PROGRAMACIÓN ORIENTADA A OBJETOS
(PRACTICA FINAL, CURSO 2007-2008)

1. INTRODUCCIÓN

1.1. Objetivo

El objetivo de este trabajo es utilizar el entorno de simulación de batallas de robots Robocode para el aprendizaje y práctica del lenguaje Java.

1.2. ¿Qué es Robocode?

Robocode es un simulador de batallas robóticas desarrollado en Java. El objetivo del juego consiste en desarrollar el código de un robot basándonos en un API (Application Program Interface) que provee el propio entorno, situar dicho robot en el campo de batalla y dejarlo luchar con otros robots hasta que sólo quede uno.

1.3. ¿Por qué Robocode?

Robocode está especialmente recomendado para la docencia de Java por varios motivos, entre los que podemos citar:

- Fue diseñado con objetivos educativos, haciendo la facilidad de uso una de sus metas.
- El aspecto competitivo del juego motiva a tratar de mejorar el código desarrollado.
- El entorno tiene gráficos atractivos y está integrado con un editor y un compilador Java.

1.4. ¿Qué podemos aprender con Robocode?

Entre los aspectos básicos que podemos aprender al programar un robot de Robocode podemos citar los siguientes:

- Leer la documentación de un API, bien a través de los propios documentos Javadoc o a través de tutoriales o ficheros de ayuda.
- Usar un API y extenderlo con nuevas clases que se integren en el mismo
- Conocer y usar conceptos básicos de la orientación a objetos como: herencia, polimorfismo, ligadura dinámica, sobrescritura, etc.

- Conocer y usar aspectos avanzados de la programación Java como: programación multihilo, gestión de eventos, clases internas, etc.

1.5. ¿Cómo obtener e instalar Robocode?

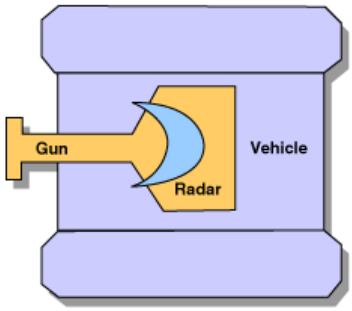
La página principal de Robocode es: http://robocode.sourceforge.net/ desde ahí podremos bajarnos el código de Robocode así como links a distintas páginas de tutoriales o ayuda. Para instalar Robocode en nuestro sistema iremos a la opción de download y nos bajaremos el fichero robocode-setup-x.x.x.jar, que se trata de un instalable Java del programa (las "x" marcan el número de versión). Podremos ejecutar el instalable haciendo doble click sobre el fichero jar siempre y cuando tengamos un entorno de ejecución Java instalado (JRE).

2. Aspectos básicos de Robocode

A continuación explicaremos algunos detalles básicos de Robocode. Esta explicación sólo pretende ser un simple resumen y se emplaza al alumno a consultar la documentación adicional que puede encontrar en la Facultad Virtual (FV) o en Internet.

2.1. Anatomía de un robot

Un robot de Robocode es en realidad un tanque compuesto de un vehículo, sobre el cual se monta un cañón. Sobre dicho cañón tenemos puesto un radar que nos servirá para detectar a los demás robots en el campo de batalla (ver figura).



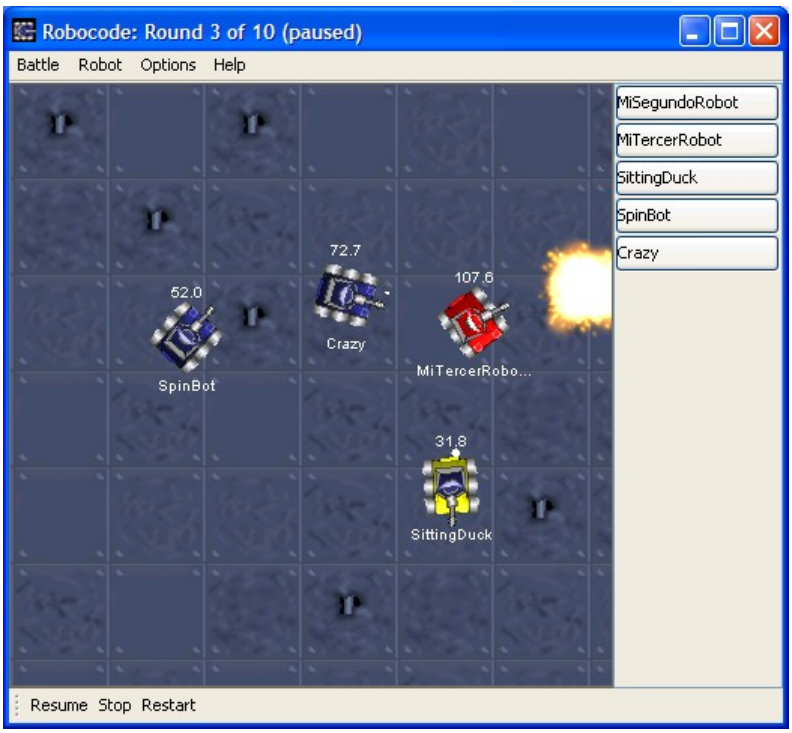
Cada uno de estos elementos puede girar de forma independiente, aunque inicialmente se encuentran todos alineados tal y como muestra la figura. Además los robots pueden moverse hacia adelante o hacia atrás.

Cada robot dispone de una cierta cantidad de energía al iniciarse la batalla. Cuando el robot gasta toda la energía es destruido. La energía se gasta cuando somos alcanzados por los proyectiles enemigos pero también cuando disparamos nuestras propias balas. Si nuestra energía llega a cero debido a nuestros propios disparos el robot queda *disabled*, lo que quiere decir que no puede hacer ninguna acción pero no está destruido. Puede salir de este estado si alguna bala lanzada previamente alcanza a otro robot.

A la hora de disparar disponemos de balas que podemos lanzar con distintos niveles de energía. Cuanto más nivel de energía tiene la bala más daños provoca. Pero hay que usar la munición con precaución ya que el hecho de disparar hace disminuir nuestra propia energía. Por otro lado, dañar o destruir robots enemigos hace que nuestra propia energía aumente.

2.2. El campo de batalla

El campo de batalla es un espacio rectangular en el cual se disponen los robots para la batalla. No existen obstáculos dentro del campo de batalla pero habrá que procurar no chocar contra los bordes del mismo ya que se nos restaría energía. Al comienzo de la partida los robots se sitúan aleatoriamente dentro del campo de batalla (ver figura).



2.3. Obteniendo información del entorno

Un robot puede obtener valiosa información del entorno que le indica cómo actuar. Esta información puede obtenerse de dos formas:

- De métodos llamados por el propio robot. Como `getX` o `getY` que devuelven la posición del robot, o `getHeading`, que nos devuelve en grados la dirección adonde apunta el frente del vehículo.

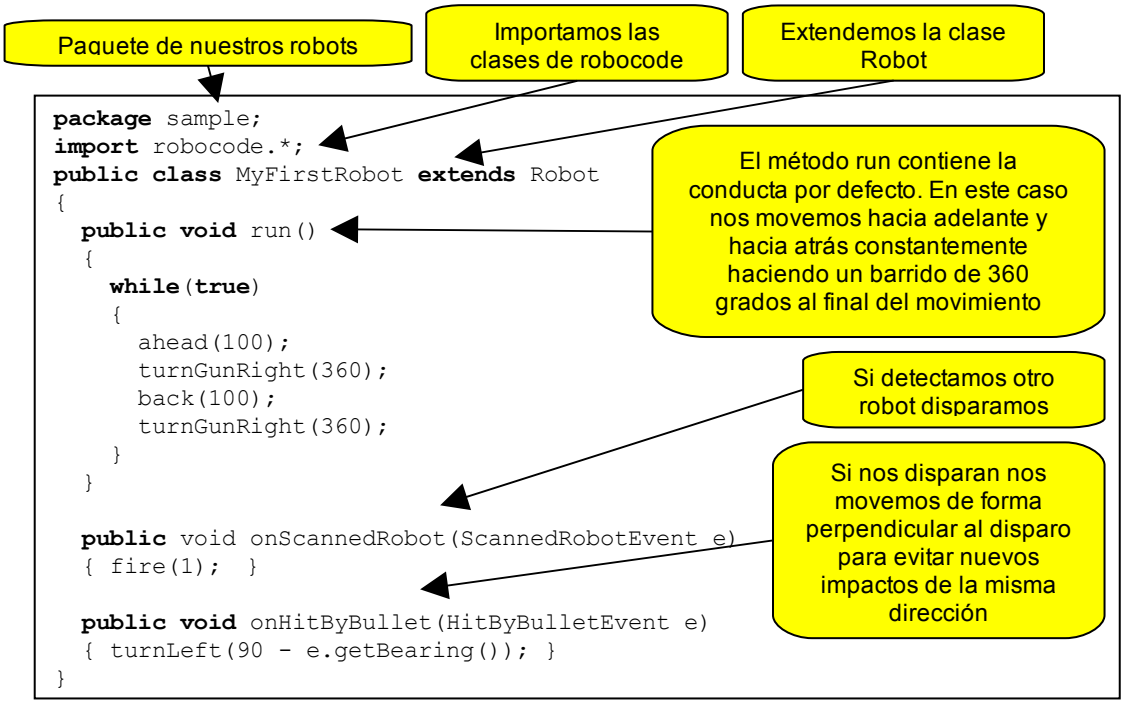
- De eventos que suceden durante la batalla. Los métodos que gestionan dichos eventos están definidos en la clase `Robot`, aunque sin incluir ninguna implementación concreta. Dichos métodos habrá que sobreescribirlos convenientemente en la clase de nuestro robot (que deberá ser una subclase de `Robot`). Entre los más importantes gestores de eventos que podemos sobreescribir tenemos `onScannedRobot` (que se activa cuando se detecta un robot enemigo), `onHitByBullet` (que se activa cuando recibimos el impacto de una bala), `onHitRobot` (que se activa cuando chocamos con otro robot) o `onHitWall` (que se activa cuando chocamos con los bordes del campo de batalla.

3. ¿Cómo programar un robot?

3.1. Código básico

A continuación se muestra el código básico de un robot que propone el propio Robocode. Lo primero que hay que hacer es importar las clases del paquete `robocode` y posteriormente heredar de la clase `Robot`. De todas formas para la práctica final se recomienda el empleo de la subclase `AdvancedRobot`, más compleja de usar pero que permite desarrollar robots más elaborados.

El método principal a sobreescribir es `run`, que marcará el funcionamiento por defecto del robot cuando no se active ningún evento. Este robot de prueba sobreescribe dos métodos de gestión de eventos: `onScannedRobot`, que se activa cuando se detecta otro robot y `onHitByBullet`, que se activa cuando somos alcanzados por un proyectil de un tanque enemigo.



3.2. Usando NetBeans

Robocode incluye un editor integrado, pero éste se limita a resaltar la sintaxis e incluir una opción de compilación. Podemos crear nuestros robots en cualquier IDE Java como podría ser NetBeans. El único requisito imprescindible es que tenemos que tener acceso a la librería de Robocode, por lo tanto después de crear el proyecto NetBeans es necesario añadir dicha librería. Una forma sencilla de hacerlo es seleccionar la vista Projects, pulsar con el botón derecho sobre la **carpeta Libraries, seleccionar la opción Add JAR/Folder y seleccionar el fichero robocode.jar** que se encuentra en el directorio libs de la instalación de Robocode.

Para que el entorno de robocode tenga constancia de que el robot existe habría que meter su fichero class en la carpeta ...\robocode\robots. Pero esto no siempre es posible (por problemas de permisos) y no dejaría de ser un engorro. Por ello, el entorno de robocode ha incluido una opción para tener en cuenta directorios de usuario. **Hay que ir a Options → Preferences → Development Options e incluir el directorio build\classes de nuestro proyecto NetBeans en donde se sitúan nuestros ficheros class**.

4. ¿QUÉ HAY QUE ENTREGAR?

4.1. Fecha límite de entrega

La **fecha límite de entrega de la práctica es el 1 de Febrero de 2008**. Es necesario entregar el código junto a una memoria. A continuación detallaremos sus características.

4.2. Memoria

Para la elaboración de la memoria se recomienda seguir las siguientes pautas: encanutillado en espiral, tipo de letra Times New Roman tamaño 12 para los textos y proporcional (Courier New) y de tamaño 8 para el código, espaciado simple e impresión a doble cara. La memoria se entregará a los profesores de prácticas en las horas indicas como tutorías con la misma fecha límite que para la entrega del código (el último día también podrá dejarse en el casillero de los profesores).

El contenido de la memoria deberá ser el siguiente: en la primera página se deberá incluir **el código del grupo de prácticas y los nombres, apellidos y login de los miembros del grupo**. Posteriormente se incluirá un análisis del robot desarrollado, los diagramas de clase y de secuencia que explican su diseño y, finalmente, el código fuente . A continuación explicaremos lo que se deberá incluir en cada apartado.

4.2.1. Parte 1: Análisis del robot desarrollado

El análisis del robot desarrollado deberá incluir los siguientes apartados:

- Descripción general de las distintas estrategias seguidas por el robot: estrategia principal del robot (esconderse en una esquina, perseguir enemigos débiles, etc.), como apuntamos, como escapamos ante un impacto recibido, etc.

- Descripción del diseño del robot explicando los patrones de diseño utilizados: Aunque generalmente es muy sencillo hacer que un robot resida en una única clase, en esta práctica se pretende fomentar el uso de patrones de diseño en el desarrollo del robot. Por ejemplo, usar el patrón estrategia para cambiar dinámicamente la estrategia del robot, el patrón estado para adecuar el comportamiento del robot a su estado interno, etc. Se buscará que el diseño sea flexible y fácilmente ampliable. También se valorará el uso e identificación de principios de diseño.

Es importante señalar que, debido a la popularidad de Robocode, existe mucho código disponible en Internet sobre robots ya realizados. **Se considerará plagio incorporar como propio un robot sacado (enteramente o que su mayor parte) de código obtenido de otras fuentes como Internet**.

4.2.2. Parte 2: Gráficos UML

La segunda parte consistirá en representar de forma gráfica y a través de diagramas UML el diseño del robot. Los diagramas deberán ir comentados, es decir, habrá qué escribir antes de cada diagrama un breve comentario en el que se explica qué se va a mostrar y los puntos más destacados del mismo.

- Diagrama de clases: Habrá que mostrar un diagrama de clases que muestre todos los detalles de las clases propias, incluyendo los detalles de los métodos, los atributos, las asociaciones, las relaciones de dependencia, etc. En la relaciones de asociación habrá que incluir adornos como nombres de rol, multiplicidades, navegabilidad, visibilidad, etc.

- Diagramas de secuencia: Para las operaciones más importantes del robot deben desarrollarse diagramas de secuencia que permitan ver su comportamiento dinámico. Los diagramas de secuencia deben ser una fiel representación de los algoritmos escritos en Java. Si por razones de legibilidad se realizan simplificaciones deberían ser indicadas en el diagrama con la correspondiente nota. Se hará especial hincapié en la utilización de las novedades que la versión 2.0 de UML introdujo en este tipo de diagramas (por ejemplo, el uso de fragmentos).

Aunque se pueda utilizar la ingeniería inversa para desarrollar los diagramas de UML se tendrá en cuenta en la nota el hecho de que se haya cuidado el corregir los errores propios que cometen las herramientas de ingeniería inversa, así como incluir detalles que estas herramientas normalmente obvian (algunas relaciones de asociación, relaciones de dependencia, etc.). También se valorará que los diagramas entregados sean fácilmente legibles y que estén organizados (que no se crucen líneas, que los adornos estén correctamente colocados, etc.)

4.2.3. Parte 3: Código fuente

El código fuente se entregará impreso en un formato que resalte la sintaxis de Java (palabras clave, comentarios, etc.). NetBeans puede imprimir pero suele tener un tamaño de letra excesivo. Se puede usar mejor la opción "File→Print to HTML", cargar el html resultante en un editor para reducir las fuentes e imprimirlo posteriormente.

Si el código comprende varias clases deben ordenarse alfabéticamente para que sea fácil localizarlas y el código de una clase debe coincidir con el inicio de página (es recomendable, por motivos de compilación también, que cada fichero .java contenga sólo una clase).

Las clases creadas deberán ir acompañadas de comentarios Javadoc y será necesario generar dicha documentación que se entregará en el directorio de prácticas (no es necesario imprimirla)

4.3. Código desarrollado

El código desarrollado deberá depositarse en el **subdirectorio P3** dentro del directorio de entrega de prácticas. La entrega se realizará siguiendo el procedimiento del CECAFI (cualquier duda o problema con la entrega comentárselas a ellos). **La entrega se realizará en el directorio del alumno marcado como portavoz del grupo**. Si pasada la fecha límite la práctica no se encuentra en ningún directorio del grupo de prácticas se considerará no entregada.

Las convenciones a utilizar en el código serán las siguientes:

- Hay que crear un proyecto NetBeans cuyo nombre sea "g + nombre del grupo". Por ejemplo, el grupo 2.3 creará un proyecto denominado "g23".

- Dentro de ese proyecto se creará un paquete "poo" (en minúsculas).

- Dentro del paquete poo se pondrá la clase que herede de Robot (o AdvancedRobot) y se denominará "R + nombre del grupo". Por ejemplo, el grupo 2.3 creará una clase denominada "R23". Es importante que en el paquete "poo" **únicamente pongáis esa clase**.

- El resto de clases de vuestro robot irán en un subpaquete de poo denominado como el grupo. Por ejemplo, el resto de clases del grupo 2.3 irán en el paquete "poo.g23" eso evitará conflictos de nombre con las clases de otros grupos.

En la facultad virtual se os ha dejado un proyecto de ejemplo que sigue las pautas antes mencionadas. En el directorio de prácticas debéis entregar el proyecto NetBeans con los fuentes (directorio scr) y los ficheros class compilados (directorio build/classes). Es importante que no os olvidéis de entregar los ficheros compilados y no comprimáis el proyecto entregado en un fichero ZIP o similar.

4.4. Herramientas disponibles

Se os recuerda que en los laboratorios de prácticas existe el software Magic Draw que os permite realizar diseños UML y que incluye capacidades para hacer ingeniería inversa. También la versión 6 de NetBeans tiene capacidades para desarrollar diagramas de UML directamente o bien mediante ingeniería inversa.

5. Valoración de la práctica

La valoración de la práctica constará de los siguientes apartados:

- **Calidad general de la documentación (1 punto)**: Presentación, si incluye todo lo pedido, si es legible y clara, etc.
- **Calidad del diseño del robot (3 puntos)**: Explicación del diseño realizado, complejidad del mismo, utilización de patrones de diseño, identificación de principios de diseño utilizados, etc.
- **Calidad de los diagramas UML y sus explicaciones (3 puntos)**: Explicación previa de los diagramas, calidad de los mismos, completitud, que no incluyan errores, que incluyan detalles que no utilizan las herramientas de ingeniería inversa, etc.
- **Capacidad de combate del robot (3 puntos)**: Se pondrá al robot a luchar contra los robots de los demás grupos. La nota irá en proporción a la posición final que el robot ocupe en la tabla.

La capacidad de combate se establece de la siguiente manera. Se hará una batalla inicial en un campo de 1000x800 con todos los robots disponibles. Esta batalla establecerá una clasificación inicial de los robots que nos permitirá formar **grupos de entre 8 y 10 robots. Cada grupo combatirá en un campo de 800x600 y la posición dentro de estos grupos significará la nota final en el apartado de combate (podrá ser 10, 7,5, 5 o 2,5)**. Con los robots que hayan obtenido la máxima nota en su grupo se hará una batalla final para decidir cuáles son los **tres mejores robots, que conllevará una bonificación en la nota final de la asignatura (sólo en la convocatoria ordinaria)**.

6. Aspectos inportantes a tener en cuenta

Los grupos de prácticas estarán formados por **tres o cuatro personas**. Cada alumno del grupo debe tener un conocimiento completo de la misma.

La no entrega de la práctica, la entrega de una práctica que no cumpla unos **requisitos mínimos (nota menor que 4)** o la entrega de una práctica manifiestamente copiada supondrá el suspenso en la asignatura. **En las prácticas copiadas tanto el original como la copia recibirán un suspenso**. También se considerará copia la entrega de un código que, en su mayor parte, haya sido obtenido directamente de Internet.

Después del examen teórico podría tener lugar, sólo si fuera necesario, un acto de defensa de la práctica al cual podrá ser llamado cualquier alumno según la lista que se publicará con las notas del examen teórico. La no asistencia o la constatación de que el conocimiento de la práctica es insuficiente implicará el suspenso de dicha práctica y, por lo tanto, de la asignatura.

En las **convocatorias de Septiembre y Diciembre** de esta asignatura la práctica a entregar será la misma. Por ello, **se conserva la nota de la práctica** obtenida en la convocatoria ordinaria, a no ser que el alumno quiera incluirle mejoras en cuyo caso deberá volverla a entregar (incluyendo la documentación). Las fechas límites de entrega en las convocatorias de Septiembre y Diciembre coincidirán con la fecha del examen teórico.