Prácticas de Programación Orientada a Objetos
Curso 2006/2007

Entrega 2

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

La fecha de entrega se indicará en **SUMA** con suficiente antelación.

Objetivo.

El objetivo de esta práctica es la aplicación del concepto de herencia para la definición de nuevas clases en la aplicación. El uso de la herencia nos permite definir unas clases a partir de otras, ofreciendo así un mecanismo de reutilización de código. Asimismo, la herencia nos permite aplicar otras propiedades de la programación orientada a objetos como el polimorfismo o la ligadura dinámica. Sin embargo, la herencia no es sólo un mecanismo de reutilización y debe aplicarse cuando existe la relación "es-un" entre la subclase y la superclase, o se puede aplicar el principio de sustitución. Finalmente, en esta entrega se implementarán dos nuevas clases que formarán parte de la biblioteca de clases de geometría de nuestra aplicación.

Trabajo.

1. Modifica la clase **`Paralelepípedo`** para incluir tres métodos que devuelvan el tamaño de los lados en los ejes X, Y y Z del paralelepípedo.

2. Implementa la clase **`Cubo`** que representa un Paralelepípedo regular, es decir, que tiene todas las caras iguales. Por tanto, un cubo es una especialización de un paralelepípedo. Para la definición de esta clase hay que tener en cuenta que un cubo se caracteriza por el punto inferior de referencia (igual que el paralelepípedo) y el tamaño del lado. Estas propiedades permiten definir algunas operaciones de un modo más eficiente que en el caso del paralelepípedo. Por tanto, en este ejercicio es necesario identificar los atributos de la clase, la definición de los constructores y qué métodos conviene o hay que redefinir debido a las características específicas de esta figura. Por último, nótese que la operación de escalado de los paralelepípedos no es aplicable a los cubos manteniendo la misma semántica. Es decir, si se escala la figura hacia el Este, también implicaría el escalado hacia Arriba y hacia el Norte, para mantener las proporciones regulares de la figura.

3. Define la clase **`CuboSolido`** que define un cubo sólido que tiene masa. Por tanto, esta figura está caracterizada por las propiedades comunes a los cubos, por la masa (que puede variar) y su densidad, que es una propiedad calculada (masa / volumen).

4. Las clases implementadas serán probadas por las **clases de prueba `TestParalelepipedo`**, **`TestCubo`** y **`TestCuboSolido`**, que formarán parte 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 que esté bien documentado. Al igual que en la entrega anterior, se definirá un lanzador para cada una de las clases con el fin de poder lanzarlas independientemente. Finalmente, en esta entrega es importante hacer pruebas que comprueben que se aplica correctamente la ligadura dinámica en las implementaciones de las subclases. Con este propósito podría utilizarse la plantilla de código *systrace* de Eclipse para incluir al comienzo de un método el código que muestra un mensaje por la consola con el nombre de la clase y operación que se están aplicando.

Entrega.

Todas las clases deberán estar 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 (5 %).

2. Identificación de las relaciones de herencia, uso adecuado de la visibilidad *protected*, reutilización de los constructores de las superclases y redefinición de métodos heredados (40 %).

3. Documentación Javadoc *significativa* en todas las declaraciones y generación de la documentación (10 %).

4. Definición adecuada de las clases de prueba (15 %).

5. Corrección y completitud de la implementación de la funcionalidad (20 %).

6. Convención de nombrado en Java y legibilidad del código entendida como nombres de identificadores significativos, uso de la palabra clave *this* para hacer más claro el código, etc. (10 %).