Programación 3

Universidad de Alicante, 2019–2020

Práctica 1

Plazo de entrega: **Hasta el domingo 29 de septiembre de 2019 a las 23:59h**.

Peso relativo de está práctica en la nota de prácticas: 10%

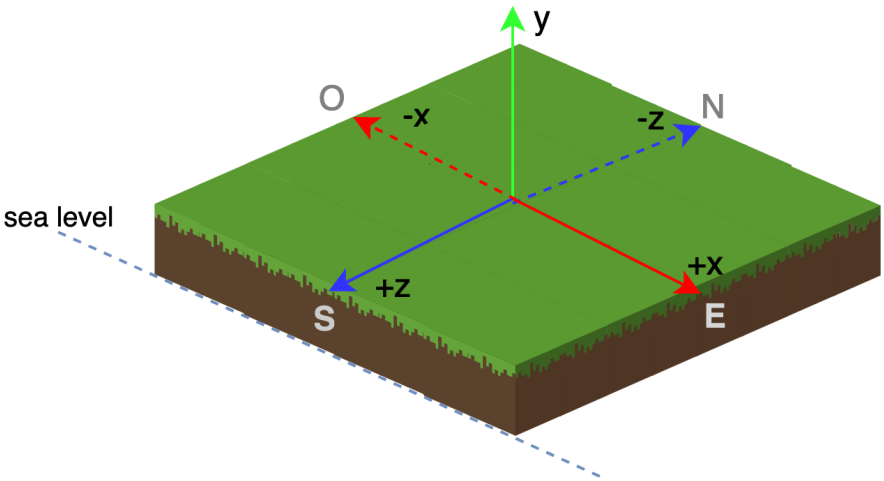
Block World (un mundo de bloques)

Introducción

En esta primera práctica realizaremos la implementación de una clase en Java a partir del código de esa misma clase en C++. El objetivo es conocer las diferencias fundamentales entre Java y C++.

La clase, perteneciente al modelo sobre el que trabajaremos durante todo el curso, se denomina *Location*. Representa una posición en un mundo tridimensional formado por bloques de diferentes materiales. El concepto de mundo de bloques está directamente inspirado en el popular juego **Minecraft**.

Una posición en un mundo *Block World* se representa mediante tres valores reales (double), **x**, **y**, **z**, que representan, respectivamente, la longitud (este-oeste), la elevación y la latitud (sur-norte) respecto al origen de coordenadas. Cada unidad de distancia equivaldría a un metro en el mundo real. Un bloque tiene unas dimensiones de un metro cúbico. Los bloques se posicionan normalmente en coordenadas con valores enteros. Como veremos más adelante, otros objetos y entidades pueden estar en posiciones con coordenadas no enteras. En la figura siguiente puedes ver una representación de este sistema de coordenadas:



- eje x : indica la distancia hacia el este (positiva) o el oeste (negativa) desde el origen, es decir, la longitud.
- eje z : indica la distancia hacia el sur (positiva) o el norte (negativa) desde el origen, es decir, la latitud.
- eje y : indica la altura (de 0 a 255, siendo 63 el nivel del mar) respecto al origen, es decir, la elevación.

Se trata de un sistema de coordenadas que sigue la regla de la mano derecha, donde el pulgar representa el eje x, el índice el eje y, y el dedo medio el eje z. Como puedes observar, los valores *x* y *z* no están limitados, sin embargo el valor y (elevación) sí lo está.

Una posición, además, siempre está asociada a un mundo determinado. En principio sólo existirá un mundo, pero más adelante podremos añadir la posibilidad de definir mundos diferentes. Estos

mundos van a estar representados por la clase `World`. Cada posición (Location) hará referencia al mundo al que pertenece.

La Figura 1 describe gráficamente en UML las clases *Location* y *World*.

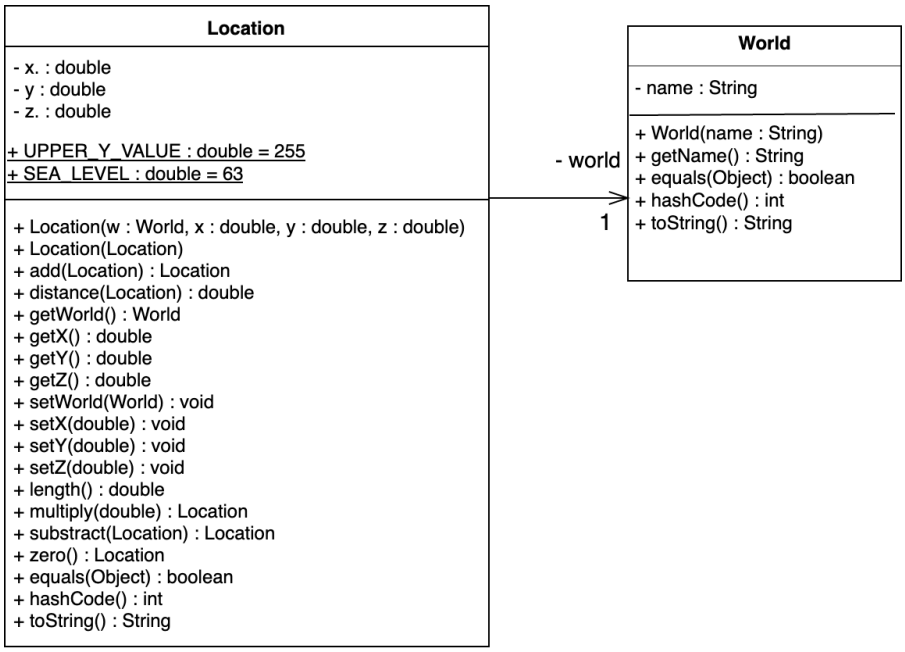


Figura 1. Diagrama de clases UML.

El código en C++ que se acompaña incluye el destructor para la clase. Dado que no existe una traducción exacta de este a Java, lo omitiremos en la implementación Java. El operador == equivale al método 'equals' de la implementación de Java. Las particularidades de la implementación del método equals en Java se comentarán en clase; si estás utilizando Eclipse puedes usar la opción *Source / Generate hashCode() and equals()* del menú, pero asegúrate de que entiendes bien el código generado y sobre todo debes revisar la implementación de *equals()* para que se ajuste a las especificaciones. Lee atentamente el fichero README.txt que acompaña al código para más información sobre la conversión de código C++ a Java.

En Java, la función *main* se debe encapsular dentro de una clase como método público y estático. Esta clase está definida en un fichero llamado *Main1.java*, que realiza las mismas operaciones que se encuentran implementadas en C++ en el fichero *main.cc*. Este será el programa principal de esta práctica. No debes crearlo tú. Sigue leyendo para saber cómo obtenerlo.

Documentación

Debéis incluir en los ficheros fuente todos los comentarios necesarios en formato Javadoc. Estos comentarios deben definirse para:

- **Ficheros**: debe incluir nombre y dni de los autores usando la anotación **@author**
- **Clases:** propósito de la clase: 3 líneas
- **Operaciones**: 1 línea para funciones triviales, y 2 líneas + parámetros de entrada, parámetros de salida y funciones dependientes para operaciones más complejas.
- **Atributos**: propósito de cada uno de ellos: 1 línea

Estructura de paquetes y directorios

En Java, la estructura de paquetes se implementa a nivel de sistema de ficheros mediante directorios. La práctica debe ir organizada en dos paquetes o directorios:

- `model` : contiene los ficheros `Location.java` y `World.java`
- `mains` contiene el fichero `Main1.java`

Puedes usar `mains/Main1.java` para comprobar si tu código al menos compila, ya que realiza llamadas a todos los métodos públicos de la clase Location. El resultado de su ejecución puede diferir un poco de la ejecución de la versíon en C++. En particular, la versión C++ crea 20 objetos de tipo Location, ¡mientras que la versión Java crea 21! Echale un vistazo al código para saber por qué.

Los ficheros fuente deben tener documentación (comentarios) en el formato de Javadoc, pero no se debe entregar los ficheros HTML que genera esta herramienta.

Pruebas unitarias (segunda sesión de prácticas)

El archivo prog3-blockworld-p1-pretest.zip contiene una carpeta `model` con pruebas unitarias. En la segunda sesión de prácticas veremos como usar estas pruebas para evaluar nuestro código.

Proyecto base para Eclipse (segunda sesión de prácticas)

Descarga el archivo prog3-blockworld-Eclipse-project.zip que contiene un proyecto Eclipse preconfigurado para las prácticas de la asignatura. El proyecto contiene la estructura de paquetes

(`model` y `mains`) que vamos a usar dentro de una carpeta de código fuente llamada `src` . Los archivos `.class` serán generados por Eclipse en una carpeta `bin` que no será visible desde el propio Eclipse. El proyecto ya contiene el fichero `mains/Main1.java` . Sigue las instrucciones de tu profesor para importarlo a Eclipse y añadir tu código al proyecto.

Requisitos mínimos para evaluar la práctica

- La práctica debe poder ejecutarse sin errores de compilación.
- Ninguna operación debe emitir ningún tipo de comentario o mensaje por salida estándar, a menos que se indique lo contrario. Evitad también los mensajes por la salida de error.

- Se debe respetar de manera estricta el formato del nombre de **todas las propiedades** (públicas, protegidas y privadas) de las clases, tanto en cuanto a ámbito de visibilidad como en cuanto a tipo y forma de escritura. En particular se debe respetar escrupulosamente la distinción entre atributos de clase y de instancia, así como las mayúsculas y minúsculas en los identificadores.
- La práctica debe estar suficientemente documentada, de manera que el contenido de la documentación que se genere mediante la herramienta *javadoc* sea significativo.

Aclaraciones

- Aunque no se recomienda, se pueden añadir los atributos y métodos privados que se considere oportuno a las clases. No obstante, eso no exime de implementar TODOS los métodos presentes en el enunciado, ni de asegurarse de que funcionan tal y como se espera, incluso si no se utilizan nunca en la implementación de la práctica.
- Cualquier aclaración adicional aparecerá en este enunciado.

Entrega de la práctica

La práctica se entrega en el servidor de prácticas del DLSI.

Debes subir allí un archivo comprimido con tu código fuente (sólo archivos .java). En un terminal, sitúate en el directorio ‘src’ de tu proyecto Eclipse e introduce la orden

```
tar czvf prog3-blockworld-p1.tgz model
```

Sube este fichero `prog3-blockworld-p1.tgz` al servidor de prácticas. Sigue las instrucciones de la página para entrar como usuario y subir tu trabajo.

Evaluación

La corrección de la práctica es automática. Esto significa que se deben respetar estrictamente los formatos de entrada y salida especificados en los enunciados, así como la interfaz pública de las clases, tanto en la signatura de los métodos (nombre del método, número, tipo y orden de los argumentos de entrada y el tipo devuelto) como en el funcionamiento de éstos. Así, por ejemplo, el método `model.Location(World, double,double,double)` debe tener exactamente cuatro argumentos, uno de tipo *World* y otros tres a continuación de tipo *double*.

Tienes más información sobre el sistema de evaluación de prácticas en la ficha de la asignatura.

Además de la corrección automática, se va a utilizar una aplicación detectora de plagios. Se indica a continuación la normativa aplicable de la Escuela Politécnica Superior de la Universidad de Alicante en caso de plagio:

> "Los trabajos teórico/prácticos realizados han de ser originales.
>
> La detección de copia o plagio supondrá la calificación de “0” en la prueba correspondiente.
>
> Se informará la dirección de Departamento y de la EPS sobre esta incidencia.
>
> La reiteración en la conducta en esta u otra asignatura conllevará la notificación al vicerrectorado
>
> correspondiente de las faltas cometidas para que estudien el caso y sancionen según la legislación vigente".