; i++) {
    heavyCrunch(inputs[i]);
    if (Thread.interrupted()) {
        // We've been interrupted: no more crunching.
        return;
    }
}
```

Thread Join

- Method join digunakan untuk menunggu thread lain selesai

```
t.join()
```

Contoh

```
public class SimpleThreads {

    // Display a message, preceded by the name of the current thread
    static void threadMessage(String message) {
        String threadName = Thread.currentThread().getName();
        System.out.format("%s: %s%n", threadName, message);
    }

    private static class MessageLoop implements Runnable {
        public void run() {
            String importantInfo[] = {
                "Mares eat oats",
                "Does eat oats",
                "Little lambs eat ivy",
                "A kid will eat ivy too"
            };
            try {
                for (int i = 0; i < importantInfo.length; i++) {
                    // Pause for 4 seconds
                    Thread.sleep(4000);
                    // Print a message
                    threadMessage(importantInfo[i]);
                }
            } catch (InterruptedException e) {
                threadMessage("I wasn't done!");
            }
        }
    }
```

```
public static void main(String args[]) throws InterruptedException {

    // Delay, in milliseconds before we interrupt MessageLoop
    // thread (default one hour).
    long patience = 1000 * 60 * 60;

    // If command line argument present, gives patience in seconds.
    if (args.length > 0) {
        try {
            patience = Long.parseLong(args[0]) * 1000;
        } catch (NumberFormatException e) {
            System.err.println("Argument must be an integer.");
            System.exit(1);
        }
    }

    threadMessage("Starting MessageLoop thread");
    long startTime = System.currentTimeMillis();
    Thread t = new Thread(new MessageLoop());
    t.start();
```

Contoh

Contoh

```
            threadMessage("Waiting for MessageLoop thread to finish");
            // loop until MessageLoop thread exits
            while (t.isAlive()) {
                threadMessage("Still waiting...");
                // Wait maximum of 1 second for MessageLoop thread to finish.
                t.join(1000);
                if (((System.currentTimeMillis() - startTime) > patience)
                      && t.isAlive()) {
                    threadMessage("Tired of waiting!");
                    t.interrupt();
                    // Shouldn't be long now -- wait indefinitely
                    t.join();
                }
            }
            threadMessage("Finally!");
        }
    }
```

Terima Kasih