



Tecnología de la Programación

Presentación de la Práctica 6

(Basado en la práctica de Samir Genaim)

Ana M. González de Miguel (ISIA, UCM)



Índice

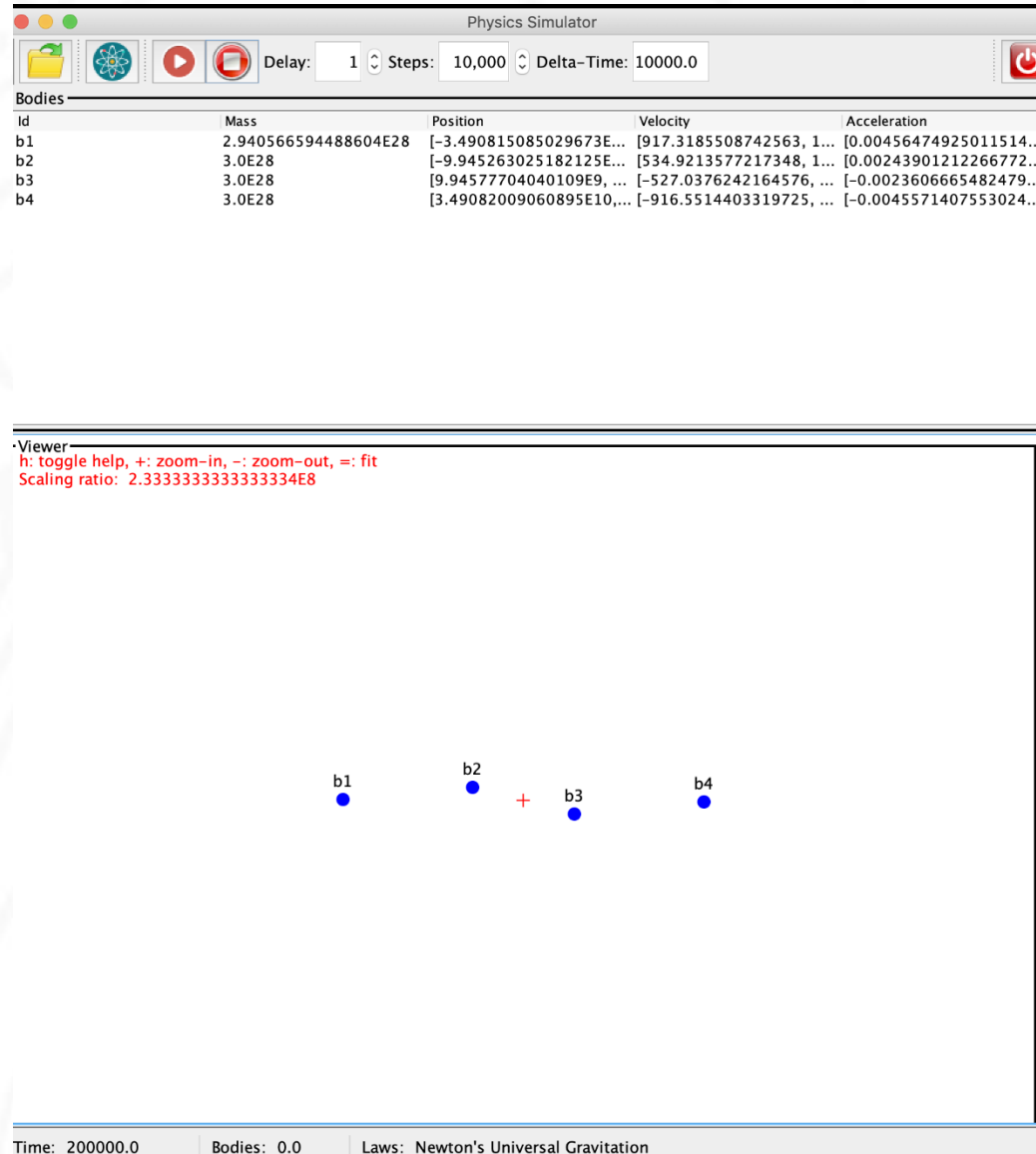
1. Introducción.
2. Solución de la Práctica 5.
3. Usando Java Threads.
4. Opcional.

1. Introducción

- ✓ Los objetivos cubiertos son: **threads** y **Swing**.
- ✓ Fecha de entrega: **6 de Mayo** a las 9:00.
- ✓ Las siguientes **instrucciones** son obligatorias:
 - Lee el enunciado completo de la práctica antes de comenzar a programar.
 - Haz una copia de la práctica 5 antes de empezar a hacer cambios para la práctica 6.
 - Sigue la estructura de paquetes y los nombres de clases que sugerimos en el enunciado.
 - Entrega un fichero **zip** del directorio del proyecto, incluyendo todos los subdirectorios excepto el directorio **bin**. **Otros formatos como 7zip, rar, etc. no serán aceptados.**

2. Solución de la Práctica 5

- ✓ En la práctica 5 hemos descrito dos maneras de implementar la funcionalidad de los botones  y 
 - En la primera, hemos usado la cola de eventos de Swing para realizar la llamada recursiva a *run_sim*. Entre una llamada y otra a *_ctrl.run(1)* Swing puede refrescar la vista y manejar las interacciones con el usuario.
 - En la segunda, hemos sugerido cambiar el método *run_sim* para simplemente llamar a *_ctrl.run(n)*. En este caso, la vista permanece bloqueada mientras el simulador ejecuta sus pasos y sóloamente podemos ver el resultado final.
- ✓ Aunque hemos obtenido un comportamiento razonable con el primer enfoque, si tenemos en cuenta la capacidad de respuesta, podemos mejorarlo usando **programación multihilo** como vamos a hacer en esta práctica.








3. Usando Java Threads

- ✓ Cambia la clase *ControlPanel* para incluir un nuevo *JSpinner Delay* (con valor mínimo 0, valor máximo 1000 y tamaño de paso 1) y su correspondiente *JLabel* anterior (ver la Figura).
- ✓ El valor de este nuevo *JSpinner* representa el retraso entre pasos consecutivos de simulación, dado que ahora la ejecución es más rápida.
- ✓ Cambia el método *run_sim* incluyendo un segundo parámetro *delay* de tipo *long* y haciendo que ejecute algo como el siguiente pseudocódigo:


```
while (n>0 && (the current thread has not been interrupted) ) {  
    // 1. execute the simulator one step, i.e., call method  
    // _ctrl.run(1) and handle exceptions if any  
    // 2. sleep the current thread for 'delay' milliseconds  
    n--;  
}
```


- ✓ Este bucle ejecuta n pasos de simulación y para si el hilo correspondiente ha sido interrumpido.
- ✓ En el paso 1, ejecuta el simulador un único paso y captura cualquier posible excepción que sea lanzada por el simulador/controlador, muéstrala al usuario en un cuadro de diálogo y finaliza el método *run_sim* inmediatamente con un *return* (como en la práctica 5).
- ✓ En el paso 2, usa *Thread.sleep* para dormir el hilo durante *delay* milisegundos.
- ✓ Recuerda que si un hilo se interrumpe mientras duerme, el flag de interrupción no se pone a *true* sino que se lanza una excepción. Por tanto, en ese caso, deberás interrumpir nuevamente el hilo cuando captures la excepción y así salir del bucle (o simplemente salir del método con *return*).

- ✓ A continuación, cambia la funcionalidad de los botones  y  para ejecutar *run_sim* en un nuevo hilo haciendo lo siguiente:
- Añade un nuevo atributo llamado *_thread* del tipo *java.util.Thread* en la clase *ControlPanel*. Hazlo volátil dado que va a ser modificado desde distintos hilos.
 - Cuando se haga click en , desactiva todos los botones a excepción de  y crea un nuevo hilo (asigna esa referencia a *_thread*) que hará lo siguiente: (1) llama a *run_sim* con el número de pasos y el delay especificados en los *JSpinner* correspondientes; (2) habilita todos los botones (al volver de *run_sim*); y (3) pon el campo *_thread* a null.
 - Cuando se haga click en , si hay un hilo ejecutándose (es decir, si *_thread* no es null) entonces interrúmpelo para salir del bucle while y terminar el hilo.

- ✓ La misma funcionalidad puede ser conseguida utilizando el atributo `_stopped` de la práctica 5 en lugar de la interrupción de hilos. En ese caso deberías declarar el campo `_stopped` como volátil.
- ✓ Pero queremos que practiques la interrupción de hilos así que no hagas la solución basada en el campo `_stopped`. Elimina este atributo de la clase *ControlPanel*.
- ✓ Cambia los métodos de observador, en todas las clases de la vista, de manera que cada vez que se modifique un campo o un componente Swing, se realice utilizando *SwingUtilities.invokeLater*. Esto es necesario porque los métodos de observador ahora se ejecutan en un hilo distinto de el EDT de Swing. Lo mismo se debe hacer al mostrarse mensajes de error en el método *run_sim*.

4. Opcional

- ✓ Implementa la funcionalidad descrita en la sección anterior utilizando un **Swing worker** en lugar de crear un nuevo hilo cada vez que se hace click en .