Instituto Politécnico Nacional

Escuela Superior de Cómputo

Programación Orientada a Objetos

Reporte de Práctica #4

Hilos

Profesor: Roberto Tecla Parra

Alumno: Calva Hernández José Manuel

Grupo: 2CM3

Ticker

Mover las letras de un texto de izquierda a derecha (o de derecha a izquierda) sin mover el texto de lugar. Se puede quitar la primer letra y pegarla (con el operador +) al final y repetir esto de forma periódica.

Objetivos

- ➢ Conocer el concepto de Hilo y Multitarea en POO.
- ➢ Conocer el concepto de subproceso.
- ➢ Implementar la interfaz Runneable para definir al método run().
- ➢ Reforzar el concepto de herencia simple en Java.
- ➢ Entender la importancia del método run() para llevar a cabo la acción del hilo
- ➢ Entender la importancia del método start y el de sleep para la ejecución de hilos.
- ➢ Entender que try llama a métodos que pueden lanzar excepciones y catch, maneja esas excepciones.

Desarrollo

Se creó una clase nueva llamada Ticker que implementó la interfaz Runnable, después se crearon objetos de tipo Thread (Hilo) y String para llevar acabo posteriormente el subproceso de recorrido de una cadena, además se creó un contador y un sleepTime de tipo int.

```
public class Ticker implements Runnable {

    Thread TickerThread;
    String s,sub;
    int sleepTime;
    char c;
```

Se creó el método constructor, con el mismo nombre que la clase Ticket, el cual se encargaría de establecer las variables s y sleep como cadena y tiempo de dormir respectivamente, además de poner al estado del hilo como recién nacido e inicializarlo.

```
public Ticker(String s, int sleep){

    this.s=s;
    sleepTime=sleep;
    TickerThread=new Thread(this);
    TickerThread.start();
}
```

Se creó el método run() el cuál contendría la acción del hilo, ésta consistía en desprender la primera letra de la palabra y concatenarla al final de la que ya se le había quitado de la inicial, esto se iría repitiendo hasta que la palabra fuera completamente recorrida. También se puso el método try que pondría en estado dormido al hilo y así ya no sería elegible para ejecución, y a catch para atrapar las excepciones que surgieran.

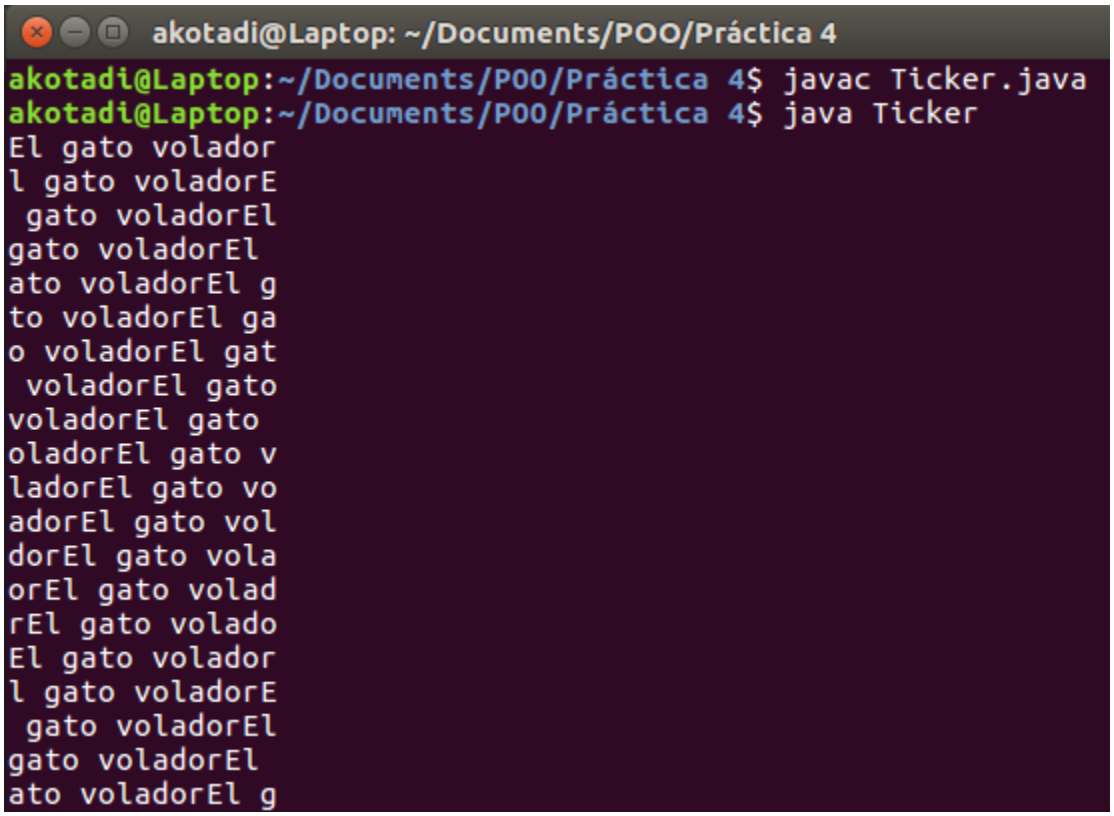
```
public void run(){

    System.out.println(s);
    while(true){
        c = s.charAt(0);
        sub = s.substring(1,s.length());
        s = sub + c;
        System.out.println(sub + c);
        try{
            Thread.sleep(sleepTime);
        } catch (Exception e) {
            return;
        }
    }
}
```

Por último se creó el método main, el cual repetiría la actividad del hilo, previamente establecido, con la frase “El gato volador” y con un tiempo de 50 milisegundos.

```
public static void main(String args[]){
    new Ticker("El gato volador", 50);
}
```

Se ejecutará desde consola con los siguientes comandos:



Conclusiones

El concepto de multitarea o multiprocesamiento es bastante sencillo de entender ya que solo consiste en hacer varias cosas a la vez sin que se vea alterado el resultado final. Al realizar estas actividades, se tomará en cuenta el tiempo de la de mayor tiempo al realizar, para que así se hagan las actividades dentro de ese lapso. Cuando se llama a start() se inicia un segundo hilo, pero el hilo principal continua. El hilo principal continúa hasta que llega al join(), donde espera a que termine el hilo segundo. Por lo que sí aplica el funcionamiento de hilos (al menos siempre está el hilo principal, más lo que se inicie con start()) aunque luego venga otro hilo.