Memoria Practica I Threads

Que

Se desea implementar una carrera de 100m lisos:
Por un lado, tenemos 8 Atletas dispuestos a correr. Cada uno tiene un atributo dorsal. Por otro lado, tenemos una clase principal Carrera. Ésta indica el pistoletazo de salida y el resultado de la carrera.

Todos los Atletas comienzan pero se quedan parados esperando el pistoletazo de salida.

Luego comienzan a correr (tardan entre 9 y 11s).

Al llegar a meta notifican a la carrera su dorsal y terminan.

La Carrera escribe “preparados” y espera 1s, luego escribe “listos” y espera 1s, finalmente escribe “ya!” y notifica a los hilos de los Atletas Cada vez que un atleta le notifica su dorsal, escribe por pantalla lo que ha tardado el atleta con la ayuda de: System.currentTimeMillis() NOTA: podéis hacer uso de una clase adicional, por ejemplo, Salida, donde déis las señales de salida y llegada.

Para que

Este ejercicio ha sido útil para practicar más los hilos en Java y dar por finalizado el temario. Es útil si queremos ver la funcionalidad de los métodos `synchronized`, además de la función `join()`, y una nueva función `System.currentTimeMillis()`. Al principio, pensaba que creaba un nuevo hilo con un contador, pero al ver la documentación, me di cuenta de que es mucho más sencillo de lo que pensaba. Simplemente devuelve la hora actual en milisegundos y para calcular el tiempo que ha tardado, restamos un tiempo por otro. Creo que es una buena manera de hacer temporizador de tiempos cortos, ya que si lo quisiéramos implementar en un programa con un temporizador de varias horas, pienso que podría dar problemas si justo es uno de esos días en los que la hora cambia o el usuario decide cambiar la hora de su sistema.

Pseudocodigo

Atleta

```
Clase Atleta extiende Hilo:
    Entero dorsal
    Objeto Carrera carrera

    Método Atleta(Entero dorsal, Carrera carrera):
        // Constructor de la clase Atleta
        Esto.dorsal = dorsal
        Esto.carrera = carrera

    Método run():
        Sincronizar (carrera):
            Intentar:
                Imprimir que dorsal esta listo
                Espera a que la carrear empiece
            Capturar InterruptedException:
                Interrumpe el hilo

        Long startTime es el timpo actual
        Espera (Entero aleatorio entre 9000 y 11000)
        Long endTime = es el timpo despues de el tiempo aleatorio
        Long tiempo = la resta de los dos tiempos
        Imprime qunato ha tardado en acabar la carrera
        carrera.notificarLlegada(dorsal, tiempo)
```

Carrera:

```
Clase Carrera:
    Entero NUM_ATLETAS
    Arreglo Atleta atletas
    Arreglo Long ganador = {0, 12000}

    Método Carrera(Entero numAtletas):
        // Constructor de la clase Carrera
        Esto.NUM_ATLETAS = numAtletas
        Esto.atletas = nuevo Arreglo Atleta[NUM_ATLETAS]

    Método prepararAtletas():
        inicia cada hilo con un for de numero de corredores

    Método correr():
        Sincronizar (esto):
            Imprimir "Preparados!"
            Espera 1 segundo
            Imprimir "Listos!"
            Espera 1 segundo
            Imprimir "Ya!\n"
            notificarTodos()

        Carrera.dormir(8500)
        Imprimir "Resultados llegada:\n"
        Para cada atleta en atletas:
            Intentar:
                Añade el atleta a el array
            Capturar InterruptedException:
                Interrumpe el hilo

    Método notificarLlegada(Entero dorsal, Long tiempo):
        Si tiempo < ganador[1]:
            ganador[0] = dorsal
            ganador[1] = tiempo

    Método obtenerGanador():
        Devolver ganador[0]

    Método estático dormir(Entero milisegundos):
        Intentar:
            Dormir el hilo (milisegundos)
        Capturar InterruptedException:
            Interrumpe el hilo
```

Practica_03(Principal)

```
Clase Practica_03:
    Método estático main(String[] args):
        Imprimir "Carrera 100M LISOS.\n"
        Carrera carrera = nuevo Carrera(8)
        Imprimir "Corredores a la pista…\n"
        carrera.prepararAtletas()
        Espera 1 segundo
        Imprimir "\nEMPIEZA LA CARRERA\n"
        carrera.correr()
        Imprimir "\nGanador de la carrera: Dorsal " +
carrera.obtenerGanador() + "!"
```

Como

Clase Atleta

Esta clase extiende de `Thread`, lo que significa que cada objeto de esta clase es un hilo. Tiene dos variables: el dorsal (int) y un objeto de la clase `Carrera` para poder llamar a sus métodos. Además, cuenta con un constructor estándar y solo un método, el cual proviene de `Thread`, llamado `run()`. Dentro de este método, hay un fragmento de código que está sincronizado (`synchronized`) para evitar condiciones de carrera. Este fragmento de código simplemente hace que el hilo espere hasta que la carrera notifique que ha comenzado, además de informar por pantalla que el hilo está preparado para correr y gestionar posibles errores del hilo.

Una vez que la carrera ha empezado, utilizando la función `currentTimeMillis()`, guarda la hora actual, duerme el hilo durante un tiempo aleatorio y luego vuelve a guardar la hora después de ese tiempo. Posteriormente, muestra por pantalla el tiempo que ha tardado y notifica a la clase `Carrera` el tiempo que ha tomado.

Clase Carrera

La clase `Carrera` se encarga del control y gestión de los atletas participantes. Tiene una variable que hace referencia al número de atletas, un array de atletas y otro array donde guarda los datos del ganador. El constructor de la carrera solicita el número de corredores y los asigna a la longitud del array.

Esta clase tiene varias funciones: `prepararAtletas()`, `correr()`, `notificarLlegada()`, `obtenerGanador()`, `dormir()`.

Algunas de estas funciones son fáciles de explicar. `dormir()` simplemente es una función que duerme cualquier hilo en ejecución y gestiona posibles errores. `prepararAtletas()` crea objetos de tipo `Atleta` según el número de atletas, los guarda en el array de atletas y llama a su método `run` con `start()`. `obtenerGanador()` devuelve el dorsal del ganador tomando la primera variable de los datos del ganador.

`correr()` tiene una parte sincronizada donde muestra por pantalla la preparación, listos y ya del inicio de una carrera, y notifica a los atletas (hilos) que la carrera ha comenzado con el método `notifyAll()`. Esto debe estar obligatoriamente dentro del bloque sincronizado para evitar condiciones de carrera. Finalmente, espera algo más de 8 segundos (esto es solo para que tarde un poco en imprimir el string `"Resultados llegada"`, es opcional) y llama al método `join()` de cada atleta para que la función no termine hasta que todos los hilos hayan terminado.

`notificarLlegada()` recibe el dorsal del atleta y el tiempo que ha tardado, comprueba si ese tiempo es mejor que el que ya está registrado y, si es así, lo registra. Este método también es `synchronized` para evitar que dos hilos accedan al array de los datos del ganador al mismo tiempo.

Clase Practica_03

Esta clase simplemente crea un objeto de tipo `Carrera`, pide el número de corredores, imprime algunos mensajes informativos por pantalla y llama a los métodos `prepararAtletas()` y `correr()` para ejecutar la carrera. Finalmente, imprime al ganador llamando a la función `obtenerGanador()`.

Conclusión

Como conclusión, creo que esta actividad ha sido interesante y me ha ayudado a mejorar mis habilidades de programación en hilos, gracias al uso de nuevas funciones, así como a la aplicación de aquellas que ya conocía previamente. También descubrí una función que no conocía, la cual devuelve el tiempo. Me pareció curioso y fácil de implementar, siendo una función simple.

Además, me gustaría saber cómo mis compañeros abordaron esta actividad, ya que creo que tiene varias formas de implementarse. Preguntar a mis amigos y obtener explicaciones sobre cómo llevaron a cabo las actividades podría ser una buena manera de aprender también. En general, este tema me resultó de dificultad normal-alta para entender, y es una lástima que justo cuando empezaba a comprenderlo mejor, cambiemos de tema. Sin embargo, espero que el próximo sea igual o más interesante que este.