

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

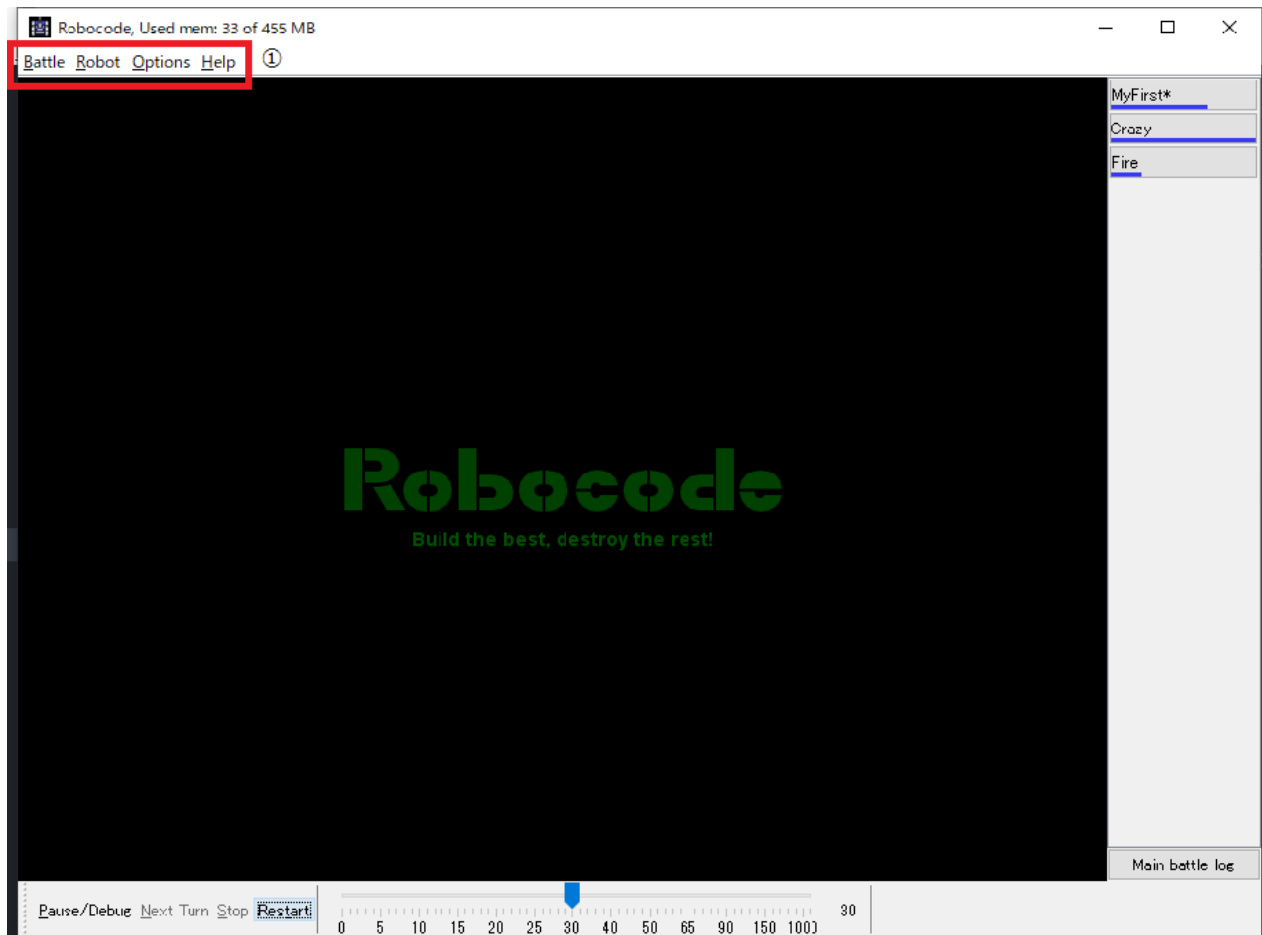
これから、プレーするゲームの名前です。インターネットで検索するときは「robocode インストール」と検索するとインストーラー(jarファイル)をダウンロードできます。

準備済みの設定

1. 文字の大きさを変更してあります。

初期画面について

起動すると下のような画面が見れます。



①メニュー

使うもの、のみを記載しています。

- 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. ほかのロボ、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:

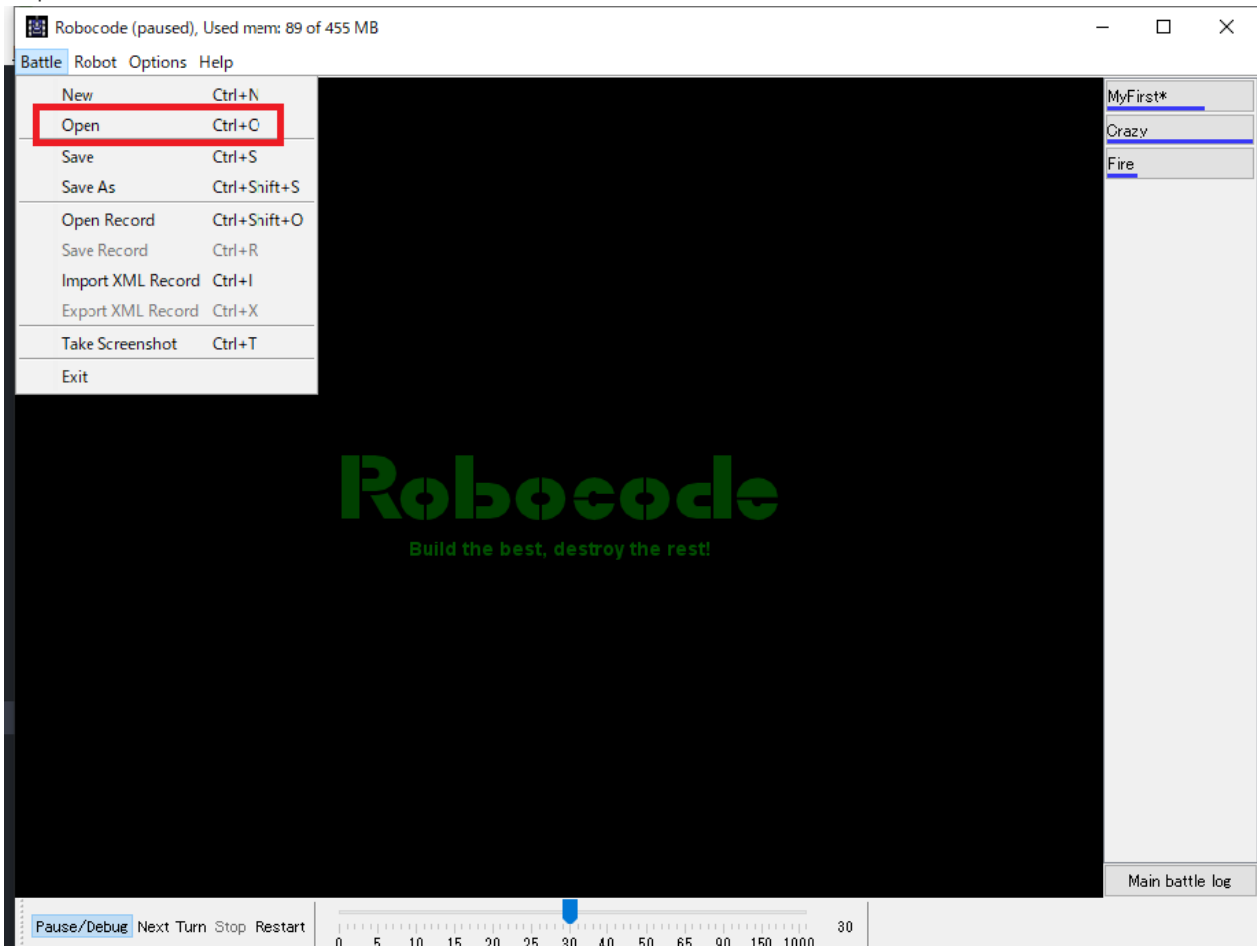
1. 環境設定(Preference),
2. Roboのキャッシュをクリア(Clean robo cache)

- Help: ヘルプ

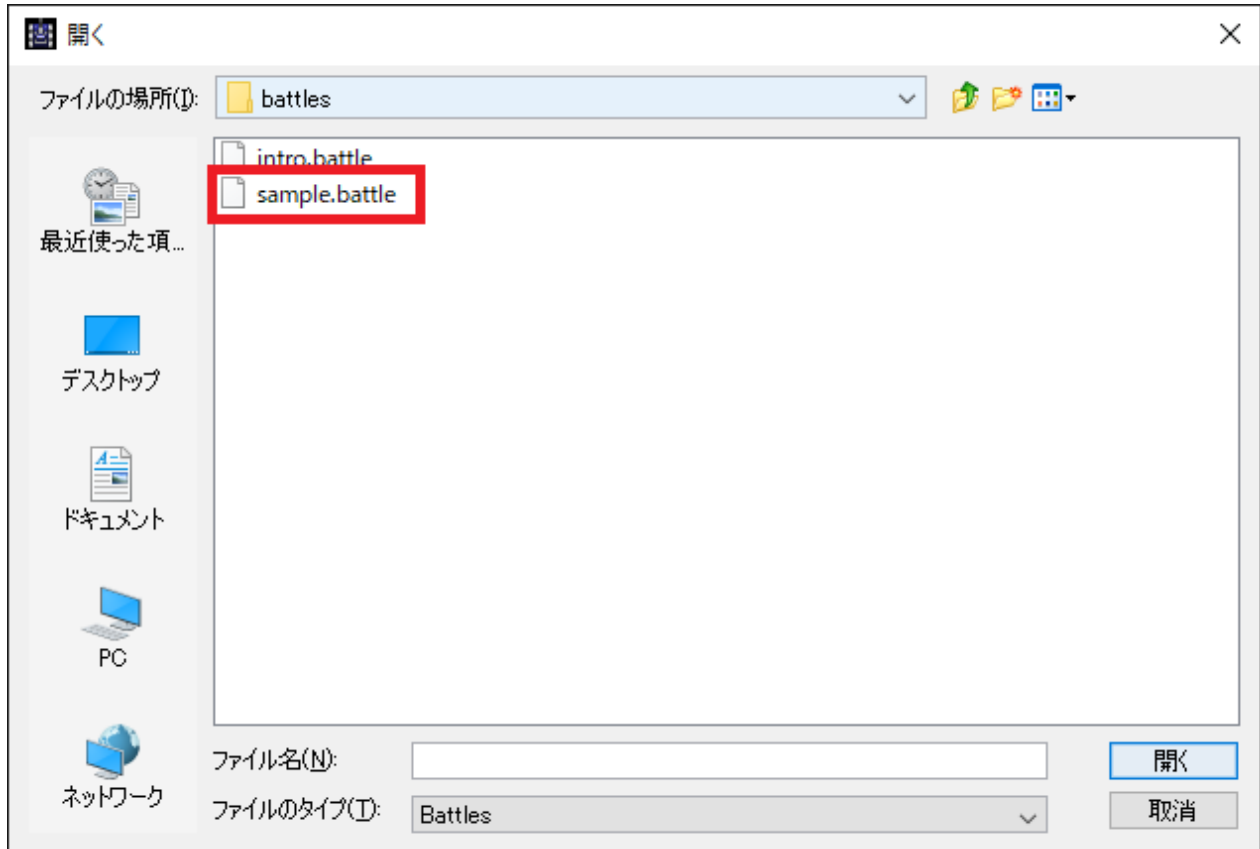
動かしてみよう

まずは、サンプルバトルを動かしてみます。下の手順で動かしてください。

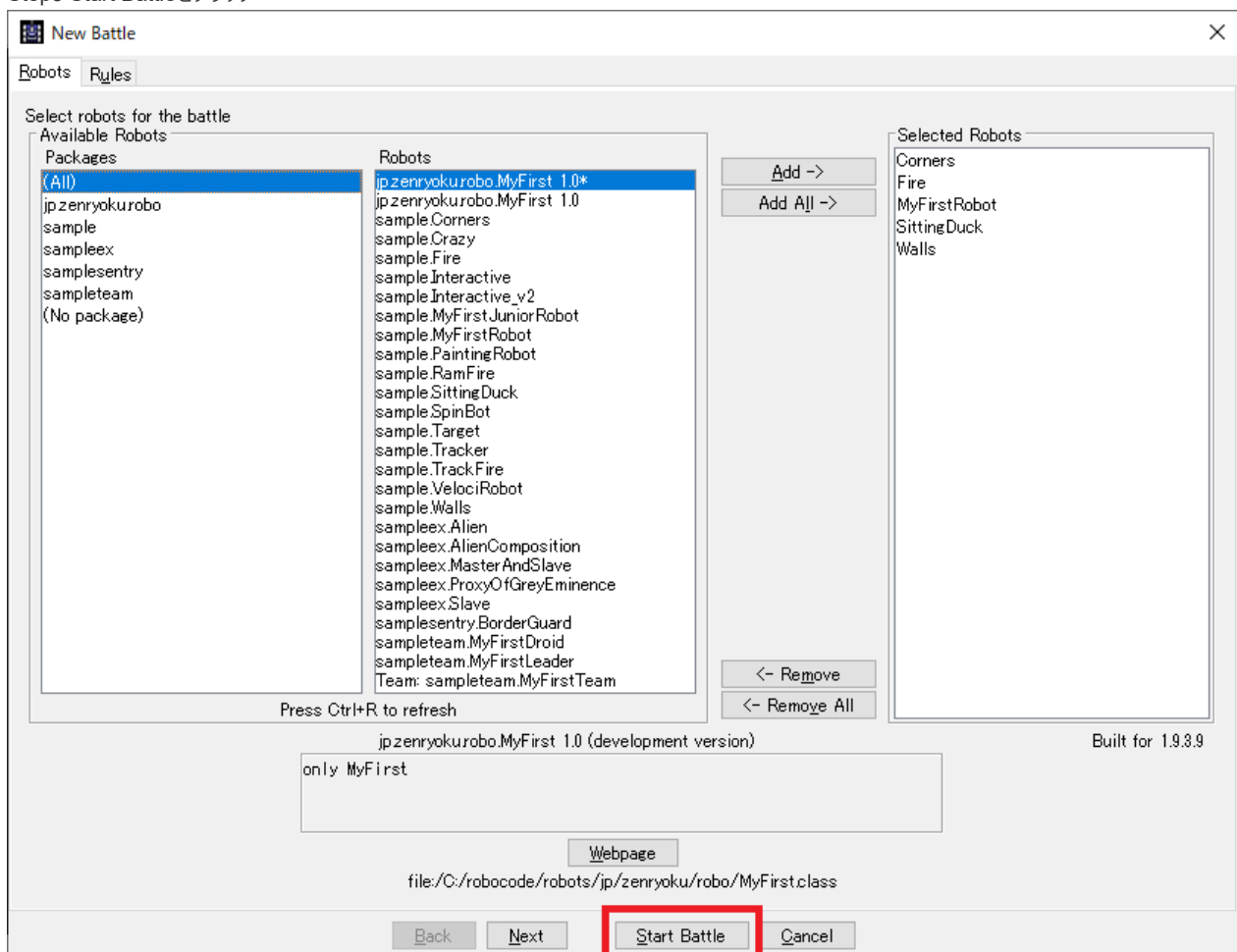
Step1 メニューから「Battle」をクリック



Step2 sample.battleファイルをクリック



Step3 Start Battleをクリック



Step4 Stopでバトルを修了する

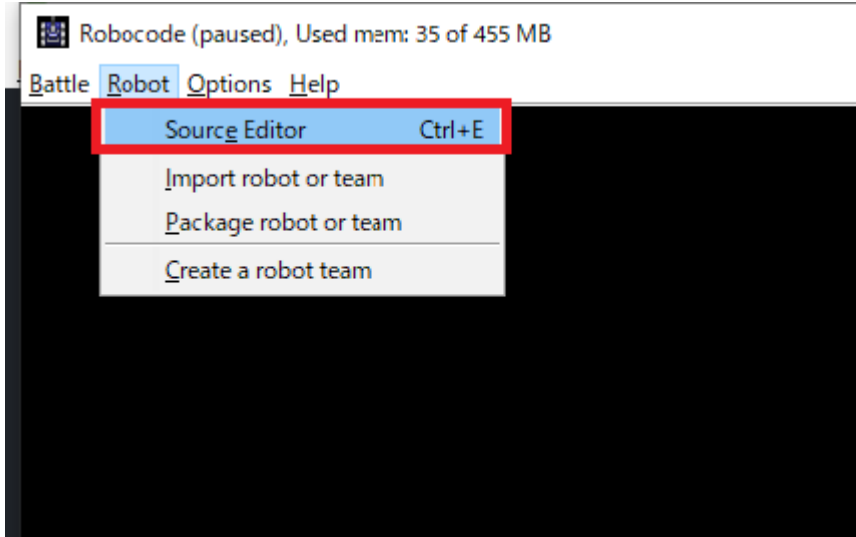


動いているのをみて、どんな動きをしているか観察してみてください。

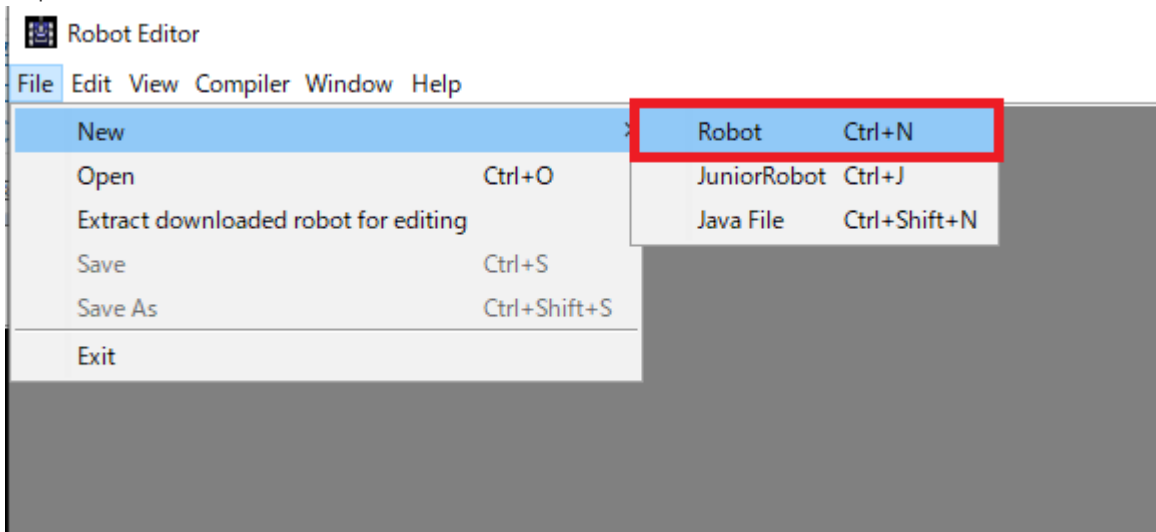
オリジナル・ロボを作ろう

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。

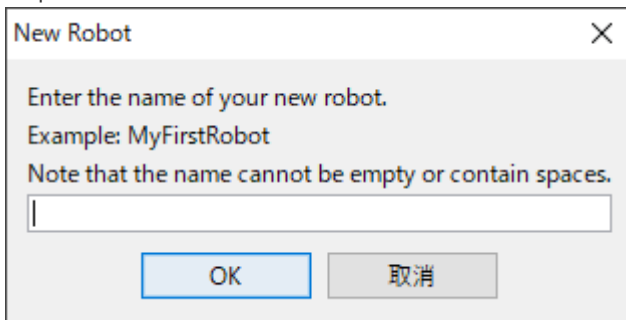
Step1 メニューから「Robo」をクリックし、「Source Editor」をクリック



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



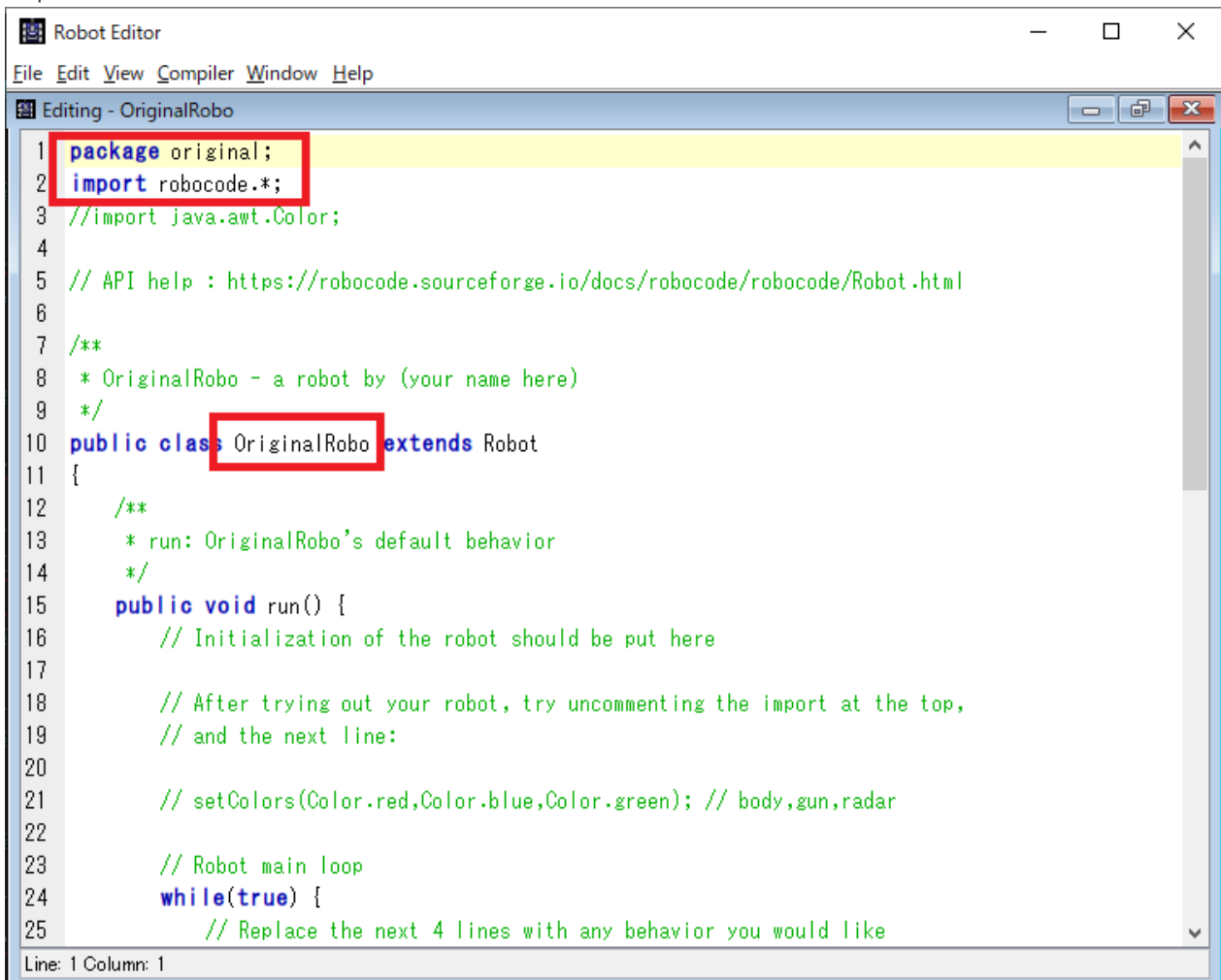
Step4 ロボのファイルを配置するパッケージの名前を入力する ※ロボの名前には「OriginalRobo」という名前を付けたので、赤線部にロボの名前が表示されています。

Package name for OriginalRobo

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.

OK 取消

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。



```
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
```

Line: 1 Column: 1

ロボの仕組みを知ろう

ロボは次の状態の時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. 100前に進む
2. 主砲を右へ360度回転
3. 100後退する
4. 主砲を右へ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);

ロボへ命令する

サンプルバトルを見たときに、ロボはどのような動きをしていたでしょうか？
少なくとも「ジャンプ」はしていませんでした。ロボに出せる「命令」は決まっています。そして、下の表のように「命令」が決まっています。

つまり「命令」を組み合わせるとオリジナルのロボを作成します。

返り値	メソッド名	振る舞い(処理の内容)
void	ahead(double distance)	ロボットを前方に移動させます。
void	back(double distance)	ロボットを後方に移動させます。

返り値	メソッド名	振る舞い(処理の内容)
void	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	fire(double power)	弾丸を発射します。
void	scan()	他のロボットを探します。
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	turnRight(double degrees)	ロボットを回転させます。