Laboratorio - Introducción Robocode

José A. Montenegro

Dpto. Lenguajes y Ciencias de la Computación
ETSI Informática. Universidad de Málaga
jmmontes@uma.es

27 de febrero de 2021

Introducción

Instalación

Batallas

Conceptos Básicos

Programación

Detalles API Robocode

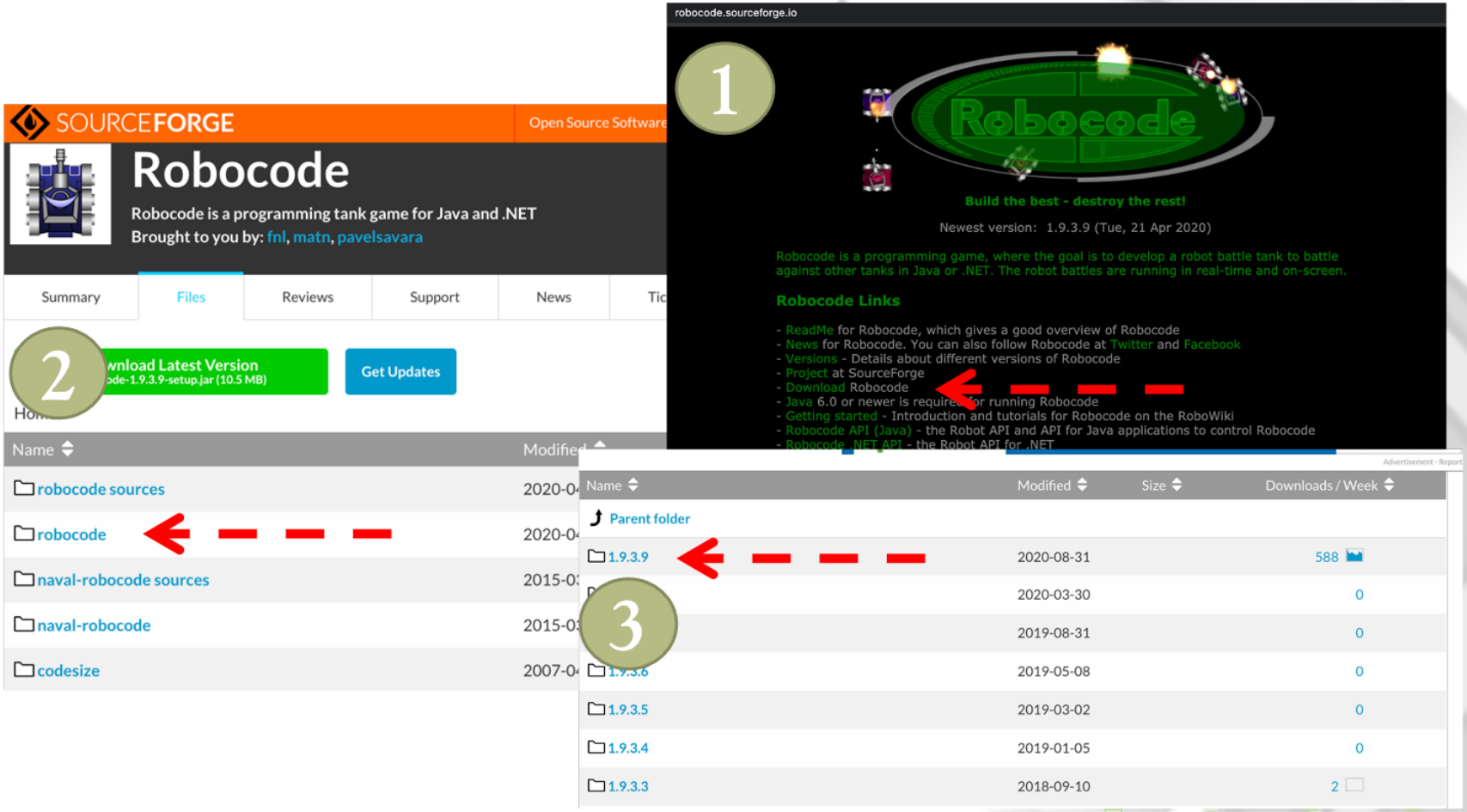
Entrega

¿Qué es Robocode?

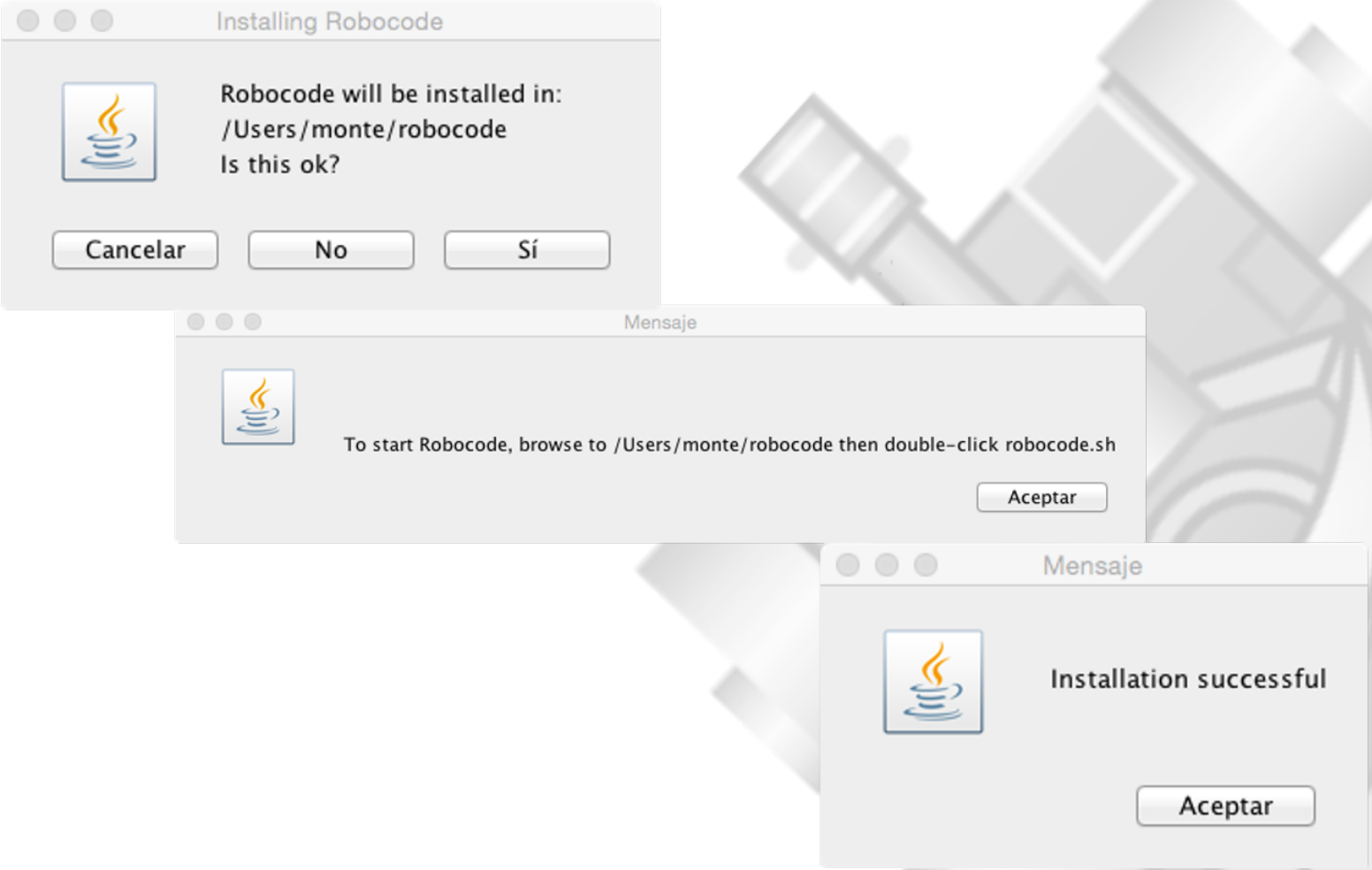
- Robocode is a programming game where the goal is to code a robot battle tank to compete against other robots in a battle arena.
- So the name Robocode is a short for "Robot code".
- The player must write the AI of the robot telling it how to behave and react on events occurring in the battle arena.
- **The motto of Robocode is: Build the best, destroy the rest!**

Descargar Robocode

http://robocode.sourceforge.net/



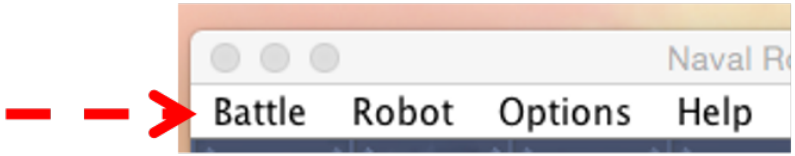
Instalación

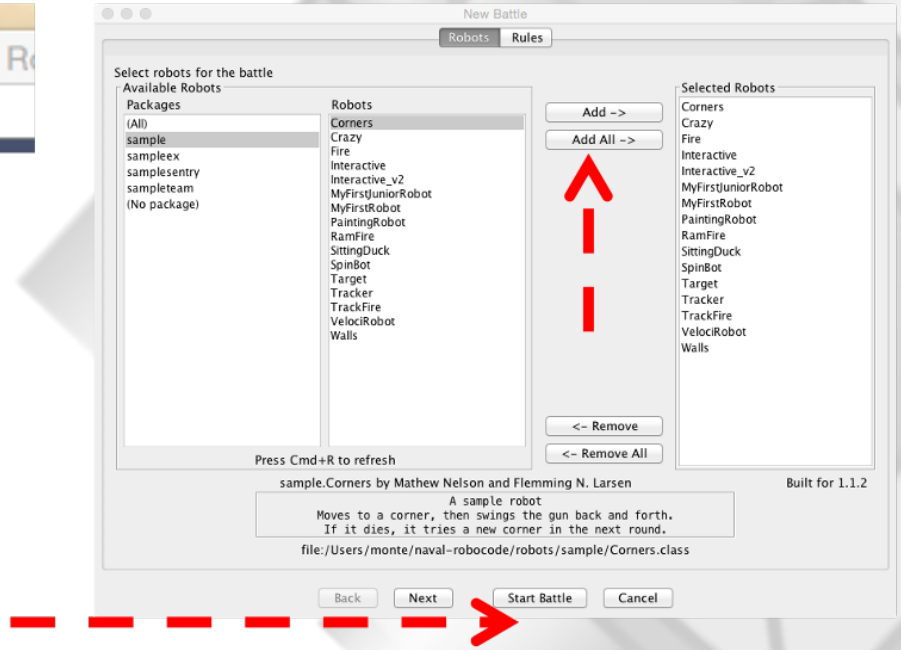


Primera Ejecución

Batallas

1. Seleccionar Battle New
2. Selecciona Robots Disponible
3. Por ejemplo: Add All
4. Next
5. Start Battle





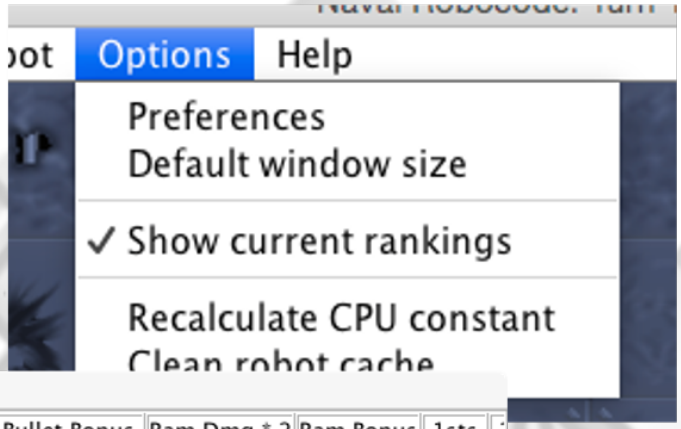
Batallas

Estado Robots

Info Batalla

Debug / Velocidad Batalla

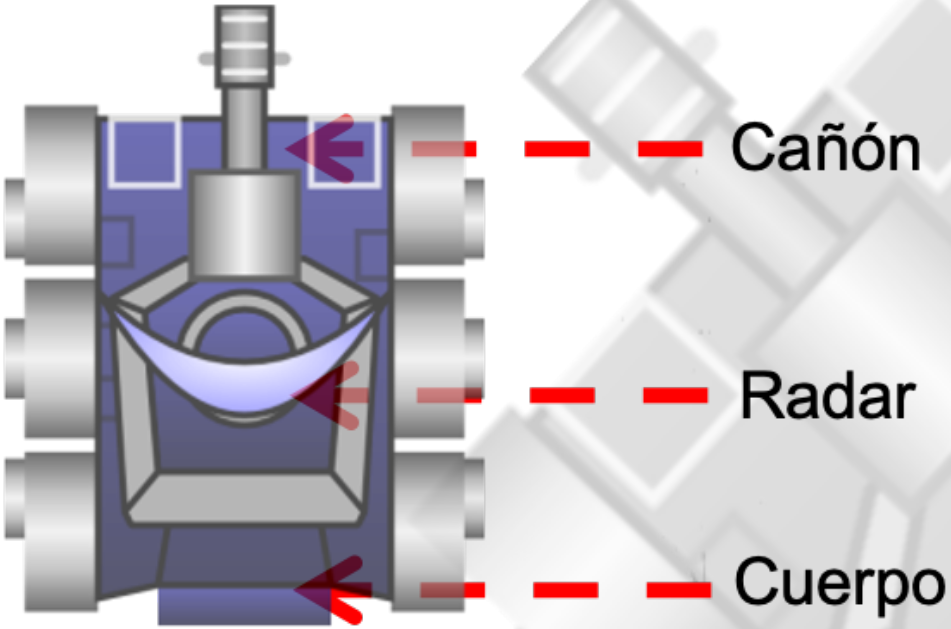
Ranking Batallas



Ranking

Rank	Robot Name	Total Score	Survival	Surv Bonus	Bullet Dmg	Bullet Bonus	Ram Dmg * 2	Ram Bonus	1sts
1st	sample.Walls	350 / 5250 (7 / 14%)	250 / 3250	600	100 / 1286	0 / 94	0 / 20	0 / 0	4
2nd	sample.SpinBot	330 / 3950 (7 / 10%)	250 / 2850	0	80 / 997	0 / 60	0 / 43	0 / 0	0
3rd	sample.TrackFire	442 / 3343 (9 / 8%)	250 / 2000	0	170 / 1200	21 / 143	0 / 0	0 / 0	0
4th	sample.Crazy	286 / 3019 (6 / 8%)	250 / 2500	0	32 / 436	0 / 11	4 / 73	0 / 0	0
5th	sample.Tracker	491 / 2405 (10 / 6%)	250 / 1600	0	192 / 661	19 / 64	30 / 80	0 / 0	0
6th	sample.VelociRobot	305 / 2226 (6 / 5%)	200 / 1350	0	68 / 464	0 / 0	37 / 412	0 / 0	0
7th	sample.RamFire	394 / 2157 (8 / 5%)	150 / 1050	0	154 / 738	19 / 25	71 / 266	0 / 77	0
8th	sample.MyFirstJuniorRo...	114 / 2123 (2 / 5%)	100 / 1700	0	8 / 361	0 / 28	6 / 35	0 / 0	0
9th	sample.MyFirstRobot	286 / 2090 (6 / 5%)	250 / 1750	0	36 / 325	0 / 5	0 / 10	0 / 0	0
10th	sample.Corners	348 / 2053 (7 / 5%)	250 / 1350	0	88 / 619	10 / 82	0 / 2	0 / 0	0
11th	sample.Fire	298 / 1921 (6 / 5%)	250 / 1500	0	48 / 410	0 / 5	0 / 6	0 / 0	0
12th	sample.Target	251 / 1856 (5 / 4%)	250 / 1850	0	0 / 0	0 / 0	1 / 6	0 / 0	0
13th	sample.PaintingRobot	288 / 1647 (6 / 4%)	250 / 1350	0	36 / 282	0 / 0	2 / 16	0 / 0	0
14th	sample.Interactive_v2	250 / 1600 (5 / 4%)	250 / 1600	0	0 / 0	0 / 0	0 / 0	0 / 0	0
15th	sample.SittingDuck	0 / 800 (0 / 2%)	0 / 800	0	0 / 0	0 / 0	0 / 0	0 / 0	0
16th	sample.Interactive	50 / 750 (1 / 2%)	50 / 750	0	0 / 0	0 / 0	0 / 0	0 / 0	0

Anatomía del Robot



Movimiento

- Aceleración (a):
  - Robots acelera con un ratio de 1 pixel por turno.
  - Robocode determina la aceleración dependiendo la distancia donde quieres moverte.
- Velocidad (v):
  - v = at. Velocidad nunca puede exceder 8 pixels/turn.
  - Robocode asume la dirección del vector del ángulo del robot. (heading)

Rotación y Colisiones

- Rotación del robot:
  - (10 - 0,75 * (velocidad)) grados/turno.
  - Cuanto más rápido te muevas, más lento giras.
- Rotación del cañón: 20 grados/turno.
- Rotación del radar: 45 grados/turno.
- Colisiones:

  Entre robots: Cada robot obtiene 0.6 de daño.
  Con la pared (AdvancedRobot): abs(velocity) * 0.5 - 1; (Nunca $<$ 0).

Disparo

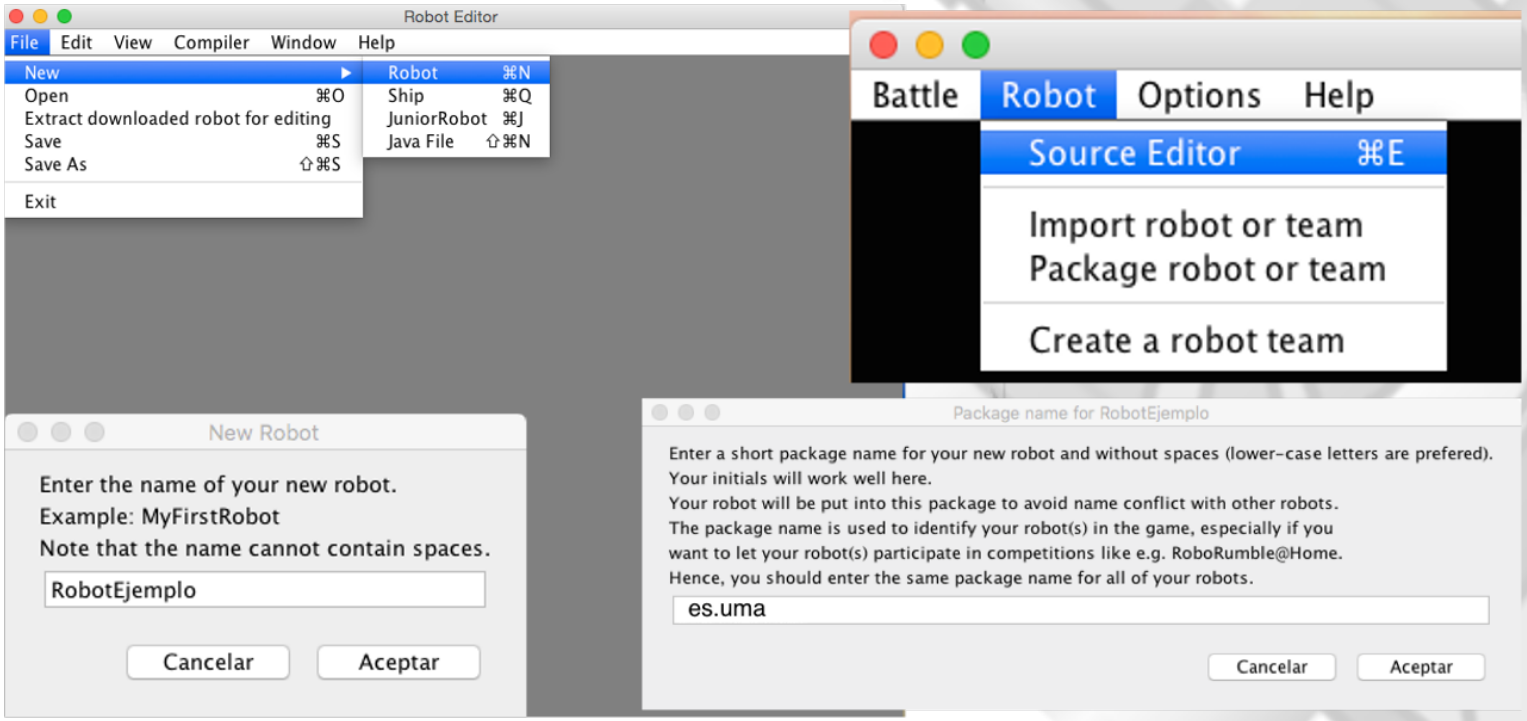
- **Potencia Disparo** (Firepower): [0.1-3]
- **Daño Causado**: 4 * potencia.
  Si potencia $>$ 1, daño adicional $=$ 2 * (potencia - 1).
- **Velocidad Disparo**: 20 - 3 * potencia.
- **Calor generado** (GunHeat): 1 $+$ potencia / 5.
  No es posible disparar si calor $>$ 0.
  Es necesario esperar ((1 $+$ (potencia / 5)) * 10)
- **Energía recuperado cuando acierto**: 3 * potencia.

Energía

- **Elementos que causan decremento energía**
  1. **Alcanzado por una bala**:
     *4 * bulletPower + (max(0, bulletPower - 1) * 2*
  2. **Chocar contra otro robot**:
     0,6
  3. **Chocar contra la pared** (AdvancedRobot):
     *abs(velocity) * 0.5 - 1*
  4. **Disparar una bala**:
     fire(x) cuesta x energía. [0.1-3]
  5. **Inactividad**
- **Elementos que causan incremento energía**
  1. **Alcanzar un robot con una bala**: *3 * firepower*

Crear un nuevo Robot

1. Utilizamos el Editor de Robot
2. File , New, Robot
3. Nombre del Robot: *RobotEjemplo*
4. Nombre del Paquete: *es.uma*



Editor Integrado

Robot Editor

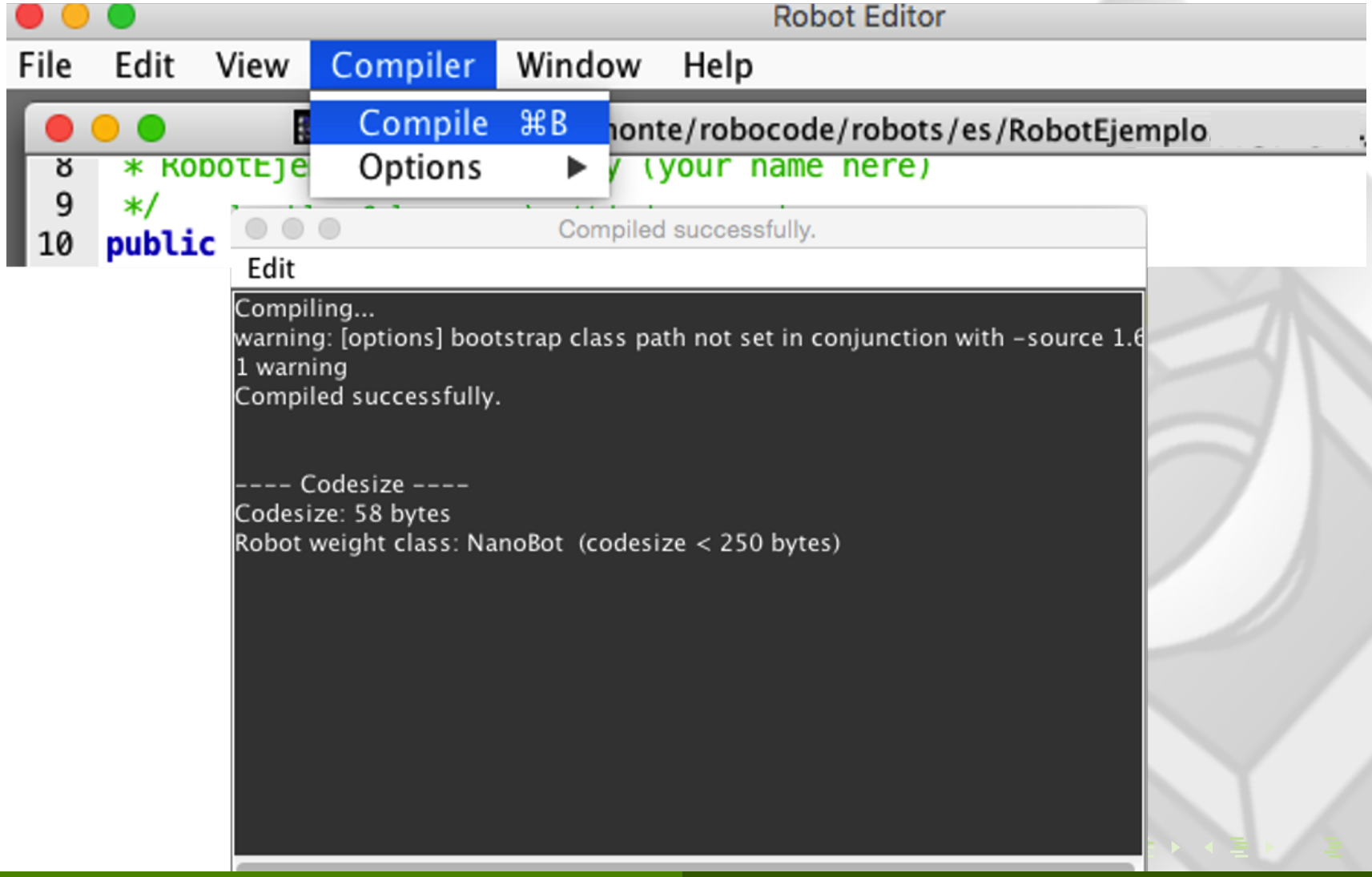
File Edit View Compiler Window Help

Editing – RobotEjemplo *

```
 * RobotEjemplo - a robot by (your name here)
 */
public class RobotEjemplo extends Robot
{
    /**
     * run: RobotEjemplo's default behavior
     */
    public void run() {
        // Initialization of the robot should be put here

        // After trying out your robot, try uncommenting the import at the top,
        // and the next line:

        // setColors(Color.red,Color.blue,Color.green); // body,gun,radar

        // Robot main loop
        while(true) {
            // Replace the next 4 lines with any behavior you would like
            ahead(100);
            turnGunRight(360);
            back(100);
            turnGunRight(360);
        }
    }

    /**
     * onScannedRobot: What to do when you see another robot
     */
```

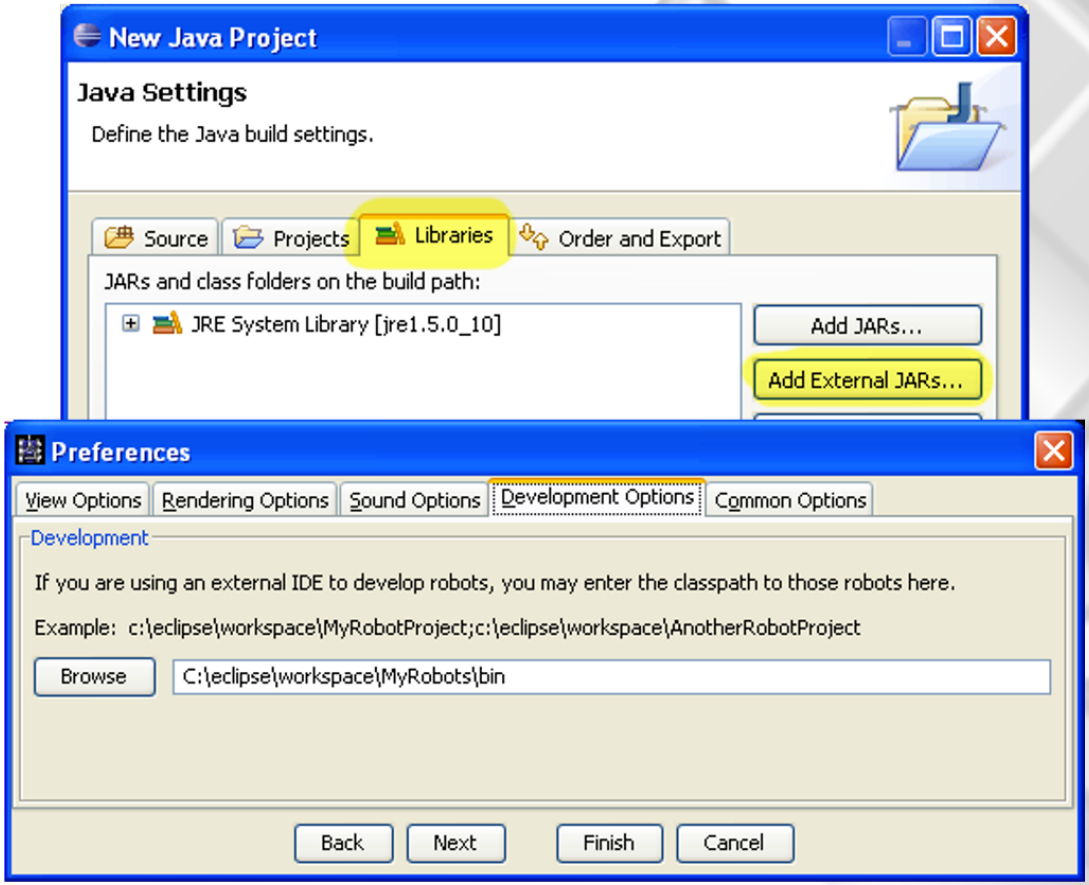
Line: 1 Column: 1

Compilar


Robot Editor
File Edit View Compiler Window Help
Compile ⌘B
Options
onte/robocode/robots/es/RobotEjemplo
* RobotEje
(your name here)
*/
public
Compiled successfully.
Edit
Compiling...
warning: [options] bootstrap class path not set in conjunction with -source 1.6
1 warning
Compiled successfully.
---- Codesize ----
Codesize: 58 bytes
Robot weight class: NanoBot (codesize < 250 bytes)

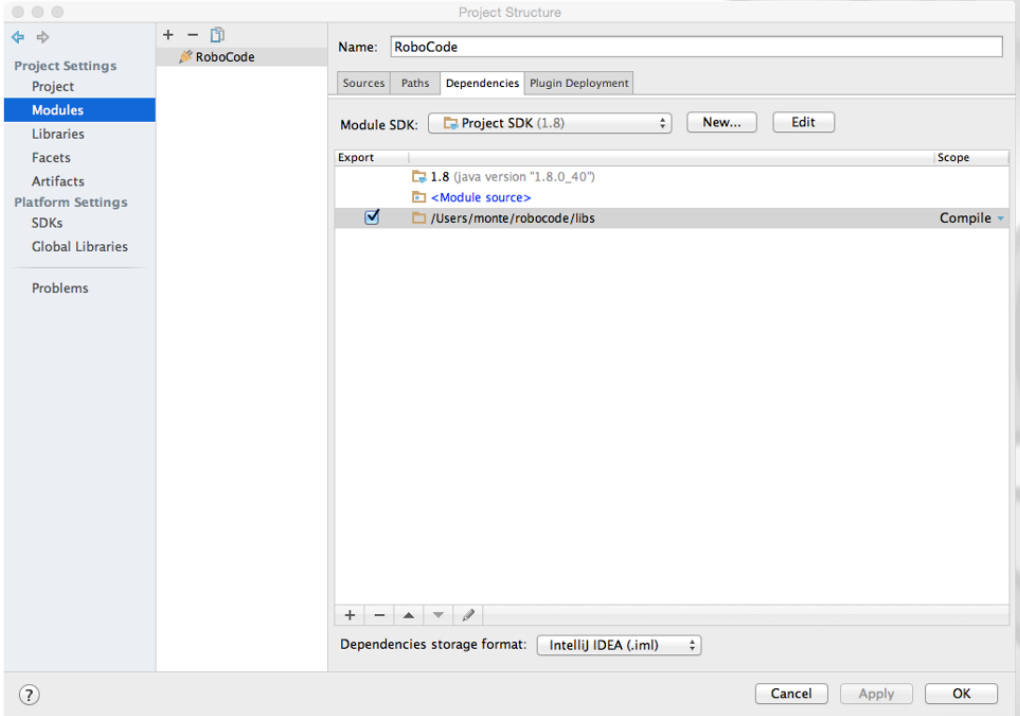
Eclipse

http://robowiki.net/wiki/Robocode/Eclipse



IntelliJ

- File
- Project Structure
- Modules
- Dependencies
- Añadir robocode/libs



Métodos API

http://robocode.sourceforge.net/docs/robocode/robocode/Robot.html

```
public class RobotEjemplo extends Robot{
        public void run() {
                // Parámetros inicialización
                // Robot cuerpo principal
                while(true) {
                        ahead(100);
                        turnGunRight(360);
                        back(100);
                        turnGunRight(360);
                }
        }
```

ahead(double distance): Mueve el robot hacia delante la distancia en pixels.

back(double distance): Mueve el Robot hacia atrás la distancia en pixels.

turnGunRight(double degrees): Gira cañon a la derecha degrees grados.

Métodos API

http://robocode.sourceforge.net/docs/robocode/robocode/Robot.html

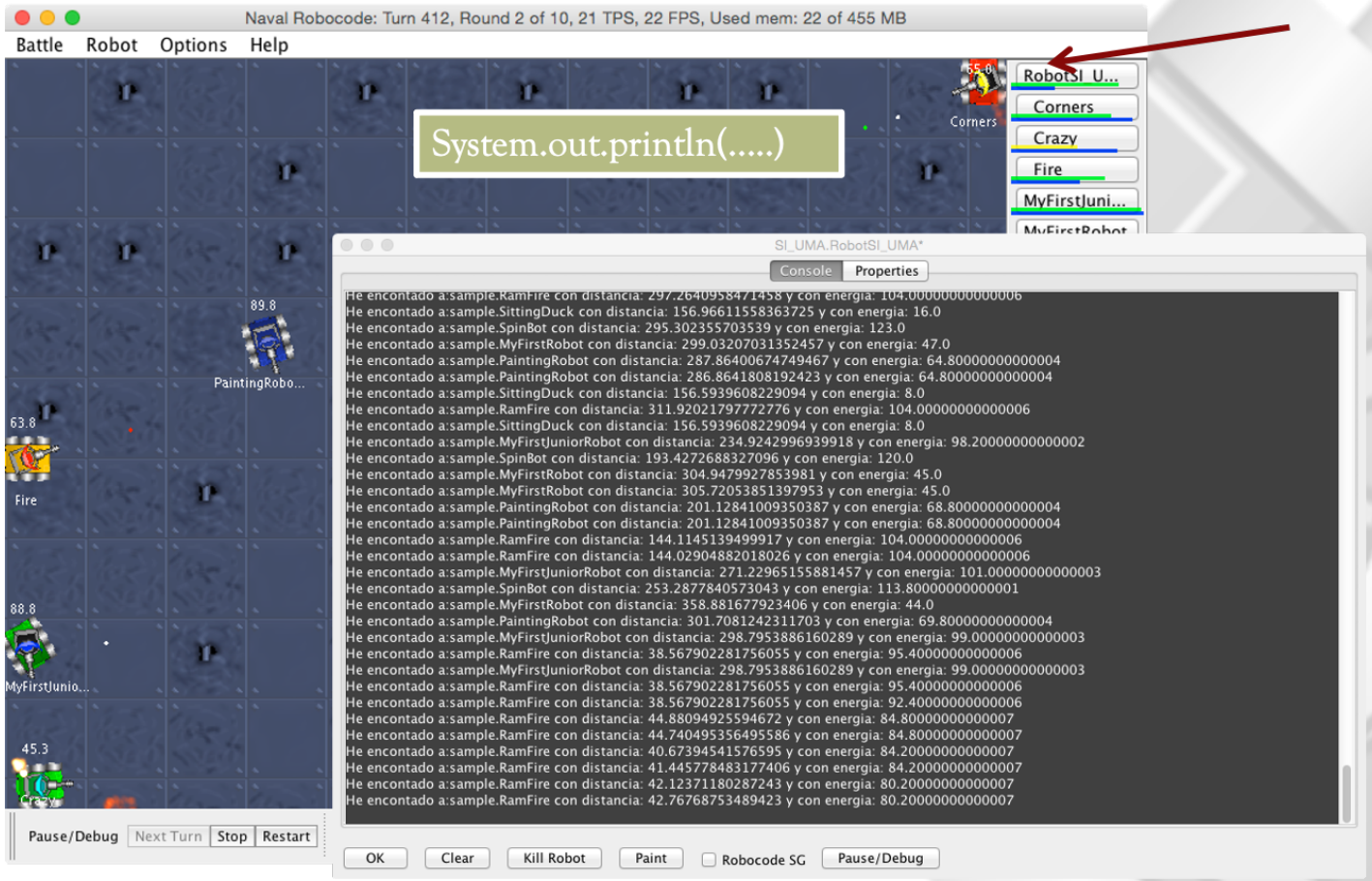
```
public void onScannedRobot(ScannedRobotEvent e) {
                fire(1);
        }
public void onHitByBullet(HitByBulletEvent e) {
                back(10);
        }
public void onHitWall(HitWallEvent e) {
                back(20);
        }
```

onScannedRobot Invocado cuando el robot escanea otro robot.

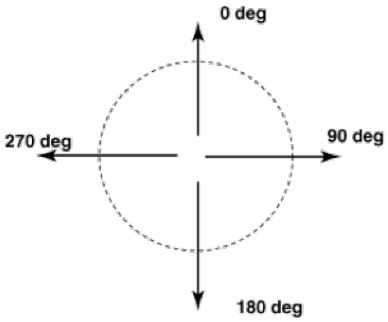
onHitByBullet Invocado cuando un robot es alcanzado por una bala.

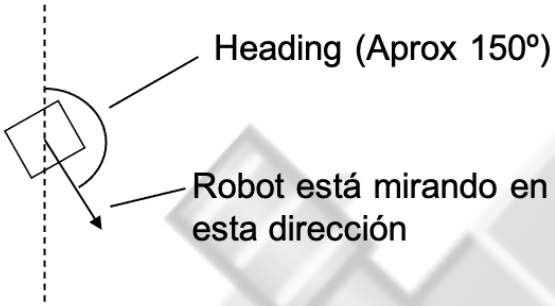
onHitWall Invocado cuando un robot choca con una pared.

Eventos


Naval Robocode: Turn 412, Round 2 of 10, 21 TPS, 22 FPS, Used mem: 22 of 455 MB
Battle Robot Options Help
System.out.println(.....)
Corners
Crazy
Fire
MyFirstJuni...
SI_UMA.RobotSI_UMA*
Console Properties
Pause/Debug Next Turn Stop Restart
OK Clear Kill Robot Paint Robocode SG Pause/Debug

Ángulos





Separación de los componentes:

- **setAdjustGunForRobotTurn**(true) El movimiento del **cañón** es independiente del giro del robot.
- **setAdjustRadarForRobotTurn**(true) El movimiento del **radar** es independiente del giro del robot.
- **setAdjustRadarForGunTurn** (true) El movimiento del **radar** es independiente del giro del **cañón**.

Cuerpo
- getHeading()
- turnLeft(angle) / turnRight(angle)

Cañon
- getGunHeading()
- turnGunRight(angle) / turnGunLeft (angle)

Radar
- getRadarHeading()
- turnRadarRight(angle) / turnRadarLeft(angle)

Ejemplo AnatomyBot

```
public class AnatomyBot extends Robot {
        public void run() {
                out.println("current heading = " + getHeading());
                // face up
                turnLeft(getHeading());
                out.println("1.heading is now = " + getHeading());
                stop();
                // face right
                turnRight(90);
                out.println("2.heading is now = " + getHeading());
                stop();
                // face down
                turnRight(90);
                out.println("3.heading is now = " + getHeading());
                stop();
                // face left
                turnRight(90);
                out.println("4.heading is now = " + getHeading());
                stop();
                // face up again
                turnRight(90);
                out.println("5.heading is now = " + getHeading());
                stop();
```

Ejemplo AnatomyBot

```
out.println("current gun heading = " + getGunHeading());
// detach gun
setAdjustGunForRobotTurn(true);
// point gun right
turnGunRight(90);
out.println("1.gun heading is now = " + getGunHeading());
stop();
// point gun down
turnGunRight(90);
out.println("2.gun heading is now = " + getGunHeading());
stop();
// point gun left
turnGunRight(90);
out.println("3.gun heading is now = " + getGunHeading());
stop();
// point gun up again
turnGunRight(90);
out.println("4.gun heading is now = " + getGunHeading());
stop();
```

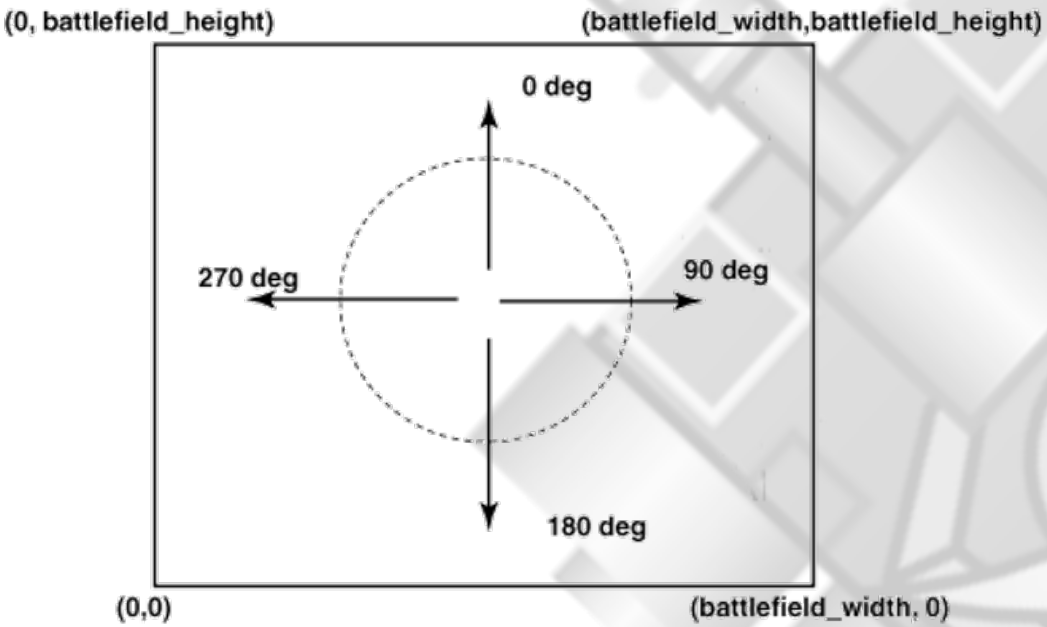
Ejemplo AnatomyBot

```
            out.println("current radar heading = " + getRadarHeading());
            // detach radar
            setAdjustRadarForRobotTurn(true);
            // point radar right
            turnRadarRight(90);
            out.println("1.radar heading is now = " + getRadarHeading());
            stop();
            // point radar down
            turnRadarRight(90);
            out.println("2.radar heading is now = " + getRadarHeading());
            stop();
            // point radar left
            turnRadarRight(90);
            out.println("3.radar heading is now = " + getRadarHeading());
            stop();
            // point radar up again
            turnRadarRight(90);
            out.println("4.radar heading is now = " + getRadarHeading());
            stop();
        }
    }
```

Información Campo de batalla

getBattleFieldWidth() Devuelve ancho campo batalla en pixels.

getBattleFieldHeight() Devuelve alto campo de batalla en pixels.



¡ Importante las coordenadas del campo de batalla !
Observa dónde está la coordenada (0,0).

Información Campo de batalla

getEnergy() Devuelve la energía del robot.

getGunHeat() Devuelve el calor del cañón.

getX() / getY() Posición X e Y del robot.
(0,0) es la esquina izquierda del campo de batalla.

getOthers() Devuelve número de oponentes.

getTime() Devuelve el tiempo de la ronda actual.

Ejercicio 1

http://robocode.sourceforge.net/docs/robocode/robocode/Robot.html

Siguiendo la información de javadoc:

1. Personalice los colores del robot. (Inicialización)
2. Modifique el comportamiento del robot según (Cuerpo Principal):
   - 2.1 La Energía que tiene el robot en ese momento.
   - 2.2 El tiempo de combate.

```
public class RobotSI_UMA extends Robot{
        public void run() {
                // Parámetros inicialización
                // Robot cuerpo principal
                while(true) {
                        ahead(100);
                        turnGunRight(360);
                        back(100);
                        turnGunRight(360);
                }
    }
}
```

Eventos

onBulletHitBullet Una bala es alcanzada por otra bala (destruida).

onBulletHit Una bala alcanza el oponente.

onBulletMissed Una bala no alcanza al oponente, hemos fallado.

onHitByBullet Una bala enemiga nos alcanza.

onHitRobot He chocado con otro robot.

onHitWall He chocado con la pared.

onScannedRobot He escaneado un robot en las medicaciones.

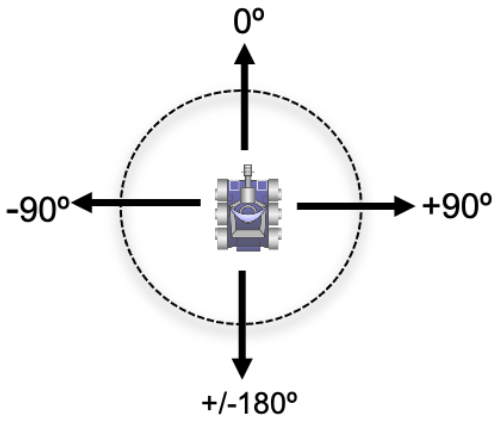
Ejemplo WallBanger

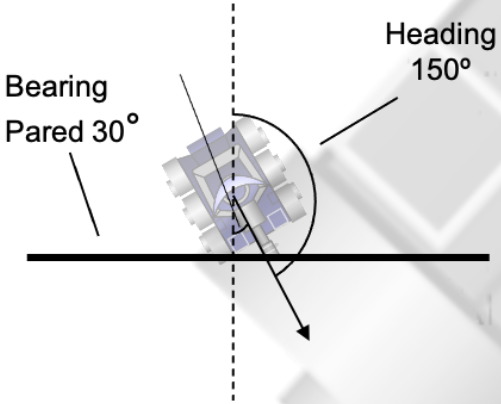
```
public class WallBanger extends Robot {
public void run() {
 out.println("Battlefield width = " + getBattleFieldWidth());
 out.println("Battlefield height = " + getBattleFieldHeight());
  while (true) {
    turnLeft(getHeading()); ahead(10000);  // face up
    turnRight(90); ahead(10000);  // face right
    turnRight(90); ahead(10000);  // face down
    turnRight(90);  ahead(10000);          // face left
        }
}
 public void onHitWall(HitWallEvent event) {
  NumberFormat f = NumberFormat.getNumberInstance();
  f.setMaximumFractionDigits(2);
  out.println("hit a wall (x, y) = (" + f.format(getX()) + ", " +
                                f.format(getY()) + ")");
 }
 public void onScannedRobot(ScannedRobotEvent e) {
  fire(1);
  out.println("Encontrado "+e.getName()+ " a distancia: "+e.getDistance());
 }
}
```

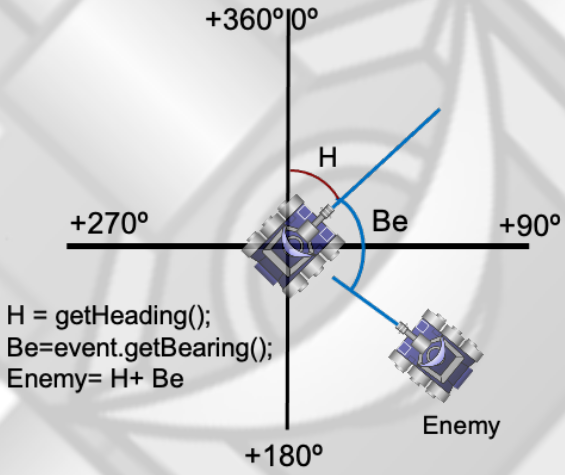
Bearing (Ángulo relativo al cuerpo del robot)

Información cuando choca con un elemento

turnRight(event.getBearing()) getBearing() Devuelve el número de grados más cercano al objeto de choque.







Ejemplo BearingBot

```
public class BearingBot extends Robot {
 public void run() {
  setAdjustRadarForGunTurn(true);
  while (true) {
                // Turn the scanner until we find an enemy robot
        turnRadarRight(10000);
    }
 }
 public void onScannedRobot(ScannedRobotEvent e) {
  NumberFormat f = NumberFormat.getNumberInstance();
         f.setMaximumFractionDigits(2);
        out.println("scanned a robot at bearing " + f.format(e.getBearing()));
        out.println("my heading is currently " + f.format(getHeading()));
        // When we scan a robot, turn toward him
        turnRight(e.getBearing());
        // shoot at him
        fire(3);
        // and ran into him
        ahead(e.getDistance());
 }
```

Ejemplo BearingBot

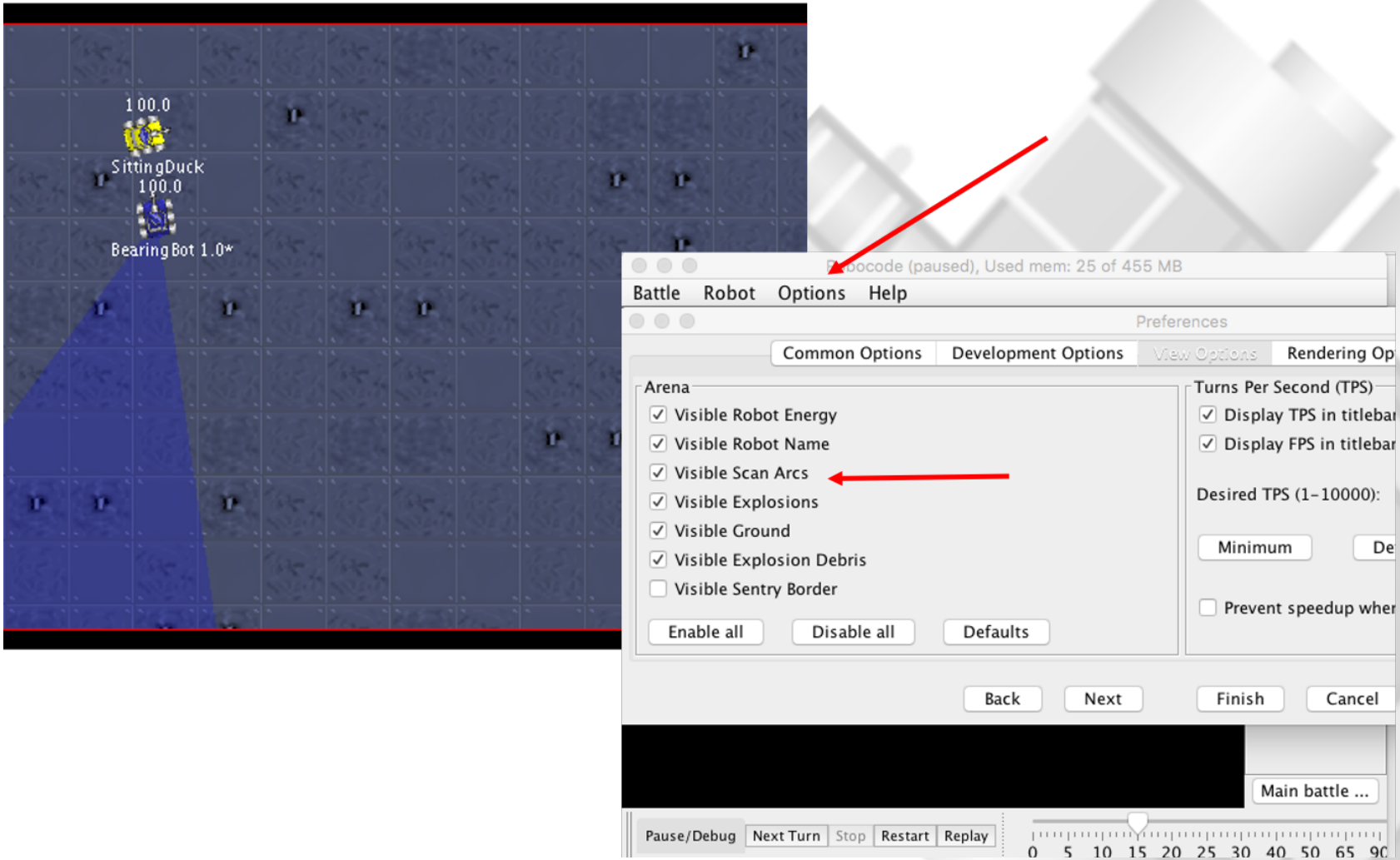
```
public void onHitRobot(HitRobotEvent e) {
        NumberFormat f = NumberFormat.getNumberInstance();
        f.setMaximumFractionDigits(2);
        out.println("hit a robot at bearing " + f.format(e.getBearing()));
        out.println("my heading is currently " + f.format(getHeading()));
        if (e.isMyFault()) {
                out.println("I hit " + e.getName());
        } else {
                out.println(e.getName() + " hit me");
        }
  }
}
```

Bearing (Información adicional en el evento)

onScannedRobot (ScannedRobotEvent e)

- e.getBearing() – [-180 - 180 grados] : 0 enfrente, 180 detrás, 90 derecha, -90 izquierda
- e.getHeading() – donde está oponente mirando: 0 norte , 90 este, 180 sur, 270 oeste
- e.getDistance() – cuánto lejos está oponente
- e.getEnergy() – energía oponente
- e.getVelocity() – velocidad oponente

Depuración Evento



Ejercicio 2

http://robocode.sourceforge.net/docs/robocode/robocode/ScannedRobotEvent.html

Siguiendo la información de javadoc:

1. Obtenga la información en el evento **onScannedRobot**, mediante el objeto **ScannedRobotEvent**:
   - 1.1 El nombre del robot localizado
   - 1.2 La distancia de dicho robot
   - 1.3 y su energía restante

```
public void onScannedRobot(ScannedRobotEvent e) {
   fire(1);
}
```

Ejercicio 3

http://robocode.sourceforge.net/docs/robocode/robocode/HitByBulletEvent.html

Siguiendo la información de javadoc:

1. Obtenga información en el evento **onHitByBullet**, mediante el objeto **HitByBulletEvent**:
   - 1.1 ¿Quién disparó la bala?
   - 1.2 ¿Qué velocidad llevaba la bala?

```
public void onHitByBullet(HitByBulletEvent e) {
   back(10);
}
```

Ejercicio 4

http://robocode.sourceforge.net/docs/robocode/robocode/HitWallEvent.html

Siguiendo la información de javadoc:

1. Obtenga información en el evento **onHitWall**, mediante el objeto **HitWallEvent**:
   1.1 Orientación en grados (x) con la pared que hemos chocado, relativo al cuerpo del robot, (-180 <= x < 180)

```
public void onHitWall(HitWallEvent e) {
                back(20);
}
```

Ejercicio 5

Siguiendo la información de javadoc:

1. Obtenga información cuando una bala de nuestro robot impacta a otro robot. (**onBulletHit**)
2. Obtenga información cuando una bala de nuestro robot no impacta a otro robot. (**onBulletMissed**)
3. Obtenga información cuando nuestro robot impacta con otro robot. (**onHitRobot**)

Entrega

Sube el código Java del robot de prácticas que resuelva los siguientes ejercicios:

1. Ejercicio 1.
2. Ejercicio 2.
3. Ejercicio 3.
4. Ejercicio 4.
5. Ejercicio 5.

José A. Montenegro Montes
Dpto. Lenguajes y Ciencias de la Computación
ETSI Informática. Universidad de Málaga
jmmontes@uma.es

UNIVERSIDAD DE MÁLAGA