

Robocodeのインストール

1. プログラムソースは、[ここからダウンロード](#)



The screenshot shows the SourceForge page for Robocode. At the top, there's a navigation bar with 'Open Source Software', 'Business Software', and 'Resources'. Below that, there's an advertisement for EIZO. The main content area features the Robocode logo and a description: 'Robocode is a programming tank game for Java and .NET. Brought to you by: [fni](#), [matn](#), [pavelsavara](#)'. Below this, there's a green progress bar indicating 'Your download will start shortly...' with a '0' in a circle. There are three buttons: 'Get Updates', 'Share This', and 'Problems Downloading?'. Below these buttons, it says 'robocode-1.9.3.9-src.zip | Scanned for malware ✓'. At the bottom, there's a section titled 'Other Useful Business Software' with a link to 'Build Intelligent Content Applications'. A red box highlights the download button for 'robocode-1.9.3.9-src.zip'.

2. インストーラーは、[ここからダウンロード](#)
3. Robotクラスのメソッドの説明(JavaDoc)は[こちら](#)
4. RobocodeのWiki(英語だけど色々書いている)

Robocodeは無料ゲーム

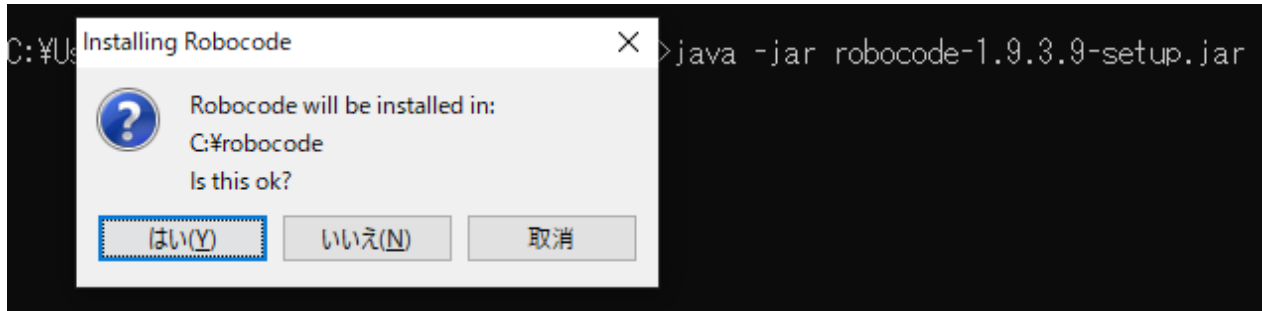
早い話がOSSということです。※オープン・ソース・ソフトウェア＝無料で使用できるアプリケーション

インストールについて

SourceForgeからダウンロードしたら下のようなファイルがダウンロードできます。

robocode-1.9.3.9-setup.jar

このファイルは実行可能JARファイルです。しかし、ダブルクリックで実行できなかったので、コマンドで実行しました。下のようにコマンドを叩きます。



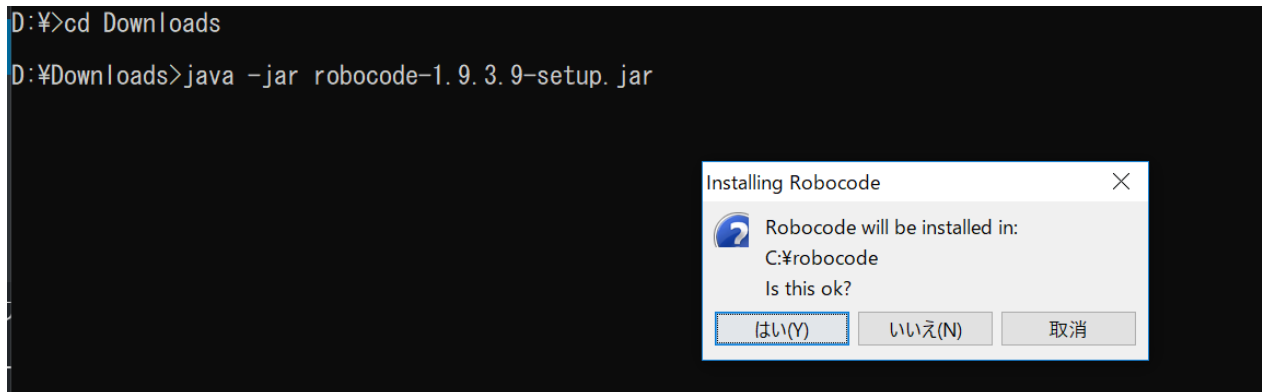
<実行コマンド>

- `java -jar robocode-1.9.3.9-setup.jar`

コマンドの内容としては、下の通りです。

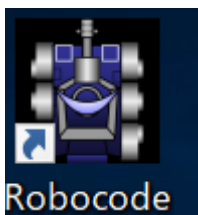
- javaコマンドでJavaを起動したりできます。
- -jarオプションをつけてjarファイルaaの実行を行います。
- このオプションをつけてJARファイルを指定すると実行できます。

下のような感じです。

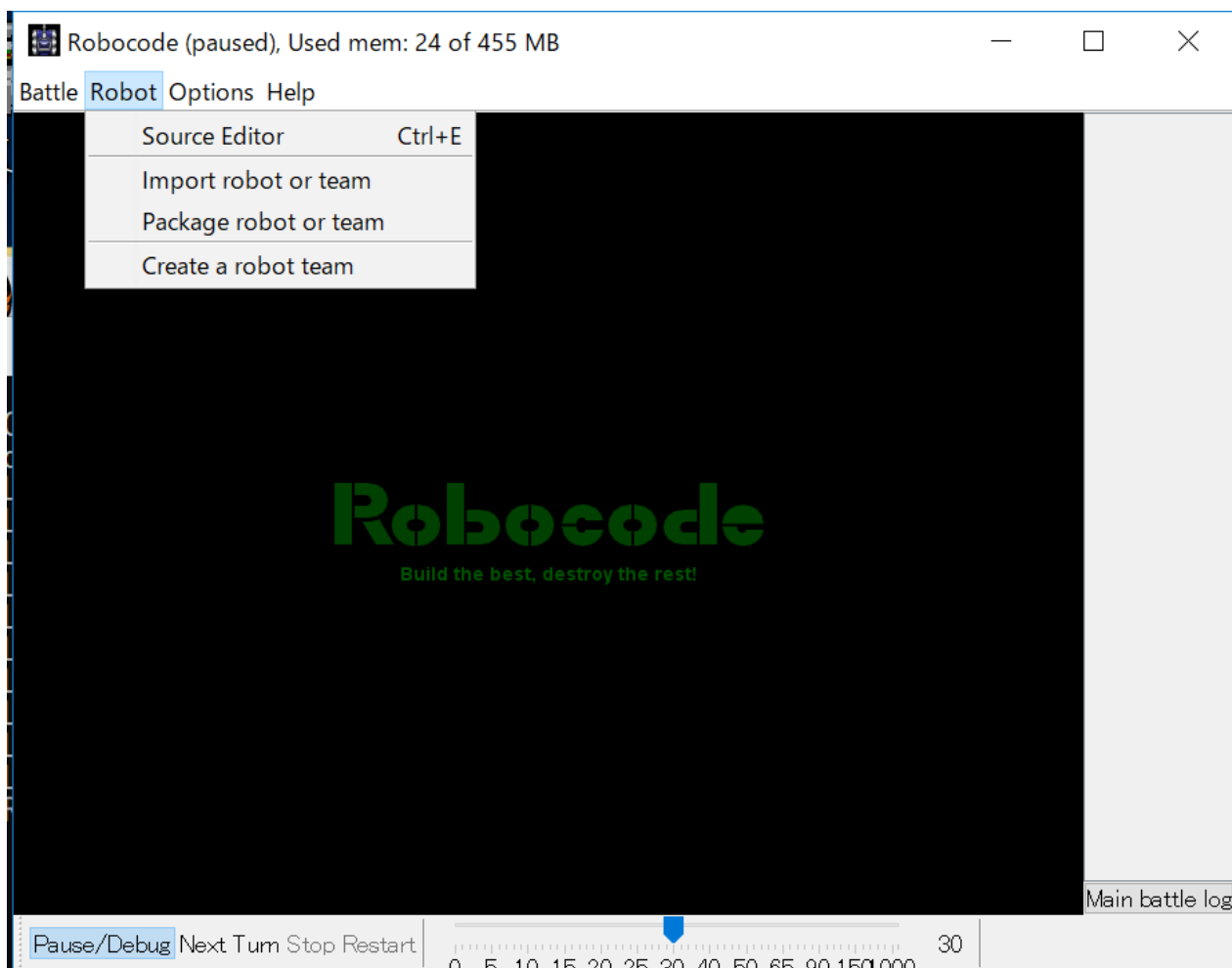


Roboを作る

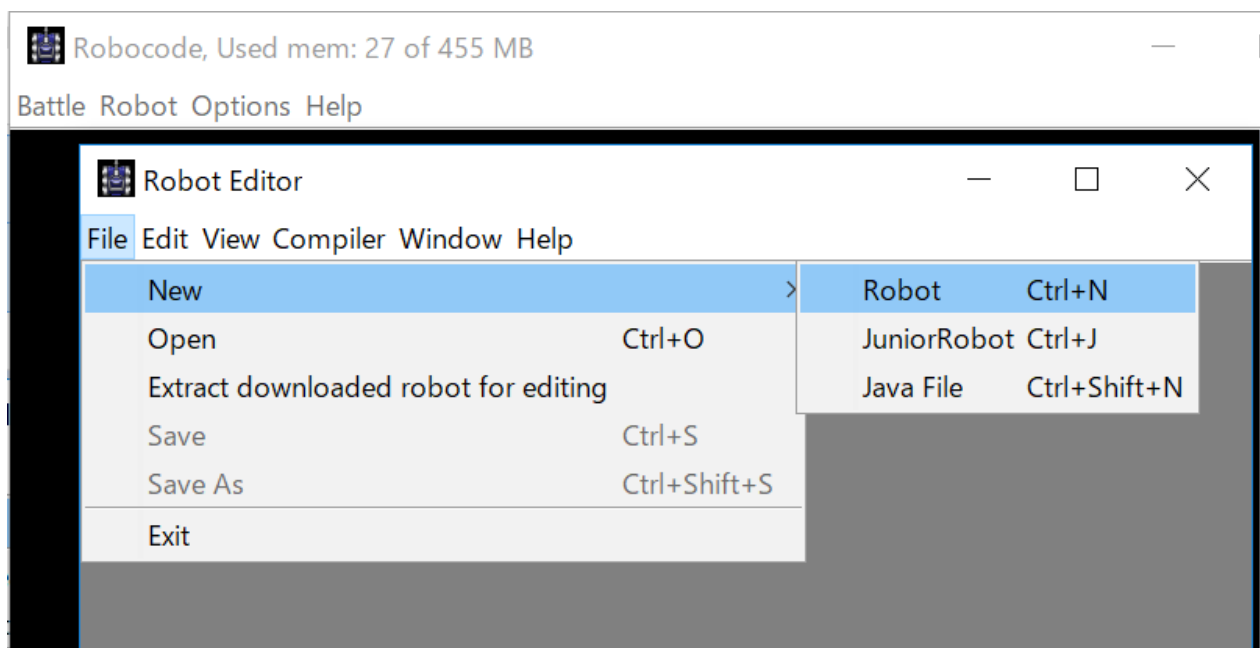
インストールしたアプリを起動します。



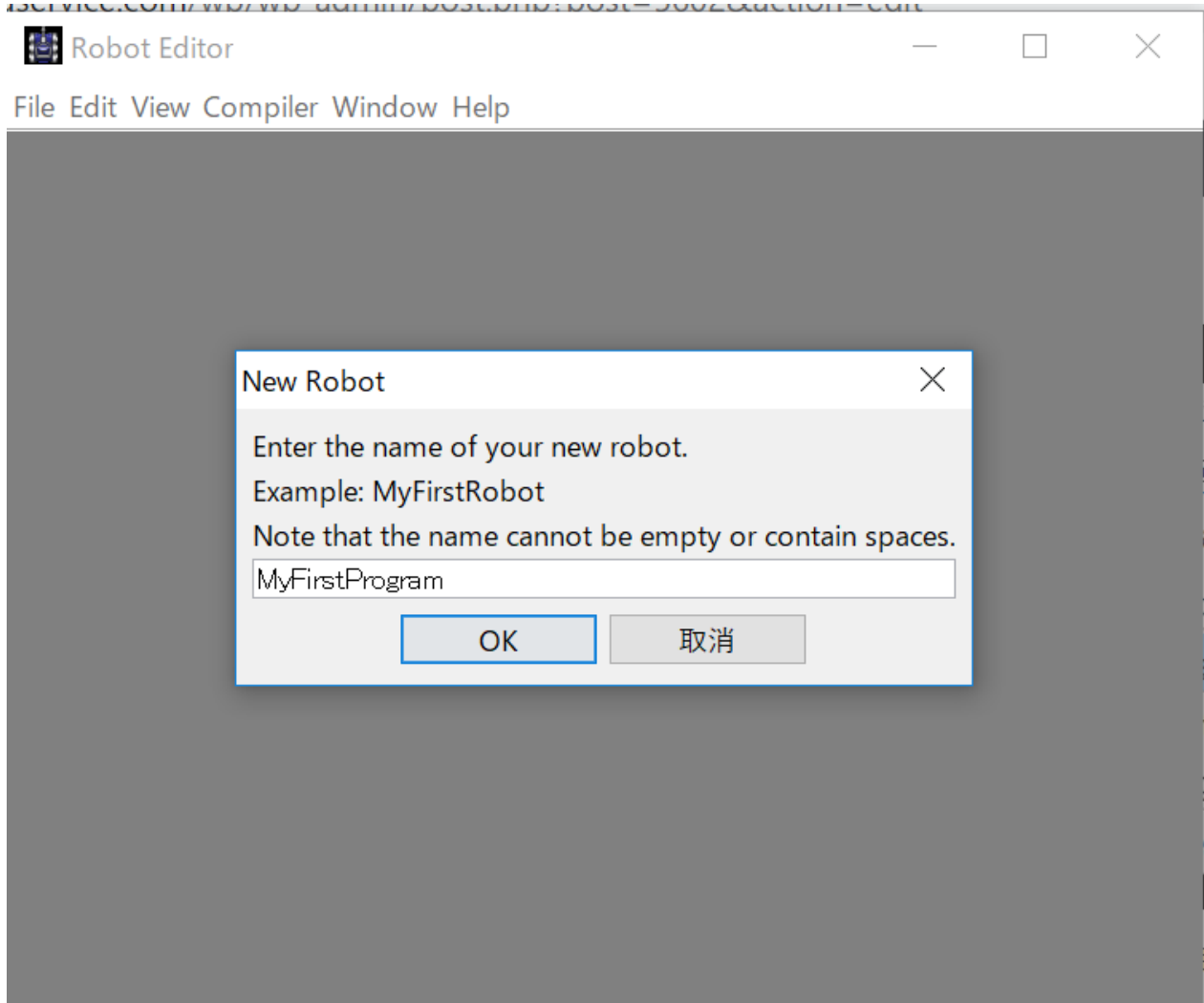
このアイコンをクリックすると起動できます。



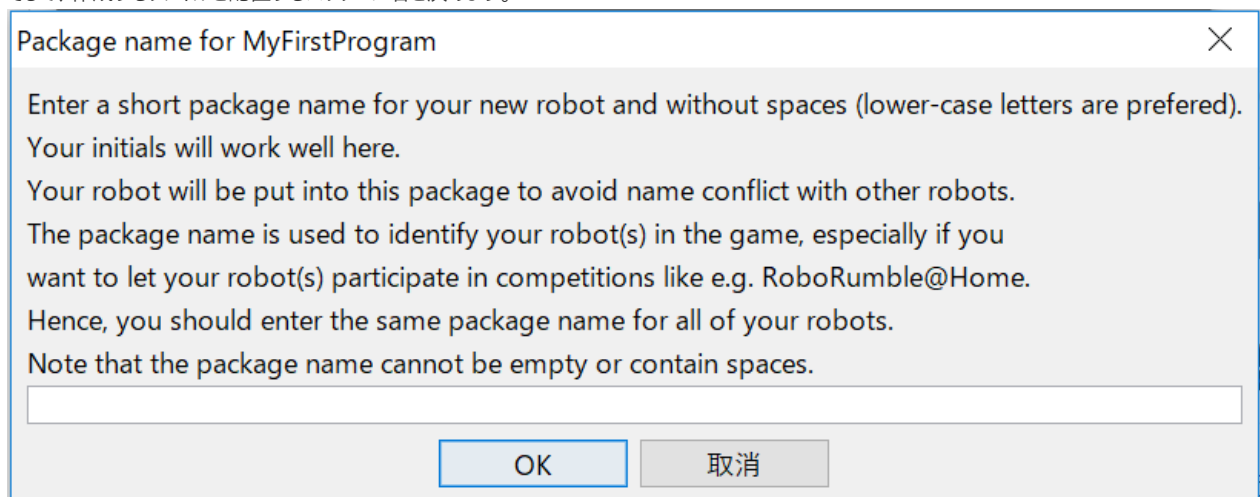
そして、プログラムを作成します。



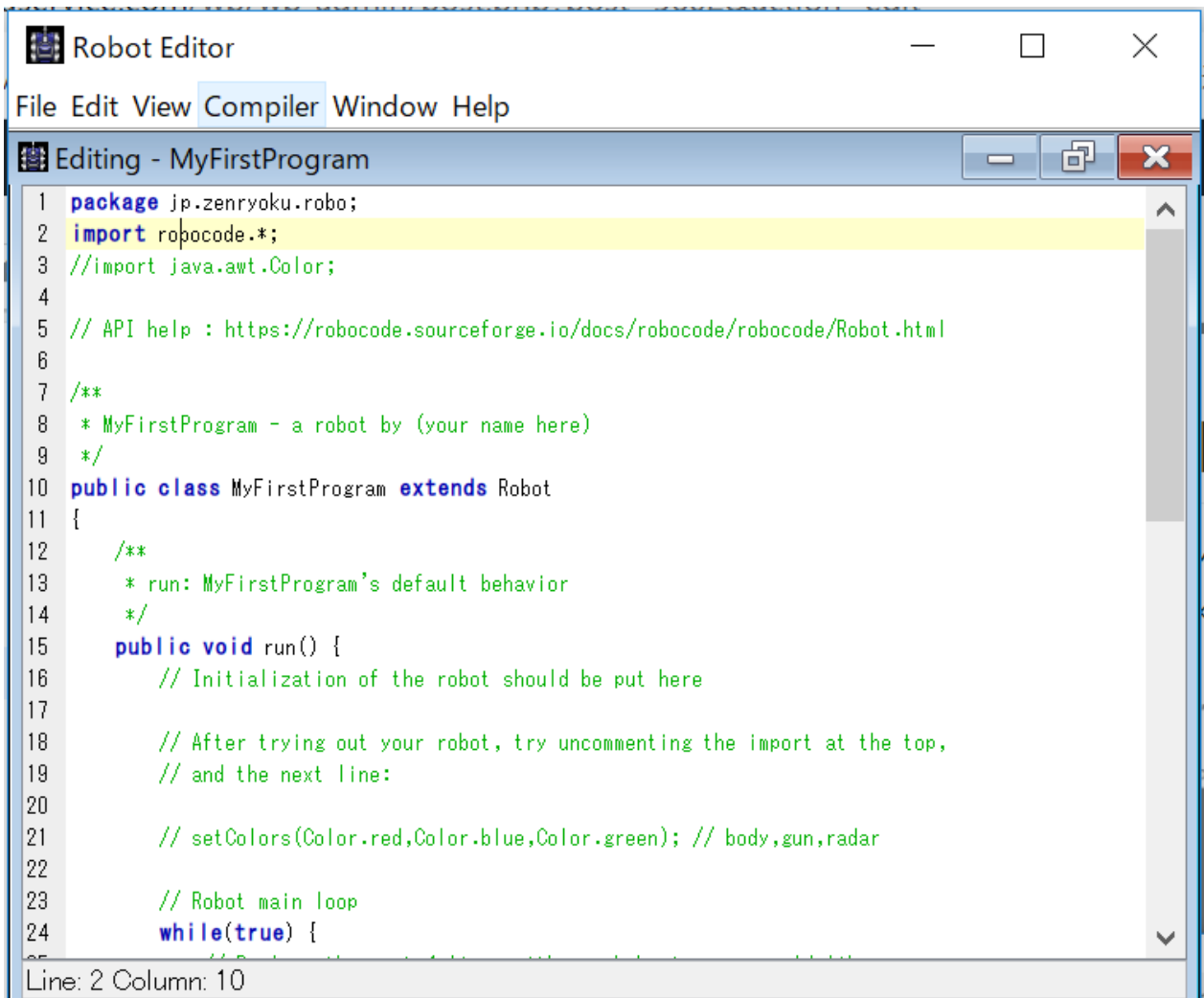
起動したら、ソースエディタを起動します。初めにプログラムのファイル名(クラス名)を決めます。



そして、作成するファイルを配置するパッケージ名を決めます。



そして、Javaコードが生成されます。

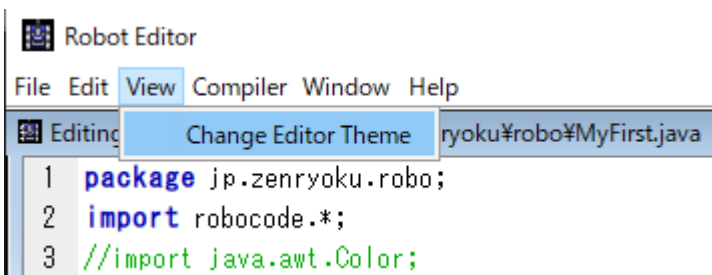


The screenshot shows the Robot Editor application window. The title bar reads "Robot Editor". The menu bar includes "File", "Edit", "View", "Compiler", "Window", and "Help". The window title is "Editing - MyFirstProgram". The code editor displays the following Java code:

```
1 package jp.zenryoku.robo;
2 import robocode.*;
3 //import java.awt.Color;
4
5 // API help : https://robocode.sourceforge.io/docs/robocode/robocode/Robot.html
6
7 /**
8  * MyFirstProgram - a robot by (your name here)
9  */
10 public class MyFirstProgram extends Robot
11 {
12     /**
13      * run: MyFirstProgram's default behavior
14      */
15     public void run() {
16         // Initialization of the robot should be put here
17
18         // After trying out your robot, try uncommenting the import at the top,
19         // and the next line:
20
21         // setColors(Color.red,Color.blue,Color.green); // body,gun,radar
22
23         // Robot main loop
24         while(true) {
```

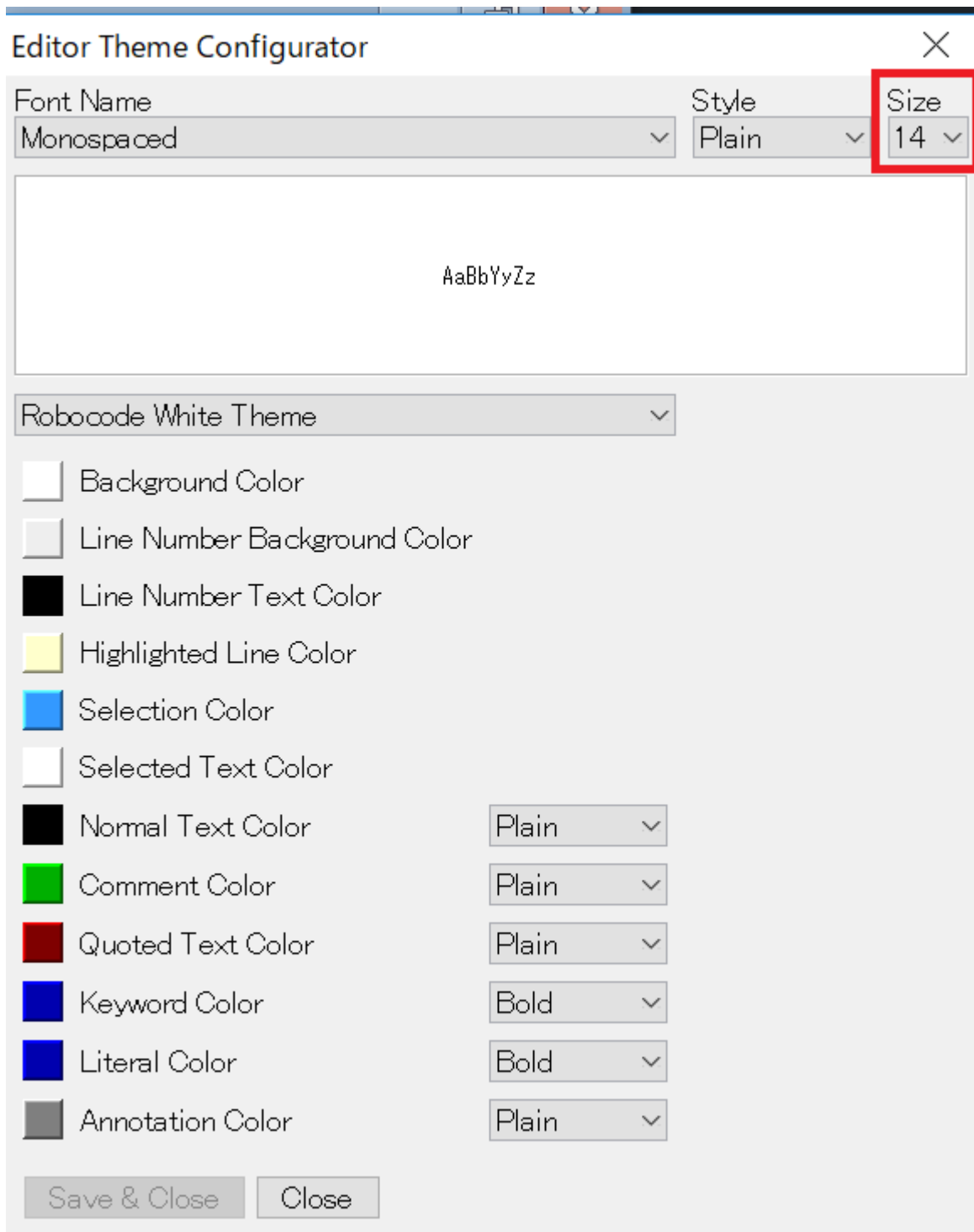
The status bar at the bottom indicates "Line: 2 Column: 10".

しかし、文字のサイズが小さいので、文字の大きさを変更します。



The screenshot shows the Robot Editor application window with the "View" menu open. The menu options are "Change Editor Theme" and "ryoku¥robo¥MyFirst.java". The code editor displays the following Java code:

```
1 package jp.zenryoku.robo;
2 import robocode.*;
3 //import java.awt.Color;
```



上のような手順で、サイズを変更します。

そして、お待ちかねプログラムコードです。

Java Code

```
package jp.zenryoku.robocode;
import robocode.*;
//import java.awt.Color;

// API help : https://robocode.sourceforge.io/docs/robocode/robocode/Robot.html

/**
```

```

* MyFirstProgram - a robot by (your name here)
*/
public class MyFirstProgram extends Robot
{
    /**
     * run: MyFirstProgram'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
     */
    public void onScannedRobot(ScannedRobotEvent e) {
        // Replace the next line with any behavior you would like
        fire(1);
    }

    /**
     * onHitByBullet: What to do when you're hit by a bullet
     */
    public void onHitByBullet(HitByBulletEvent e) {
        // Replace the next line with any behavior you would like
        back(10);
    }

    /**
     * onHitWall: What to do when you hit a wall
     */
    public void onHitWall(HitWallEvent e) {
        // Replace the next line with any behavior you would like
        back(20);
    }
}

```

全体を眺めると、慣れていない人には「なんじゃこりゃー」となるかもしれませんが、一部分のみを見てみましょう。runメソッドです。

```

/**
 * run: MyFirstProgram'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);
    }
}

```

```
}  
}
```

そして、コメントを見えます、英語です。。。

こんな時はGoogle翻訳で訳しましょう。

- `run: MyFirstProgram's default behavior`

英語 ▼

↔

日本語 ▼

run:
MyFirstProgram's
default behavior

×

実行 : MyFirstProgramの
デフォルトの動作
Jikkō: MyFirstProgram no deforuto no
dōsa





[Google 翻訳で開く](#)

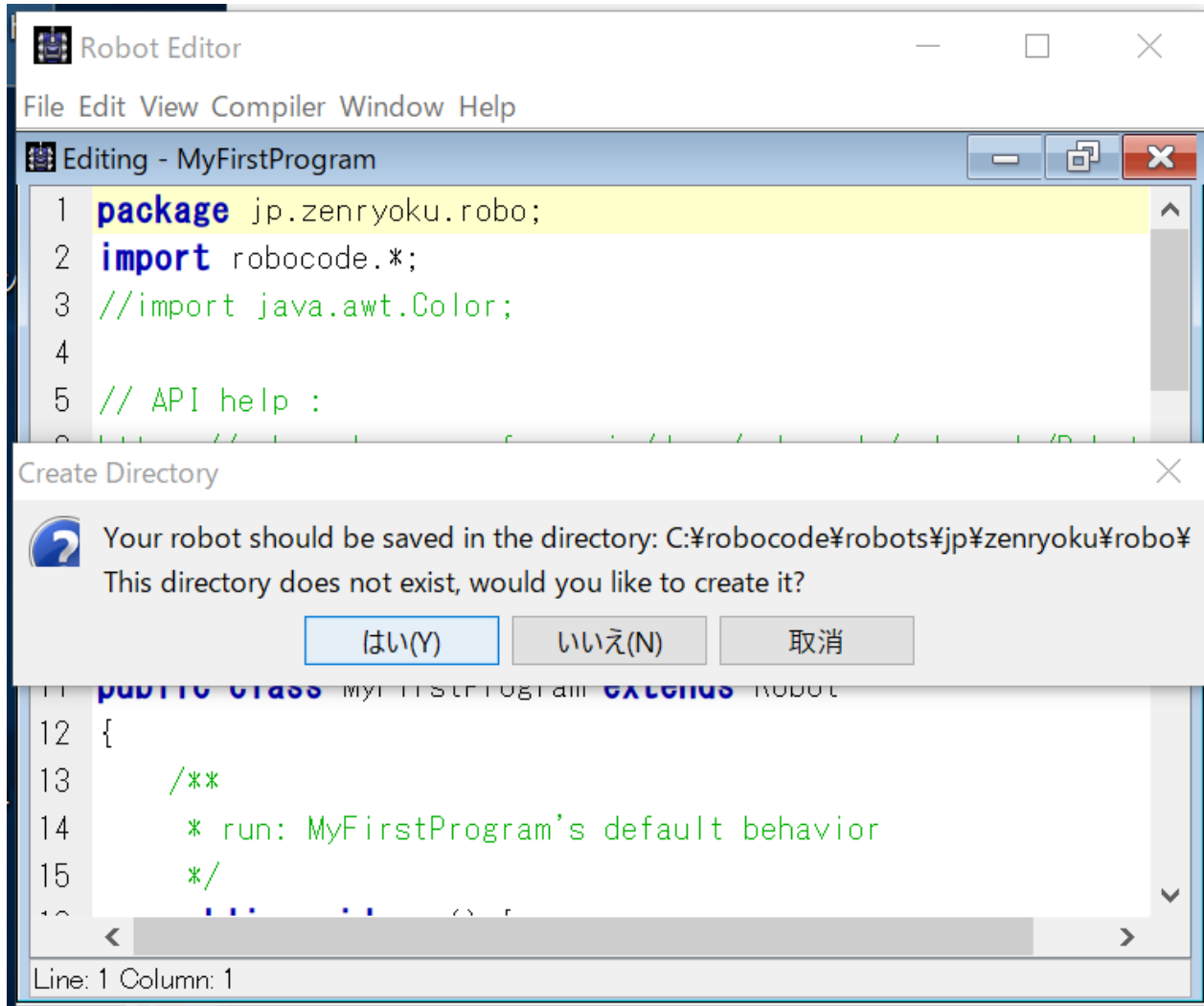
[フィードバック](#)

この様に、分解してみていけば、ある程度は何とかなります。

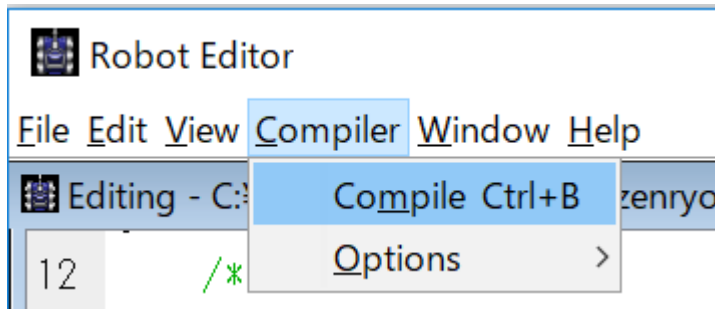
Javaの基本などを学習するのによいと思います。

[Javaの学習フロー\(こんな順序で学習するとよいのでは?\)](#)

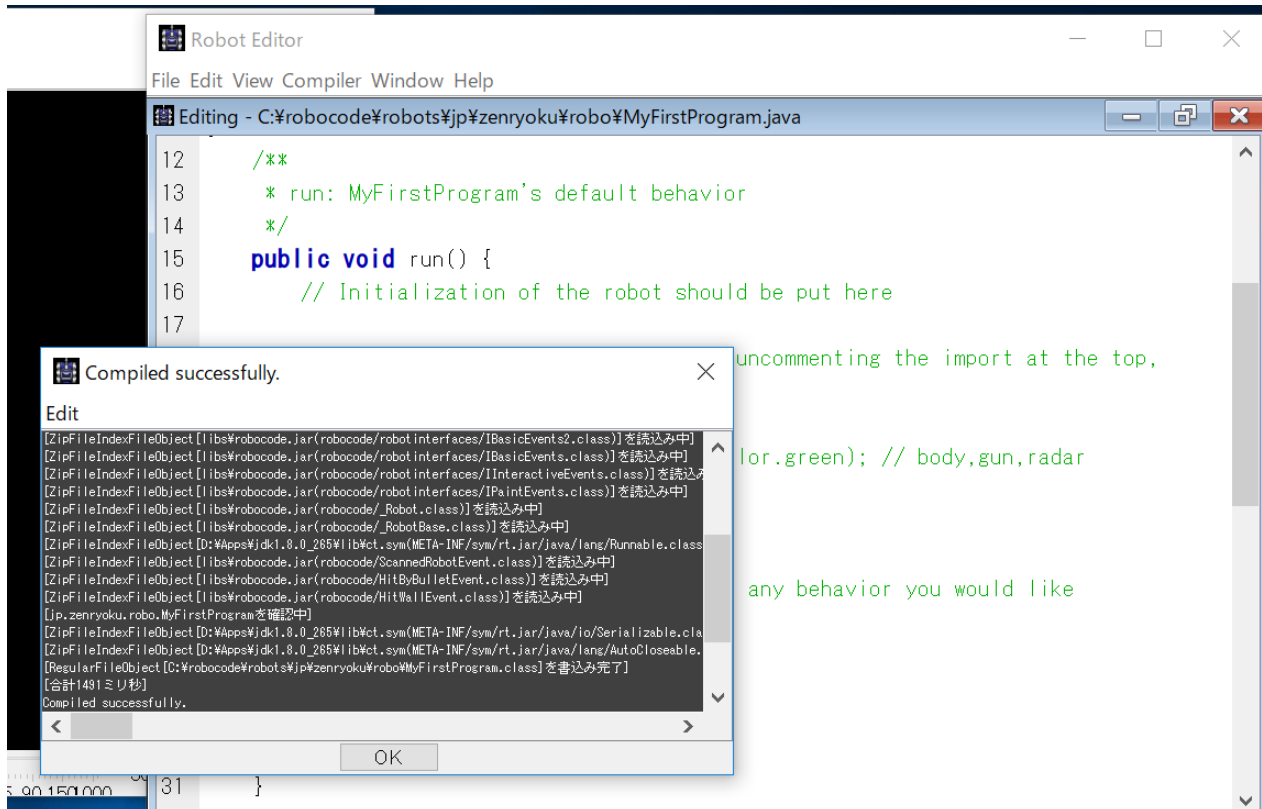
そして、作成したコードは保存しないと残りません。



作成したコードは、コンパイルして。。



結果の確認。。



プログラムの実行をしてみます。ここでのポイントは、デフォルトの動き「runメソッドの動きに注目する」ということです。

他のメソッドのコメントを見ると以下のようにになっています。実装されているメソッドは4つありました。

1. run: MyFirstProgram's default behavior -> MyFirstProgramのデフォルトの動作
2. onScannedRobot: What to do when you see another robot -> 別のロボットを見たときの対処方法
3. onHitByBullet: What to do when you're hit by a bullet -> 弾丸に当たったときの対処方法
4. onHitWall: What to do when you hit a wall -> 壁にぶつかったときの対処方法



JavaDocをみる

JavaDocを見てどんなメソッドがあるか確認します。やはり、英語です。そして翻訳機能に頼ります。

<原文>

The basic robot class that you will extend to create your own robots.
Please note the following standards will be used:
heading - absolute angle in degrees with 0 facing up the screen, positive clockwise. $0 \leq \text{heading} < 360$.
bearing - relative angle to some object from your robot's heading, positive clockwise. $-180 < \text{bearing} \leq 180$
All coordinates are expressed as (x,y).
All coordinates are positive.
The origin (0,0) is at the bottom left of the screen.
Positive x is right.
Positive y is up.

<翻訳>

独自のロボットを作成するために拡張する基本的なロボットクラス。
次の標準が使用されることに注意してください。
見出し-絶対角度（度単位）。0は画面を上に向け、正の時計回りです。
 $0 \leq \text{機首方位} < 360$ 。
方位-ロボットの機首方位からのオブジェクトに対する相対角度、時計回りに正。
 $-180 < \text{方位} \leq 180$
すべての座標は（x、y）として表されます。
すべての座標は正です。
原点（0,0）は画面の左下にあります。
正のxは正しいです。
正のyが上がっています。

日本語が少しおかしいですが、大体のところは。。。
いや、補足します。

正のxは正しいです。
正のyが上がっています。

の部分に関して、「Xは正しい -> 右方向」「yは縦方向」に置き換えて読みましょう。※つたない英語力なので細かい部分はご容赦ください。

ここまでで、基本的なRobocodeの動きを確認する事ができると思います。

あとは、JavaDocを見ながら(翻訳しながら)使えそうなメソッドを使用してみることです。

ちなみに、自分が気になったメソッドは下のものです。

- [fireBullet\(double power\)](#)

弾を打つとエネルギーを消費するが大きなダメージを与えられるようです。

こんなところで。。。導入編を終わりにしたいと思います。

コードを変更して動かす

自動生成されたコードで動かした場合は、すでに確認したので、次は自分で修正したコードで動かしてみようと思います。

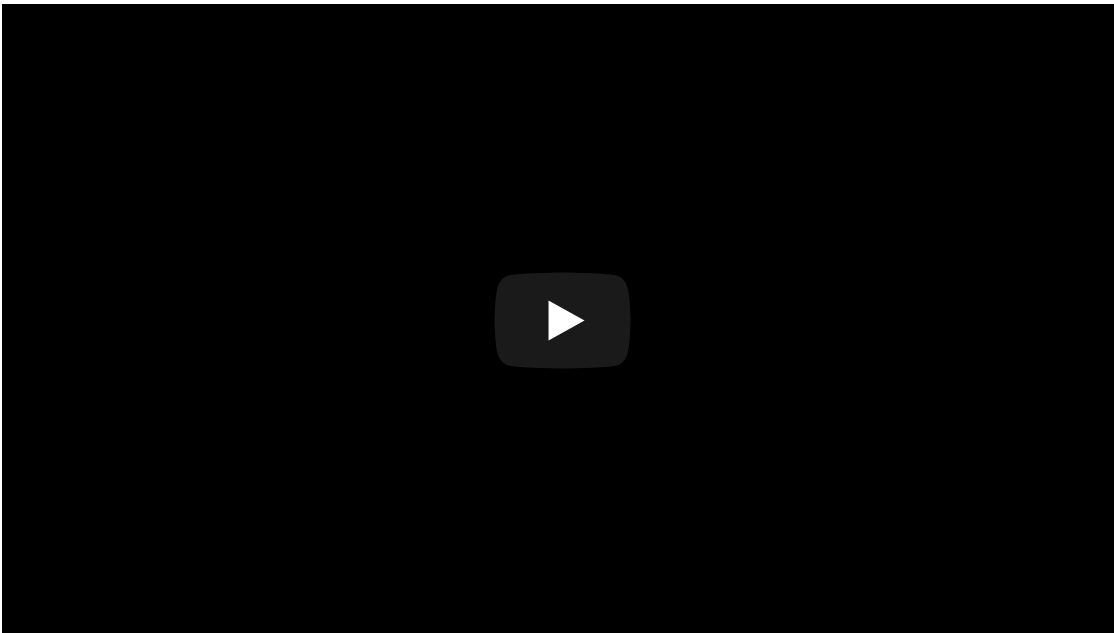
```
public void run() {  
    // Initialization of the robot should be put here  
  
    // After trying out your robot, try uncommenting the import at the top,  
    // and the next line:  
  
    // setColors(Color.red,Color.blue,Color.green); // body,gun,radar  
  
    // Robot main Loop  
    while(true) {  
        // Replace the next 4 lines with any behavior you would like  
        ahead(100);  
        turnGunRight(360);  
        back(100);  
        turnGunRight(360);  
    }  
}
```

このコードは、デフォルトの動きを設定しています。なので、この動きを変えてみようと思います。

```
// Robot main loop  
while(true) {  
    // Replace the next 4 lines with any behavior you would like  
    ahead(1000);  
    turnGunRight(90);  
    back(500);  
    turnGunRight(90);  
}
```

1. aheadの引数を100から1000に修正
2. turnGunRightの引数を360から90に修正
3. backも同様に500に修正

実行してみます。



これでは、戦いになりませんね。。。

次に考えること

ある程度の操作ができることはわかりました。
他のメソッドに関しても同様に修正して動かしてみましょう。

そして、自分が気になったのは、壁にぶつかったときの動きを何とかしたいと思ったので、そのように実装します。

壁にぶつかったとき

コメントを頼りにします。

修正するコードは下のものです。

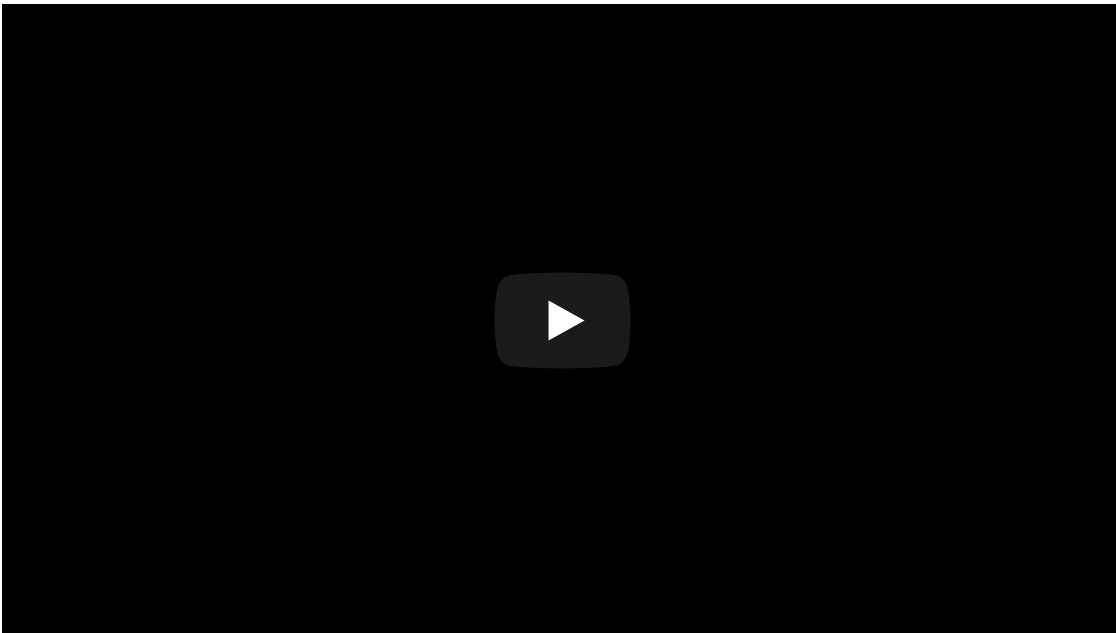
```
/**
 * onHitWall: What to do when you hit a wall
 */
public void onHitWall(HitWallEvent e) {
    // Replace the next line with any behavior you would like
    back(20);
}'''
```

コードを言葉にすると「壁にぶつかったとき(onHitWall())、20下がる(back(20))」とあるので、向きを変える方向で、修正したいと思います。

それっぽいメソッドを[JavaDoc](<https://robocode.sourceforge.io/docs/robocode/robocode/Robot.html>)から探します。
次のようなメソッドがあったので、早速実装します。
![<http://zenryokuserice.com/wp/wp-content/uploads/2021/01/robocode-14.png>]

```
/**
```

- onHitWall: What to do when you hit a wall
- ```
*/
public void onHitWall(HitWallEvent e) {
 // Replace the next line with any behavior you would like
 turnRight(90);
}'''
```



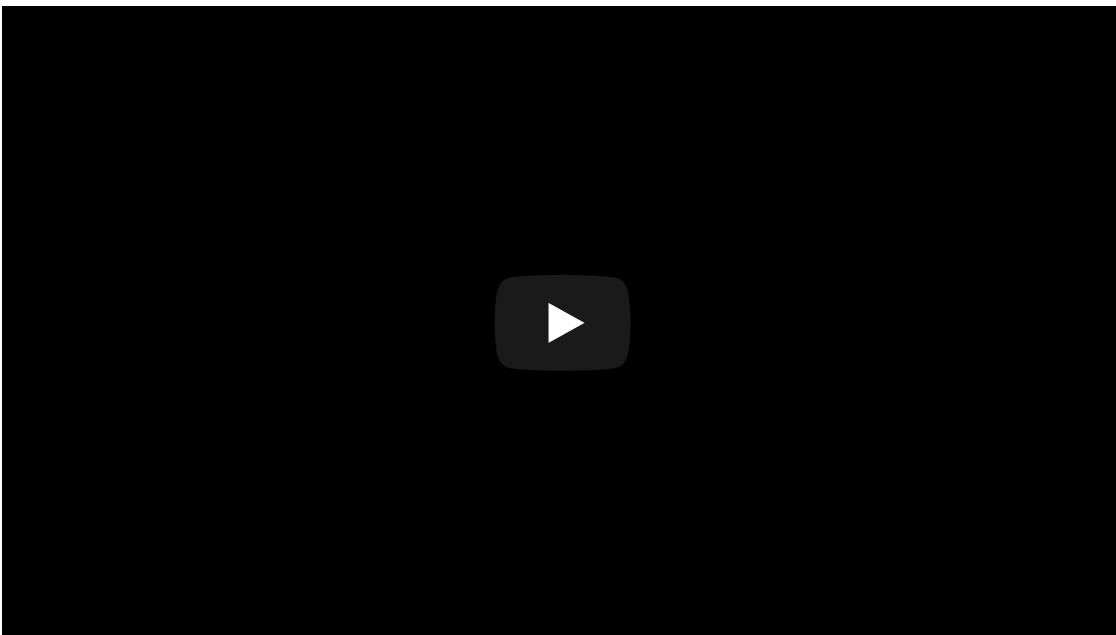
しかし、右に曲がるだけではイマイチでした。

## 情報収集をする

---

実装した各イベントに対応するメソッドに渡されている引数の値を表示します。

コンソールに出力する方法はこちらの動画にアップしております。



各メソッドの処理内容が知りたいときには[JavaDocを見るのが一番](#)です。

## スキャンしたとき

< 引数にあるクラス(Event) >

[ScannedRobotEvent](#)

< 使用できるメソッド例 >

```
out.println("Bearing: " + e.getBearing()); out.println("Distance: " + e.getDistance()); out.println("Energy: " + e.getEnergy());
```

## 弾に当たったとき

< 引数にあるクラス(Event) >

[HitByBulletEvent](#)

< 使用できるメソッド例 >

```
out.println("Bearing: " + e.getBearing());
out.println("Bearing#power: " + e.getBullet().getPower());
out.println("Heading: " + e.getHeading());
```

### ### 壁に当たったとき

< 引数にあるクラス(Event) >

[\[HitWallEvent\]](https://robocode.sourceforge.io/docs/robocode/robocode/HitWallEvent.html) (<https://robocode.sourceforge.io/docs/robocode/robocode/HitWallEvent.html>)

< 使用できるメソッド例 >

```
out.println("Bearing: " + e.getBearing());
out.println("BearingRadians: " + e.getBearingRadians());
...
```

< 出力内容(一部抜粋) >

```
*** onScannedRobot ***
Bearing: -33.82512154957675
Distance: 281.41020026543913
Energy: 0.20000000000001172
*** onHitByBullet ***
Bearing: -33.47367916773932
Bearing#power: 1.0
Heading: 121.41321926877121
*** onHitWall ***
Bearing: 115.11310156348944
BearingRadians: 2.009102634465523
```

## APIを読んでいて発見したこと

1. `turnRight(e.getBearing())` を実行すれば、ロボットは弾丸が飛んできた方向を向くことになります。

## 実行可能なメソッド(コマンド)

| 返り値    | メソッド名                                 | 振る舞い(処理の内容)                              |
|--------|---------------------------------------|------------------------------------------|
| void   | <code>ahead(double distance)</code>   | ロボットを前方に移動させます。                          |
| void   | <code>back(double distance)</code>    | ロボットを後方に移動させます。                          |
| void   | <code>doNothing()</code>              | このロボットの今回の順番では、何も動作を行いません。               |
| void   | <code>finalize()</code>               | システムによって呼び出され、ロボットのクリーンアップ (終結処理) を行います。 |
| void   | <code>fire(double power)</code>       | 弾丸を発射します。                                |
| Bullet | <code>fireBullet(double power)</code> | 弾丸を発射します。                                |

| 返り値    | メソッド名                                         | 振る舞い(処理の内容)                                                                                 |
|--------|-----------------------------------------------|---------------------------------------------------------------------------------------------|
| double | getBattleFieldHeight()                        | 現在のバトルフィールドの高さを取得します。                                                                       |
| double | getBattleFieldWidth()                         | 現在のバトルフィールドの幅を取得します。                                                                        |
| double | getEnergy()                                   | ロボットの現在のエネルギーを戻します。                                                                         |
| double | getGunCoolingRate()                           | 大砲の冷却速度を戻します。                                                                               |
| double | getGunHeading()                               | 大砲の向きを 360 度形式で戻します。                                                                        |
| double | getGunHeat()                                  | 大砲の現在の熱さを戻します。                                                                              |
| double | getHeading()                                  | ロボットの現在の向きを 360 度形式で戻します。                                                                   |
| double | getHeight()                                   | ロボットの高さを戻します。                                                                               |
| String | getName()                                     | ロボットの名前を戻します。                                                                               |
| int    | getNumRounds()                                | 現在のバトルのラウンド数を戻します。                                                                          |
| int    | getOthers()                                   | 敵ロボットの残り台数を戻します。                                                                            |
| double | getRadarHeading()                             | レーダーの向きを 360 度形式で戻します。                                                                      |
| int    | getRoundNum()                                 | 現在のラウンドが、バトルの第何ラウンドかを戻します (1～getNumRounds() の範囲)                                            |
| long   | getTime()                                     | 現在のゲーム時刻を戻します。                                                                              |
| double | getVelocity()                                 | このロボットの移動速度を戻します。                                                                           |
| double | getWidth()                                    | ロボットの幅を戻します。                                                                                |
| double | getX()                                        | ロボットの X 座標の位置を戻します。                                                                         |
| double | getY()                                        | ロボットの Y 座標の位置を戻します。                                                                         |
| void   | onBulletHit(BulletHitEvent event)             | このロボットが発射したいずれかの弾丸が他のロボットに当たったとき、このメソッドが呼び出されます。                                            |
| void   | onBulletHitBullet(BulletHitBulletEvent event) | このロボットが発射したいずれかの弾丸が他の弾丸に当たったとき、このメソッドが呼び出されます。                                              |
| void   | onBulletMissed(BulletMissedEvent event)       | このロボットが発射したいずれかの弾丸がはずれたとき (壁に当たったとき)、このメソッドが呼び出されます。                                        |
| void   | onDeath(DeathEvent event)                     | このメソッドは、ロボットが死んだときに呼び出されます。<br>このイベントが通知されるようにするには、ロボットのコードでこのメソッドをオーバーライドする必要があります。        |
| void   | onHitByBullet(HitByBulletEvent event)         | このロボットに弾丸が命中したとき、このメソッドが呼び出されます。                                                            |
| void   | onHitRobot(HitRobotEvent event)               | このロボットが他のいずれかのロボットと衝突したとき、このメソッドが呼び出されます。                                                   |
| void   | onHitWall(HitWallEvent event)                 | このロボットが壁に衝突したとき、このメソッドが呼び出されます。                                                             |
| void   | onRobotDeath(RobotDeathEvent event)           | このメソッドは、他のいずれかのロボットが死んだときに呼び出されます。<br>このイベントが通知されるようにするには、ロボットのコードでこのメソッドをオーバーライドする必要があります。 |



| 返り値  | メソッド名                                                          | 振る舞い(処理の内容)                                  |
|------|----------------------------------------------------------------|----------------------------------------------|
| void | onScannedRobot(ScannedRobotEvent event)                        | このロボットが他のロボットを発見したとき、このメソッドが呼び出されます。         |
| void | onWin(WinEvent event)                                          | このロボットがバトルに勝ったとき、このメソッドが呼び出されます。             |
| void | resume()                                                       | stop() による停止中の動作があれば、その動作を再開します。             |
| void | run()                                                          | 各ロボットのメイン・メソッド。                              |
| void | scan()                                                         | 他のロボットを探します。                                 |
| void | setAdjustGunForRobotTurn(boolean newAdjustGunForRobotTurn)     | ロボットが回転するときに、大砲が自動的に逆方向に回転するように設定します。        |
| void | setAdjustRadarForGunTurn(boolean newAdjustRadarForGunTurn)     | 大砲が回転するときに、レーダーが自動的に逆方向に回転するように設定します。        |
| void | setAdjustRadarForRobotTurn(boolean newAdjustRadarForRobotTurn) | ロボットが回転するときに、レーダーが自動的に逆方向に回転するように設定します。      |
| void | setColors(Color robotColor, Color gunColor, Color radarColor)  | このメソッドは、ロボットの色を設定するために呼び出します。                |
| void | stop()                                                         | 動作をすべて停止し、後で resume() 呼び出しを使って再開できるよう、保存します。 |
| void | stop(boolean overwrite)                                        | 動作をすべて停止し、後で resume() 呼び出しを使って再開できるよう、保存します。 |
| void | turnGunLeft(double degrees)                                    | ロボットの大砲を回転させます。                              |
| void | turnGunRight(double degrees)                                   | ロボットの大砲を回転させます。                              |
| void | turnLeft(double degrees)                                       | ロボットを回転させます。                                 |
| void | turnRadarLeft(double degrees)                                  | ロボットのレーダーを回転させます。                            |
| void | turnRadarRight(double degrees)                                 | ロボットのレーダーを回転させます。                            |
| void | turnRight(double degrees)                                      | ロボットを回転させます。                                 |