

# Robocode – die letzte Lektion

## 1 Bevor Du beginnst

- Stelle sicher, dass Du Java und Robocode wie im Dokument vom ersten Tag **Robocode auf Windows installieren.docx** beschrieben, installiert hast.
- Stelle sicher, dass Du das Dokument **Erste Schritte mit Robocode** durchgearbeitet hast. Wir bauen auf dem dort entwickelten Roboter auf
- Stelle sicher, dass Du das Dokument **Robocode - wir entwickeln den Roboter weiter** durchgearbeitet hast.

## 2 Was ist der Plan für heute?

Heute werden wir unseren **Day3** Roboter erschaffen, wir nennen ihn auch RevengeBot (Rache Roboter). Wir wollen unseren **Day2** Roboter in mehrfacher Hinsicht verbessern. Das Ziel ist, dass unser **Day3** Roboter unseren **Day2** Roboter schlagen wird.

Beginnen wir mit der Codierung!

## 3 Starte Robocode und verwende den Code-Editor

### 3.1 Start Robocode

- Starte jetzt Robocode wie es im Dokument vom ersten Tag, **Robocode auf Windows installieren.docx**, beschrieben ist.
- Jetzt wähle vom **Robot Menu** den **Source Editor** (Quellcode Editor) aus, um den Java Editor zu starten. Auch das ist im Dokument vom ersten Tag beschrieben

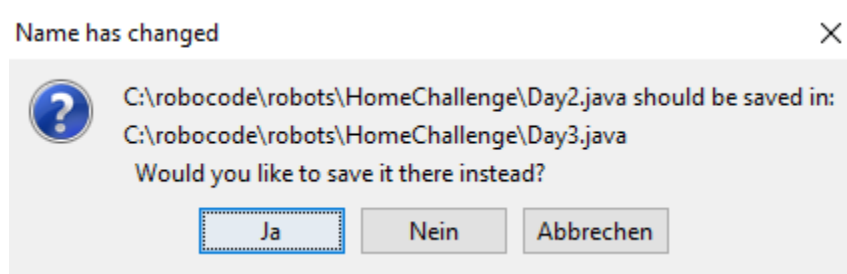
### 3.2 Erstelle Deinen Day3 Robot

- Wähle **File > Open**
- Öffne den **HomeChallenge** Ordner und darin öffne die Datei **Day2.java**
- Ändere im Editor die Zeile

```
public class Day2 extends JuniorRobot  
zu
```

```
public class Day3 extends JuniorRobot
```

- Speichere die Datei, und wähle **Ja** aus, um die Datei neu als **Day3.java** zu speichern.



## 4 Wir planen unseren neuen Roboter

Heute werden wir einen Roboter programmieren, der entwickelt wird, um den **Day2** Roboter zu besiegen. Wir beginnen damit, alles aufzulisten, was wir über das Verhalten unseres **Day2 - Roboters** wissen und wie wir dieses Verhalten besiegen können.

Day2 Robot Verhalten	Wie können wir das schlagen?
Day2 erwartet, dass das Ziel stillsteht	<b>Wie bewegen uns immer</b> Wir werden uns immer bewegen und ständig andere Roboter suchen (scannen)
Day2 beginnt sich zu bewegen, sobald er getroffen ist	<b>Grosse Kugeln</b> Wir werden nicht so schnell und so oft schiessen, aber wir werden viel grössere Kugeln verwenden, wenn wir schiessen
Day2 steht still, wenn er schießt	<b>Stationäres Ziel</b> <ul style="list-style-type: none"> <li>• Wenn wir einen Feind im Radar sehen, schiessen wir</li> <li>• Wenn wir von einer Kugel getroffen werden, werden schiessen wir sofort in die Richtung zurück, aus der die Kugel kam,</li> </ul>
Day2 schießt weiter, wenn er einen Feind erkennt	<b>Wir weichen aus</b> Sobald wir getroffen wurden und zurückgeschossen haben, gehen wir schnell aus dem Weg, Day2 schießt weiter und verschwendet so seine Energie

Ob Du es glaubst oder nicht, aber die obige Liste ist das, was Computerprogrammierer ein Anforderungsdokument nennen und ist etwas, das echte Programmierer tun, bevor sie Code schreiben. Verstehe das Problem, beschreibe wie Du es lösen willst, und schreibe erst dann den Code.

## 5 Bereite den Code vor

Wir löschen zuerst eine Menge Code, den wir nicht mehr verwenden, aus dem Day2 Roboter:

Lösche diesen Import vom Anfang der Datei (keine knifflige Mathematik für diesen Roboter):

```
import java.awt.geom.Point2D;
```

Löschen die folgenden 3 Variablen aus dem Abschnitt Variablendeklaration

```
private boolean bScannedEnemy = false;
```

```
private boolean bGoForward = true;  
private int iEnemyX = -1;  
private int iEnemyY = -1;
```

Bereinige die `run()` - Methode, damit sie wie folgt aussieht:

```
public void run() {  
}
```

Bereinige die `onScannedRobot()` - Methode, damit sie wie folgt aussieht:

```
public void onScannedRobot() {  
}
```

Bereinige die `onHitByBullet()` - Methode, damit sie wie folgt aussieht:

```
public void onHitByBullet() {  
}
```

Keep the direction switch code in the `onHitWall()` method but delete the rest, it should look like this:

```
public void onHitWall() {  
    bGoForward = !bGoForward;  
}
```

Lösche die gesamte `absoluteBearing()` - Methode, aber denke daran, nicht die allerletzte geschweifte Klammer `}` in der Datei zu löschen (diejenige, die unsere `Day3`-Klasse schliesst).

Speichere und kompiliere den Code (wie wir das schon früher beschrieben haben). Du solltest keine Fehler sehen. Jetzt hast Du einen `Day3` Roboter, der absolut nichts tut.

Jetzt beginnen wir mit der Umsetzung unserer Anforderungen! Wir programmieren weiter.

## 6 Day3 Roboter, auch RevengeBot genannt, Implementation

Wir programmieren alle Anforderung, die in der obigen Tabelle aufgeführt ist, nacheinander:

### 6.1 Anforderung 1: Wir bewegen uns immer

Wir tun dies in der Run-Methode:

```
public void run() {
    // Always Move
    // Bewege Dich immer
    turnGunRight(5);
    if (bGoForward) {
        ahead(10);
    }
    else {
        back(10);
    }
}
```

### 6.2 Anforderung 2 & 3: Grosse Kugeln & Stationäres Ziel

Bevor wir dies tun, müssen wir zwei Dinge sicherstellen: Wenn Du die die **fire()** - **Methode** aufrufst, kannst Du ihr einen Wert von 0.1 bis 3 (in Schritten von 0.1) geben. Je höher die Zahl, desto langsamer bewegt sich die Kugel, desto mehr Energie verbraucht sie, desto langsamer ist die Feuerrate (das heisst weniger Kugeln pro Runde), aber desto mehr Schaden richtet die Kugel an. Wir drehen die Stärke auf 3, auf das Maximum.

Wenn Du von einer Kugel getroffen wirst, kannst Du eine Variable **hitByBulletAngle** verwenden. Der Wert dieser Variablen gibt den Winkel (in Grad) an, woher die Kugel stammt. Ändere nun die Methoden **onScannedRobot()** und **onHitByBullet()** in:

```
public void onScannedRobot() {
    // Big Bullets
    // Grosse Kugeln
    fire(3);
}

public void onHitByBullet() {
    // Stationary Target
    // Stationäres Ziel
    turnGunTo(hitByBulletAngle);
    // Big Bullets
    // Grosse Kugeln
    fire(3);
}
```

*Um es uns später leichter zu machen zu verstehen warum wir etwas so programmiert haben, fügen wir auch jetzt wieder Kommentarzeilen ein!*

### 6.3 Anforderung 4: Wir weichen aus

Schliesslich müssen wir unsere Ausweichmassnahmen kodieren. Wir werden zwei Methoden anwenden, die es uns ermöglichen, schnell vorwärts zu gehen und uns zu drehen oder rückwärts zu gehen und uns zu drehen. Fügen der `onHitByBullet()` Methode die folgenden unterstrichenen Codezeilen hinzu:

```
public void onHitByBullet() {
    // Stationary Target
    // Stationäres Ziel
    turnGunTo(hitByBulletAngle);
    // Big Bullets
    // Grosse Kugeln
    fire(3);
    // Evasive Action
    // Wie weichen aus
    if (bGoForward) {
        turnAheadRight(50, 45);
    }
    else {
        turnBackLeft(50, 45);
    }
}
```

Das wars! Speichere und kompiliere Deinen Day3 Code. Sind alle Fehler behoben, kannst du einen neuen Kampf mit einem Day2 – und einem Day3 Roboter beginnen. Normalerweise wird Dein Day3 Roboter den Day2 Roboter jedes Mal schlagen! Schau Dir das Log eines zwei Runden Kampes an, Day3 hat 306 Punkte, Day2 nur 48 Punkte.

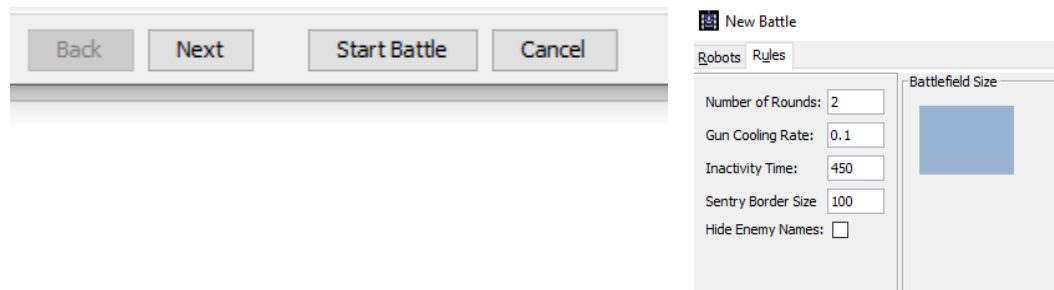
Results for 2 rounds

Rank	Robot Name	Total Score	Survival	Surv Bonus	Bullet Dmg	Bullet Bonus	Ram Dmg * 2	Ram Bonus	1sts	2nds	3rds
1st	HomeChallenge.Day3*	306 (86 %)	100	20	155	31	0	0	2	0	0
2nd	HomeChallenge.Day2*	48 (14 %)	0	0	48	0	0	0	0	2	0

Save OK

Der Day3 Roboter wird sogar einen guten Kampf gegen einen Day1 Roboter führen.

Zur Erinnerung: Unten im 'New Battle' Schirm kannst Du den 'Next' Knopf drücken um die Anzahl Runden, Grösse des Schlachtfeldes und weitere Parameter definieren!



Wir haben das Ende der heutigen Ausbildung erreicht. Wir geben Dir ein paar Ressourcen, mit Information zur Vertiefung. Morgen werden wir Dir die ultimative Herausforderung geben.

## 7 Weitere Ressourcen

Eine Erinnerung von gestern, die Dokumentation für die JuniorRobot-Klasse, die Dir alle Variablen und Methoden zeigt, die Du verwenden kannst, findest Du hier:

<https://robocode.sourceforge.io/docs/robocode/index.html?robocode/JuniorRobot.html>

Es gibt auch **Robot** – und **AdvancedRobot** – **Klassen**, die Du erweitern kannst, diese Klassen geben Dir Zugriff auf viel mehr Variablen und Methoden, sind aber komplizierter wenn Du damit arbeiten willst:

<https://robocode.sourceforge.io/docs/robocode/index.html?robocode/Robot.html>

<https://robocode.sourceforge.io/docs/robocode/index.html?robocode/AdvancedRobot.html>

Und schließlich die RoboWiki, eine grosse Sammlung von Ressourcen für Robocode Roboter-Coder:

[http://robowiki.net/wiki/Main\\_Page](http://robowiki.net/wiki/Main_Page)

Nur für den Fall, dass das Wiki nicht erreichbar ist, versuche es auf der Website des Webarchivs zu finden

[https://web.archive.org/web/20190828182533/http://www.robowiki.net/wiki/Main\\_Page](https://web.archive.org/web/20190828182533/http://www.robowiki.net/wiki/Main_Page)

Oder alternativ auch hier [http://robowiki.duckdns.org/index.php/Main\\_Page](http://robowiki.duckdns.org/index.php/Main_Page)