

# CODEVS for STUDENT

## ゲームの目的

このゲームは、フィールドの一定の高さまでブロックを積み上げないようにしながら、相手のフィールドをブロックでいっぱいにするゲームです。ブロックは条件を満たすと消えます。消えたときの条件次第では相手に消えないブロックが送られ、妨害できます。うまくブロックを積み上げ消しすることで、相手より長く生存することを目指しましょう。

## ゲームの画面



図1. ゲーム画面のスクリーンショット

### 1... フィールド

フィールドは正方形のグリッドで構成される長方形です。

### 2... ブロック

ブロックは数値を持っており、条件を満たすと消えます。

### 3...お邪魔ブロック

消すことができないブロックです。

### 4...パック

ブロックの塊です。フィールドにはパック単位でブロックを設置します。

### 5...ネクストパック

次のターンのパックが表示されます。

### 6...デンジャーライン

このラインまでブロックが積み上がるとそのプレイヤーの負けです。

# 基本ルール

ゲームを理解するために必要なルールが記載されています。

AI実装およびシミュレーション実装などに必要な事項は、「ルールの詳細」をご覧ください。

## ゲームの流れ

AIはどのようにパックを投下するかを決めます。

パックの投下時に指定できるのは次の情報になります。

- パックの投下位置
- パックの回転数

パックを投下してから次のパックを投下するまでの間を「ターン」と呼びます。

1ターンは下記のように構成されます。

1. パックの投下 (AIからの入力)
2. 下記の終了条件を満たしていれば終了、そうでなければ 3 へ
  - a. 思考時間の制限を超えた
  - b. 不正な入力をした
3. 落下するブロックがあれば全て落下
4. 消滅するブロックがあれば全て消滅
5. 落下するブロックが再び現れたなら 3 へ、無いなら 6 へ
6. 下記の終了条件を満たしていればゲーム終了、そうでなければ 7 へ
  - a. ターン数が一定数に達した
  - b. デンジャーラインまでブロックが積み上がった
7. お邪魔ブロックの挿入処理
8. 次のターンへ

## ブロック

ブロックには数値が振られています。この数値はブロックの消滅時に考慮されます。（詳細はブロックの消滅の項を参照）

## お邪魔ブロック

お邪魔ブロックは消滅しないブロックです。消滅しない以外はブロックと同じ扱いです。

## ブロックの消滅

ブロックは条件を満たすと消滅します。

ブロックの消滅条件は下記のようになります。

- 縦・横・斜めのいずれかの方向にブロックが1個以上連続している。
- 連続したブロックが持つ数値の和が一定の値である。

以上の条件すべてを満たしたブロックは消滅します。またこの時、和の対象となった範囲を消滅範囲と呼びます。

## ブロックの落下

ブロックはフィールドの底または積まれたブロックに達するまで落下します。ブロックが消滅した場合、上に乗っているブロックはすべて落下します。

## チェイン

1ターンのうちにブロックの落下と消滅が交互に発生します。落下と消滅の繰り返しをチェインと呼びます。

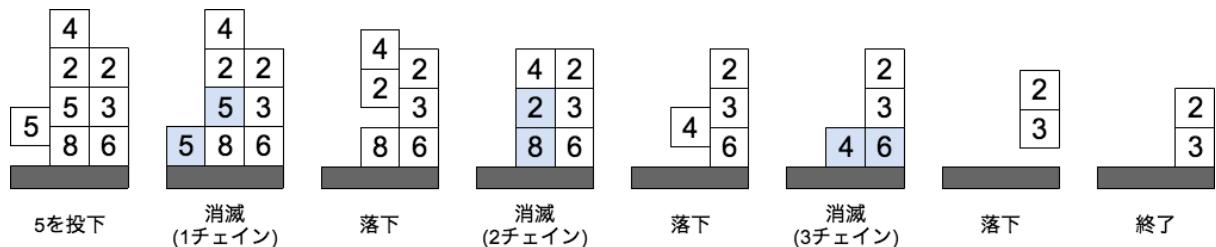


図2. 3チェインの例 (和が10の場合)

## ブロックの消滅カウント

1ターン中に消滅の対象となったブロックの数です。1つのブロックが複数の消滅範囲に含まれた場合、重複してカウントされます。実際にフィールドから消滅したブロックの数とは異なることに注意してください。

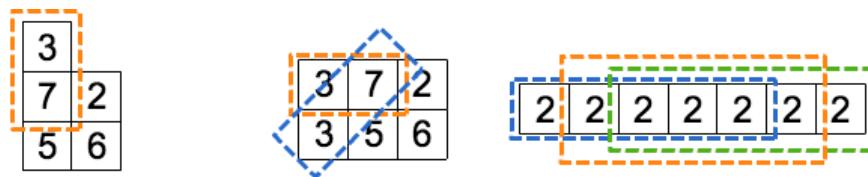


図3. 消滅カウントの例 (和が10の場合)

## 対戦相手への妨害

対戦相手の妨害として、相手のネクストパックにお邪魔ブロックを挿入できます。また自分のネクストパックに挿入予定のお邪魔ブロックを減らす（相殺と呼称）こともできます。  
(お邪魔ブロックの挿入と相殺については「ルールの詳細」で解説します)

## スコア

スコアはチェイン数とブロックの消滅カウントから特定の式によって算出されます。（計算式については「ルールの詳細」で解説します）

## ゲームの終了条件

下記の条件のいずれかを満たした場合、ゲームが終了します。

- ターン終了時、デンジャーラインまでブロックが積み上がった
- 思考時間の制限を超えた
- ターン数が一定数を超えた
- 不正な入力をした

## 勝敗について

**勝利**：ターン終了時、相手がデンジャーラインまでブロックが積み上がった

相手のAIが思考時間の制限を超えた

相手のAIが不正な入力をした

**敗北**：ターン終了時、自分がデンジャーラインまでブロックが積み上がった

自分のAIが思考時間の制限を超えた

自分のAIが不正な入力をした

また下記の条件のとき、獲得スコアの多い方が勝利となります。

- 自分と相手が同ターンでデンジャーラインまでブロックが積み上がった
- 自分と相手のAIが同ターンで思考時間の制限を超えた
- ターン数が一定数を超えた

ただし、獲得スコアが同じ場合は、そのゲームは**引き分け**となります。

## 思考時間の制限について

思考時間とは、AIが行動を決定するために計算する時間のことを言います。（思考時間の制限は「ルールの詳細」で解説します）

# ルールの詳細

AIの実装およびゲームのシミュレーションする上で必要な細かな仕様が記載されています。

## ターンについて

1ゲームは最大500ターンです。

## パックについて

パックは3×3の正方形の枠で、枠の中にブロックがランダムに配置されます。

## ブロックについて

ブロックの数値は1~9までの数が振られます。

すべてのターンを通して、1~9のブロックはそれぞれ180個与えられます。

## 消滅条件の和について

消滅条件である和の値は10です。

## お邪魔ブロックについて

お邪魔ブロックは、「消滅条件の和+1」の数値（つまり11）が振られたブロックとして入力で与えられます。

## パックの投下

パックの投下位置と回転数を指定して、パックをフィールドに投下します。指定した投下位置と回転数が不正だった場合、ゲームは直ちに終了します。

## パックの投下位置

パックの投下位置は、-2~9まで指定できます。投下位置の指定には以下の制約があります。

投下位置	指定可能条件
-2	パックの左2列がすべて空白の場合
-1	パックの左1列がすべて空白の場合
0~7	いつでも指定可能

8	パックの右1列がすべて空白の場合
9	パックの右2列がすべて空白の場合

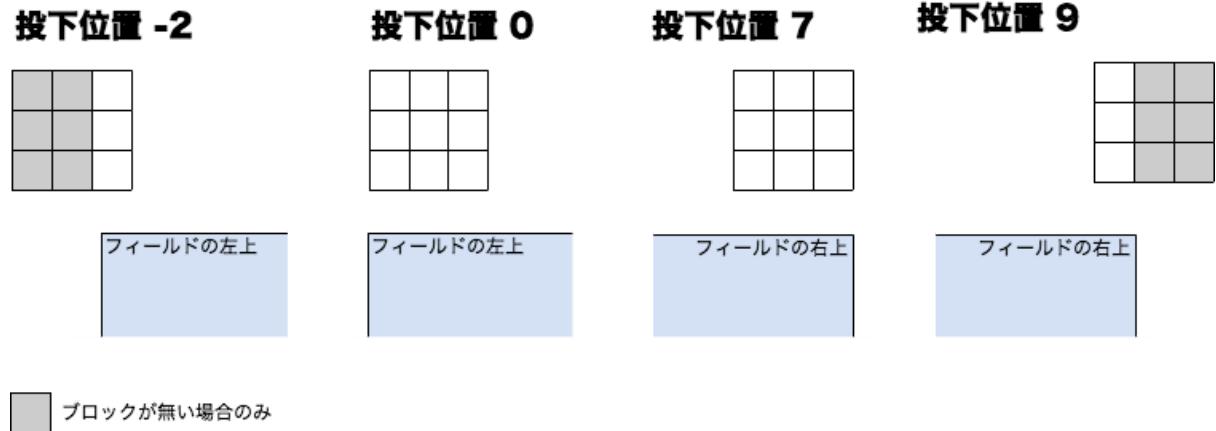


図4. 投下位置の例

### パックの回転数

パックの回転数は、0~3の整数を指定できます。回転数1につき時計回りに90°回転します。



図. ブロックの回転数の例

### スコアについて

ブロックを消滅させたりチェインを繋げたりすることで、スコアを獲得します。スコアは下記の式で算出されます。

$$\text{スコア} = \sum_{i=1}^C \text{floor}(1.3^i) \cdot \text{floor}\left(\frac{E_c}{2}\right)$$

※チェイン数  $C$ 、1チェインの中で消した消滅カウント  $E_c$  とする

※  $\text{floor}(x)$  は  $x$  の小数点以下の切り捨てを表す

## お邪魔ブロックの挿入

お邪魔ブロックの挿入は下記の流れで行われます。

1. (パックの投下)
2. (ブロックの消滅&落下処理)
3. お邪魔ブロックをお邪魔ストックに追加
4. お邪魔ストックの相殺
5. お邪魔ブロックをネクストパックに挿入
6. お邪魔ストックの持ち越し
7. (ターン終了)
8. (システムからの入力)
9. (パックの投下)

## お邪魔ブロックをお邪魔ストックに追加

スコアを獲得することで、お邪魔ブロックが相手にストックされます。これをお邪魔ストックと呼びます。ストックされるお邪魔ブロックの数は下記の式で決定します。

$$\text{お邪魔ブロック数} = \text{floor}(\text{スコア}/5)$$

※ チェイン数  $C$ 、1チェインの中で消した消滅カウント  $E_c$  とする

※  $\text{floor}(x)$  は  $x$  の小数点以下の切り捨てを表す

### お邪魔ブロックの計算例

2ブロックずつ消滅させた3チェインの時：

#### 獲得スコア

$$\begin{aligned} & \text{floor}(1.3^1) \cdot \text{floor}(\frac{2}{2}) + \text{floor}(1.3^2) \cdot \text{floor}(\frac{2}{2}) + \text{floor}(1.3^3) \cdot \text{floor}(\frac{2}{2}) \\ &= \text{floor}(1.3) \cdot \text{floor}(1) + \text{floor}(1.69) \cdot \text{floor}(1) + \text{floor}(2.197) \cdot \text{floor}(1) \\ &= 1 \cdot 1 + 1 \cdot 1 + 2 \cdot 1 \\ &= 4 \end{aligned}$$

#### 追加されるお邪魔ブロック

$$\begin{aligned} & \text{floor}(4/5) \\ &= \text{floor}(0.8) \\ &= 0\text{個} \end{aligned}$$

1チェイン目2個、2チェイン目4個、3チェイン目8個のブロックを消滅させた時：

#### 獲得スコア

$$\begin{aligned} & \text{floor}(1.3^1) \cdot \text{floor}(\frac{2}{2}) + \text{floor}(1.3^2) \cdot \text{floor}(\frac{4}{2}) + \text{floor}(1.3^3) \cdot \text{floor}(\frac{8}{2}) \\ &= \text{floor}(1.3) \cdot \text{floor}(1) + \text{floor}(1.69) \cdot \text{floor}(2) + \text{floor}(2.197) \cdot \text{floor}(4) \\ &= 1 \cdot 1 + 1 \cdot 2 + 2 \cdot 4 \\ &= 11 \end{aligned}$$

## 追加されるお邪魔ブロック

$$\begin{aligned} & \text{floor}(11/5) \\ &= \text{floor}(2.2) \\ &= 2\text{個} \end{aligned}$$

## お邪魔ブロックの相殺

両者のお邪魔ストックを比較し、お邪魔ストックの多いプレイヤーには両者のお邪魔ストックの差分を残し、お邪魔ストックの少ないプレイヤーはお邪魔ストックを0にします。これをお邪魔ブロックの相殺と呼びます。

### 例1.

自分のお邪魔ストック10個、相手のお邪魔ストックが5個  
⇒ 自分のお邪魔ストック5個、相手のお邪魔ストック0個

### 例2.

自分のお邪魔ストック8個、相手のお邪魔ストックが10個  
⇒ 自分のお邪魔ストック0個、相手のお邪魔ストック2個

### 例3.

自分のお邪魔ストック0個、相手のお邪魔ストックが2個  
⇒ 自分のお邪魔ストック0個、相手のお邪魔ストック2個

## お邪魔ブロックをネクストパックに挿入

ストックされたお邪魔ブロックは、ネクストパックの空白部分に挿入されます。お邪魔ブロックは、左上から右下へ順番に挿入されます。

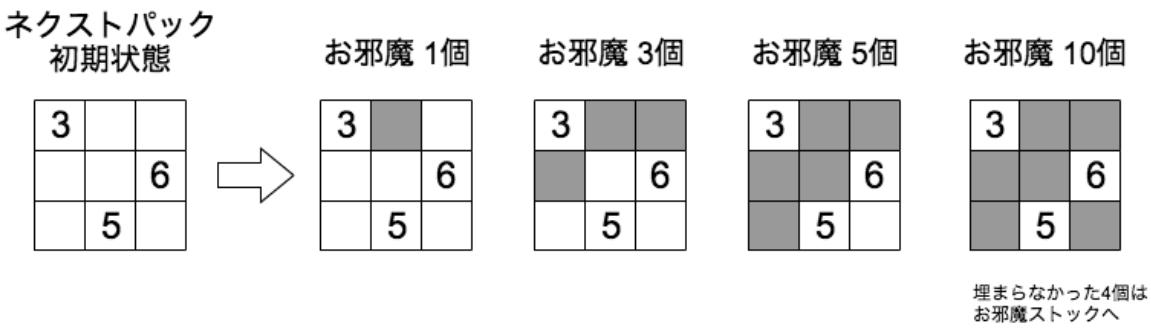


図6. お邪魔ブロックの挿入例

## お邪魔ストックの持ち越し

ネクストパックに挿入しきれなかったお邪魔ストックは次のターンに持ち越されます。

## 思考時間の制限について

思考時間は1試合5分までと制限されています。

## AIの出力

AIは、決められた書式に従って標準出力にテキストを出力することでゲームに対して行動を指定してください。標準出力に出力するテキストを以降、命令文と呼称します。

また命令文はそのターンのシステムの入力を受け取った後に出力してください。

### よくある質問

Q. 標準出力に出力したのに読み取ってくれないのはなぜ？

A. 言語仕様に従った標準出力のフラッシュを行ってみてください

Q. 1ターン目の直後に「内部エラー」と表示され終了してしまいます

A. 命令文の末尾に空行が出力されてしまっていないか確認してください

## AI名について

システムはまずAI名を受け付けます。AI名となる文字列を標準出力に出力してください。

毎ターンの命令文の出力の際は不要です。

### AI名の例

```
sample_ai
```

## 命令文の書式

命令文は下記の形で出力してください。

```
[投下位置] [パックの回転数]
```

例：投下位置2、パックの回転数0の場合

```
2 0
```

## システムからの入力

システムからの入力はゲーム開始時とターンごとに行われます。したがって、1ターン目はゲーム開始時の入力とターンごとの入力が連続して送られます。また、以降の説明は下記を前提とします。

W : フィールドの幅

H : フィールドの高さ

T : 投下するパックの一辺の長さ

S : 消滅のために作るべき和の値

N : 与えられるパックの数 (=ターン数)

## ゲーム開始時の入力

ゲーム開始時の入力は下記の形で与えられます。

[W] [H] [T] [S] [N]  
[1ターン目に投下されるパック情報]  
[2ターン目に投下されるパック情報]  
...  
[Nターン目に投下されるパック情報]

## パック情報

パック情報は  $T \times T$  行列で 1 行目が最も天井に近い行、  $T$  行目が最も底に近い行になります。各行には、その行のブロックのモチ数値が半角スペース区切りで与えられます。0 は空白を意味します。それぞれのパック情報の最終行にはENDと書かれた行が与えられます。

### パック情報の例 (3 × 3 のパック)

```
0 0 0  
2 0 0  
0 8 1  
END
```

## ターンごとの入力

ターンごとの入力は下記の形で与えられます。

[現在のターン数]  
[自分の残りの思考時間。単位はミリ秒]  
[自分のお邪魔ストック]  
[前のターン終了時の自分のフィールド情報]  
[相手のお邪魔ストック]  
[前のターン終了時の相手のフィールド情報]

## 現在のターン数

与えられる入力のターン数です。1ターン目が0、最終ターンはN-1です。

## 自分のお邪魔ストック、相手のお邪魔ストック

ネクストパックにお邪魔ブロックを挿入する前のお邪魔ストックの数が入ります。お邪魔ストックが1以上の場合、このターンに降るパックにお邪魔ブロックが追加されることを示しています。お邪魔ブロックの挿入位置は「お邪魔ブロックをネクストパックに挿入」の項を参考にして計算してください。

## フィールド情報

フィールド情報は  $H \times W$  行列で 1 行目が最も天井に近い行、  $H$  行目が最も底に近い行になります。各行には、その行のブロックのモチ数値が半角スペース区切りで与えられます。0 は空白を意味します。最終行にはENDと書かれた行が与えられます。

## フィールド情報の例 (16 × 10 のフィールド)

## 入力例

ゲームの例として、最大3ターンのゲームの場合の入力を示します。

また、「#」から改行までの文字列は、わかりやすさのためのコメントです。実際の入力にはありません。

```

10 16 3 10 3          # 横10x縦16, 3x3のパックで、合計10になると消える
0 0 1
0 0 0
0 5 1
END
0 0 0
5 5 0
3 0 0
END
0 2 0
0 0 0
2 0 0
END          # 開始時入力ここまで
0          # そのまま続けて1ターン目の入力が行われる
300000      # 残り思考時間
0          # 自分のパックに追加されるお邪魔ブロックの数
0 0 0 0 0 0 0 0 0 0  # 前のターン終了時の自分のフィールド
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

```





```
0 0 0 0 0 0 0 0 0 0  
0 0 0 0 5 0 0 0 0 0  
0 0 0 0 1 0 1 0 3 0  
END
```

以上の入力では、以下の3つのパックがこの順番で与えられていることになります。

1ターン目	2ターン目	3ターン目
5	1	2

図7. 入力例のパック

## 更新履歴

2016/10/17：ルール公開

# CODE VS for STUDENT

## コンテストの詳細

### コンテストの期日について

#### 予選期間

2016年10月17日(月) 12:00:00 ~ 2016年11月14日(月) 23:59:59

#### 決勝について

日時：2016年12月10日(土)

会場：チームラボ本社

### コンテストの流れ

参加者のコンテスト参加の簡単な流れは次のようになります。

1. Webページで会員登録
2. 専用ページからコンテスト用クライアントをダウンロード  
(サンプルコードもあります。ぜひご利用ください)
3. 各自の環境でクライアント起動
4. AIの作成
5. 作成したAIでコンテストに参加
  - a. サーバーにAIを提出して対戦 (サブミット対戦機能)
  - b. 各自の環境でAIを起動して対戦 (オンライン対戦機能)
6. AIの改良
7. 4から6を予選終了期間まで繰り返す
8. 予選終了後、NormalおよびHardランキングの学生のみに絞った上位8名にランクインすると決勝進出
9. 運営から指定する日までに決勝用のAIを提出
10. 決勝当日を迎える

### 参加における必要な環境について

このコンテストはAIに使用する言語に制限はありません<sup>1</sup>。ただし、決勝進出者にはAIを提出していただく必要があるため、下記の条件を満たしていただけないとより良いです。

- 外部から起動できる
- 標準入出力を扱える
- 有償のライブラリ、または環境構築に時間がかかるライブラリの使用を避ける

---

<sup>1</sup> オンライン対戦機能を利用した場合に限る。

また専用のクライアントはJavaで実装されているため、Javaが動作する環境をご用意ください。

## 予選について

CODE VS for STUDENT では、参加者同士のAIを対戦させて、その勝敗でランキングを競います。1試合3ゲーム2本先取となります。1ターンの制限時間は20秒とします。他の参加者のAIと対戦させる方法は下記の2種類あります。

- サブミット対戦機能
- オンライン対戦機能

それぞれメリットとデメリットがあります。ご都合に合わせてご利用ください。

### サブミット対戦機能

サブミット対戦機能とは、AIのソースコードを提出することで、CODEVS運営事務局が管理するサーバー上でAIを起動し、自動的に対戦を行うモードのことです。一度提出したAIは取り下げることは出来ません。

また運営が用意した環境のため、プログラミング言語に制限があります。サポートしている言語は以下のものに限られます。ライブラリなどは言語標準のものに限られます。

- Java (1.8.0\_66)
- C++ (gcc 4.8.4)
- C# (mono 4.2.1)
- Ruby (2.1.8)
- Python (3.5.2)

また、提出できるソースコードは1ファイルのみです。複数ファイルを扱う場合は、プログラミング言語欄の「zip」でお試しください。ただし「zip」機能は動作を完全にサポートしているものではありません。ご注意ください。

### オンライン対戦機能

オンライン対戦機能とは、各自の環境でAIを起動し、インターネット経由で他の参加者のマッチングを行い、対戦させるモードのことです。各自の環境でAIを起動させるため、AIに使用するプログラミング言語に制限はありません。またこの機能は**サブミット対戦機能を利用して、AIがサーバー上で動作した方のみ使用可能**となります。ご利用の際はご注意ください。

### レートについて

ランキングは、対戦の勝敗によって、AIの強さを表す値（レート）が増減します。参加直後はレートは1500です。勝敗が決したとき、レートは下記のような計算で増減します。

## 前提

$R_A$  : Aのレート

$R_B$  : Bのレート

$R'_A$  : 対戦後のAのレート

$R'_B$  : 対戦後のBのレート

## それぞれの勝利する確率

$$E_A = 1/(1 + 10^{(R_B - R_A)/400})$$

$$E_B = 1/(1 + 10^{(R_A - R_B)/400})$$

## AがBに勝った時

$$\Delta R_A = 32 \cdot (1 - E_A)$$

$$\Delta R_B = 32 \cdot (0 - E_B)$$

(ただし、小数点以下は切り捨て)

$$R'_A = R_A + \Delta R_A$$

$$R'_B = R_B + \Delta R_B$$

## AとBが引き分けた時

$$\Delta R_A = 32 \cdot (0.5 - E_A)$$

$$\Delta R_B = 32 \cdot (0.5 - E_B)$$

(ただし、小数点以下は切り捨て)

$$R'_A = R_A + \Delta R_A$$

$$R'_B = R_B + \Delta R_B$$

## ランキングについて

ランキングに表示されている「レート」は、Hard、Normalそれぞれでランキング1位とランキング最下位の値から相対的についた指標です。ランキング1位は常に「1000」、ランキング最下位は常に「0」となります。

## ランキングの確定について

2016年11月14日 23:59:59時点でのランキング結果を予選結果とします。その時点で終了していない対戦はランキング結果には反映されません。AIの改善がレートに反映されるには時間がかかりますので、期限より早めの提出を推奨します。

## 決勝について

決勝戦は、NormalおよびHardランキングの学生のみに絞った上位8名、計16名が出場権を得ます。社会人の方は決勝出場権はありません。決勝進出を辞退した参加者がいた場合、9

位以降の参加者を順次繰り上げとします。詳細は予選終了後、決勝進出者に連絡を予定していますので、そちらをお待ち下さい。

## 更新履歴

2016/10/17 テキスト公開

# CODE VS クライアント トラブルシューティング

## jnlpファイル(クライアント)を開けない

- javaをインストールしてください。

## Javaのコンパイルができない

- 日本語が記述されたソースコードだと、コンパイルに失敗する場合があります。ソースコードの文字コードに合わせて、エンコーディング指定してみてください。

```
javac -encoding UTF-8 SampleAI.java
```

## ローカル対戦の実行に失敗する

- 赤チームコマンド・青チームコマンドの欄には、AIの実行コマンドを入力してください。

### C++の例

```
./a.exe
```

### Java例

```
java MyAI
```

- クライアントをAIのプログラムと同じディレクトリに移動してみてください。
- クライアントをコマンドラインから実行してみてください。  

```
javaws codevs.jnlp
```
- 実行ファイルのパスは正しいですか？  
相対パスの場合は、クライアントが置かれているディレクトリからの相対パスを入力してください。例えば、クライアントが置かれているディレクトリの中の sample というディレクトリに実行ファイルがある場合は、相対パスは次のようになります。  

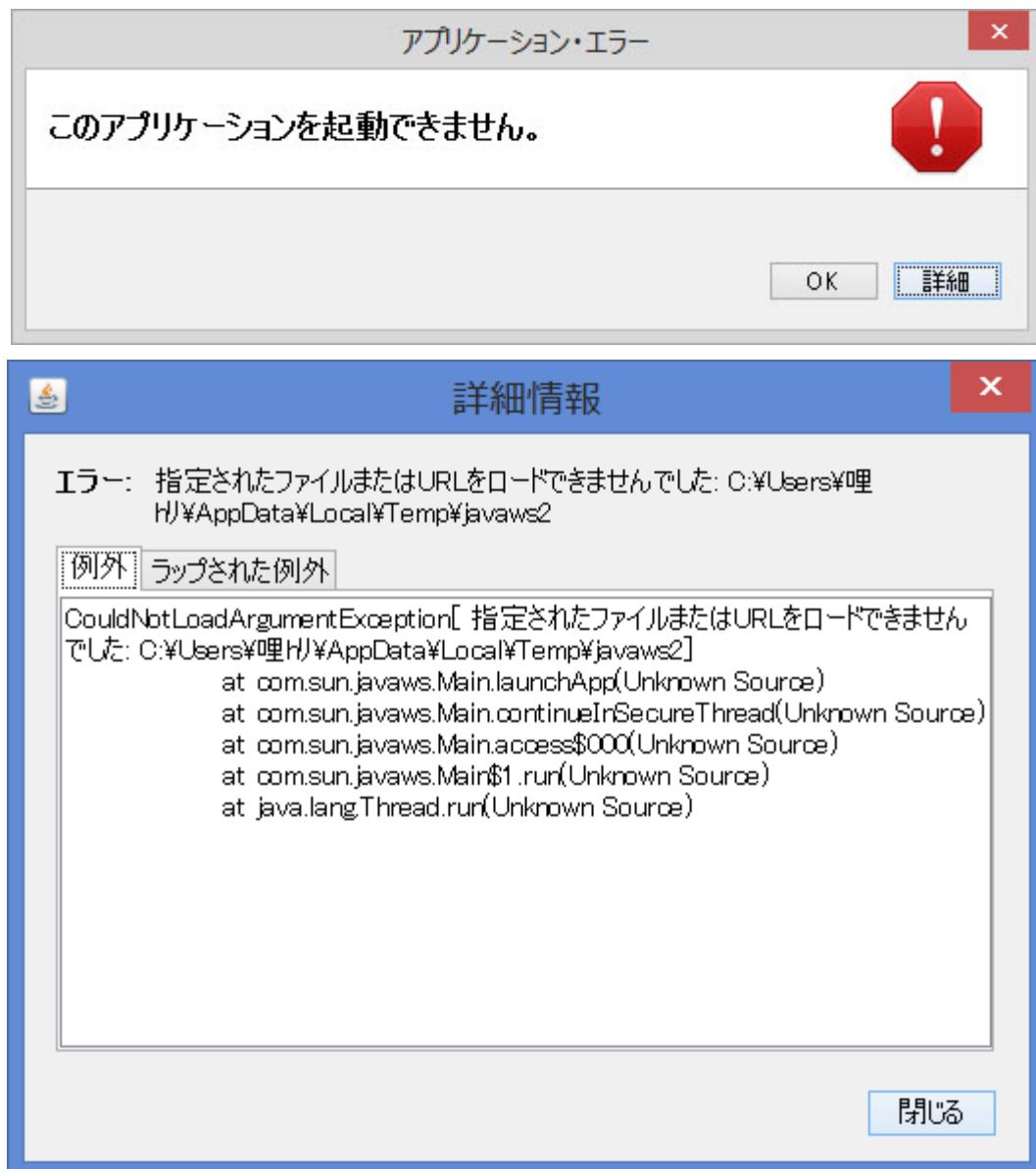
```
./sample/a.exe
```
- ウィンドウの下にカレントディレクトリを表示されていますので、そちらもご確認ください。(クライアントの位置とカレントディレクトリは必ずしも一致しません)

## Java関連

- javaのパスを通してみてください。
- クライアントに記述する実行コマンドの先頭に「java」を入れてみてください。
- 末尾に「.class」を入れている場合は削除しましょう。

```
java SampleAI
```

このようなエラーが出た場合



- ユーザー名が日本語ではありませんか？  
Oracleの仕様で日本語が含まれるパスにあるjarは起動することができません。  
また、Windowsはデフォルトで「%USERPROFILE%\AppData\Local\Temp」が一時

ファイル保存場所として指定されており、日本語のユーザー名がパスに入っています。したがって、日本語を含まないパス（例えばC直下に英数字のみの新しいフォルダを作成）を指定することでこの問題は解決します。

## jnlpから起動するとエラーが表示されないことがある

- 表示させるには、コンソールで `javaws -viewer` でJavaコントロールパネルを起動し、詳細→Javaコンソール→コンソールを表示するにチェック

