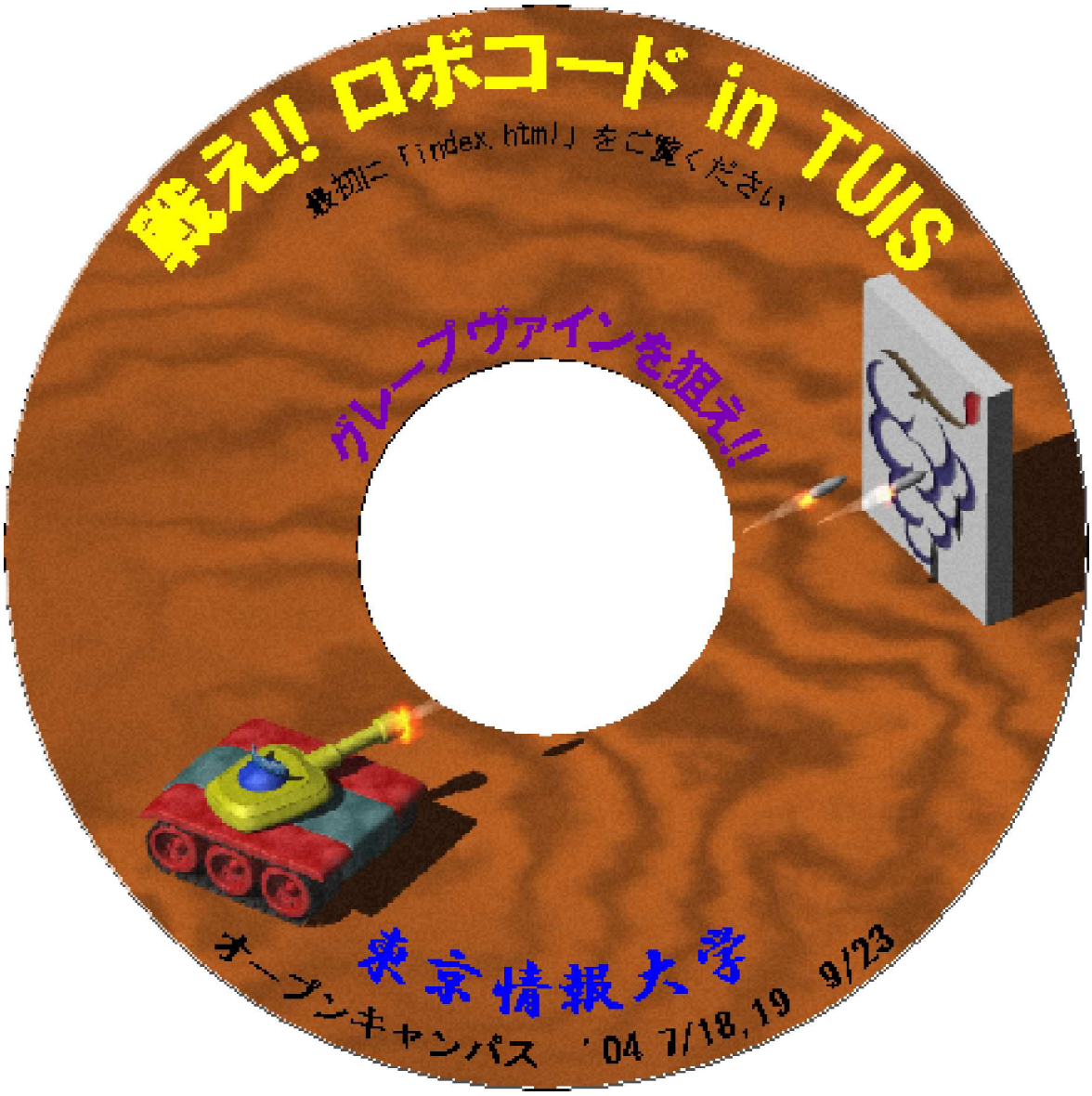
東京情報大学　情報システム学科　井関

ロボコード自作用資料v1.4.1



Java 入門（基礎編）

１．命令文

Javaでは，一つの命令を一つの文で書きます．文の終わりは必ず **;**（セミコロン）で終わります．
複数の文を **{ }** でまとめて，一つのブロックを作ることができます．

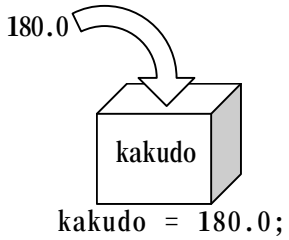
２．数と式

今回プログラムで使用するデータは実数（小数点のついた数）と**論理値**です．実数は足し算(+)，引き算(-)，掛け算(*)，割り算(/)などの計算ができます．　例 1.0+2.0*3.2

論理値は真(**true**)か偽（**false**）のどちらかになります．例えば 3.0<2.0 の論理値は **false**です（3.0は2.0より小さくない）．4.0>=3.2の論理値は **true**です（4.0は3.2と等しいかまたは大きい）．また，二つの値が等しいことは **==** であらわします．3.0==2.0 は **false** ですが，3.0==(6.0/2.0)は **true** です．論理値自身も**または**(or)や**かつ**(and)などの計算ができますが，今回は使用しません．3.0<2.0，4.0>=3.2，3.0==2.0 は**論理式**と呼ばれます．

３．変数

変数は値を保存する入れ物です．今回は実数を入れる入れ物（**実数変数**）を使います．変数は最初に変数名を宣言しないと使用できません．例えば実数変数の宣言は　**double　kakudo;**　というようにします．**kakudo**　が変数名です．このように宣言すると **kakudo** に値（実数）を入れることができます．変数に値を入れるには **=** を使用します．



４．制御文

プログラムの流れを変える機能が制御文です．制御文には **if文**と **while文**があります．他にもいくつかありますが，この2つが分かればどんなプログラムでも作ることが可能です．

if文:

```
if (論理式) {
        論理式が正しい（trueの）ときこの部分が実行される；
}
else {
        論理式が正しくない（falseの）ときこの部分が実行される；
}
```

while文:

```
while (論理式) {
        論理式が正しい（trueである）限り，この部分が実行され続ける；
}
```

５．例題　　“//”以降はプログラムと関係のないコメントを書くことができます．

```
double  kakudo;                     // 実数変数 kakudoを宣言
kakudo = ........;                  // kakudoを計算する．
if (kakudo > 180.0) {               // kakudoが180度より大きかったら...
      kakudo = kakudo - 360.0;      // kakudoから360度引いて，それを改めてkakudoに代入
}                                   // else は省略
if (kakudo < -180.0) {              // kakudoが-180度より小さかったら...
      kakudo = kakudo + 360.0;      // kakudoに360度足して，それを改めてkakudoに代入
}
```

ロボコードの命令（基礎編）

1）ロボット（戦車）は「ボディ」「砲塔（ガン）」「レーダ」からできています.
2）「ボディ」「砲塔」「レーダ」は別々に動かすことができます.
3）ただし, 最初は下の構造物が動くとその上に載っているものも一緒に動く設定になっています.
　例）砲が旋回するとその上に載っているレーダーも一緒に旋回します.
4）「レーダー」が敵を捕らえると，onScannedRobot(e) というプログラム（関数）に制御が移ります.
5）onScannedRobot(e) に渡される e という変数の中にはレーダーで捕らえた敵の情報が入っています

1．ロボット制御命令

ahead(100)	100の距離だけ前進
back(50)	50の距離だけ後退
turnLeft(60)	60度の左旋回（ボディ）
turnRight(90)	90度の右旋回（ボディ）
turnGunLeft(30)	砲を30度左へ旋回
turnGunRight(45)	砲を45度右へ旋回
turnRadarLeft(180)	レーダーを180度左へ旋回
turnRadarRight(360)	レーダーを360度右へ旋回
setAdjustGunForRobotTurn(true)	砲がボディと一緒に旋回しないようにする
setAdjustRadarForGunTurn(true)	レーダーが砲と一緒に旋回しないようにする
fire(2)	エネルギー2の弾を発射する（0～3まで指定可）
setColors(Color.yellow, Color.yellow, Color.red)	ボディ，砲塔，レーダの色を黄，黄，赤に.

2．敵ロボットの情報を得る命令

onScannedRobot(e) の中で使用される命令

e.getBearing()	自分のボディの方向を基準とする, 敵ロボットへの相対角度を得る
e.getDistance()	敵までの距離を得る
e.getHeading()	敵ロボットのボディが向いている方向の角度（絶対角度）を得る
e.getVelocity()	敵ロボットの速度を得る

絶対角度:　上方向を基準にして右回りに測った角度
相対角度:　自分のボディが向いている方向を基準にして右回りに測った角度

3．自分のロボットの情報を得る命令

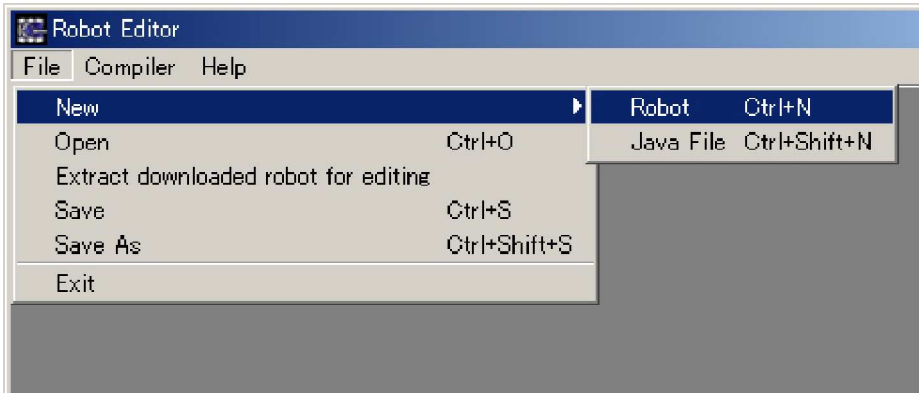
getX()	現在位置の X座標を得る
getY()	現在位置の Y座標を得る
getEnergy()	残りエネルギーを得る
getHeading()	ボディが向いている方向の角度（絶対角度）を得る
getGunHeading()	砲が向いている方向の角度（絶対角度）を得る
getRadarHeading()	レーダーが向いている方向の角度（絶対角度）を得る

プログラムの作成

１．エディタの起動

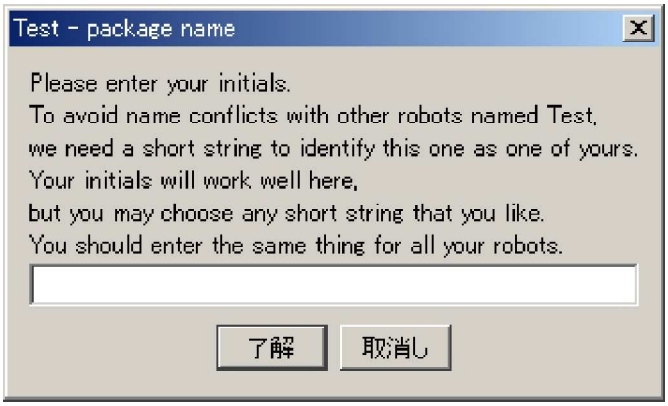
ロボコードの「Robot」メニューから「Editor」を選択します.

エディタウィンドウの「File」メニューから「New」→「Robot」を選択します.



ロボットの名前を入力. 最初の一文字目を大文字にします（何でも良い）.

パッケージ名を小文字で入力（通常はイニシャルを入力）.



2．プログラムの入力

最初に以下のようなプログラムが自動的に作られます.

```
package fi;                    //パッケージ名．最初に入れたイニシャル（気にしない）
import robocode.*;             //ロボコードの機能を使う
//import java.awt.Color;       //カラーを使いたい場合は，コメントを解除する（//を取る）

/**                                        //    /*    */ で囲まれた部分もコメントになる
 * Booo - a robot by (your name here)
 */
public class Booo extends Robot       //このロボットは既に作られている Robotという部品を利用して作る
{
	/**
	 * run: Booo's default behavior
	 */
	public void run() {                    //メインプログラム
		// After trying out your robot, try uncommenting the import at the top,
		// and the next line:
		//setColors(Color.red,Color.blue,Color.green);//ロボットに色をつけたい場合は,コメントを解除
		while(true) {                  //whileの中が trueなので永久ループ（永久に回る）
			// Replace the next 4 lines with any behavior you would like
			ahead(100);            //100の距離だけ前進
			turnGunRight(360);     //砲塔を360度回転．砲塔に乗っているレーダも一緒に回る．
			back(100);             //100の距離だけ後退
			turnGunRight(360);     //砲塔を360度回転．砲塔に乗っているレーダも一緒に回る．
		}
	}

	/**
	 * onScannedRobot: What to do when you see another robot
	 */
	//レーダが敵を捕らえると自動的にこのプログラム（関数）が呼び出される
	public void onScannedRobot(ScannedRobotEvent e) {       // eの中には敵の情報が入っている
		fire(1);
	}

	/**
	 * onHitByBullet: What to do when you're hit by a bullet
	 */
	//弾に当たると自動的にこのプログラム（関数）が呼び出される
	public void onHitByBullet(HitByBulletEvent e) {         // eの中には敵の情報が入っている
		turnLeft(90 - e.getBearing());                  //敵のいる方角から左90度方向へターン
	}
}
```

修正しよう

3行目　//import java.awt.Color;　の // を削除.

16行目　//setColors(Color.red,Color.blue,Color.green);　の //を削除.好きな色を指定しよう.

　　色の指定には Color.yellow, Color.white, Color.black, Color.magenta, Color.cyan などもつかえる.

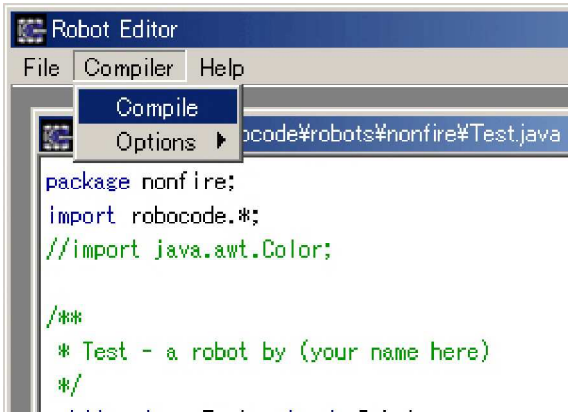
fire(1) を fire(3) に変える.

public void onHitByBullet(HitByBulletEvent e) 以下3行（}まで）は全て削除.　/* */ で囲っても良い

３．プログラムのコンパイル

注）コンパイルとは人間の書いたプログラムを，コンピュータが理解できる形式に変換することです．

「Compiler」メニューから「Compile」を選らびます．



ファイルを保存するかどうかの問い合わせがあるので，「はい」を選んでファイルを保存します．（一番最初はファイルが無いので，それを作るかどうかも聞かれる）

コンパイルが正しく行われた場合は以下のようなメッセージが表示されます．

コンパイルエラーが起きた場合は，エラー内容を確認してエディタに戻って修正します．（良く分からない場合は，手を挙げてアシスタントに聞いてください）

コンパイルが正しく行われた場合は，メイン画面の 「Battle」メニューから「New」を選ぶと作成したロボットが新しく表示されます．もしロボットが表示されない場合は，「Ｆ５」キーを押してください．

よくあるエラー

1．文末の ；（セミコロン）を ：（コロン）と書いた.
2．プログラム名（関数名）の綴り間違い．名前が長いので間違えないように!!
3．漢字の空白を間違えて入れないように!!

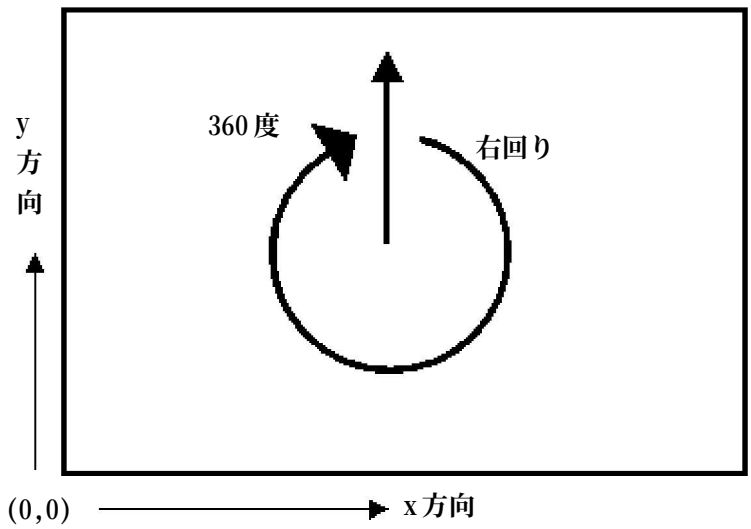
課題：四角や三角に動くロボットを作ってみよう．

ヒント： run() 関数を次のように書き換えて_____ に適当な角度を入れてみよう．

```
public void run() {
        setColors(Color.red, Color.blue, Color.green);    // 色は自由に決めて良い
        while(true) {
                ahead(100);
                turnGunRight(360);
                turnRight(____);
        }
}
```

４．座標系

角度は，上方が0度の右回り 360 度形式．座標は左下が原点となる．



５．レーダを使って相手を狙うロボットを作ろう．

１）まず，「ボディ」，「砲塔」，「レーダ」が別々に動くようにする．

run()関数の　setColor()の前に

```
setAdjustGunForRobotTurn(true);
setAdjustRadarForGunTurn(true);
```

を追加.

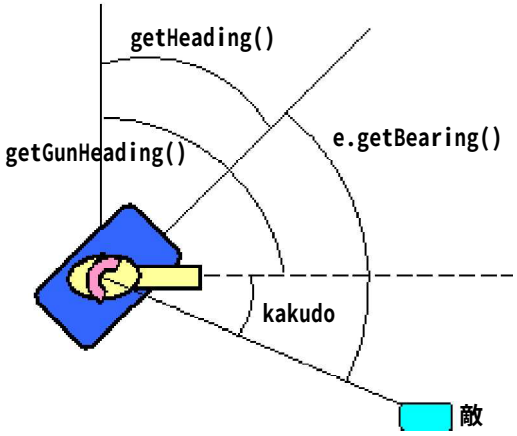
２）run()関数中の turnGunRight(360) を　turnRadarRight(360) に変える．

```
public void run() {
        // After trying out your robot, try uncommenting the import at the top,
        // and the next line:
        setAdjustGunForRobotTurn(true);
        setAdjustRadarForGunTurn(true);
        setColors(Color.yellow,Color.magenta,Color.cyan);       // 色は自由に決めて良い
        while(true) {
                ahead(100);
                turnRadarRight(360);                            // ____ に角度を入れる．
                turnRight(_____);
        }
}
```

３）敵のいる方角 kakudo を求める．

kakudo = getHeading() ___ e.getBearing() ____ getGunHeading();

___ には ＋または－が入ります．



４） onScannedRobot(e)関数の中で，上記で求めた式を使って敵を狙う．

```
public void onScannedRobot(ScannedRobotEvent e) {
        double  kakudo;                 // 角度を計算する変数を宣言
        kakudo = ..........;            // 上記の式で，角度を計算
        turnGunRight(kakudo);           // 計算した角度だけ，砲塔を回転
        fire(3);                        // 発射!!
}
```

５）このロボットの問題点は？

第２ラウンド：アドバンスドロボット

今までのロボットでは, 一度に一つのことしかできません. たとえば, 前進しながら回転し, かつレーダを回して敵を探すと言ったことができません. しかし, AdvancedRobotという部品を使えば, 上記のようなことができるようになります. AdvancedRobotでは行動を予約し, 最後に通常の行動を起こすことによって, 予約した行動を同時に行うことができます.

「前進を予約→ボディの回転を予約→レーダの回転」 を行うと, 円運動(前進しながらボディを回転)しながら, レーダで索敵ができます. 行動を予約するには, 通常の行動命令の前に set を付けて, その次に文字を大文字にします. たとえば,「前進を予約→ボディの回転を予約→レーダの回転」は,

```
while (true) {
      setAhead(100000);          //前進を予約. 数字は適当に大きな数を指定.
      setTurnRight(10000);       //ボディの回転を予約.
      turnRadarRight(360);       //レーダを回す. 予約された命令も同時に実行される.
}
```

とします.

０．予約型のアドバンスドロボット制御命令

setAhead(100)	100の距離だけ前進を予約
setBack(50)	50の距離だけ後退を予約
setTurnLeft(60)	60度の左旋回（ボディ）を予約
setTurnRight(90)	90度の右旋回（ボディ） を予約
setTurnGunLeft(30)	砲の30度左旋回を予約
setTurnGunRight(45)	砲の45度右旋回を予約
setTurnRadarLeft(180)	レーダーの180度左旋回を予約
setTurnRadarRight(360)	レーダーの360度右旋回を予約

アドバンスドロボットを作ろう!!

１．AdvancedRobot を使うための宣言．

プログラムの先頭の　public class ロボット名 extends Robot　の Robot を AdvancedRobot に変える.

２．円運動ロボット

run()関数の while(true) の部分を次のように変更しよう!! どんな動きになるだろう.

```
while (true) {
      setAhead(100000);   .
      setTurnRight(10000);
      turnRadarRight(360);
}
```

課題：今まで勉強したことを使って，自分のロボットを自由に作ろう!!　正解はありません．変な動きの方が強いかも！！

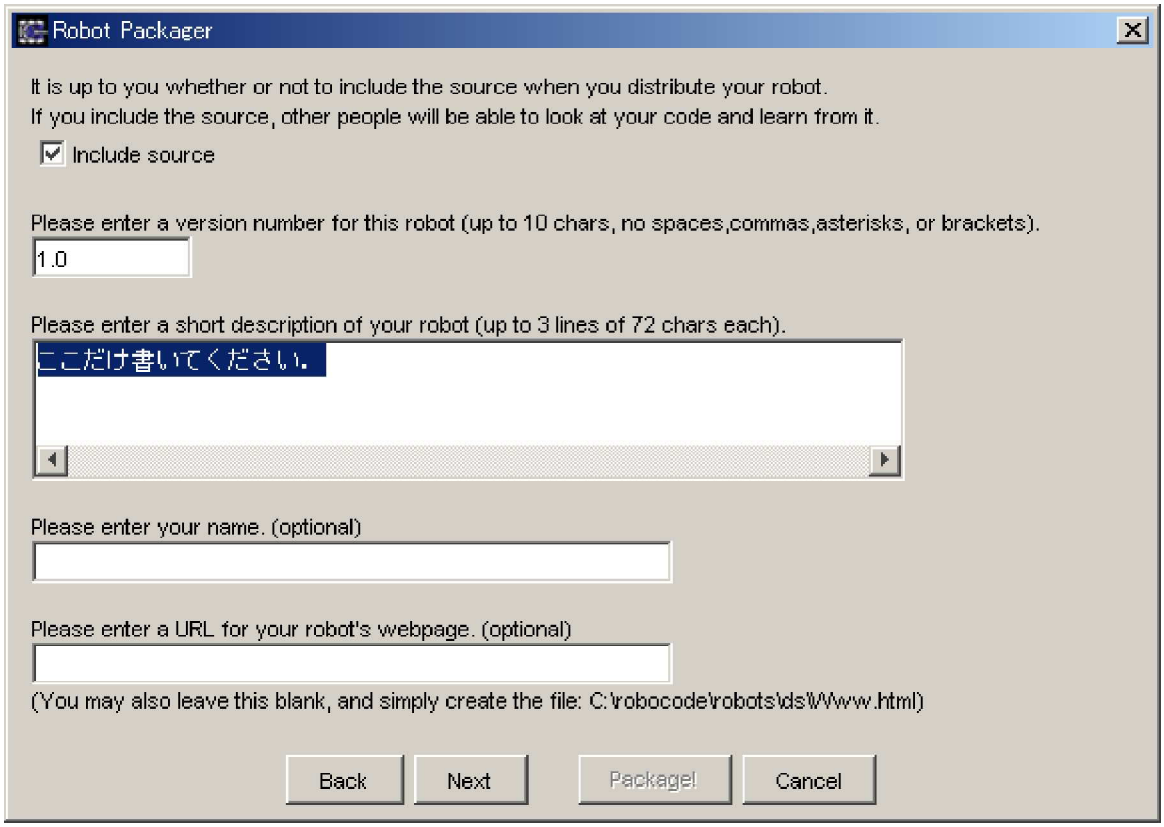
みんなでリーグ戦をやろう!!

自分のロボットができたら, リーグ戦に参加しよう!! リーグ戦に参加するには先ずロボットを jarファイルにしなければなりません. jarファイルは Javaのプログラムを一つにまとめて配布し易いようにするためのファイル形式です.（中身はzipです：詳しい人用）

１．Jar ファイルの作り方

「Robot」メニュから「Package robot for upload」を選択します.

jar ファイルにするロボットを選んで,「Next」ボタンを押します.
ロボットの説明文（適当でよい）だけ書いて「Next」ボタンを押します.



jar ファイルができる場所を確認して「Next」→「Package!」ボタンを押せば, jar ファイルができます.

２．ロボット（Jar ファイル）の登録と結果の参照

ロボットファイルの登録と結果の参照は
http://www.infosys.tuis.ac.jp/xoops/modules/roboleague/ から行います.
ゲストの場合，登録する際にはパスワードを指定しますが，そのパスワードで「変更（編集)」「削除」ができますので，絶対に忘れないでください.

もっと強いロボットを作る．（まだまだ続くロボット作成）

１．インファイター突撃ロボット
相手に向かってぶつかって行くロボットを作ってみよう．

２．アウトファイターロボット
相手から離れて戦うロボットを作ってみよう．反重力（反発）移動を考えてみよう．

３．砲撃回避ロボット
いつもは止まっていて，相手のエネルギーが減った（弾を撃った）瞬間に横に移動するロボットを作ってみよう．でも，移動した先に撃たれたら ……

４．ランダムロボット
ランダムにふらふら移動するロボットを作ってみよう．

５．壁衝突回避ロボット
壁に衝突しないロボットを作ってみよう．

６．予測砲撃ロボット
相手の動きを予測して砲撃するロボットを作ってみよう．

７．対 Level_10 ロボット
Level_10ロボットを撃破しよう．

８．技を盗め!!
他人のロボットのソースコードを読んで技を盗もう．達人のソースコードに勝る教科書はありません！