**情報システム学科**

**オープンキャンパス**

**Javaプログラミング体験**

**～Robocodeで遊ぼう～**

**本日の体験メニューについて**

プログラミング言語のJavaを使ってゲームをします。

そして、皆でバトルします。

Robocode

これから、プレーするゲームの名前です。

Roboodeは、Javaというプログラミング言語でロボット戦車を作り、戦わせます。

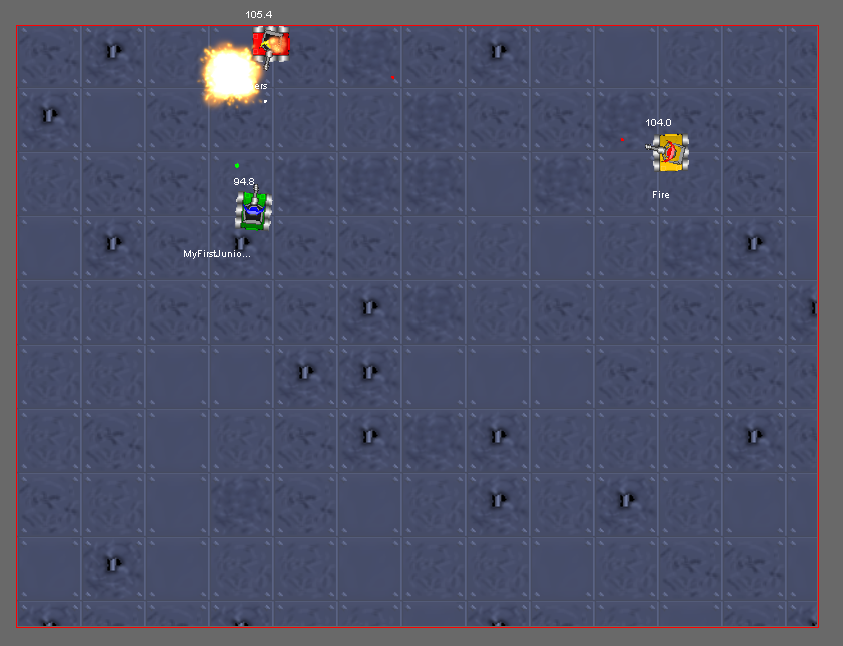
自分で作成したロボ(戦車)と友達が作成したロボ(戦車)で対戦することもできます。

【ゲームの内容】

・仮想の戦場で、複数のロボット戦車を戦わせ勝敗を競うシュミレーションゲームです。

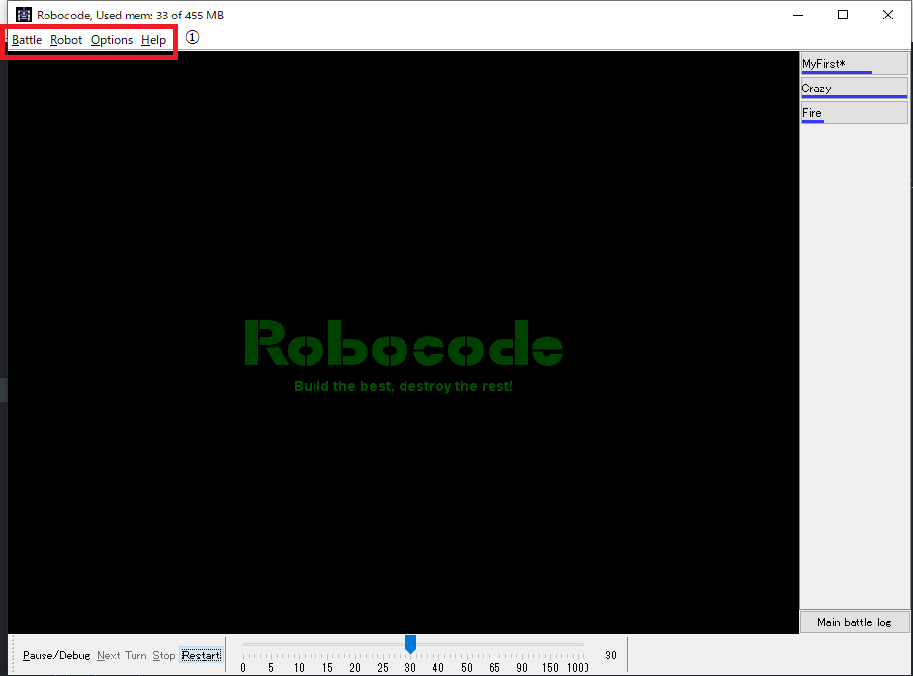
・友達の作成したロボとバトルが可能。

＜ゲームのイメージ＞



★初期画面について★

Robocodeを起動すると下のような画面が見ることができます。



★メニュー★

Battle:  
 1. 新規バトル(New)

2. バトルを開く(Open)

\* Robot:

1. ソースエディタ(Source editor)

2. 他のロボ、チーム取り込み(import robot or team)

3. JARファイルの作成(Package robot or team)

\* Options:

1. 環境設定(Preference),

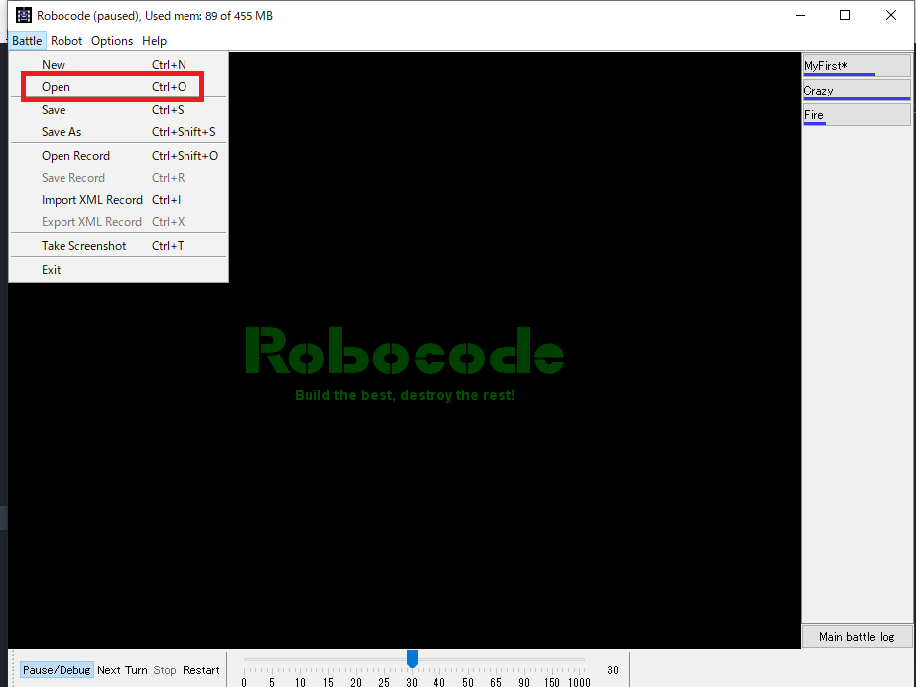
2. Roboのキャッシュをクリア(Clean robo cache)

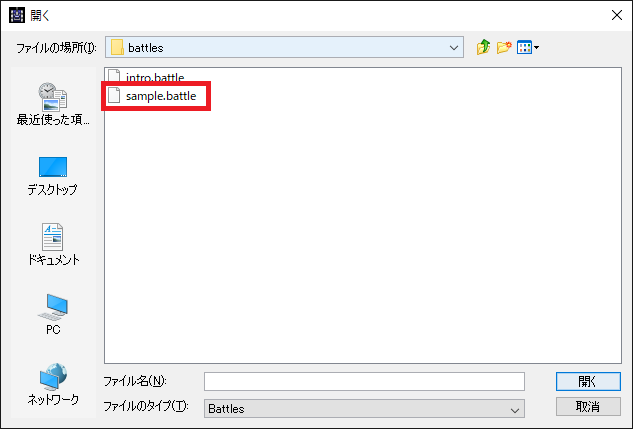
\* Help: ヘルプ

★★動かしてみよう★★

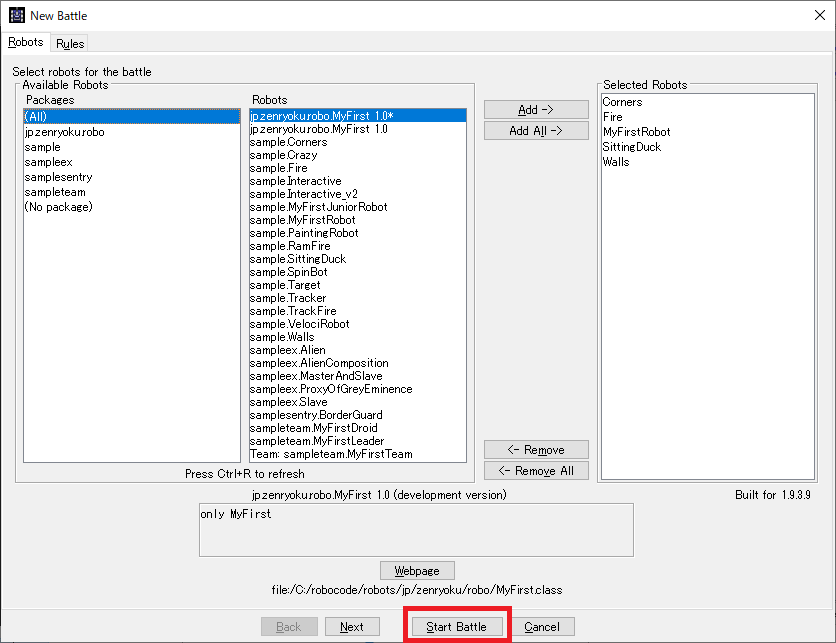
まずは、サンプルバトルを動かします。下の手順で動かしてください。

**★Step1★** メニューから「Battle」をクリック、「Open」をクリック

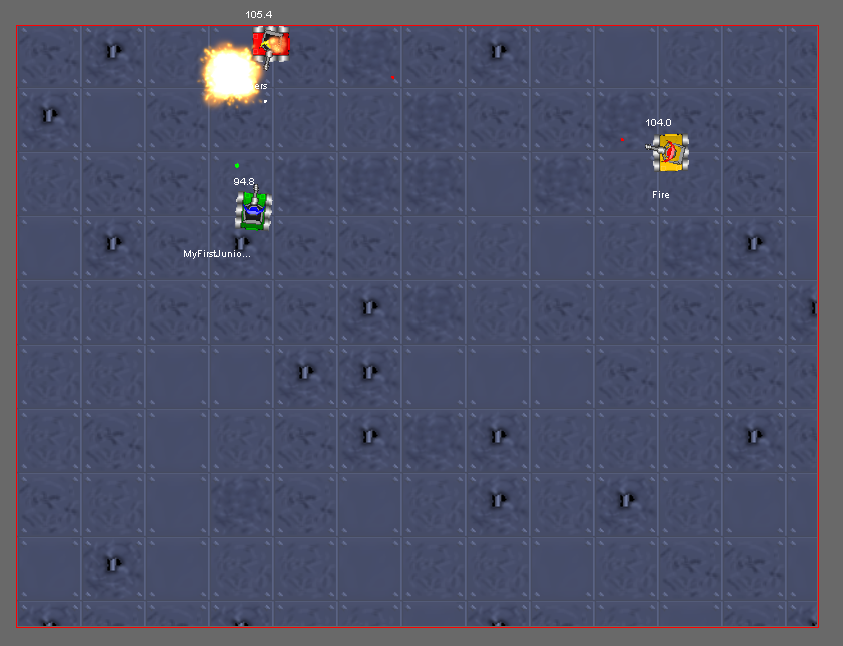


**★Step2★** sample.battleファイルをクリックして開く

**★Step3★** RobotsをAddをクリックして、Start Battleをクリック

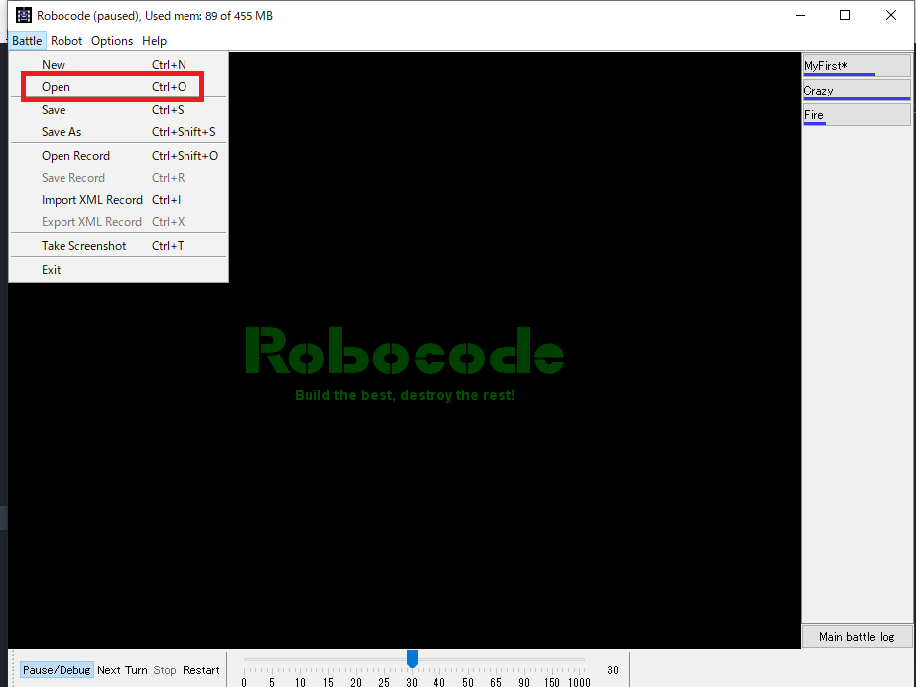


**★Step4★** Stopでバトルを終了する



★★観察する★★

動いている様子をみて、どんな動きをしているか観察してください。



★観察ポイント★

1. 通常時の動きはどんな動きか？
2. 弾を打っているときはどんな時か？
3. 敵の弾に当たった時はどんな動きか？
4. 壁にぶつかったときはどんな動きをするか？

上の観察ポイントは、これから作成するRoboの動きを決めるのに重要な部分です。

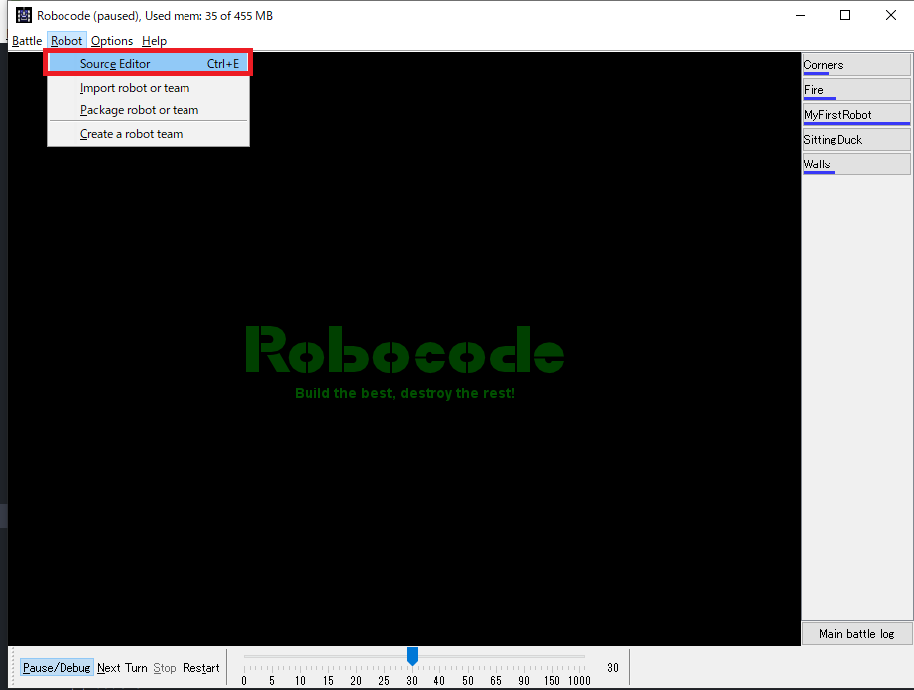
「通常時」というのが少しわかりずらいと思いますが、そこは実際に動かして、予想してみてください。

プログラムのコードを見てみるとはっきりします。

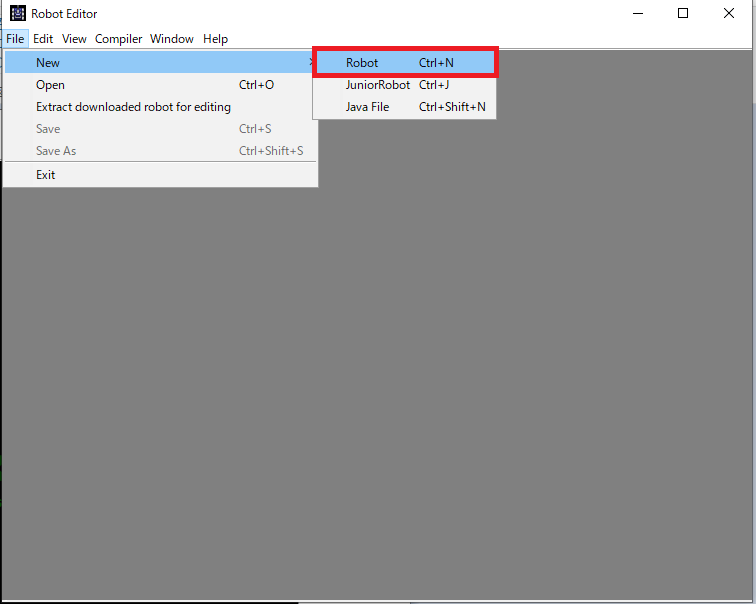
★★オリジナル・ロボを作ろう★★

オリジナルのロボを作ります。下の手順でロボの作成準備をしてください。

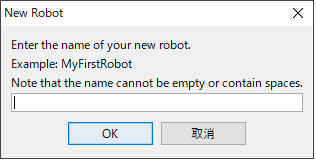
**★Step1★** メニューから「Robo」をクリックし、「Source Editor」をクリック



**★Step2★** ソースエディタを開き、New -> Roboをクリック



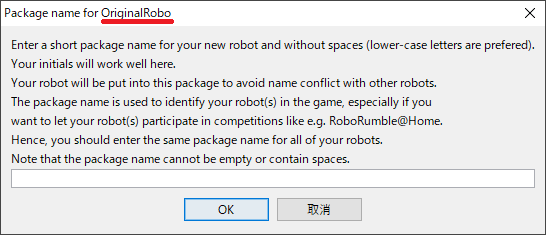
**★Step3★** ロボの名前を入力する



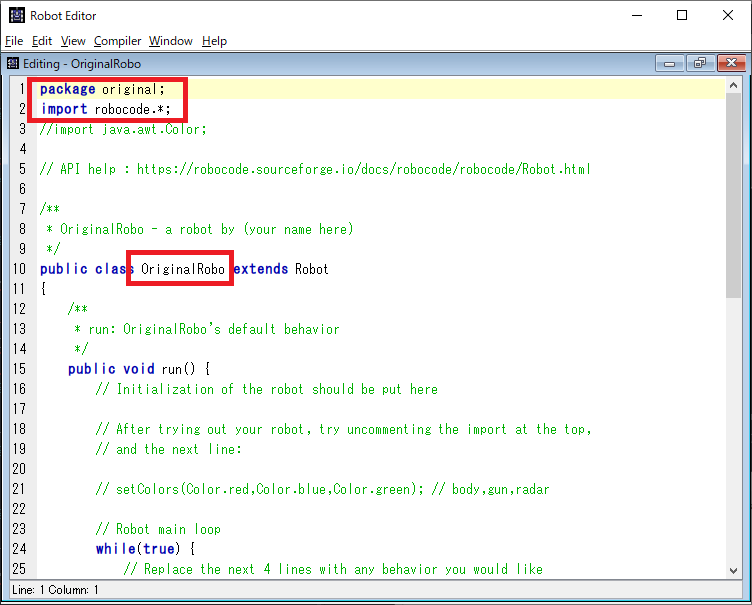
**★Step4★** ロボのファイルを配置するパッケージの名前を入力する

※ロボの名前には「OrignalRobo」という名前をつけたので、赤線部にロボの名前

が表示されています。



**★Step5★** テンプレートのロボが作成されるので、オリジナルのロボに改造しましょう。



★★プログラミングについて★★

Roboodeでは、Javaというプログラミング言語を使ってロボを作ります。

このゲームは、オープン・ソース・ソフトウェア(OSS)という部類に入り、基本的に、ダウンロード、改造ともに自由にやってよいものです。

ですが「オープン・ソース：ライセンス」というものもあり、ある程度の規制があります。

製作者に迷惑が掛からないようにするためです。

そして、それぞれのライセンスで規制する内容が異なるので注意して下さい。

有名なのは、下のものがあります。

* Apache License(アパッチライセンス) ※Android OS
* MIT license(MITライセンス) ※MIT = マサチューセッツ工科大学
* GNU (General Public License)

インターネット上にはオープン・ソース・ソフトウェアが沢山あります。ゲームから仕事用のアプリケーションまで、色々なソフト(アプリ)があります。

★★Javaプログムを読む★★

プログラム言語の処理は、上から下に流れます。どのプログラムでも同じことです。

例として、作成したRoboのコードを見てみましょう。

作成されたRoboのコードは、初めにrun()メソッドが動きます。

★補足★

* 「//」と書いている行はコメント行といいます。この行はプログラムとして読み込まれません。

★★Roboのコード(runメソッド) ★★

ほとんどが、コメント行です。赤枠の中にコードが書いてあります。

/\*\*

\* run: OriginalRobo'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);

}

}

★処理の内容(命令の内容★

**while(true) { … }** は無限ループを示します。

ahead(100); 　　　はRoboを100前進させます。

turnGunRight(360); はRoboの大砲の向きを右へ360回転させます。

back(100); 　　　 はRoboを100後退させます。

turnGunRight(360); はRoboを大砲の向きを右へ360回転させます。

そして、「（）」の中を「引数」と呼びます。

自分のロボを作成できたら、バトルに参加させて、動きを観察してみよう。

★★ロボの仕組みを知ろう★★

ロボは次の状態の**動きを定義**することで、どのように動くか決まります。

* 通常時
* 敵を検知した時
* 自分が弾に当たった時
* 壁にぶつかった時

それぞれの動きを定義しているのが、次のメソッドというものです。

* 通常時: → run()
* 敵を検知した時: → onScannedRobot()
* 自分が弾に当たった時: → onHitByBullet()
* 壁にぶつかった時: → onHitWall()

★通常時の動き★

ロボを作成したばかりの状態ではロボは通常時に下のように動きます。

1. 前に100進む

2. 大砲を右へ360度回転

3. 100後退する

2. 大砲を右へ360度回転

★敵を検知した時★

ロボを作成したばかりの状態ではロボは敵を検知した時に下のように動きます。

1. 弾を発射する: fire(1);

★自分が弾に当たった時★

ロボを作成したばかりの状態ではロボは自分が弾に当たった時に下のように動きます。

1. 10後退する: back(10);

★壁にぶつかった時★

ロボを作成したばかりの状態ではロボは壁にぶつかった時に下のように動きます。

1. 20後退する: back(20);

★★ やってみよう ①★★

通常時の動きを、下のように変更してみよう。

1. 前に100進む

2. 大砲を右へ360度回転 -> 50前に進む

3. 100後退する

4. 大砲を右へ360度回転 -> 50後退する

★★やってみよう ②★★

通常時の動きを、下のように変更してみよう。

1. 前に100進む -> 大砲を右へ180度回転

2. 前に50進む

3. 100後退する -> 大砲を左へ180度回転

4. 50後退する

★やってみたことを理解する★

「やってみよう①と②」で、通常時の動きを変更しました。

＜デフォルト(通常時)の動き: Javaのコード＞

1. 前に100進む: ahead(100);

2. 大砲を右へ360度回転: turnGunRight(360);

3. 100後退する: back(100);

4. 大砲を右へ360度回転: turnGunRight(360);

どのように、動きが変わったでしょうか？

この要領で、各状態の動きを定義してみて下さい。

* 通常時: → run()　※ここを実装しました。
* 敵を検知した時: → onScannedRobot()
* 自分が弾に当たった時: → onHitByBullet()
* 壁にぶつかった時: → onHitWall()

次の表に、ロボに命令できることを一覧にしたので、参考にしてください。

＜命令表＞

|  |  |  |
| --- | --- | --- |
| 戻り値 | 命令名(メソッド名) | 振る舞い(処理の内容) |
| void(戻り値なし) | ahead(double distance) | ロボットを前方に移動させます。 |
| void(戻り値なし) | back(double distance) | ロボットを後方に移動させます。 |
| void(戻り値なし) | doNothing() | 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(戻り値なし) | fire(double power) | 弾丸を発射します。 |
| void(戻り値なし) | scan() | 他のロボットを探します。 |
| void(戻り値なし) | setColors(  Color robotColor,  Color gunColor,  Color radarColor) | ロボットの色を設定するために呼び出します。 |
| void(戻り値なし) | stop() | 動作をすべて停止し、呼び出しを使って再開できるよう、保存します。 |
| void(戻り値なし) | turnGunLeft(double degrees) | ロボットの大砲を左に回転させます。 |
| void(戻り値なし) | turnGunRight(double degrees) | ロボットの大砲を右に回転させます。 |
| void(戻り値なし) | turnLeft(double degrees) | ロボットを左に回転させます。 |
| void(戻り値なし) | turnRight(double degrees) | ロボットを右に回転させます。 |

以上で本日の体験は終了です。

自宅に帰ってからでもRobocodeをインストールして遊ぶことができます。是非、最強のロボを作ってみてください。

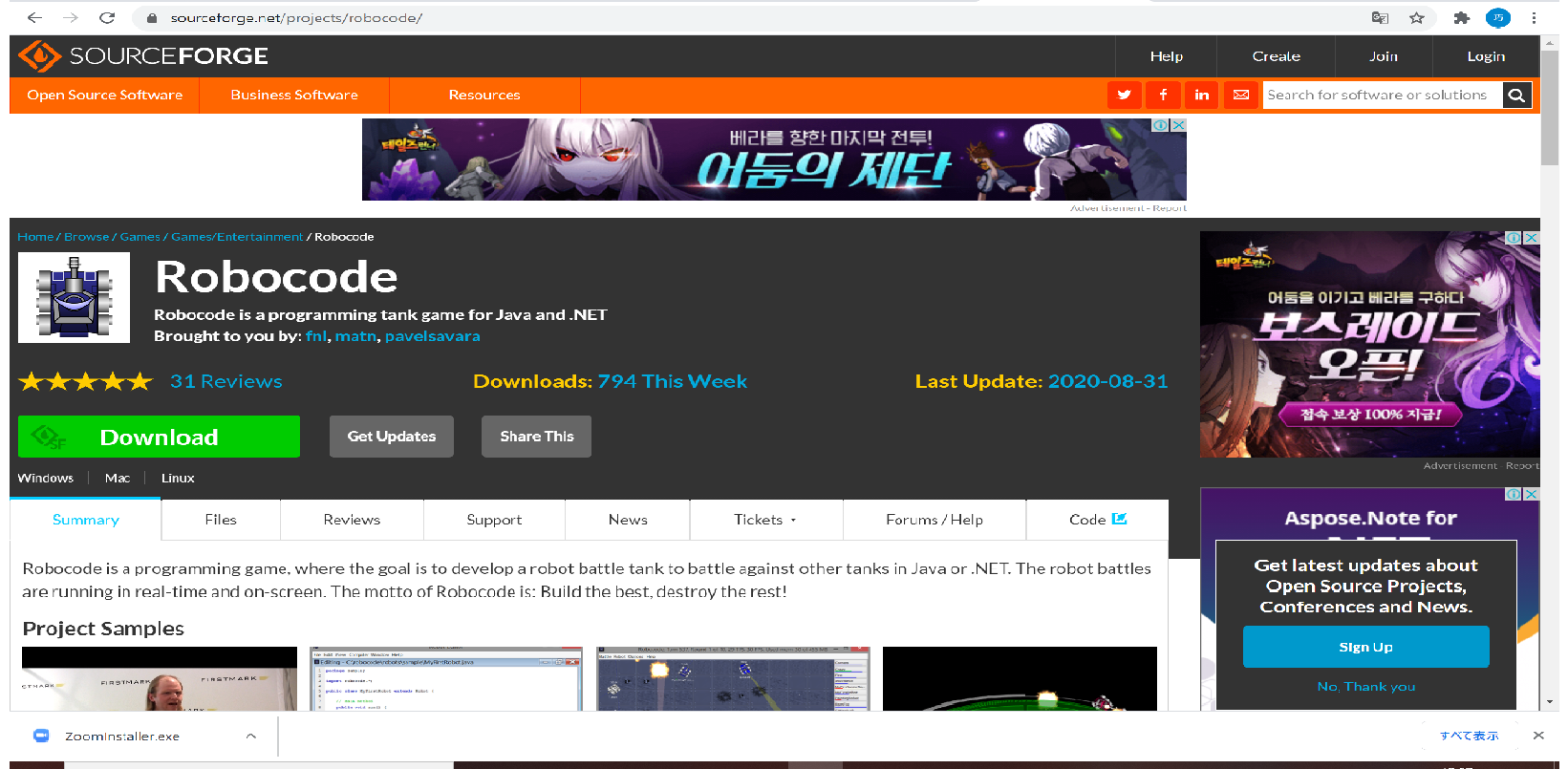
インターネットで検索するときは

「robocode sorceforge.net」と検索するとインストールのためのファイルを見つけることができます。

インストール方法など「robocode インストール」などの文言で検索すれば、インストール方法の書いてある記事を見つけることができます。

【ファイルのダウンロード画面】

URL:　 https://sourceforge.net/projects/robocode/



お疲れ様でした！