

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。インターネットで検索するときは
「robocode インストール」と検索するとインストールの手順を見つけることができます。

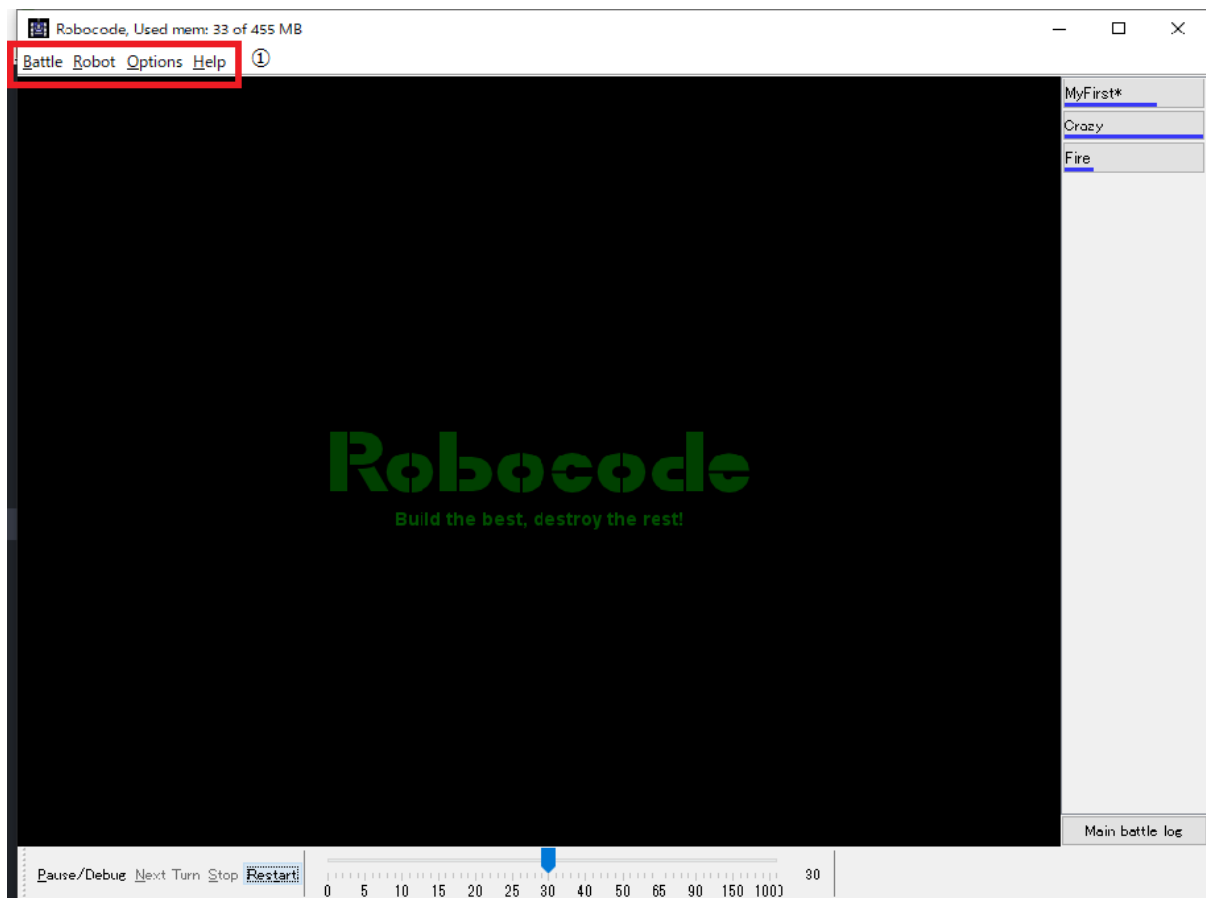
Robocode は、Java というプログラミング言語でロボット戦車を作り、戦わせます。
自分で作成したロボ(戦車)と友達が作成したロボ(戦車)で対戦することもできます。
メニューから「Package robot or team」を選択して JAR ファイルというものを作成し
友達の作成したロボ(戦車)を取り込み戦わせるという方法です。

【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。
- ・JAR ファイルをインポートすることで、友達の作成したロボとバトルが可能。

★初期画面について★

Robocode を起動すると下のような画面を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:

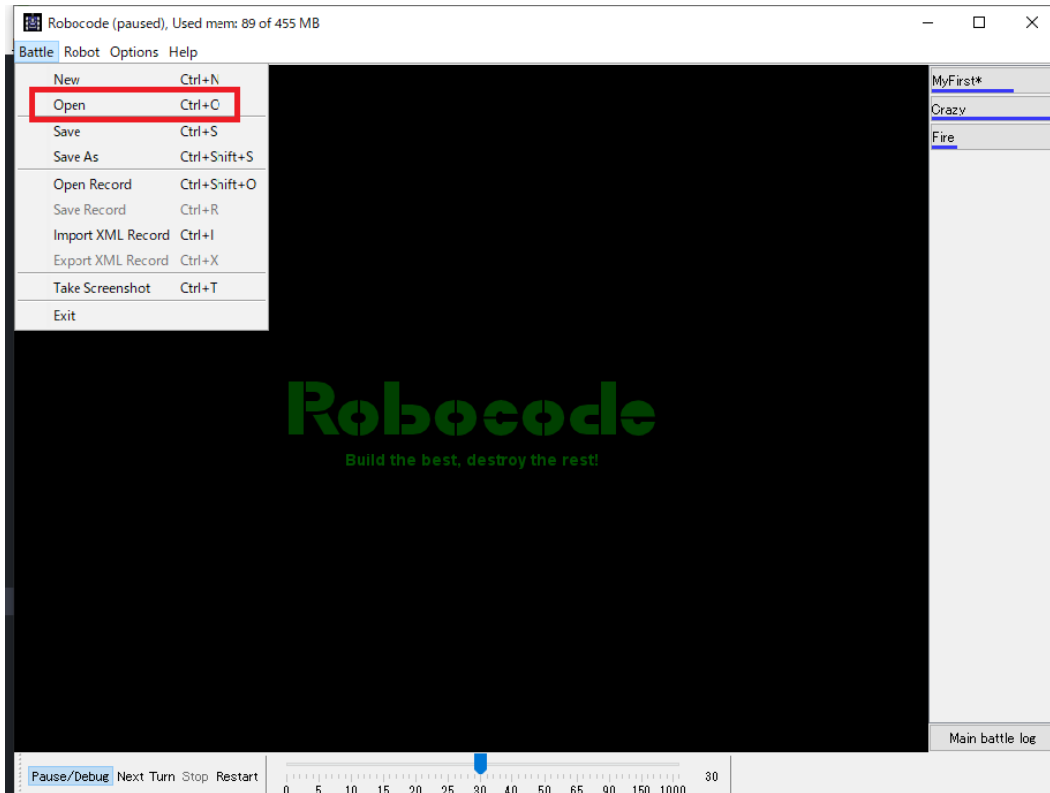
1. 環境設定(Preference),
2. Robo のキャッシュをクリア(Clean robo cache)

* Help: ヘルプ

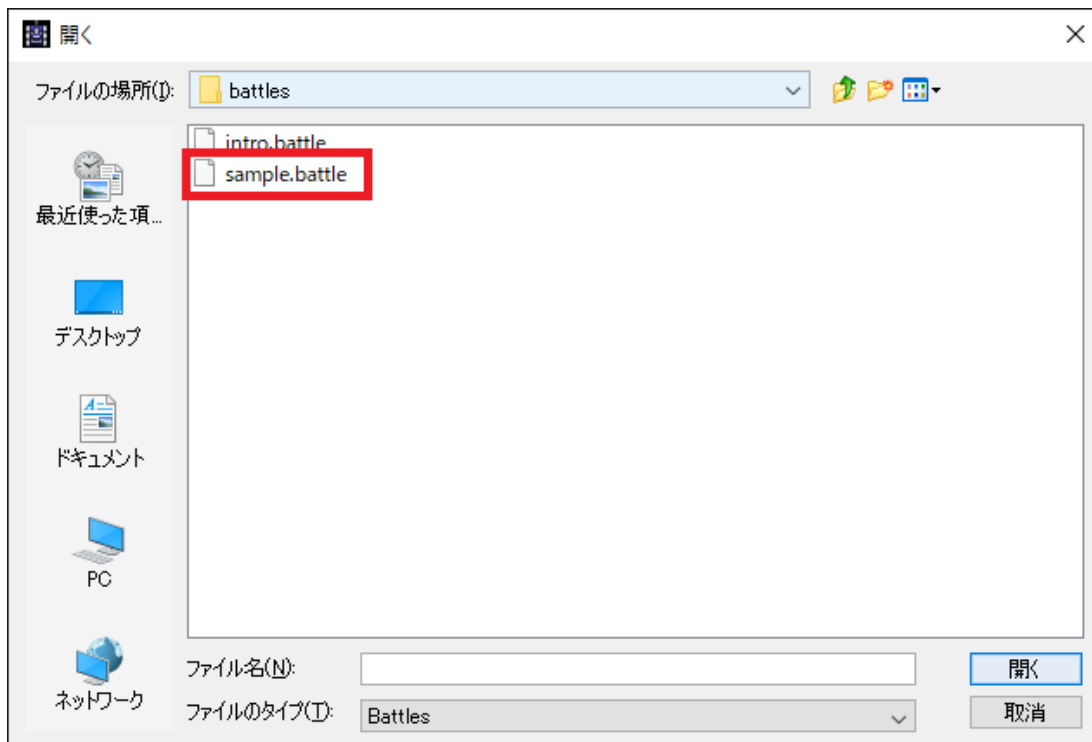
★★動かしてみよう★★

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。

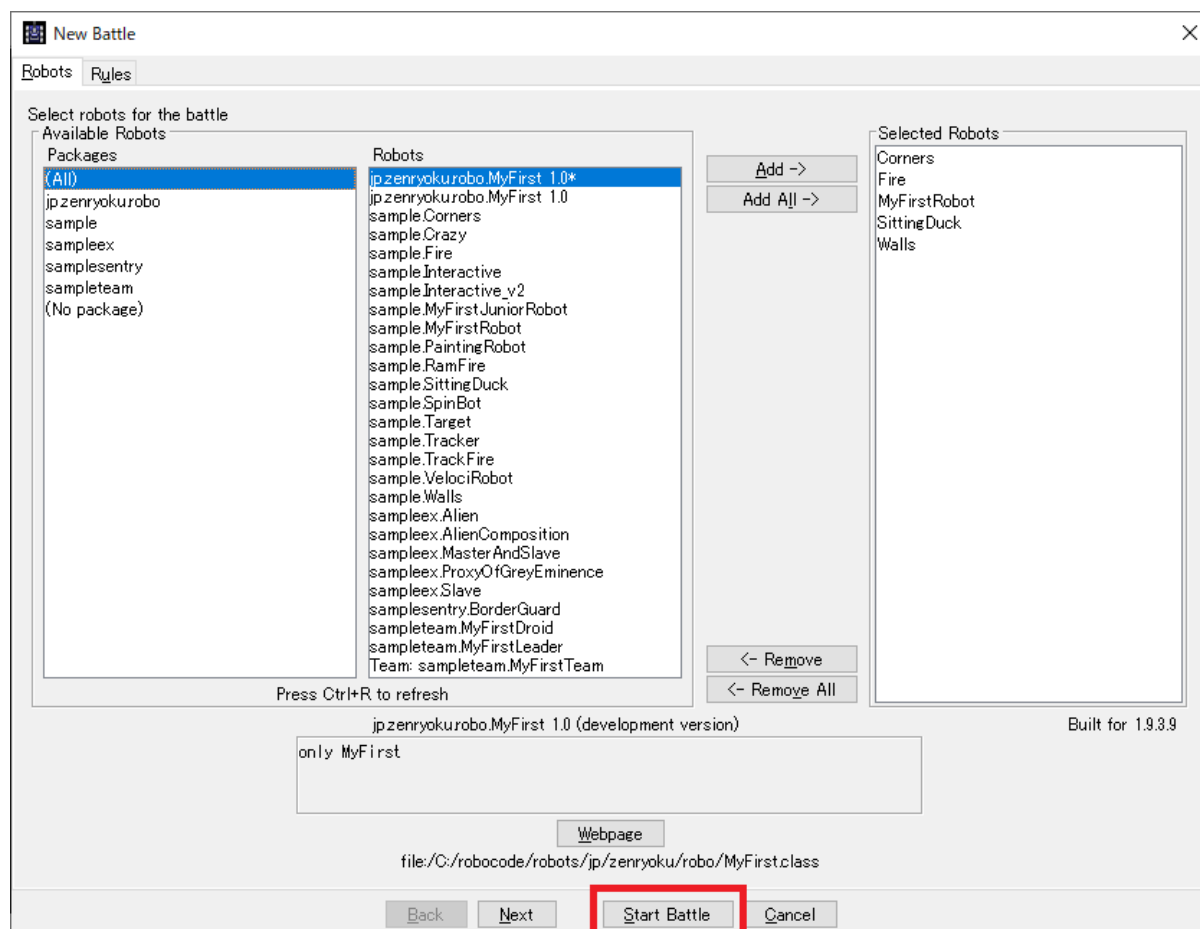
★Step1★ メニューから「Battle」をクリック、「Open」をクリック



★Step2★ sample.battle ファイルをクリック



★Step3★ Robotsを追加して、Start Battle をクリック

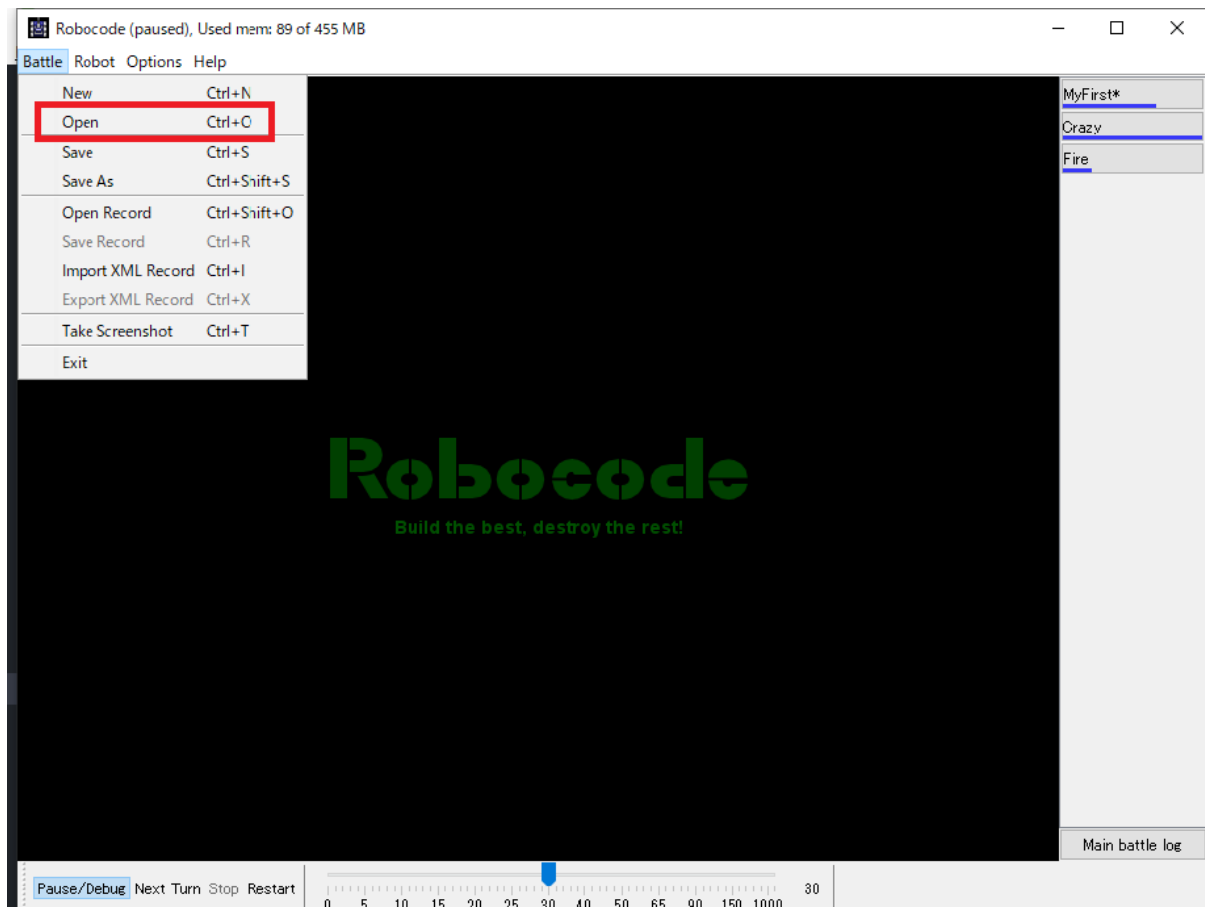


★Step4★ Stop でバトルを終了する



★★観察する★★

動いているのをみて、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？

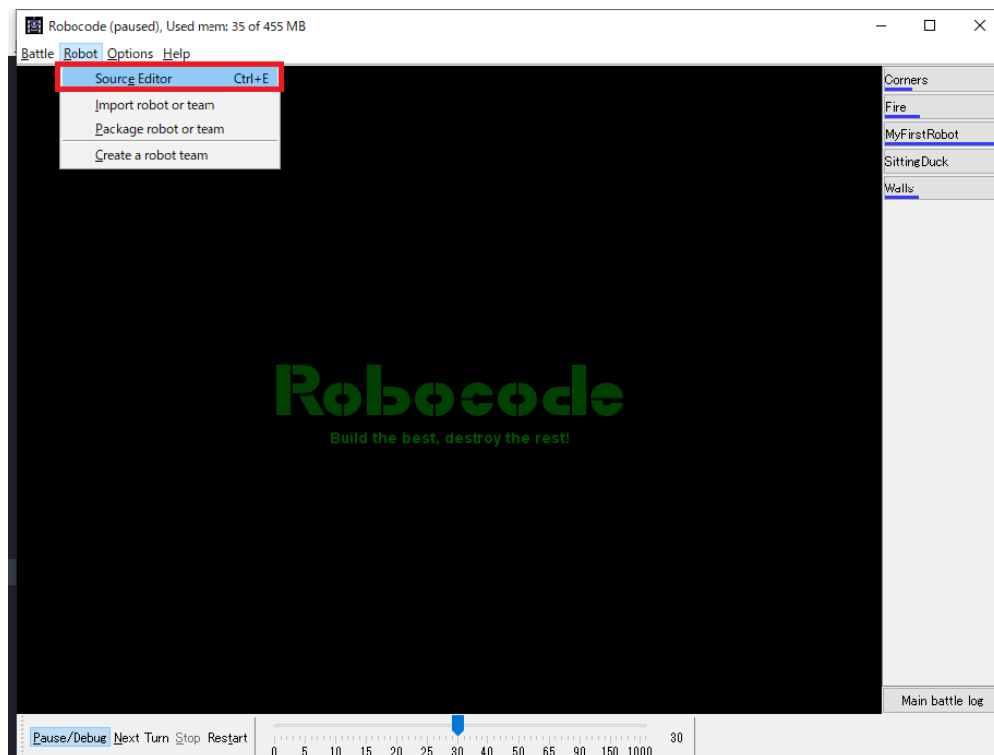
上の観察ポイントは、これから作成する Robo の動きを決めるのに重要な部分です。「通常時」というのが少しわかりずらいと思いますが、そこは実際に動かして、予想してみてください。

プログラムのコードを見てみるとはっきりします。

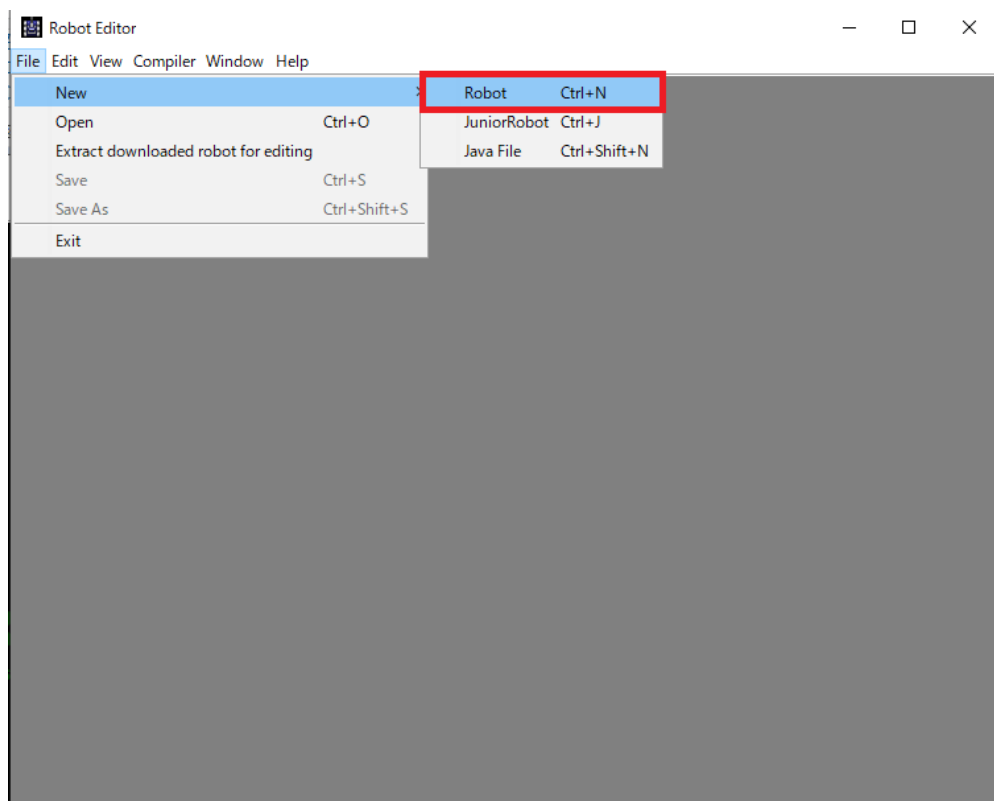
★★オリジナル・ロボを作ろう★★

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。

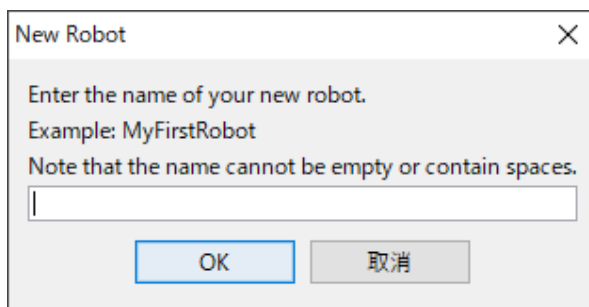
★Step1★ メニューから「Robo」をクリックし、「Source Editor」をクリック



★Step2★ ソースエディタを開き、New -> Robo をクリック



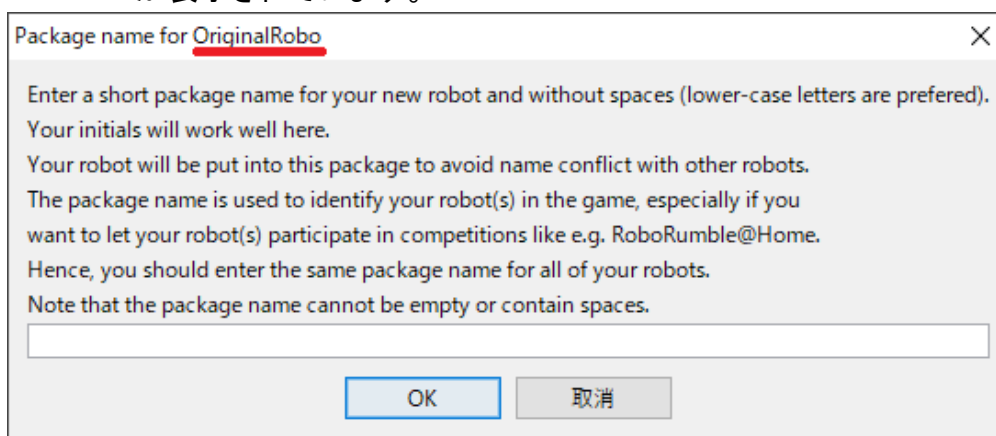
★Step3★ ロボの名前を入力する



A dialog box titled "New Robot" with a close button (X) in the top right corner. The text inside says: "Enter the name of your new robot. Example: MyFirstRobot. Note that the name cannot be empty or contain spaces." Below the text is a text input field. At the bottom are two buttons: "OK" and "取消" (Cancel).

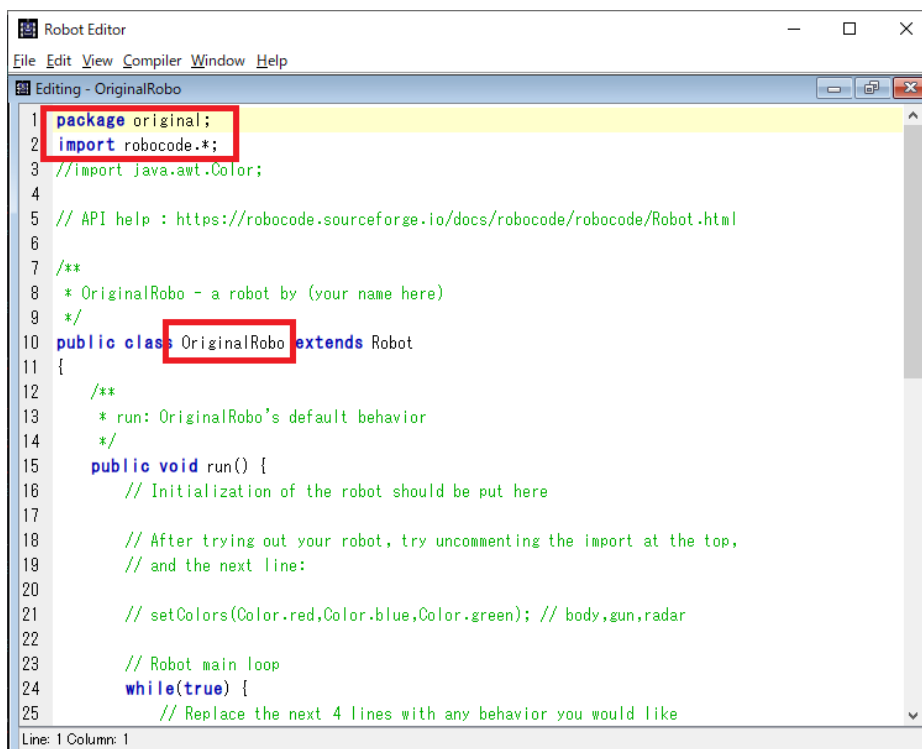
★Step4★ ロボのファイルを配置するパッケージの名前を入力する

※ロボの名前には「OriginalRobo」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。



A dialog box titled "Package name for OriginalRobo" with a close button (X) in the top right corner. The text inside says: "Enter a short package name for your new robot and without spaces (lower-case letters are preferred). Your initials will work well here. Your robot will be put into this package to avoid name conflict with other robots. The package name is used to identify your robot(s) in the game, especially if you want to let your robot(s) participate in competitions like e.g. RoboRumble@Home. Hence, you should enter the same package name for all of your robots. Note that the package name cannot be empty or contain spaces." Below the text is a text input field. At the bottom are two buttons: "OK" and "取消" (Cancel).

★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。



A screenshot of the "Robot Editor" window. The title bar says "Robot Editor". The menu bar includes "File", "Edit", "View", "Compiler", "Window", and "Help". The main window is titled "Editing - OriginalRobo". It shows a Java code editor with the following code:

```
1 package original;
2 import robocode.*;
3 //import java.awt.Color;
4
5 // API help : https://robocode.sourceforge.io/docs/robocode/robocode/Robot.html
6
7 /**
8  * OriginalRobo - a robot by (your name here)
9  */
10 public class OriginalRobo extends Robot
11 {
12     /**
13      * run: OriginalRobo'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) {
25             // Replace the next 4 lines with any behavior you would like
```

At the bottom of the window, it says "Line: 1 Column: 1".

★★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。

有名なものは、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 プログラムを読む★★

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
- 敵をスキャンした時
- 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()`
- 敵をスキャンした時: → `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

★敵をスキャンした時★

ロボを作成したばかりの状態ではロボは敵をスキャン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ください。

- 通常時: → `run()` ※ここを実装しました。
- 敵をスキャンした時: → `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。

お疲れ様でした！