卒業研究報告書

題目

Java を用いた神経衰弱アプリの開発

指導教員

石水　隆　講師

報告者

17–1–037–0181

入江 峻

近畿大学理工学部情報学科

令和 4 年 1 月 24 日提出

概要

神経衰弱は, 裏側に伏せられたカードをプレイヤー順にめくり, 同じ数字や絵柄であれば自分の持ちカードとし, 全てカードを取り終えた後, 最もカードを多く取得していたプレイヤーの勝利となる, 記憶力が鍵とするカードゲームである.

神経衰弱はルールが簡単で, カードがあれば老若男女遊べることができ, 集中力や記憶力が重要とするため, 脳を育てる知育用のゲームとしても用いられる. そこで本研究では, 集中力や記憶力を育む目的で神経衰弱アプリケーションを開発する. 本研究で作成するアプリケーションは初心者でも遊びやすいように, 一度表にされた数字を再度表にした際にヒントを与える機能や CPU に難易度調整機能を付ける.

目次

1 序論

1.1 本研究の背景

神経衰弱は記憶量が勝敗に強く影響するゲームである. そのため, 計算機相手に神経衰弱をする場合に, 計算機であれば一度めくられた全てのカードを確実に覚えることができ, 時間経過にしても忘れることが無いのに, 対して, 人間には記憶力というものが不確定要素であるため, 計算機対人間では人間側が圧倒的に不利となってしまう. しかし, 計算機側の記憶力を調整することによって人間側にも優位になる. そこで本研究では計算機側の記憶力を調整し, 初心者にも遊びやすい神経衰弱アプリを作成する.

1.2 戦略

神経衰弱の戦略については，Zwick と Paterson が取るカードの枚数の期待値を最大にする戦略について, 篠田が 2 人で対戦する際に相手より多くのカードを取る確率を最大にする戦略 [1] [2] について述べている. [1] では, 各状態での最善の戦略は, その時点までにお互いが何枚カードを取得しているかで決まる. [2] では, 自分が取得するカードの枚数と相手が取得するカードの枚数との差の期待値を最大化する戦略を求めている. プレイヤーは過去に何枚カードを取得したかは気にする必要はなく, プレイヤーはその時点で場にあるカードの状態から最善の戦略を考える必要がある.

神経衰弱はカードをめくる場合，すでに判明しているカードの中に同じ数字のペアがあればそれをめくり，できるだけ沢山のカードを取るのが通常の戦略である．だが，戦略の選択として, 残りのカード枚数や既知のカードの枚数の偶奇大きく依存するため, わざと合わないカードを取る「パス」となる手が有効となる場合もある.[1] また，坂元と篠田は，ある特定のカードを取った方が勝ちとする特別ルールを加えたときに，勝率を最大にする戦略について考察している [3].

1.3 既存のアプリケーション

既知の神経衰弱アプリとしては,[4][6] [7] 等がある.[4] は対 CPU 戦が可能で難易度調整はできないが, カード枚数を 3 パターン設定可能であり,[6][7] は一人用の神経衰弱であり, 連続取得回数や残り制限時間に応じて得点化される. ユーザインタフェースは使いやすく仕上がっているものの, 一度表になったカードのヒント機能に対応しているものは無い.

1.4 本研究の目的

前説で述べた通り, 既知のアプリケーション [4][6][7] はヒント機能などには対応していない. そこで本研究では集中力や記憶力を育む神経衰弱アプリの開発を目指し, 一度表になったカードのヒント機能などを実装することにより, より初心者にも遊びやすいアプリケーションを作成する.

1.5 本報告書の構成

本報告書の構成は以下の通りである.2 章で本研究対象である神経衰弱について説明する. 続く 3 章で, 本研究で作成した神経衰弱プログラムについて述べる.4 章において結果を示し, また考察を行う. 最後に 5 章で結

論及び今後の課題を述べる.

2　神経衰弱について

本章では, 本研究対象である神経衰弱について説明する.

2.1　神経衰弱の概要

神経衰弱は, 裏側に伏せられたカードをプレイヤー順にめくり, 同じ数字や記号であれば自分の得点とし, 全てカードを取り終えた後, 最もカードを多く取得していたプレイヤーの勝利となる.

2.2　神経衰弱のルール

本節では, 本研究での神経衰弱のルールについて述べる.

- ジョーカーを含めた 54 枚のカードを裏向きにしてランダムに並べる.
- プレイヤーはカードを 2 枚表にし, 同じ数字または記号であれば得点とし, もう一度プレイできる.2 枚が異なる場合, カードを裏側にし次のプレイヤーの順番となる.
- 1 組 2 点で他の要素によって得点が増減することはない.
- 全てのカードが表になるまで行い, 得点が多いプレイヤーの勝ちとなる.

3　研究内容

本研究では,Java を用いて神経衰弱アプリケーションを作成する. 本研究で作成するプログラムは, めくられたカードのうち一定枚数のカードのみを覚えることができる. 記憶可能上限までカードを覚えている時, 古いカードから忘れていき, 新しいカードを覚える. 記憶神経衰弱の難易度をカードの記憶方法の調節で開発していく. そこで, カードの記憶方法が異なる CPU 同士で対戦させ, 様々な勝率を求め, 初心者にも遊びやすい難易度に調整していく. 本研究で作成するアプリケーションは記憶したカードの中に数字が揃うものがあれば必ずそのカードをめくり,「パス」は行わない.

4　神経衰弱プログラム

本章では本研究で作成した神経衰弱アプリケーションについて説明する. 付録に本研究で作成した神経衰弱アプリケーションのソースコードを示す.

本研究で作成した神経衰弱アプリケーションは,CPU との対戦が行えるようになっている.

4.1　プログラムの仕様

本プログラムは全てマウス操作で行う. 図 1, 図 2 にプログラムの起動の様子を示す. プログラムを実行し, はじめからをクリックし, 次に難易度の選択を行う. Level1 が記憶枚数 6 枚,Level2 が記憶枚数 10 枚,Level3 が記憶枚数 14 枚となっており, 難易度選択をしなかった場合, 初期設定である, 全て記憶する CPU となる. プレ

イヤー側が先行でカードをクリックすると表になり,2 枚同じカードをめくった場合, もう 1 ターンプレイすることができる. 1 枚目を表にした際, その表にした数字が既に場で表になっていた場合, 一度でたカードとヒントが与えられる. カードが異なった場合, 相手にターンが渡り, 裏側のカードが無くなるまでこれを続け, ゲーム終了となる.

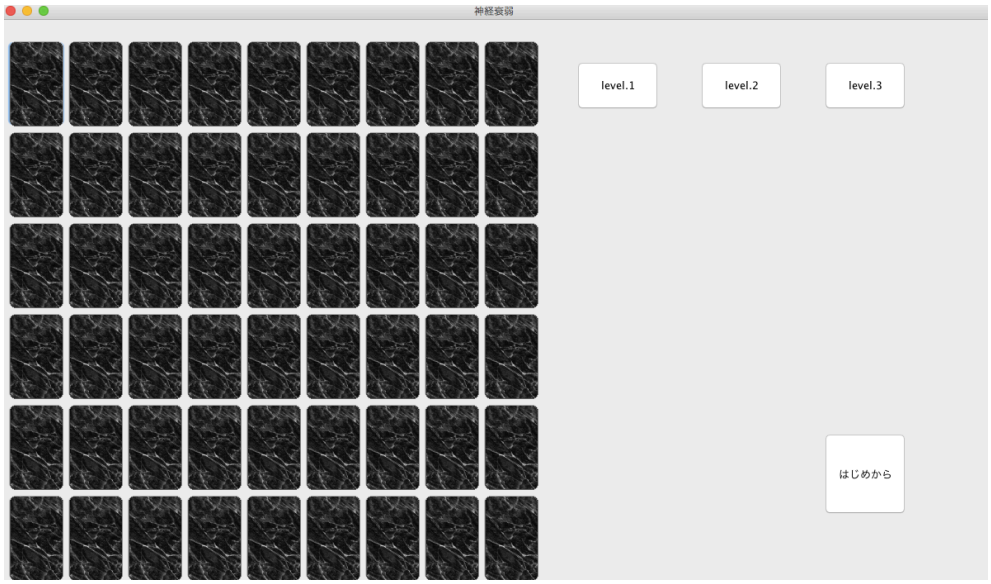

図 1　プログラム起動の様子

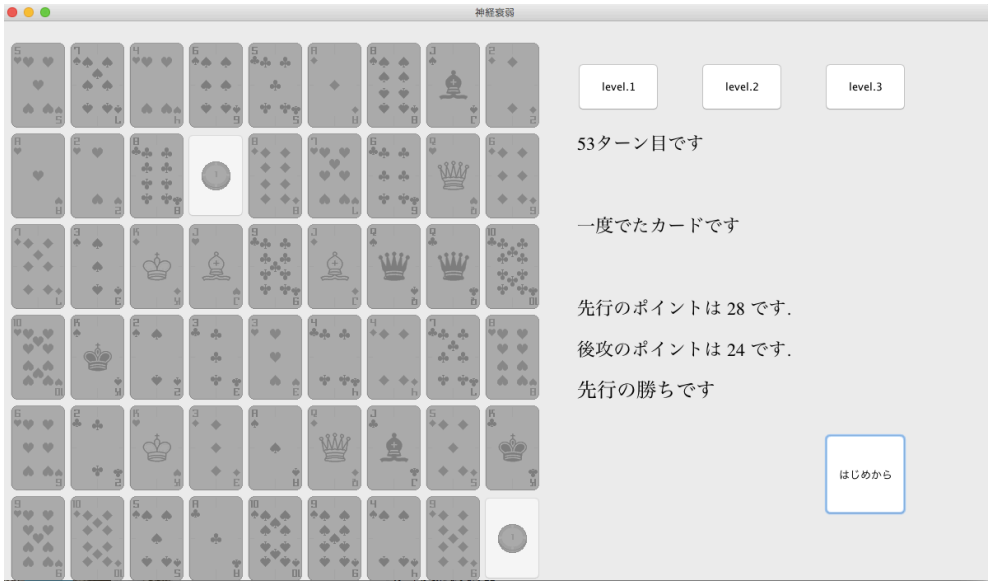

図 2　ゲーム終了の様子

4.2 SinkeiSuijaku.java

SinkeiSuijaku クラスは本研究で作成したプログラムの中心部分である.3 にクラス図を記載する.

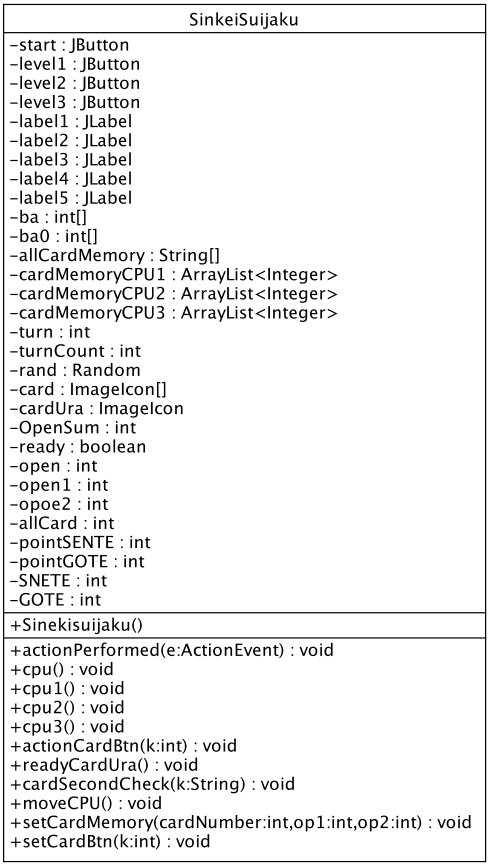

図 3　SinkeiSuijaku クラスのクラス図

4.2.1 actionPerformed メソッド

actionPerformed メソッドは, ボタンをクリックするとアクションイベントが発生する. プレイヤー側の操作の部分である.

4.2.2 cpu メソッド

cpu メソッドは, 全てのカードを記憶する CPU である. 難易度を選ばなかった場合, こちらの難易度となる.

4.2.3 cpu1 メソッド

cpu1 メソッドは, 表になったカードを 6 枚だけ記憶する CPU である. 難易度 level.1 を選んだ場合, こちらの難易度となる.

4.2.4 cpu2 メソッド

cpu2 メソッドは, 表になったカードを 10 枚だけ記憶する CPU である. 難易度 level.2 を選んだ場合, こちらの難易度となる.

4.2.5 cpu3 メソッド

cpu3 メソッドは, 表になったカードを 14 枚だけ記憶する CPU である. 難易度 level.3 を選んだ場合, こちらの難易度となる.

4.2.6 actionCardBtn メソッド

actionCardBtn メソッドは, カードを表にする動作のメソッドである.

4.2.7 readyCardUra メソッド

readyCardUra メソッドは, 表にしたカードを裏側に戻す動作のメソッドである.

4.2.8 cardSecondCheck

cardSecondCheck メソッドは, 一度表になったカードが再度表になった際にヒントを与える機能である.

4.2.9 moveCPU メソッド

moveCPU メソッドは, 指定された CPU メソッドを動かすメソッドである.

4.2.10 setCardMemory メソッド

setCardMemory メソッドは, 表にしたカードの記憶をするメソッドである

4.2.11 setCardBtn メソッド

setCardBtn メソッドは, カードが表になった際にカードの画像を貼り付けるメソッドである.

4.3 SinkeiSuijakuWindow.java

SinkeiSuijakuWindow クラスは神経衰弱アプリのウィンドウを表示させるクラスである.

5 結果と考察

本研究で作成したアプリの実行の様子を図 4 と図 5 に示す. 図 4 で一度 8 の数字が表になり, 図 5 で再度別の 8 の数字が表になった際に, 一度でたカードとヒントを表示する機能により, 対戦中に表になったカードをより思い出そうとする記憶力の手助けができると考えられる.

また本研究では,CPU が記憶する枚数に応じて CPU の難易度がどう変化するかを検証するために, めくられた数字を全て記憶する CPU と一定の枚数のみを記憶する CPU とで各 100 回ずつ対戦させ, 各 CPU の平均取得枚数を表 1 に示す. 記憶枚数を増やすことによって取得できる枚数も増えていることがわかり, カードの記憶枚数の増減は CPU の難易度と強い関わりがあることがわかる. また, 初心者に最適な難易度を検証す

るためには, 被験者を集めて統計を取る必要があると考えられる.

表 1　平均取得枚数 (試行回数 100 回)

記憶枚数	平均取得枚数
6	8.64 枚
10	12.12 枚
14	14.16 枚

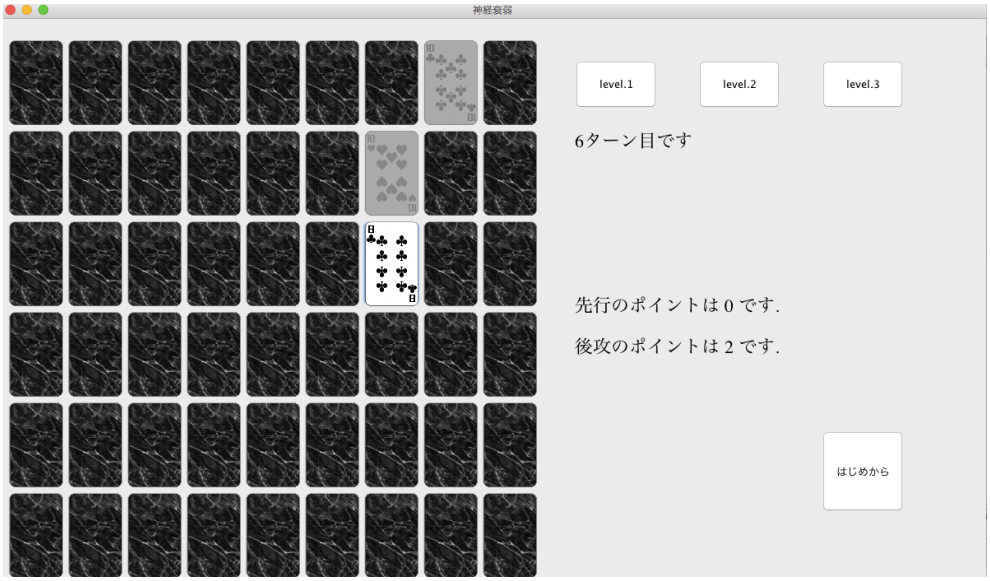

図 4　アプリ画面実行の様子 1

6　結論・今後の課題

本研究では Java を用いて神経衰弱のアプリケーションを開発した.CPU の難易度調整をカードの記憶枚数で作成したが,CPU の表にするカードにランダム要素が絡むので, 必ずしも初心者に易しい結果になっているとは言えない. 今後の課題としては, カードの記憶枚数以外の調整を行い, より遊びやすい難易度を作成することである. またより使いやすくするためのユーザインターフェースの改善等が今後の課題である.

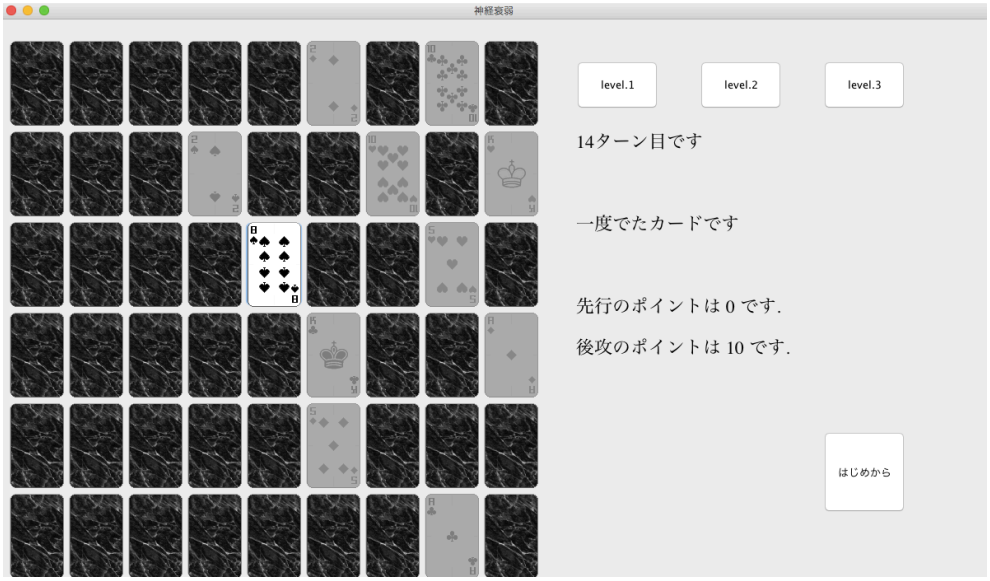

図 5　アプリ画面実行の様子 2

謝辞

本研究を行うにあたって, 石水隆講師から卒研レジュメや卒業論文の推敲, 資料の提供など様々なご指導を受けました. ここに感謝の意を表します.

参考文献

[1] 篠田正人:Wining strategy of the memory game, IPSJ Symposium Serise Vol.2008, No11,pp.181–188(2008)

[2] U.Zwick and M.S.Paterson : The memory games, Theoretical Computer Science 110, pp. 169–196 (1993)
https://www.sciencedirect.com/science/article/pii/030439759390355W

[3] 坂元香菜美，篠田正人：特別なカードを含む神経衰弱の勝率最大化戦略，情報処理学会 研究報告，Vol.2010-GI-24, No.7, pp.1–8 (2010)
https://ipsj.ixsq.nii.ac.jp/ej/?action=pages_view_main&active_action=repository_view_main_item_detail&item_id=69715&item_no=1&page_id=13&block_id=8

[4] Sonya Marcarelli : 神経衰弱。, AppleStore, 2021 年 11 月 15 日,
https://apps.apple.com/jp/app/神経衰弱/id540679659

[5] abe3 : 神経衰弱 (絵合わせ), Google Play, 2018 年 10 月 12 日,
https://play.google.com/store/apps/details?id=net.abe3.concentration.fruit

[6] Qualia Systems Inc. : 神経衰弱できるもん　おすしやさん, Apple Store, 2021 年 11 月 30 日,
https://apps.apple.com/jp/app/id458072561

[7] yumearu Co.,Ltd. : あそぼう神経衰弱　脳トレ記憶ゲーム, Apple Store, 2019 年,
https://apps.apple.com/jp/app/id1518708952

付録 A　ソースプログラム

本研究で作成した神経衰弱プログラムのソースを以下に示す..

```
  package sinkeisuijaku;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.Random;
import java.util.ArrayList;

public class SinkeiSuijaku extends JFrame implements ActionListener {
        // プレイするカード
        private JButton[] cardBtn = new JButton[54];
        // ゲームスタートボタン
        private JButton start = new JButtonはじめから("");
        // 表示文章
        private JLabel label1 = new JLabel();
        private JLabel label2 = new JLabel();
        private JLabel label3 = new JLabel();
        private JLabel label4 = new JLabel();
        private JLabel label5 = new JLabel();
        private JButton level1 = new JButton("level.1");
        private JButton level2 = new JButton("level.2");
        private JButton level3 = new JButton("level.3");
        // 初期化用
        private int[] ba = { 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 };
        // カードの番号を保存する場所
        private int[] ba0 = { 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 };
        // カードの数字を保存する場所
        private String[] ba2 = { "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" };

        // 全てのカードの記憶
        private String[] allCardMemory = { "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" };
```

```
// 記憶CPU2
private ArrayList<Integer> cardMemoryCPU2 = new ArrayList<>();
private ArrayList<Integer> cardMemoryCPU3 = new ArrayList<>();
private ArrayList<Integer> cardMemoryCPU4 = new ArrayList<>();
private int cpuLevel = 0;
// 先手後手
private int turn;
// ターン数
private int turnCount = 0;
private Random rand = new Random();
// カードの画像
private ImageIcon[] card = new ImageIcon[55];
private int openSum = 0;
private boolean ready = false;
// 表にした枚数
private int open = 0;
// 表にしたカードの
private int open1 = 0;
private int open2 = 0;
// カードの残り枚数
private int allCard = 54;
// 先手の得点
private int pointSENTE = 0;
// 後手の得点
private int pointGOTE = 0;
private int bug = 0;
private static final int SENTE = 1;
private static final int GOTE = -1;
// /Users/is/Documents/workspace/sinkeisuijaku/img
// カードの裏
private ImageIcon cardUra = new ImageIcon("/Users/is/
Documents/workspace/sinkeisuijaku/img/Card.png", "-1");

// ウィンドウの定義
public SinkeiSuijaku() {
        // クラブ
        card[1] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku/
        img/CardClub1.png", "1");
        card[2] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
        /img/CardClub2.png", "2");
        card[3] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
        /img/CardClub3.png", "3");
        card[4] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
        /img/CardClub4.png", "4");
        card[5] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
        /img/CardClub5.png", "5");
        card[6] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
        /img/CardClub6.png", "6");
        card[7] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku/
        img/CardClub7.png", "7");
        card[8] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
        /img/CardClub8.png", "8");
        card[9] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
```

```
/img/CardClub9.png", "9");
card[10] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardClub10.png", "10");
card[11] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardClubJ.png", "11");
card[12] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardClubQ.png", "12");
card[13] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardClubK.png", "13");
// ダイアモンド
card[14] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardDiamond1.png", "1");
card[15] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardDiamond2.png", "2");
card[16] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardDiamond3.png", "3");
card[17] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardDiamond4.png", "4");
card[18] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardDiamond5.png", "5");
card[19] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardDiamond6.png", "6");
card[20] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardDiamond7.png", "7");
card[21] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardDiamond8.png", "8");
card[22] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardDiamond9.png", "9");
card[23] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardDiamond10.png", "10");
card[24] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardDiamondJ.png", "11");
card[25] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardDiamondQ.png", "12");
card[26] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardDiamondK.png", "13");
// ハート
card[27] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardHeart1.png", "1");
card[28] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardHeart2.png", "2");
card[29] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardHeart3.png", "3");
card[30] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardHeart4.png", "4");
card[31] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardHeart5.png", "5");
card[32] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardHeart6.png", "6");
card[33] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardHeart7.png", "7");
card[34] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardHeart8.png", "8");
```

```
card[35] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardHeart9.png", "9");
card[36] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardHeart10.png", "10");
card[37] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardHeartJ.png", "11");
card[38] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardHeartQ.png", "12");
card[39] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardHeartK.png", "13");
// スペード
card[40] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardSpade1.png", "1");
card[41] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardSpade2.png", "2");
card[42] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardSpade3.png", "3");
card[43] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardSpade4.png", "4");
card[44] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardSpade5.png", "5");
card[45] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardSpade6.png", "6");
card[46] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardSpade7.png", "7");
card[47] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardSpade8.png", "8");
card[48] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardSpade9.png", "9");
card[49] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardSpade10.png", "10");
card[50] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardSpadeJ.png", "11");
card[51] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardSpadeQ.png", "12");
card[52] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/CardSpadeK.png", "13");
card[53] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/chip1.png", "14");
card[54] = new ImageIcon("/Users/is/Documents/workspace/sinkeisuijaku
/img/chip1.png", "14");
// カード配置のための変数
int x = 10;
int y = 25;
int yc = 8;
setLayout(null);
// はじめからのボタン
start.setSize(100, 100);
start.setLocation(1000, 500);
start.addActionListener(this);
// 文章のサイズフォント,
level1.setSize(100, 60);
level1.setLocation(700, 50);
```

```
        level1.addActionListener(this);
        level2.setSize(100, 60);
        level2.setLocation(850, 50);
        level2.addActionListener(this);
        level3.setSize(100, 60);
        level3.setLocation(1000, 50);
        level3.addActionListener(this);
        label1.setSize(1000, 100);
        label1.setLocation(700, 100);
        label1.setFont(new Font("Serif", Font.PLAIN, 22));
        label5.setSize(1000, 100);
        label5.setLocation(700, 200);
        label5.setFont(new Font("Serif", Font.PLAIN, 22));
        label2.setSize(1000, 100);
        label2.setLocation(700, 300);
        label2.setFont(new Font("Serif", Font.PLAIN, 22));
        label3.setSize(1000, 100);
        label3.setLocation(700, 350);
        label3.setFont(new Font("Serif", Font.PLAIN, 22));
        label4.setSize(1000, 100);
        label4.setLocation(700, 400);
        label4.setFont(new Font("Serif", Font.PLAIN, 24));
        add(start);
        add(level1);
        add(level2);
        add(level3);
        add(label1);
        add(label2);
        add(label3);
        add(label4);
        add(label5);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setTitle神経衰弱("");
        setSize(1200, 700);
        setVisible(true);
        // カードの配置
        for (int i = 0; i < 54; i++) {
                cardBtn[i] = new JButton();
                cardBtn[i].setSize(70, 104);
                cardBtn[i].setLocation(x, y);
                if (i == yc) {
                        y += 110;
                        x = 10;
                        yc += 9;
                } else {
                        x += 72;
                }
                cardBtn[i].setIcon(cardUra);
                cardBtn[i].addActionListener(this);
                add(cardBtn[i]);
        }
        repaint();
}
```

```
@Override
public void actionPerformed(ActionEvent e) {
        int num;
        // スタート
        if (e.getSource() == start) {
                while (openSum < 54) {
                        num = rand.nextInt(54);
                        if (ba[num] == 0) {
                                openSum++;
                                ba[num] += openSum;
                                ba2[num] =
                                card[openSum].getDescription();
                                System.out.print(openSum +
                                ":" + num + "=");
                                System.out.println(ba2[num]);
                                turn = SENTE;
                                ready = false;
                        }
                }
        } else if (e.getSource() == level1) {
                cpuLevel = 1;
        } else if (e.getSource() == level2) {
                cpuLevel = 2;
        } else if (e.getSource() == level3) {
                cpuLevel = 3;
        }
        readyCardUra();
        for (int k = 0; k < 54; k++) {
                if (e.getSource() == cardBtn[k]) {
                        actionCardBtn(k);
                }
        }
}

// 一度表にしたカードは全て取るそれ以外はランダムパスなし...
public void cpu() {
        int random = 0, random2 = 0;
        ready = true;
        System.out.println(turn + ":の番ですCPU1");
        // 場にあるカードが既に枚記憶しているならその枚を取得する22
        for (int o1 = 0; o1 < 54; o1++) {
          if (!(allCardMemory[o1] == "0") &&
          !(allCardMemory[o1] == "-1")) {
                        for (int o2 = 0; o2 < 54; o2++) {
                          if ((allCardMemory[o1] == allCardMemory[o2])
                          && !(o1 == o2) && ready) {
                                        actionCardBtn(o1);
                                        actionCardBtn(o2);
                                        ready = false;
                                        break;
                                }
                        }
```

```
                }
        }
        // 場に同じ枚記憶が無い場合枚をランダムに取る21
        if (ready) {
                do {
                        random = rand.nextInt(54);
                        bug++;
                        if (bug == 100000) {
                                System.out.println("owari");
                                break;
                        }

                } while (allCardMemory[random] == "-1");

                actionCardBtn(random);
                for (int ro = 0; ro < 54; ro++) {
                  if ((allCardMemory[ro] == allCardMemory[random])
                  && !(ro == random)) {
                                actionCardBtn(ro);
                                ready = false;
                                break;
                        }
                }
                // ランダムで取ったカードが記憶に無い場合もう枚もランダムで取る,1
                if (ready) {
                        do {
                                random2 = rand.nextInt(54);
                        } while (!(allCardMemory[random2] == "0") &&
                        !(random == random2));
                        actionCardBtn(random2);
                }
        }
}

// 記憶カード直前の枚6.
public void cpu1() {
        int random = 0, random2 = 0;
        ready = true;
        System.out.println(turn + ":の番ですCPU2");
        // 場にあるカードが既に枚記憶しているならその枚を取得する22
        for (int o1 = 0; o1 < cardMemoryCPU2.size(); o1++) {
          if (!(allCardMemory[cardMemoryCPU2.get(o1)] == "0") &&
          !(allCardMemory[cardMemoryCPU2.get(o1)] == "-1")) {
            for (int o2 = 0; o2 < cardMemoryCPU2.size();
            o2++) {
                        if (((allCardMemory[cardMemoryCPU2.get(o1)]
                        == allCardMemory[cardMemoryCPU2.get(o2)])
                        && !(o1 == o2)
                                        && ready)) {
                                actionCardBtn(o1);
                                actionCardBtn(o2);
                                ready = false;
                                break;
```

```
                                }
                        }
                }
        }
        // 場に同じ枚記憶が無い場合枚をランダムに取る21
        if (ready) {
                do {
                        random = rand.nextInt(54);
                        System.out.println"ランダム一つ目です""" + random);
                } while (!(allCardMemory[random] == "0"));

                actionCardBtn(random);
                System.out.println("uuu1");
                for (int ro = 0; ro < cardMemoryCPU2.size(); ro++) {
                  if ((allCardMemory[cardMemoryCPU2.get(ro)]
                  == allCardMemory[random])
                  && !(cardMemoryCPU2.get(ro)
                  == random)) {
                                System.out.println(random + "です"");
                                actionCardBtn(cardMemoryCPU2.get(ro));
                                ready = false;
                                break;
                        }
                }
                // ランダムで取ったカードが記憶に無い場合もう枚もランダムで取る,1
                if (ready) {
                        do {
                                random2 = rand.nextInt(54);
                        } while (!(allCardMemory[random2] == "0") &&
                        !(allCardMemory[random2] == "-1") &&
                        !(random == random2));
                        System.out.println"セーフ"("");
                        actionCardBtn(random2);
                }
        }
}

// cpu2
public void cpu2() {
        int random = 0, random2 = 0;
        ready = true;
        System.out.println(turn + ":の番ですCPU3");
        // ランダムで取ったカードが記憶に無い場合もう枚もランダムで取る,1
        for (int o1 = 0; o1 < cardMemoryCPU3.size(); o1++) {
          if (!(allCardMemory[cardMemoryCPU3.get(o1)] == "0") &&
          !(allCardMemory[cardMemoryCPU3.get(o1)] == "-1")) {
            for (int o2 = 0; o2 < cardMemoryCPU3.size();
            o2++) {
                        if (((allCardMemory[cardMemoryCPU3.get(o1)]
                        == allCardMemory[cardMemoryCPU3.get(o2)])
                        && !(o1 == o2)
                                        && ready)) {
                                actionCardBtn(o1);
```

```
                                        actionCardBtn(o2);
                                        ready = false;
                                        break;

                                }
                        }
                }
        }
        // 場に同じ枚記憶が無い場合枚をランダムに取る21
        if (ready) {
                do {
                        random = rand.nextInt(54);
                        System.out.println ランダム一つ目です("" + random);
                } while (!(allCardMemory[random] == "0") &&
                !(allCardMemory[random] == "-1"));

                actionCardBtn(random);
                for (int ro = 0; ro < cardMemoryCPU3.size(); ro++) {
                  if ((allCardMemory[cardMemoryCPU3.get(ro)] ==
                  allCardMemory[random])
                  && !(cardMemoryCPU3.get(ro
                  ) == random)) {
                                System.out.println(random + です"");
                                actionCardBtn(cardMemoryCPU3.get(ro));
                                ready = false;
                                break;
                        }
                }
                // ランダムで取ったカードが記憶に無い場合もう枚もランダムで取る,1
                if (ready) {
                        do {
                                random2 = rand.nextInt(54);
                        } while (!(allCardMemory[random2] == "0") &&
                        !(random == random2));
                        System.out.println セーフ("");
                        actionCardBtn(random2);
                }
        }
}

// cpu3
public void cpu3() {
        int random = 0, random2 = 0;
        ready = true;
        System.out.println(turn + ":の番ですCPU3");
        // ランダムで取ったカードが記憶に無い場合もう枚もランダムで取る,1
        for (int o1 = 0; o1 < cardMemoryCPU4.size(); o1++) {
          if (!(allCardMemory[cardMemoryCPU4.get(o1)] == "0") &&
          !(allCardMemory[cardMemoryCPU4.get(o1)] == "-1")) {
                        for (int o2 = 0; o2 < cardMemoryCPU3.size(); o2++) {
                          if (((allCardMemory[cardMemoryCPU4.get(o1)]
                          == allCardMemory[cardMemoryCPU4.get(o2)]
                          ) && !(o1 == o2)
```

```
                                        && ready)) {
                                actionCardBtn(o1);
                                actionCardBtn(o2);
                                ready = false;
                                break;
                        }
                }
        }
        // 場に同じ枚記憶が無い場合枚をランダムに取る21
        if (ready) {
                do {
                        random = rand.nextInt(54);
                        System.out.println"ランダム一つ目です"("" + random);
                } while (!(allCardMemory[random] == "0") &&
                !(allCardMemory[random] == "-1"));

                actionCardBtn(random);
                for (int ro = 0; ro < cardMemoryCPU4.size(); ro++) {
                  if ((allCardMemory[cardMemoryCPU4.get(ro)]
                  == allCardMemory[random])
                  && !(cardMemoryCPU4.get(ro)
                  == random)) {
                                System.out.println(random + "です"");
                                actionCardBtn(cardMemoryCPU4.get(ro));
                                ready = false;
                                break;
                        }
                }
                // ランダムで取ったカードが記憶に無い場合もう枚もランダムで取る,1
                if (ready) {
                        do {
                                random2 = rand.nextInt(54);
                        } while (!(allCardMemory[random2] == "0")
                        && !(random == random2));
                        System.out.println"セーフ"("");
                        actionCardBtn(random2);
                }
        }
}

// カードを表にする動作
public void actionCardBtn(int k) {
        // 枚目を表にする1
        if (open == 0) {
                turnCount++;
                System.out.println(turnCount + "ターン目です"");
                label1.setText(turnCount + "ターン目です"");
                open1 = k;
                open++;
                cardSecondCheck(ba2[k]);
                setCardMemory(open1, -100, -100);
                ba0[open1] = 1;
```

```
        System.out.println(turn + ":" + 一枚目です":" + ba2[open1]);
        setCardBtn(k);
        // 枚目を表にする2
} else if (open == 1) {
        open2 = k;
        open++;
        setCardMemory(open2, -100, -100);
        ba0[open2] = 1;
        System.out.println(turn + ":" + 二枚目です":" + ba2[open2]);
        setCardBtn(k);
        // 枚目と枚目が同じ場合12
        if (ba2[open1] == ba2[open2] && open == 2) {
                System.out.println正解("");
                // 得点追加
                if (turn == 1)
                        pointSENTE += 2;
                else if (turn == -1)
                        pointGOTE += 2;
                ba0[open1] = 2;
                ba0[open2] = 2;
                cardBtn[open1].setEnabled(false);
                cardBtn[open2].setEnabled(false);
                allCardMemory[open1] = "-1";
                allCardMemory[open2] = "-1";
                open = 0;
                open1 = 0;
                open2 = 0;
                allCard -= 2;
                if (turn == -1)
                        cpu1();
                // ゲーム終了
                if (allCard == 0) {
                        if (pointSENTE > pointGOTE) {
                          System.out.println先行(":" + pointSENTE +後攻
                          ",:" + pointGOTE + " 先行の勝ちです.");
                          label4.setText先行(":" + pointSENTE + ",後攻
                          :" + pointGOTE + " 先行の勝ちです.");
                        } else if (pointSENTE < pointGOTE) {
                          System.out.println先行(":" + pointSENTE +後攻
                          ",:" + pointGOTE + " 後攻の勝ちです.");
                          label4.setText先行(":" + pointSENTE + ",後攻
                          :" + pointGOTE + " 後攻の勝ちです.");
                        } else if (pointSENTE == pointGOTE) {
                          System.out.println先行(":" + pointSENTE +後攻
                          ",:" + pointGOTE + " 引き分けです.");
                          label4.setText先行(":" + pointSENTE + "後攻
                          ,:" + pointGOTE + " 引き分けです.");
                        }
                        System.out.printlnゲームを終了します("");
                }
                // 枚目と枚目が異なる場合12
        } else {
                System.out.println不正解("");
```

```
                        ba0[open1] = 0;
                        ba0[open2] = 0;
                        open = 0;
                        open1 = 0;
                        open2 = 0;
                        // ターン交代
                        if (turn == 1) {
                                turn = GOTE;
                                System.out.println先行のポイントは(" " +
                                pointSENTE + " です後攻のターンに移ります.");
                                label2.setText先行のポイントは(" " +
                                pointSENTE + " です.");
                                moveCPU();
                        } else if (turn == -1) {
                                turn = SENTE;
                                System.out.println後攻のポイントは(" " +
                                pointGOTE + " です先行のターンに移ります.");
                                label3.setText後攻のポイントは(" " +
                                pointGOTE + " です.");
                        }
                }
        }
}

// 表にしたカードを裏にする
public void readyCardUra() {
        if (open1 == 0 && open2 == 0) {
                for (int n = 0; n < 54; n++) {
                        if (ba0[n] == 0) {
                                cardBtn[n].setIcon(cardUra);
                        }
                }
        }
}

// 一度表になったカードが再度表になった場合知らせてくれるヒント機能
public void cardSecondCheck(String k) {
        label5.setText("");
        for (int b = 0; b < 54; b++) {
                if (allCardMemory[b] == k) {
                        System.out.println一度でたカードです("");
                        label5.setText一度でたカードです("");
                }
        }
}

public void moveCPU() {
        if (cpuLevel == 0)
                cpu();
        else if (cpuLevel == 1) {
                cpu1();
        } else if (cpuLevel == 2) {
                cpu2();
```

```
        } else if (cpuLevel == 3) {
                cpu3();
        }
}

// 表にしたカードの記憶
public void setCardMemory(int cardNumber, int op1, int op2) {
        if ((op1 > 0 && op2 > 0)) {
                allCardMemory[op1] = "-1";
                allCardMemory[op2] = "-1";
                cardMemoryCPU2.remove(cardMemoryCPU2.indexOf(op1));
                cardMemoryCPU2.remove(cardMemoryCPU2.indexOf(op2));
                cardMemoryCPU3.remove(cardMemoryCPU3.indexOf(op1));
                cardMemoryCPU3.remove(cardMemoryCPU3.indexOf(op2));
                cardMemoryCPU3.remove(cardMemoryCPU4.indexOf(op1));
                cardMemoryCPU3.remove(cardMemoryCPU4.indexOf(op2));
        } else {
                allCardMemory[cardNumber] = ba2[cardNumber];
                if (cardMemoryCPU2.size() > 6) {
                        cardMemoryCPU2.remove(0);
                }
                if (cardMemoryCPU3.size() > 10) {
                        cardMemoryCPU3.remove(0);
                }
                if (cardMemoryCPU4.size() > 14) {
                        cardMemoryCPU4.remove(0);
                }

                cardMemoryCPU2.add(cardNumber);
                System.out.println(cardMemoryCPU2);
                cardMemoryCPU3.add(cardNumber);
                System.out.println(cardMemoryCPU3);
                cardMemoryCPU4.add(cardNumber);
                System.out.println(cardMemoryCPU4);
        }
}

// カードを表にする
public void setCardBtn(int k) {
        System.out.printlnカードが表になります("" + ba[k]);
        switch (ba[k]) {
        case 1:
                cardBtn[k].setIcon(card[1]);
                break;
        case 2:
                cardBtn[k].setIcon(card[2]);
                break;
        case 3:
                cardBtn[k].setIcon(card[3]);
                break;
        case 4:
                cardBtn[k].setIcon(card[4]);
                break;
```

```
case 5:
        cardBtn[k].setIcon(card[5]);
        break;
case 6:
        cardBtn[k].setIcon(card[6]);
        break;
case 7:
        cardBtn[k].setIcon(card[7]);
        break;
case 8:
        cardBtn[k].setIcon(card[8]);
        break;
case 9:
        cardBtn[k].setIcon(card[9]);
        break;
case 10:
        cardBtn[k].setIcon(card[10]);
        break;
case 11:
        cardBtn[k].setIcon(card[11]);
        break;
case 12:
        cardBtn[k].setIcon(card[12]);
        break;
case 13:
        cardBtn[k].setIcon(card[13]);
        break;
case 14:
        cardBtn[k].setIcon(card[14]);
        break;
case 15:
        cardBtn[k].setIcon(card[15]);
        break;
case 16:
        cardBtn[k].setIcon(card[16]);
        break;
case 17:
        cardBtn[k].setIcon(card[17]);
        break;
case 18:
        cardBtn[k].setIcon(card[18]);
        break;
case 19:
        cardBtn[k].setIcon(card[19]);
        break;
case 20:
        cardBtn[k].setIcon(card[20]);
        break;
case 21:
        cardBtn[k].setIcon(card[21]);
        break;
case 22:
        cardBtn[k].setIcon(card[22]);
```

```
        break;
case 23:
        cardBtn[k].setIcon(card[23]);
        break;
case 24:
        cardBtn[k].setIcon(card[24]);
        break;
case 25:
        cardBtn[k].setIcon(card[25]);
        break;
case 26:
        cardBtn[k].setIcon(card[26]);
        break;
case 27:
        cardBtn[k].setIcon(card[27]);
        break;
case 28:
        cardBtn[k].setIcon(card[28]);
        break;
case 29:
        cardBtn[k].setIcon(card[29]);
        break;
case 30:
        cardBtn[k].setIcon(card[30]);
        break;
case 31:
        cardBtn[k].setIcon(card[31]);
        break;
case 32:
        cardBtn[k].setIcon(card[32]);
        break;
case 33:
        cardBtn[k].setIcon(card[33]);
        break;
case 34:
        cardBtn[k].setIcon(card[34]);
        break;
case 35:
        cardBtn[k].setIcon(card[35]);
        break;
case 36:
        cardBtn[k].setIcon(card[36]);
        break;
case 37:
        cardBtn[k].setIcon(card[37]);
        break;
case 38:
        cardBtn[k].setIcon(card[38]);
        break;
case 39:
        cardBtn[k].setIcon(card[39]);
        break;
case 40:
```

```
                        cardBtn[k].setIcon(card[40]);
                        break;
                case 41:
                        cardBtn[k].setIcon(card[41]);
                        break;
                case 42:
                        cardBtn[k].setIcon(card[42]);
                        break;
                case 43:
                        cardBtn[k].setIcon(card[43]);
                        break;
                case 44:
                        cardBtn[k].setIcon(card[44]);
                        break;
                case 45:
                        cardBtn[k].setIcon(card[45]);
                        break;
                case 46:
                        cardBtn[k].setIcon(card[46]);
                        break;
                case 47:
                        cardBtn[k].setIcon(card[47]);
                        break;
                case 48:
                        cardBtn[k].setIcon(card[48]);
                        break;
                case 49:
                        cardBtn[k].setIcon(card[49]);
                        break;
                case 50:
                        cardBtn[k].setIcon(card[50]);
                        break;
                case 51:
                        cardBtn[k].setIcon(card[51]);
                        break;
                case 52:
                        cardBtn[k].setIcon(card[52]);
                        break;
                case 53:
                        cardBtn[k].setIcon(card[53]);
                        break;
                case 54:
                        cardBtn[k].setIcon(card[54]);
                        break;
                }
        }
}

package sinkeisuijaku;

public class SinkeiSuijakuWindow {
        private static SinkeiSuijaku sinkeisuijaku;
```

```
        public static void main(String args[]) {
                sinkeisuijaku = new SinkeiSuijaku();
                sinkeisuijaku.setVisible(true);
        }
}
```