# Simulación de objetos físicos con comportamientos autónomos

## Índice de contenido



## 1. Requisitos software

- Eclipse http://www.eclipse.org/downloads/
- VisualVM incluida en jdk versión 6 o superiores
- Repositorio de proyectos Github con sistema de control de versiones Git donde se monitorice el desarrollo incremental.

## 2. Objetivos específicos

- Conocer una aplicación concurrente y su motivación
- Conocer la concurrencia aplicada en tratamiento de eventos gráficos en Java
- Implementar código concurrente para simular objetos físicos con comportamientos autónomos independientes
- Creación y destrucción de hilos en aplicaciones concurrentes Java
- Definición y comprobación de invariantes/postcondiciones como técnica que garantiza la imposición de diseño de seguridad
- Sincronización de métodos modificadores como técnica que garantiza la imposición de diseño de seguridad

## 3. Enunciado

Se quiere desarrollar un juego de billar donde cada bola de la mesa tiene un comportamiento independiente de las demás. En una primera versión de la aplicación, disponible en https://gist.github.com/clopezno/6b5a192596f0c7e3745f, se ha incluido la siguiente funcionalidad:

- Creación de componentes gráficos de la interfaz de usuario, un tablero (`Board.java`) para visualizar un conjunto de bolas (`Ball.java`) y una ventana que contiene el tablero (`Billards.java`). Inicialmente no se han añadido bolas al tablero (ver Ilustración 1)
- El movimiento de las bolas es en línea recta en función de un ángulo (`Ball.fi`) y una velocidad constante (`Ball.v`)


*Ilustración 1: Interfaz gráfica de la aplicación*

**Se pide:**

- Continuar el desarrollo de esta versión implementando la activación y parada del movimiento independiente de las bolas utilizando hilos de Java. Consultar las etiquetas **//TODO** disponibles en el código fuente para evolucionar de forma incremental a la nueva versión.
- El número de bolas que hay que añadir al tablero debe ser **xx + 3**, donde **xx** es el número del grupo de práctica asignado por el profesor.
- Definir y comprobar las postcondiciones de los métodos modificadores. Se tiene que utilizar asertos de Java para modelar el invariante *"las bolas no pueden salirse fuera del tablero"*

**Nota:** El código propuesto tiene objetivos docentes relacionados con programación concurrente acorde a los conocimientos disponibles por alumnos de sexto semestre del Grado de Ingeniería Informática de la UBU. No tiene pretensiones de pasar a explotación, ya que sería más apropiado usar otras clases del API de la interfaz gráfica de Java que se salen fuera del alcance del objetivo docente buscado.

## 4. Productos a entregar

1. Url del repositorio del proyecto público en Github donde debe existir una revisión/commit por cada una de las etiquetas **//TODO** resueltas.
    - Nombre del proyecto público en Github **2018PCTR_DXX**, donde **D** es **L**(unes), **M**(iércoles) y **O**(online) y **XX** es el número del grupo de práctica asignado por el profesor.
    - Incluir a todos los miembros del grupo de prácticas como participantes del proyecto.
    - Recordad el vídeo_tutorial de la asignatura de Gestión de proyectos de control de versiones.
2. Una distribución ejecutable en un fichero llamado **2018XXbillar.jar**, donde **xx** es el número del grupo de práctica asignado por el profesor.

## 5. Bibliografía


- <<Capítulo 1>> Doug Lea. *Programación concurrente en Java: Principios y patrones de diseño*. 2ª ed. PEARSON EDUCACION, 2000. http://gee.cs.oswego.edu/dl/cpj/.
- Oracle. «Lesson: Concurrency (The Java™ Tutorials > Essential Classes)». 2013. http://docs.oracle.com/javase/tutorial/essential/concurrency/index.html.
- Oracle. «VisualVM». Accedido marzo 13, 2015. http://docs.oracle.com/javase/8/docs/technotes/guides/visualvm/index.html.