Prácticas de Programación Orientada a Objetos
Curso 2006/2007

Entrega 1

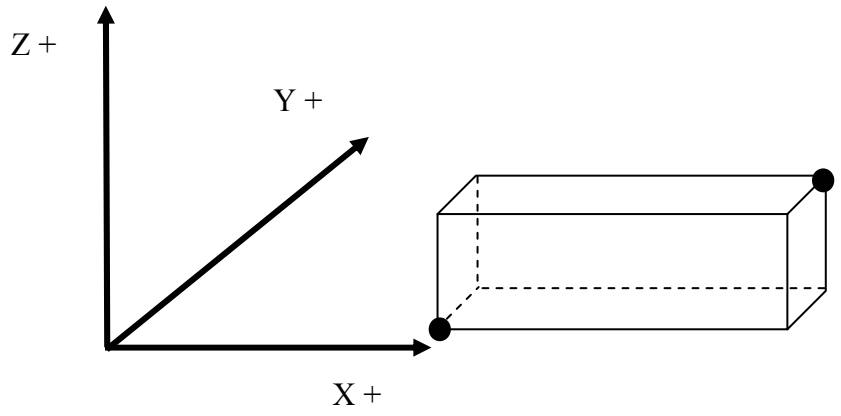
El **plazo de entrega** finaliza a las **dos semanas** de la presentación del ejercicio en clase

Objetivo.

El objetivo de esta práctica es comenzar la implementación de las clases de apoyo que nos permitirán trabajar con los conceptos de geometría del videojuego. La funcionalidad que se indica en este enunciado no está cerrada, es decir, que en posteriores entregas podrá ser necesario extenderla e incluso modificarla.

Introducción.

El juego se desarrolla en un **espacio tridimensional**. Las figuras que se utilizarán en el videojuego serán paralelepípedos (poliedros de 6 caras, iguales y paralelas dos a dos), y cubos, que será implementada en próximas entregas. Con el fin de simplificar la situación de las figuras en el espacio supondremos que las caras del paralelepípedo son paralelas a los planos que definen los ejes de coordenadas, tal como muestra la siguiente figura:



Dadas las restricciones expresadas anteriormente, un paralelepípedo podría caracterizarse por los puntos que definen la esquina *menor* en los tres ejes y la esquina *mayor*. Por otro lado, se establece la restricción de que las figuras en el espacio sólo podrán desplazarse en la dirección de los ejes, definiéndose, por tanto, 6 sentidos de movimiento: NORTE (Y+), SUR (Y-), ESTE (X+), OESTE (X-), ARRIBA (Z+) y ABAJO (Z-), que corresponden a una visión del espacio sobre el plano XY.

Previo.

Crea un **proyecto** de programación Java en **Eclipse** con nombre "practicas-poo". Organiza el código fuente en un paquete con nombre "cubos" que contendrá a su vez el paquete "geometria", donde introduciremos el código de esta entrega.

Trabajo.

1. Implementa el **enumerado `Sentido`**, que represente los sentidos de movimiento de las figuras en el espacio.

2. Programa la **clase `Punto3D`** que caracteriza los puntos en el espacio tridimensional. Los atributos de esta clase son las tres coordenadas del punto. Para cada coordenada define sendos métodos get/set. Asimismo, implementa tres métodos de incremento (`incX`, `incY`, `incZ`) que admitan valores negativos de incremento, es decir, que permitan decrementar el valor, y un método `desplazamiento` para mover el punto en un sentido de movimiento. Finalmente define tres constructores para la clase, un constructor por defecto que establezca todas las coordenadas a cero, uno que tome los valores de las tres coordenadas y un constructor de copia que obtenga los valores de otro punto.

3. Define la **clase `Paralelepipedo`** que está caracterizada por los puntos *inferior* y *superior*, según las restricciones expresadas en esta entrega. La clase ofrecerá dos métodos de consulta para estos puntos, que deberán preservar la integridad de la clase. Además, la clase deberá ofrecer la siguiente funcionalidad:
    a. Un método que devuelva el volumen de la figura (`getVolumen`), que se calcula como el producto de los 3 lados.
    b. Otro método que permita comprobar si un punto está dentro de la figura (`estaDentro`).
    c. Una serie de métodos para determinar si dos figuras se encuentran en `interseccion`, si existe `inclusion` de una figura en otra, si `esContiguo` en alguno de los sentidos de movimiento (comparten la misma cara) o en cualquier cara (debe ofrecerse una implementación sobrecargada de este método).
    d. Seis métodos que devuelvan el mayor y menor valor de coordenada de la figura en cada eje (`mayorX`, `menorX`, etc.)
    e. Un método para agrandar o reducir en un tanto por ciento la figura en alguno de los sentidos de movimiento (`escalar`)
    f. Un método de `desplazamiento` en un sentido.
    g. Y por último, un constructor que tome como parámetros los dos puntos de referencia y otro que tome el punto inferior y el tamaño de los tres lados.

4. Las clases implementadas serán probadas por las **clases de prueba `TestPunto3D`** y **`TestParalelepipedo`**, respectivamente, que se situarán dentro del paquete "cubos.geometria.tests". La implementación de las pruebas es a criterio de cada grupo, aunque se valorará que el código de prueba sea significativo y esté bien documentado. Finalmente, se definirá un lanzador para cada una de las clases, con el fin de poder lanzarlas independientemente.

Entrega.

Todas las clases deberán ser documentadas adecuadamente en **javadoc** y deberá generarse la **documentación** para cada entrega. El **responsable** del grupo (el primer alumno del grupo) deberá dejar la práctica en su zona de contenidos de la asignatura. La falta de entrega en el plazo indicado supondrá un 5 % de penalización sobre la nota final de la práctica.

Criterios de evaluación.

1. Visibilidad de las declaraciones, y métodos de acceso y modificación de los atributos (10 %).

2. Documentación Javadoc *significativa* en todas las declaraciones y generación de la documentación (15 %). Nótese que la semántica de algunas operaciones no se ha definido con precisión en el enunciado, por ejemplo si el método desplazar permite valores negativos. Por tanto, es importante documentar adecuadamente el funcionamiento de los métodos.

3. Interpretación adecuada de la semántica de referencia de Java (15 %).

4. Reutilización de código, especialmente en el caso de los constructores y métodos sobrecargados (10 %).

5. Definición adecuada de las clases de prueba (10 %).

6. Corrección y completitud de la implementación de la funcionalidad (25 %).

7. Convención de nombrado en Java, legibilidad del código, nombres de identificadores significativos y uso de la palabra clave *this* para hacer más legible el código (15 %).