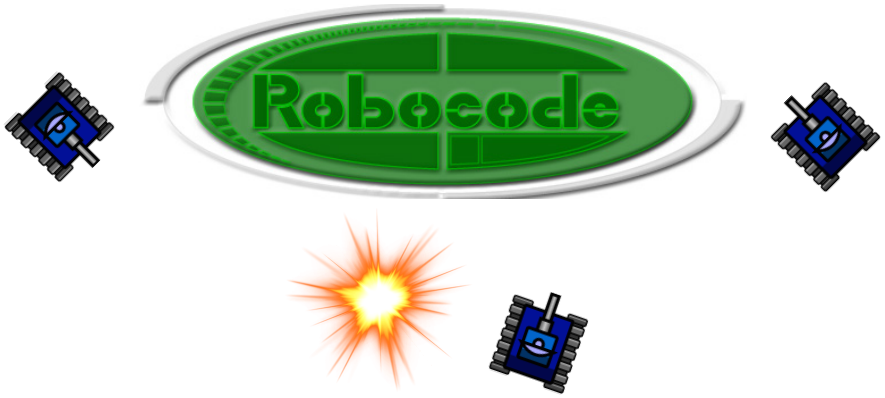
TALLER "Competencia de Bots"



Tareas:

- Explicación del problema de la “Competencia de Bots”
- Desarrollo de la solución
- Competencia de los bots desarrollados

Explicación de Robocode y objetivos del juego.

RoboCode es un simulador de combates entre tanques robóticos o robots desarrollado por IBM Alphaworks. El robot debe recorrer el entorno para evitar ser alcanzado por los disparos de sus oponentes y tratando de no chocar contra las paredes. Para ganar, el robot debe localizar a sus adversarios y dispararles hasta que todos ellos sean destruidos.

Los robots comienzan la simulación con una determinada cantidad de energía, que se puede ir perdiendo por:

- Recibir un disparo de un robot enemigo.
- Chocar contra la pared o un robot enemigo.
- Por disparar balas.

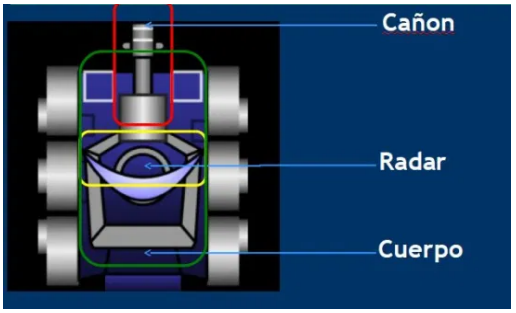
Sólo se puede ganar energía alcanzando con un disparo a un adversario. Si un robot se queda sin energía debido a que realizó muchos disparos sin éxito, entonces queda inhabilitado. Si una bala impacta contra alguno de sus enemigos, la cantidad de energía del robot se incrementa. Pero si un robot se queda sin energía debido a que fue alcanzado por una bala o por chocar contra la pared, entonces este se destruye.

Un robot puede rotar su cuerpo, su arma, o el radar. Todas las rotaciones llevan su tiempo. Además, los robots pueden moverse hacia delante o hacia atrás, a una velocidad constante.

Un robot puede disparar su arma con distintas potencias, a mayor potencia se consumirá más energía, pero hará más daño al impactar al enemigo.

Todos los robots están equipados con un único sensor, el radar, el cual es la única manera que tiene el robot de obtener información sobre sus adversarios. El radar devuelve la posición del enemigo, su orientación, y su energía. Así mismo, el robot también es consciente de: su posición, su orientación, su energía, la orientación de su arma y su radar.

Los robots de **RoboCode** durante la simulación pueden reaccionar ante determinados eventos, como son: detectar un adversario, ser alcanzado por una bala, chocar con un adversario o chocar con una pared.



PARTE I

La actividad de esta parte del taller consiste en modificar el *proyecto* **RobocodeLabo** para que soporte estrategias de guerra *enchufables,* usando **interfaces**. Esta modificación le aportará al proyecto características de extensibilidad, modularidad, desacoplamiento y buen diseño.

El proyecto **RobocodeLabo** provee una estrategia de guerra básica implementada en la clase **LaboRobot**.

Concretamente, hay que:

- **Desarrollar una estrategia propia, basándose en el patrón de diseño Strategy, haciendo uso de herencia, polimorfismo , clases abstractas e interfaces, según lo considere adecuado. Para esto, modifique la clase LaboRobot, la cual hereda las características y el comportamiento de un *JuniorRobot***
  *(*https://robocode.sourceforge.io/docs/robocode/robocode/JuniorRobot.html)

Algunos métodos de JuniorRobot

Método	Descripción
ahead(int distance)	Mueve el robot hacia adelante en píxeles.
back(int distance)	Mueve el robot hacia atrás en píxeles.
bearGunTo(int angle)	Gira el arma al ángulo especificado (en grados) relativo al cuerpo del robot.

doNothing(int turns)	Saltea el número especificado de turnos.
fire(double power)	Dispara una bala con el poder de bala espacificado,entre 0.1 y 3 (3 es el máximo).
setColors(int bodyColor, int gunColor,int radarColor, int bulletColor,int scanArcColor)	Establece los colores del robot.
turnAheadLeft(int distance, int degrees)	Mueve el robot hacia adelante en píxeles y gira el robot a la izquierda en grados al mismo tiempo.
turnAheadRight(int distance, int degrees)	Mueve el robot hacia adelante en píxeles y gira el robot a la derecha en grados al mismo tiempo.
turnBackLeft(int distance, int degrees)	Mueve el robot hacia atrás en píxeles y gira el robot a la izquierda en grados al mismo tiempo.
turnBackRight(int distance, int degrees)	Mueve el robot hacia atrás en píxeles y gira el robot a la derecha en grados al mismo tiempo.
turnGunLeft(int degrees)	Gira el arma a la izquierda en grados.
turnGunRight(int degrees)	Gira el arma a la derecha en grados.
turnGunTo(int angle)	Gira el arma al ángulo especificado (en grados).
turnLeft(int degrees)	Gira el robot a la izquierda en grados.
turnRight(int degrees)	Gira el robot a la derecha en grados.
turnTo(int angle)	Gira el robot al ángulo especificado (en grados).

Algunos campos de JuniorRobot

Campo	Descripción
int energy	Energía actual del robot, donde 100 es lleno de energía y 0 sin energía (muerto).
int fieldHeight	Contiene la altura del campo de batalla.
int fieldWidth	Contiene el ancho del campo de batalla.
int gunBearing	Ángulo actual del cañón respecto a su cuerpo (en grados).
int gunHeading	Ángulo actual del cañón del robot (en grados).
boolean gunReady	Flag que especifica si el arma está lista para disparar.
int heading	Ángulo del rumbo actual del robot (en grados).
int hitByBulletAngle	Último ángulo desde el cual el robot fue golpeado por una bala (en grados).
int hitByBulletBearing	Último ángulo desde el cual el robot fue golpeado por una bala (en grados) en comparación con su cuerpo.
int hitRobotAngle	Último ángulo desde el cual el robot ha golpeado a otro robot (en grados).
int hitRobotBearing	Último ángulo desde el cual el robot ha golpeado a otro robot (en grados) en comparación con su cuerpo.
int hitWallAngle	Último ángulo donde este robot ha golpeado una pared (en grados).
int hitWallBearing	Último ángulo donde este robot ha golpeado una pared (en grados) en comparación con su cuerpo.
int others	Número actual de otros robots en el campo de batalla.

int robotX	Ubicación horizontal actual del robot (en píxeles).
int robotY	Ubicación vertical actual del robot (en píxeles)
int scannedAngle	Ángulo actual al robot más cercano escaneado (en grados).
int scannedBearing	Ángulo actual al robot más cercano escaneado (en grados) en comparación con su cuerpo.
int scannedDistance	Distancia actual al robot más cercano escaneado (en píxeles).
int scannedEnergy	Energía actual del robot más cercano escaneado.
int scannedHeading	Direccion actual del robot más cercano escaneado (en grados).
int scannedVelocity	Velocidad actual del robot más cercano escaneado.

Tabla de puntuaciones al terminar las batallas

Results for 10 rounds

Rank	Robot Name	Total Score	Survival	Surv Bonus	Bullet Dmg	Bullet Bonus	Ram Dmg * 2	Ram Bonus	1sts	2nds	3rds
1st	sample.Tracker	2889	800	160	1616	266	47	0	8	0	2
2nd	sample.RamFire	1990	350	0	1230	22	306	83	0	7	3
3rd	sample.MyFirstRo...	763	350	40	343	9	22	0	2	3	5

- **Total Score:** La suma de todas las puntuaciones
- **Survival Score:** Cada robot vivo, gana 50 puntos por cada robot que muere
- **Last Survival Bonus:** El último robot vivo, gana 10 puntos por cada robot muerto antes que él
- **Bullet Damage:** 1 punto por cada punto de daño realizado a los enemigos
- **Bullet Damage Bonus:** Cuando un robot mata a un enemigo gana un 20% de todo el daño que le ha hecho.
- **Ram Damage:** 2 puntos por cada punto de daño realizado con impactos (por colisión)
- **Ram Damage Bonus:** Cuando un robot mata a un enemigo por impacto, gana un 30% de todo el daño que le ha causado

El proyecto **RobocodeLabo** está disponible para su descarga del sitio de la cátedra: https://catedras.info.unlp.edu.ar.

Para desarrollar el taller, realice las siguientes acciones:

1. Importe el proyecto **RobocodeLabo** en Eclipse.
2. Para probar **LaboRobot:**
    a. Ejecutar el script **robocode.sh** (linux) o **robocode.bat** (windows), ubicado en la raíz del proyecto. Se abrirá una aplicación con un menú superior.
    b. Ir a la opción **Battle > New** y seleccione 2 robots para que se enfrenten en el campo de batalla. Los robots disponibles son aquellos cuyas compilaciones se encuentran en la carpeta "robots" del proyecto. Es posible agregar el directorio donde se compilará su robot a través de **Options > Preferences > Development Options**.
3. Realice las modificaciones necesarias y visualice la estrategia en el campo de batalla.
4. **Recuerde modificar el paquete y nombre de su robot**.