トランプゲームを作ってみよう (JavaFX 編)

文責　：　高知大学名誉教授　中村　治

ここでは二人でプレイするカード・ゲーム「ジャーマン・ホイスト」を JavaFX で作ってみます。

ジャーマン・ホイストはホイストのバリエーションの一つで、2 人でプレイするように工夫されたものです。松田道弘著「トランプゲーム辞典」東京堂出版に載っています。

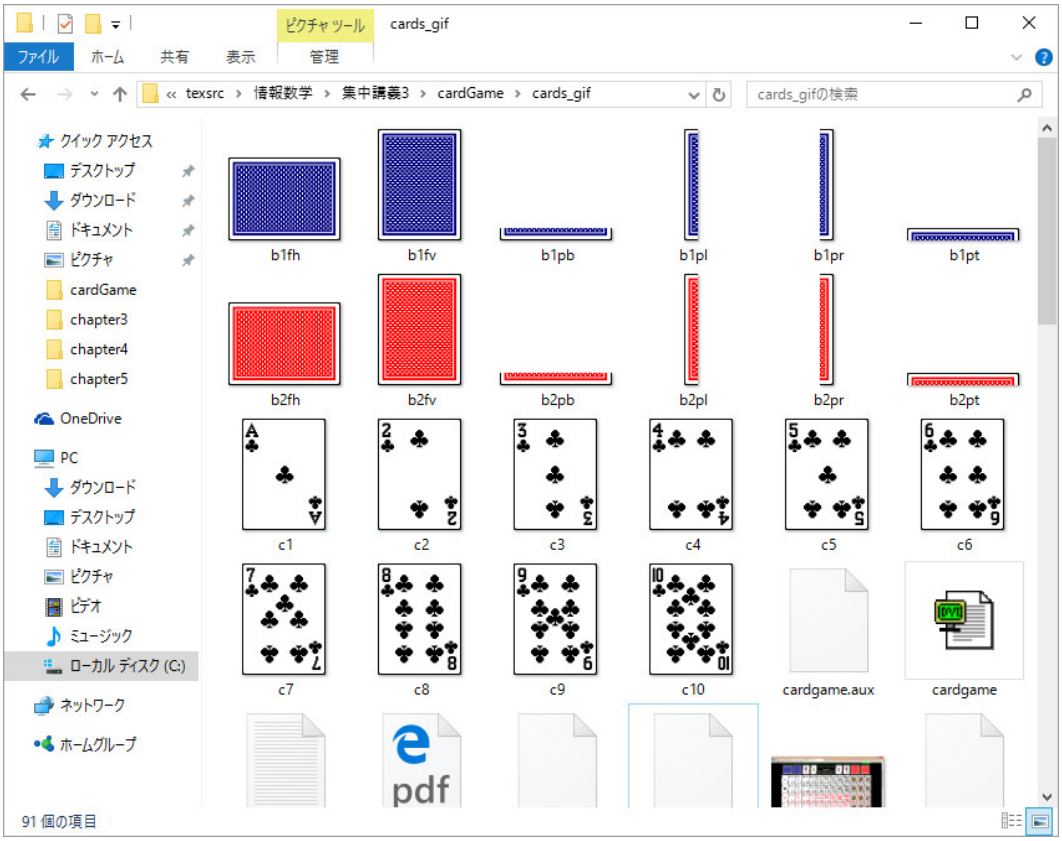
ルールは

1. 人数は 2 人
2. 52 枚のカードを使用する
3. カードの順位は A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2 の順です
4. 手札は各人 13 枚で、残りはストックとしてテーブルに置き、ストックのトップカードだけを表向きにします。このカードのスーツが切り札になります。
5. ディーラーでない人がオープニング・リードをします。
6. リードされたスーツは必ず出さなければなりません。 リードされたスーツを持っていなければ、どのカードを出しても良いです。
7. 最強の切り札を出した人か、または切り札が場に出ていなければリードされたスーツの一番高いランクのカードを出した人がトリックを取ります。
8. トリックを取った人は、ストックのトップ・カード（切り札）を取って手札に入れ、相手はその下のカード（裏向き）を取ります。相手には見せません。
9. 各プレーヤーの手札は 13 枚に戻りました。この時、トリックを取った人は、次のリードをする前に、ストックの一番上のカードを表向きにします。
10. 第 l トリックを取った人がリードして、このトリックの勝者がストックの表向きのカードを取り、相手がその下のカードを取ります。
11. この手順をストックが無くなるまで続け、その後は手札が無くなるまでトリック争奪のプレイを続けます。
12. プレイが終わると取得トリックの数を比べ、勝った人はその差を得点します。トリック数が同数の時は両者無得点です。

です。

このジャーマン・ホイストのプログラムを Java で作成し、コンピュータと対戦できるようにします。

インターネットで無料素材のトランプの画像（.gif）を手に入れます。



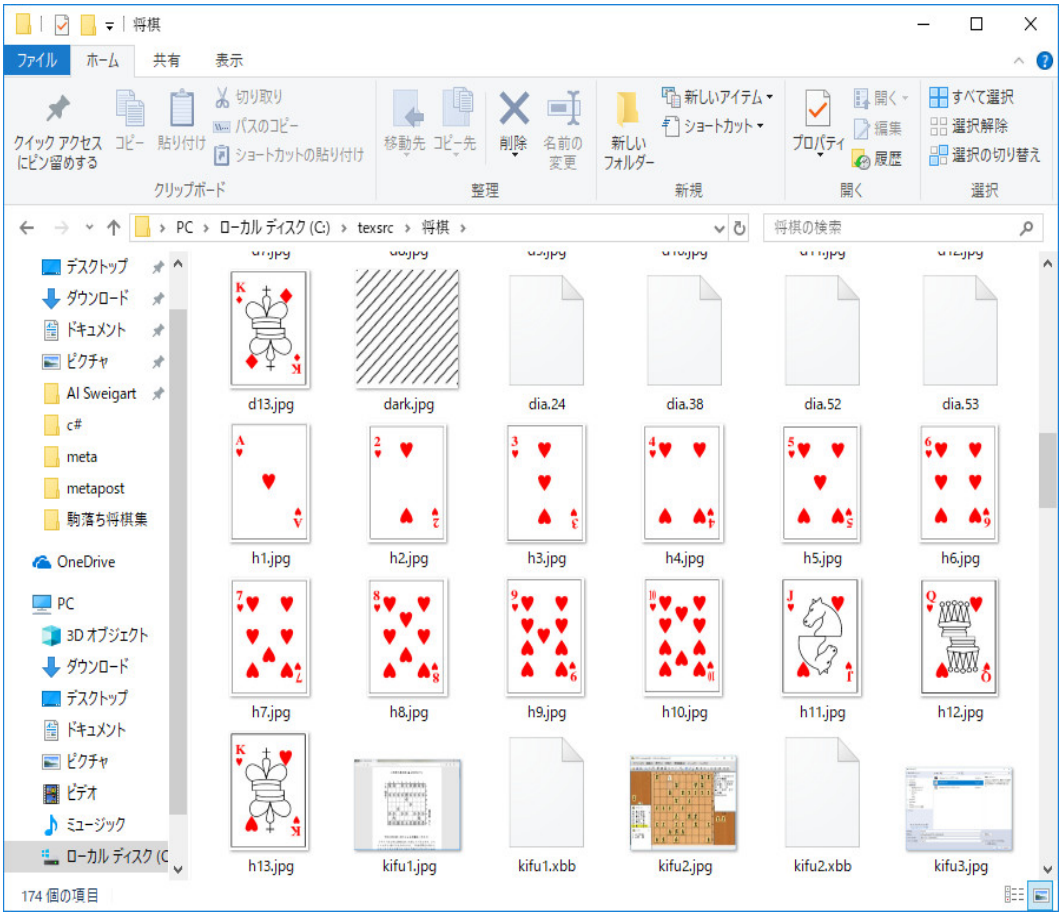
自分だけで楽しむのでなければ、GIMP などで、例えば、スーツと数字だけの簡単なオリジナルな画像を作れば良いです。

LaTeX と metapost と GIMP を使って、トランプの画像（.gif）を自分で作る方法を「情報数学の資料」のページの「Python による音声データの有効利用と LaTeX による将棋・囲碁カード作成補助の C#のソフト作成とカード画像作成」

`http:/www.cc.kochi-u.ac.jp/~tyamag/jyohou/kifumodify.pdf`

の pdf ファイルの最後に説明してあります。私は著作権を主張しませんから、それをそのまま使ってもいいですし、自分で好みの画像に改良して使ってもいいです。

私の作った metapost のプログラムでは次のような画像が作れます。

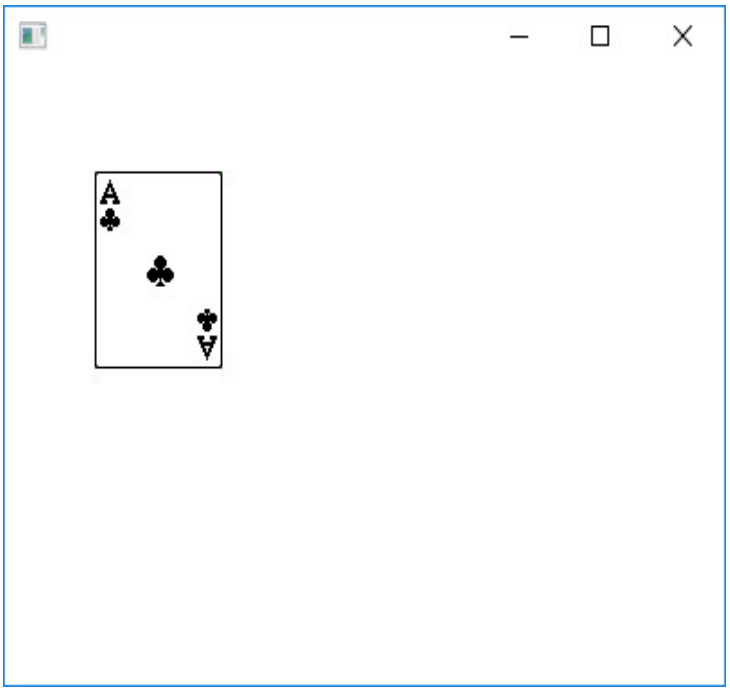


やっつけ仕事ですから、スーツのスペードやクラブやハートの形や大きさや絵札の絵が気に入らなければ、自分でプログラムを修正してください。

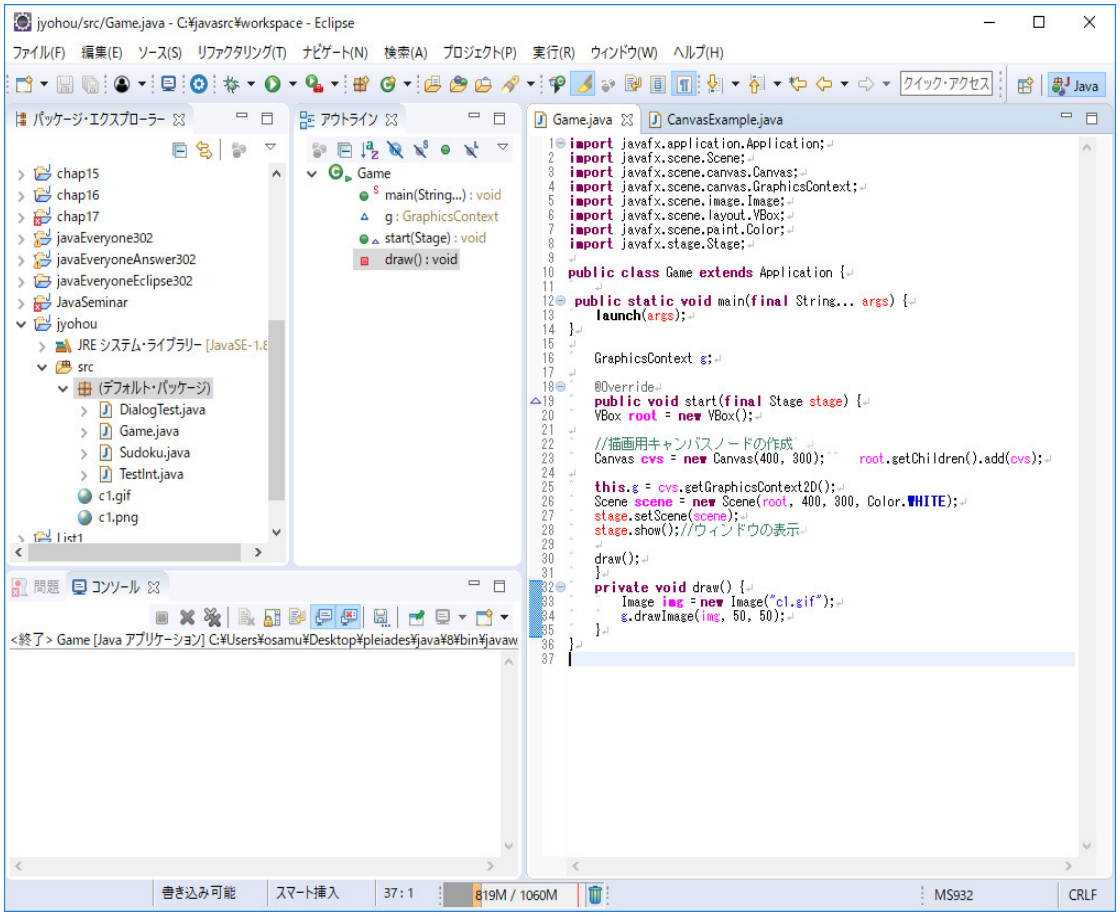
```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.image.Image;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.stage.Stage;

public class Game extends Application {

  public static void main(final String... args) {
    launch(args);
  }

  GraphicsContext g;
```

```
  @Override
   public void start(final Stage stage) {
     VBox root = new VBox();

     //描画用キャンバスノードの作成
     Canvas cvs = new Canvas(400, 300);
     root.getChildren().add(cvs);
     this.g = cvs.getGraphicsContext2D();
     Scene scene = new Scene(root, 400, 300, Color.WHITE);
     stage.setScene(scene);
     stage.show();//ウィンドウの表示

     draw();
   }
   private void draw() {
     Image img = new Image("c1.gif");
     g.drawImage(img, 50, 50);
   }
}
```

で、カードを表示できます。



ここで、Eclipse を使っている時は



のように、デフォルトパッケージに c1.gif をコピーしておく必要があります。Eclipse はインポートすべきファイルなど教えてくれて便利ですが、使いかたになれる必要があります。`*.gif` だけでなく `*.png` なども使えます。

コマンドラインで、コンパイルするには、エディタ Atom などで、プログラミングして、

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

のようにコンパイルします。この場合、画像ファイルは Game.java のある所に置いておきます。TeraPad でプログラムを作ると上手くいきませんでした。Python や Ruby に比べるとかなり厄介です。ここまで、出来るようになるのに数時間必要でした。

ここからは、私の Ruby/Tk のプログラムを真似して、Java のプログラムを作っていきます。Ruby と違って、Java は純なオブジェクト指向言語なので、そのまま翻訳すればいいというわけにはいきません。VC++ と Python と JavaScript で作った「ジャーマン・ホイスト」は非オブジェクト指向プログラムだし、Ruby のプログラムはクラスはいっぱい作りましたが、いかにも素人が作ったというグローバル変数を多用した変なプログラムでした。プロは小森裕介著「なぜ、あなたは Java でオブジェクト指向開発ができないか」技術評論社の Lesson 6 の「ババ抜き」のプログラムのように、カード・ゲーム「ジャーマン・ホイスト」を分析し、クラスにすべきものを洗い出し、クラス図やシーケンス図を描いて、全体の構想を描いてから、プログラミングするのでしょうが、私は素人ですからそんなめんどくさいことはしません。「なぜ、あなたは Java でオブジェクト指向開発ができないか」を頭の片隅に置きながら、思いついたことを順に泥縄式に、一つ一つ正

しいか確かめながら作っていきます。せいぜい 500 行ぐらいのプログラムになる予定ですから、これで大丈夫なはずです。実はカード・ゲーム「ジャーマン・ホイスト」を Java で作るのは初めてではありません。カード・ゲームのプログラムを初めて作ったのは、何年も前に、Java アプレットとして、非オブジェクト指向で、Python のプログラムのようなものを作りました。Java アプレットがインターネットで閲覧できなくなったので、VC++, Python, JavaScript, Ruby と新しい言語を学ぶたびに、その言語で独力でプログラムが作れるか、練習のために「数独」のヒントを表示するプログラムとカード・ゲーム「ジャーマン・ホイスト」のプログラムを作ってきました（勉強した順番は Java を勉強する前に、C++ を勉強していましたが)。今まで通り小さな目標を設定しながら、段階的にプログラミングしていきます。

任意のカードを表示できるようにしましょう。最低限のクラス定義を使う方法もありますが、Java はオブジェクト指向言語なので、練習のため、沢山クラスを定義して、使ってみましょう。まず Card のクラスを作ります。カードには、スーツ（♣, ♢, ♡, ♠）とランク（A,2,3,4,5,6,7,8,9,10, J,Q,K）と表の画像と裏の画像の属性があります。カード・ゲーム「ジャーマン・ホイスト」ではジョーカーを使いません。従って、次のようなクラスを作ります。

```
import javafx.scene.image.Image;

public class Card {
  public String suit;
  public int rank;
  public Image image;
  public Image bkimage;

  Card(String suit, int rank, Image image, Image bkimage) {
    this.suit = suit;
    this.rank = rank;
    this.image = image;
    this.bkimage = bkimage;
  }
}
```

をファイル名 Card.java で作ります。

次にすべてのカードを保持するクラス CardDeck を

```
import java.util.ArrayList;
import java.util.List;
import javafx.scene.image.Image;

public class CardDeck {
  private List<Card> deck = new ArrayList<Card>();

  public CardDeck() {
    Image backImage = new Image("b1fv.gif");
    String[] suitStr = {"c", "d", "h", "s"};
```

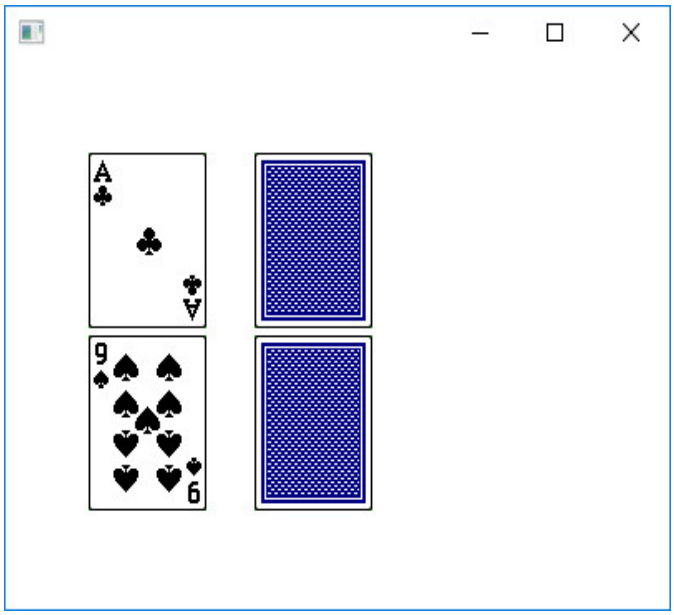
```
    for (String s: suitStr) {
      for (int r=1; r<=13; r++) {
        StringBuffer string = new StringBuffer();
        string.append(s);
        if (r<=10) {
          string.append(String.valueOf(r));
        } else if (r ==11) {
          string.append("J");
        } else if (r ==12) {
          string.append("Q");
        } else if (r ==13) {
          string.append("K");
        }
        string.append(".gif");
        Image image = new Image(string.toString());
        Card card = new Card(s, r, image, backImage);
        deck.add(card);
      }
    }
  }
  public Card getCard(int index) {
    return deck.get(index);
  }
  public void shuffle() {
    int num = deck.size();
    int pos;
    for (int count=0; count<num*5; count++) {
      pos = (int)(Math.random()*num);
      Card pickedCard = (Card)deck.remove(pos);
      deck.add(pickedCard);
    }
  }
  public Card draw() {
    Card card = (Card)deck.remove(0);
    return card;
  }
}
```

をファイル名 CardDeck.java で作ります。52 枚の Card を作り、メソッド shuffle() と getCard(int index) と draw() を作っています。getCard(int index) と draw() とどっちが必要になるか（便利か）判断がつかないので、両方作っています。クラス CardDeck の役割は、ディーラーがカードを配り終わると役割が終了します。

この二つのクラスの定義が間違ってないかクラス Game を

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.image.Image;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.stage.Stage;

public class Game extends Application {
  public static void main(final String... args) {
    launch(args);
  }

  GraphicsContext g;

  @Override
  public void start(final Stage stage) {
    VBox root = new VBox();
    Canvas cvs = new Canvas(400, 300);
    root.getChildren().add(cvs);

    this.g = cvs.getGraphicsContext2D();
    Scene scene = new Scene(root, 400, 300, Color.WHITE);
    stage.setScene(scene);
    stage.show();
    draw();
  }
  private void draw() {
    CardDeck carddeck = new CardDeck();
    Card card = carddeck.getCard(0);
    Image img = card.image;
    g.drawImage(img, 50, 50);
    Image bkimg = card.bkimage;
    g.drawImage(bkimg, 150, 50);
    carddeck.shuffle();
    card = carddeck.draw();
    img = card.image;
    g.drawImage(img, 50, 150);
    bkimg = card.bkimage;
    g.drawImage(bkimg, 150, 150);
  }
}
```

と修正します。実行すると



です。実行するたびに、下に表示されるカードは異なります。上手くいっているみたいです。

クラス Game を作り直して、ゲームができるようにします。プロのプログラマーではないので、 私のようなアマチュアのプログラマーは今までに作ったことのないプログラムを、計画性なしに、思いついた順に、適当にプログラミングしています。プログラムが長くなりますが、楽なことをし ないで、勉強したことを片っ端から使って、本当に理解できているか確かめて行きます。古い役に立たなくなった Java の参考書はいっぱいありますが、私の現在の Java の参考書は、立木秀樹、有賀妙子著「すべての人のための Java プログラミング　第 3 版」共立出版ですが、数学の学生さんが勉強するには、例題が面白いですが、初めて Java を勉強する人が独学で勉強するには、ちょっときついかもわかりません。 しかも、この本の情報だけでは、自分で、「数独」や「カードゲーム」のプログラミングをするのは無理で、この本の情報を元にインターネットで `private List<Card> deck = new ArrayList<Card>();` の使いかたなど調べながら、何とかプログラミングしています。Java は Python や Ruby に比べると勉強すべきことの範囲が広いです。

ディーラーとディーラーでない人と二人の人がゲームをするので、この二人もクラスにしてみます。共通の属性があるので、クラス Player、Dealer、User のひな型をまず作ります。ディーラーとディーラーでない人と二人とも配られた手札を持っていて、順番にある規則に従ってカードを出していくので、クラス Player の最初のひな型として

```
import java.util.ArrayList;
import java.util.List;

public class Player {
  private List<Card> hand = null;

  public Player() {
    hand = new ArrayList<Card>();
  }
```

```
  public void getCard(Card card) {
    hand.add(card);
  }
  public Card draw() {
    Card card = (Card)hand.remove(0);
    return card;
  }
}
```

と定義し、ファイル名 Player.java で保存します。次に、ディーラーとディーラーでない人と二人のクラスをクラス Player を継承したクラスとして作ります。現時点では、単にクラス Player を継承しただけのクラスとします。ディーラーはカードをシャッフルし、配る人ですが、そんな機能は付けません。多分、クラス Game で配ることを想定しています。プログラムは現実通り作る必要はないです。

```
public class Dealer extends Player {
public Dealer() {
super();
}
}
```

と

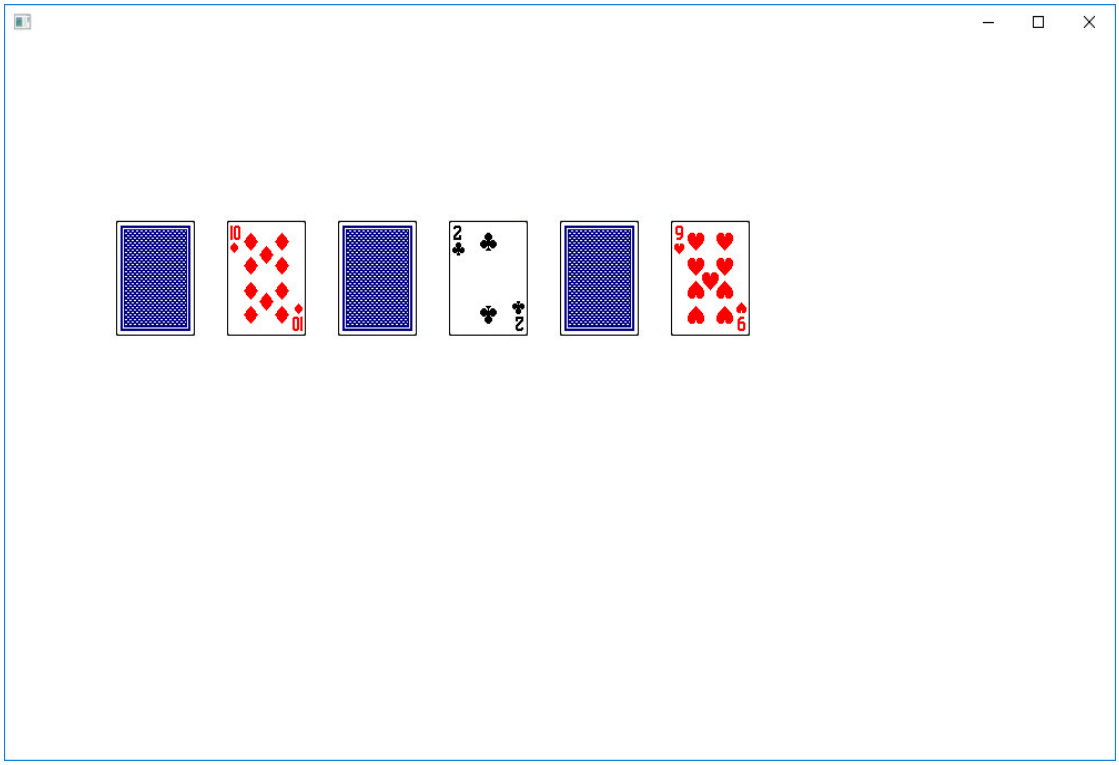
```
public class User extends Player{
public User() {
super();
}
}
```

と定義しておきます。必要な機能は、必要になれば追加していきます。ここまでのプログラミングが間違っていないか確かめるために、Game を修正します。メソッド run() で Dealer と User を登録し、それぞれに手札を 13 枚配ります。また、キャンバスの大きさを 1000 × 600 にします。

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.image.Image;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class Game extends Application {

  public static void main(final String... args) {
    launch(args);
  }
```

```
  GraphicsContext g;
  Dealer dealer;
  User user;

  @Override
  public void start(final Stage stage) {
    VBox root = new VBox();
    Canvas cvs = new Canvas(1000, 600);
    root.getChildren().add(cvs);

    this.g = cvs.getGraphicsContext2D();
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.show();
    run();
  }
  private void run() {
    dealer = new Dealer();
    user = new User();
    CardDeck carddeck = new CardDeck();
    carddeck.shuffle();
    for (int i=0; i<13; i++) {
      dealer.getCard(carddeck.getCard(2*i));
      user.getCard(carddeck.getCard(2*i+1));
    }
    for (int i=0; i<3; i++) {
      Card card = dealer.draw();
      Image bkimg = card.bkimage;
      g.drawImage(bkimg, 100+200*i, 150);
      card = user.draw();
      Image img = card.image;
      g.drawImage(img, 200+200*i, 150);
    }
  }
}
```

実行すると



です。正しいみたいです。クラス Dealer と User に、保持している手札を表示するメソッド display() を定義します。まず、クラス Player を

```
import java.util.ArrayList;
import java.util.List;

import javafx.scene.canvas.GraphicsContext;

public class Player {
  protected List<Card> hand = null;
  GraphicsContext gc;

  public Player(GraphicsContext g) {
    hand = new ArrayList<Card>();
    this.gc = g;
  }
  public void getCard(Card card) {
    hand.add(card);
  }
  public Card draw() {
    Card card = (Card)hand.remove(0);
    return card;
  }
}
```

とコンストラクターを修正し、GraphicsContext gc を保持すっるようにします。クラス Dealer を

```
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.image.Image;

public class Dealer extends Player {
  public Dealer(GraphicsContext g) {
    super(g);
  }
  public void display() {
    int len = hand.size();
    for (int i=0; i<len; i++) {
      Card card = hand.get(i);
      Image bkimg = card.bkimage;
      gc.drawImage(bkimg, 45+72*i, 65);
    }
  }
}
```

と修正し、保持しているカードを裏返して画面の上部に表示します。クラス User を

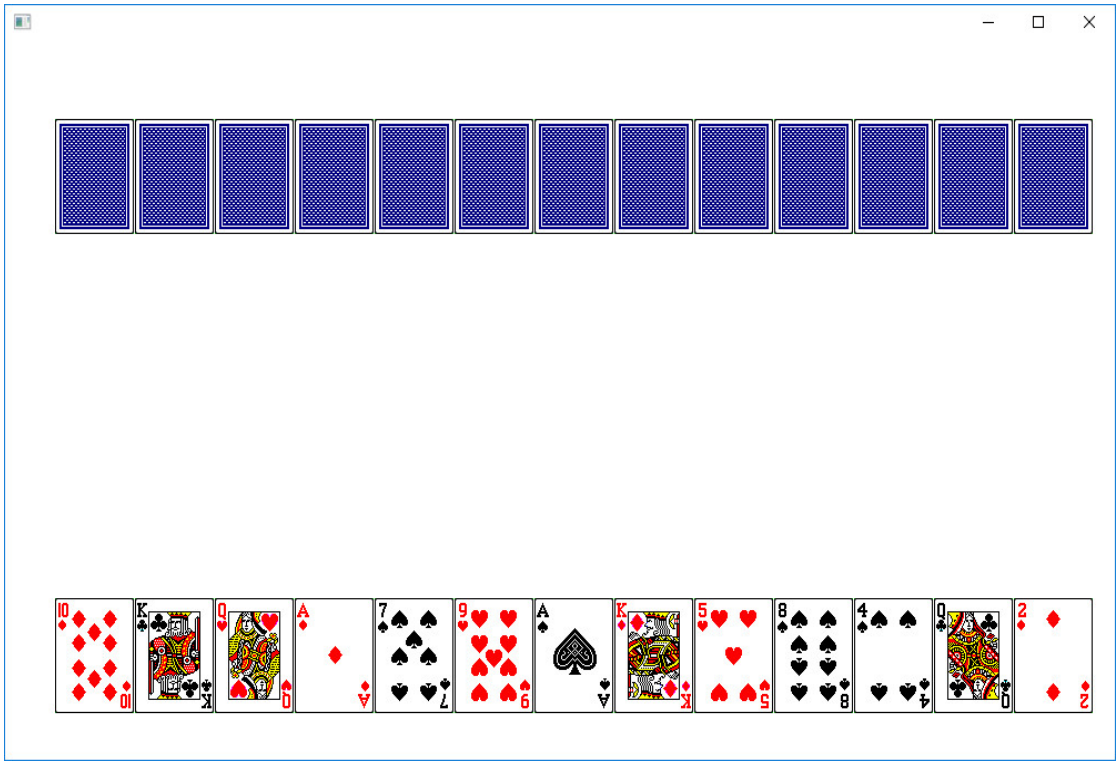
```
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.image.Image;

public class Dealer extends Player {
  public Dealer(GraphicsContext g)
    super(g);
  }
  public void display() {
    int len = hand.size();
    for (int i=0; i<len; i++) {
      Card card = hand.get(i);
      Image img = card.image;
      gc.drawImage(img, 45+72*i, 65);
    }
  }
}
```

と修正し、保持しているカードを表にして画面の下部に表示します。修正が間違っていないか、クラス Game を修正して確かめます。クラス Game を

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
```

```
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class Game extends Application {

  public static void main(final String... args) {
launch(args);
  }

  GraphicsContext g;
  Dealer dealer;
  User user;

  @Override
  public void start(final Stage stage) {
    VBox root = new VBox();
    Canvas cvs = new Canvas(1000, 600);
    root.getChildren().add(cvs);

    this.g = cvs.getGraphicsContext2D();
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.show();
    run();
  }
  private void run() {
    dealer = new Dealer(g);
    user = new User(g);
    CardDeck carddeck = new CardDeck();
    carddeck.shuffle();
    for (int i=0; i<13; i++) {
      dealer.getCard(carddeck.getCard(2*i));
      user.getCard(carddeck.getCard(2*i+1));
    }
    dealer.display();
    user.display();
  }
}
```

と修正します。メソッド draw() を run() に変えました。実行すると



です。画面の上部にディーラーの手札が、下部にユーザーの手札が表示されています。正しいみたいです。ストックのクラス Stock も作ります。

```
import java.util.ArrayList;
import java.util.List;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.image.Image;

public class Stock {
  protected List<Card> cards = null;
  GraphicsContext gc;

  public Stock(GraphicsContext g) {
    cards = new ArrayList<Card>();
    this.gc = g;
  }
  public void getCard(Card card) {
    cards.add(card);
  }
  public Card draw() {
    Card card = (Card)cards.remove(0);
    return card;
  }
  public void display() {
```

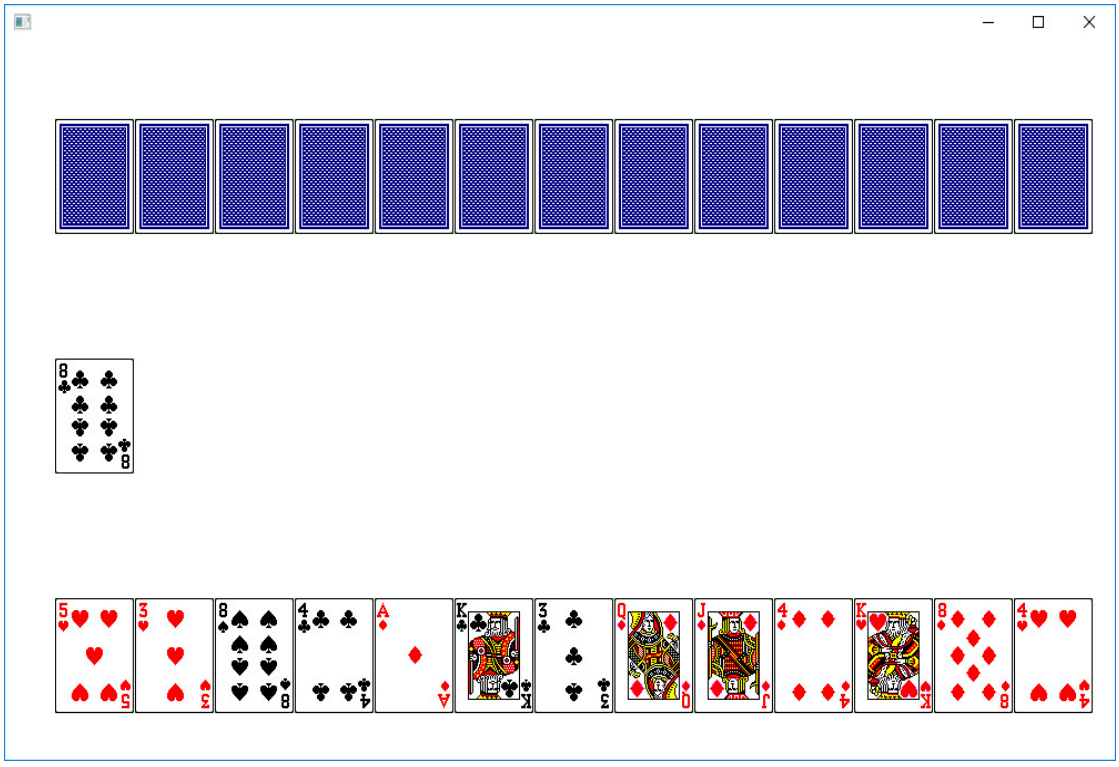
```
    int len = cards.size();
    if (len > 0) {
      Card card = cards.get(0);
      Image img = card.image;
      gc.drawImage(img, 45, 265);
    }
  }
}
```

クラス Stock が間違っていないか、クラス Game を修正して確かめます。クラス Game を

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class Game extends Application {

  public static void main(final String... args) {
    launch(args);
  }

  GraphicsContext g;
  Dealer dealer;
  User user;
  Stock stock;
  @Override
  public void start(final Stage stage) {
    VBox root = new VBox();
    Canvas cvs = new Canvas(1000, 600);
    root.getChildren().add(cvs);

    this.g = cvs.getGraphicsContext2D();
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.show();
    run();
  }
  private void run() {
    dealer = new Dealer(g);
    user = new User(g);
    stock = new Stock(g);
```

```
    CardDeck carddeck = new CardDeck();
    carddeck.shuffle();
    for (int i=0; i<13; i++) {
      dealer.getCard(carddeck.getCard(2*i));
      user.getCard(carddeck.getCard(2*i+1));
    }
    for (int i=26; i<52; i++) {
      stock.getCard(carddeck.getCard(i));
    }
    dealer.display();
    user.display();
    stock.display();
  }
}
```

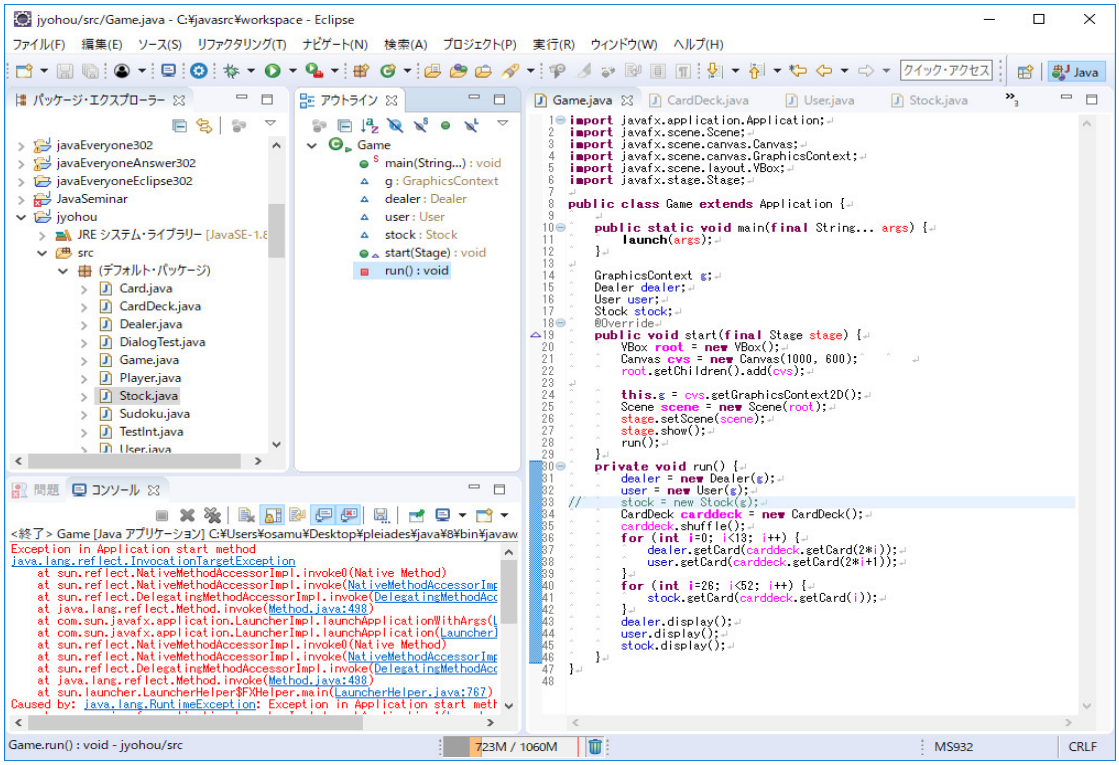
と修正します。実行すると



です。間違ってないみたいです。これがゲームが始まったばかりの初期画面です。ここからゲームが始まります。ところで、メソッド run() で、

```
    stock = new Stock(g);
```

を最初忘れていました。実行すると



です。

```
java.lang.reflect.InvocationTargetException
```

と表示されています。どこで間違えたか見当もつきませんが、インターネットには親切な人が、

これって何が起きてるの??

結論から申し上げると、**java** エンジニアの方なら一度は経験しているであろう、**NullPointerException** が発生してます。エラーにはそんなこと、一言も書いてませんけどね …（笑）なので、何も入っていない値を呼び出しているのが原因です。

と書き込んでくれています。これをヒントにしばらく考えると、stock を生成していないことに気付きます。Java は他の言語にくらべ、デバッグも難しいですが、インターネットで検索すれば、何とか乗り切ることが出来ます。

ディーラーがゲットしたカードとユーザーがゲット したカードを保持するクラスを作ります。

```
import java.util.ArrayList;
import java.util.List;
import javafx.scene.canvas.GraphicsContext;

public class GetCard {
  protected List<Card> cards = null;
  GraphicsContext gc;

  public GetCard(GraphicsContext g) {
    cards = new ArrayList<Card>();
```

```
    this.gc = g;
  }
  public void getCard(Card card) {
    cards.add(card);
  }
}
```

を基底クラスとし、

```
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.image.Image;

public class DealerGetCard extends GetCard {
  public DealerGetCard(GraphicsContext g) {
    super(g);
  }
  public void display() {
    int len = cards.size();
    for (int i=0; i<len; i++) {
      Card card = cards.get(i);
      Image img = card.image;
      gc.drawImage(img, 45+18*i, 165);
    }
  }
}
```

と

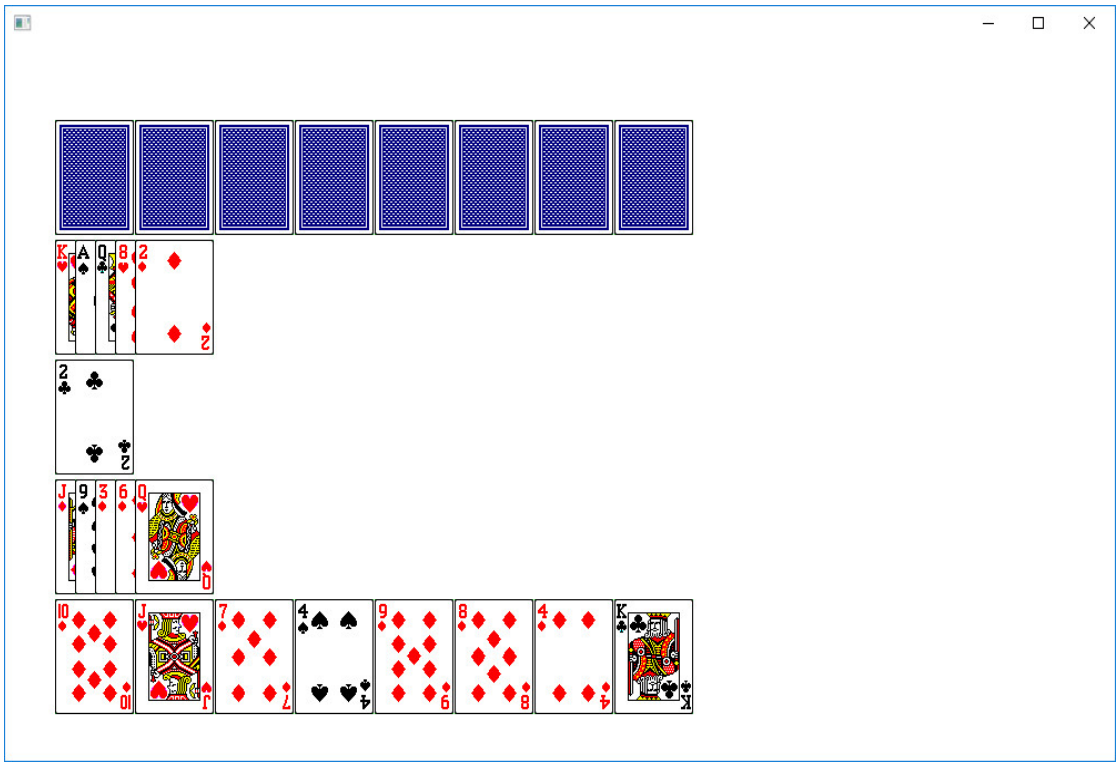
```
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.image.Image;

public class UserGetCard extends GetCard {
  public UserGetCard(GraphicsContext g) {
    super(g);
  }
  public void display() {
    int len = cards.size();
    for (int i=0; i<len; i++) {
      Card card = cards.get(i);
      Image img = card.image;
      gc.drawImage(img, 45+18*i, 365);
    }
  }
}
```

を定義します。ここまでが間違っていないか確かめるために、クラス Game を修正します。

```
public class Game extends Application {

  public static void main(final String... args) {
    launch(args);
  }

  GraphicsContext g;
  Dealer dealer;
  User user;
  Stock stock;
  DealerGetCard dealergetcard;
  UserGetCard usergetcard;

  @Override
  public void start(final Stage stage) {
    VBox root = new VBox();
    Canvas cvs = new Canvas(1000, 600);
    root.getChildren().add(cvs);

    this.g = cvs.getGraphicsContext2D();
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.show();
    run();
  }
  private void run() {
    dealer = new Dealer(g);
    user = new User(g);
    stock = new Stock(g);
    dealergetcard = new DealerGetCard(g);
    usergetcard = new UserGetCard(g);
    CardDeck carddeck = new CardDeck();
    carddeck.shuffle();
    for (int i=0; i<13; i++) {
      dealer.getCard(carddeck.getCard(2*i));
      user.getCard(carddeck.getCard(2*i+1));
    }
    for (int i=26; i<52; i++) {
      stock.getCard(carddeck.getCard(i));
    }
    for (int i=0; i<5; i++) {
      dealergetcard.getCard(dealer.draw());
      usergetcard.getCard(user.draw());
```

```
    }
     dealer.display();
     user.display();
     stock.display();
     dealergetcard.display();
     usergetcard.display();
  }
}
```

実行すると



です。ここまでは間違っていないみたいです。ディーラーがリードしたカードを保持するクラスを作ります。

```
import java.util.ArrayList;
import java.util.List;

import javafx.scene.canvas.GraphicsContext;
import javafx.scene.image.Image;

public class LeadCard {
  protected List<Card> cards = null;
  GraphicsContext gc;

  public LeadCard(GraphicsContext g) {
```

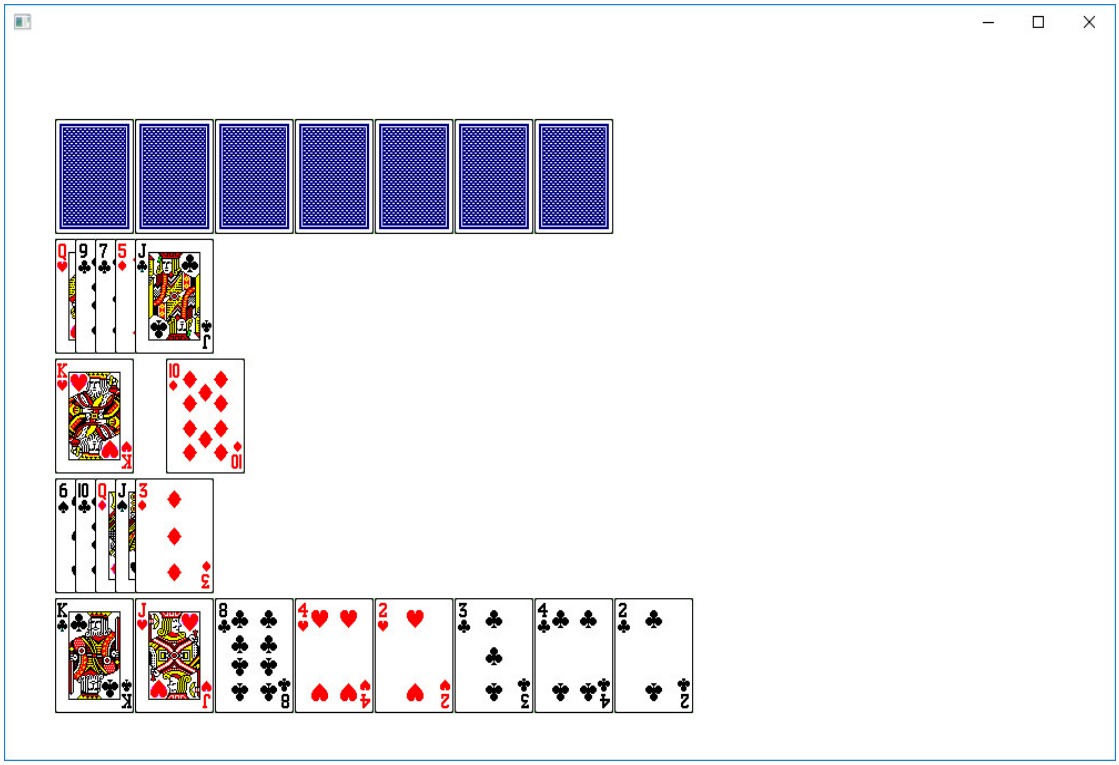
```
    cards = new ArrayList<Card>();
    this.gc = g;
  }
  public void getCard(Card card) {
    cards.add(card);
  }
  public void display() {
    int len = cards.size();
    for (int i=0; i<len; i++) {
      Card card = cards.get(i);
      Image img = card.image;
      gc.drawImage(img, 145, 265);
    }
  }
}
```

ここまでが間違っていないか確かめるために、クラス Game を修正します。

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class Game extends Application {

  public static void main(final String... args) {
    launch(args);
  }

  GraphicsContext g;
  Dealer dealer;
  User user;
  Stock stock;
  DealerGetCard dealergetcard;
  UserGetCard usergetcard;
  LeadCard leadcard;

  @Override
  public void start(final Stage stage) {
    VBox root = new VBox();
    Canvas cvs = new Canvas(1000, 600);
    root.getChildren().add(cvs);
```

```
    this.g = cvs.getGraphicsContext2D();
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.show();
    run();
  }
  private void run() {
    dealer = new Dealer(g);
    user = new User(g);
    stock = new Stock(g);
    dealergetcard = new DealerGetCard(g);
    usergetcard = new UserGetCard(g);
    leadcard = new LeadCard(g);
    CardDeck carddeck = new CardDeck();
    carddeck.shuffle();
    for (int i=0; i<13; i++) {
      dealer.getCard(carddeck.getCard(2*i));
      user.getCard(carddeck.getCard(2*i+1));
    }
    for (int i=26; i<52; i++) {
      stock.getCard(carddeck.getCard(i));
    }
    for (int i=0; i<5; i++) {
      dealergetcard.getCard(dealer.draw());
      usergetcard.getCard(user.draw());
    }
    leadcard.getCard(dealer.draw());
    dealer.display();
    user.display();
    stock.display();
    dealergetcard.display();
    usergetcard.display();
    leadcard.display();
  }
}
```

実行すると



です。ここまでは意図したとおりに動いているみたいです。これで、ゲームの間、表示すべきカードはすべて準備出来ました。ここからは、ゲームを進めていくうえで必要なプログラミングをしていきます。

クラス Game を

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class Game extends Application {

  public static void main(final String... args) {
    launch(args);
  }

  GraphicsContext g;
  Dealer dealer;
  User user;
  Stock stock;
  DealerGetCard dealergetcard;
  UserGetCard usergetcard;
```

```
  LeadCard leadcard;

  @Override
  public void start(final Stage stage) {
    VBox root = new VBox();
    Canvas cvs = new Canvas(1000, 600);
    root.getChildren().add(cvs);

    this.g = cvs.getGraphicsContext2D();
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.show();
    run();
  }
  private void run() {
    dealer = new Dealer(g);
    user = new User(g);
    stock = new Stock(g);
    dealergetcard = new DealerGetCard(g);
    usergetcard = new UserGetCard(g);
    leadcard = new LeadCard(g);
    CardDeck carddeck = new CardDeck();
    carddeck.shuffle();
    for (int i=0; i<13; i++) {
      dealer.getCard(carddeck.getCard(2*i));
      user.getCard(carddeck.getCard(2*i+1));
    }
    for (int i=26; i<52; i++) {
      stock.getCard(carddeck.getCard(i));
    }
    dealer.display();
    user.display();
    stock.display();
    dealergetcard.display();
    usergetcard.display();
    leadcard.display();
  }
}
```

と修正し、これをもとにゲームの進め方のプログラムをメソッド run() に書き込んでいきます。実行すると

です。ディーラーでない人から始めるので、ユーザーが最初のリードをします。クラス Game のメソッド run() の最後に

```
g.setFont(new Font("courier", 40));
g.setFill(Color.BLUE);
g.setTextAlign(TextAlignment.CENTER);
g.fillText("カードを選んで下さい", 400, 315);
```

を追加します。実行すると

です。

画面下部に表示されているカードをマウスでクリックして、リードするカードを選べるようにします。クラス Game のメソッド start(Stage pstage) に

```
this.g = cvs.getGraphicsContext2D();
cvs.setOnMouseClicked((event)->{
  buttonPressed(event.getX(), event.getY());
});
```

を追加し、

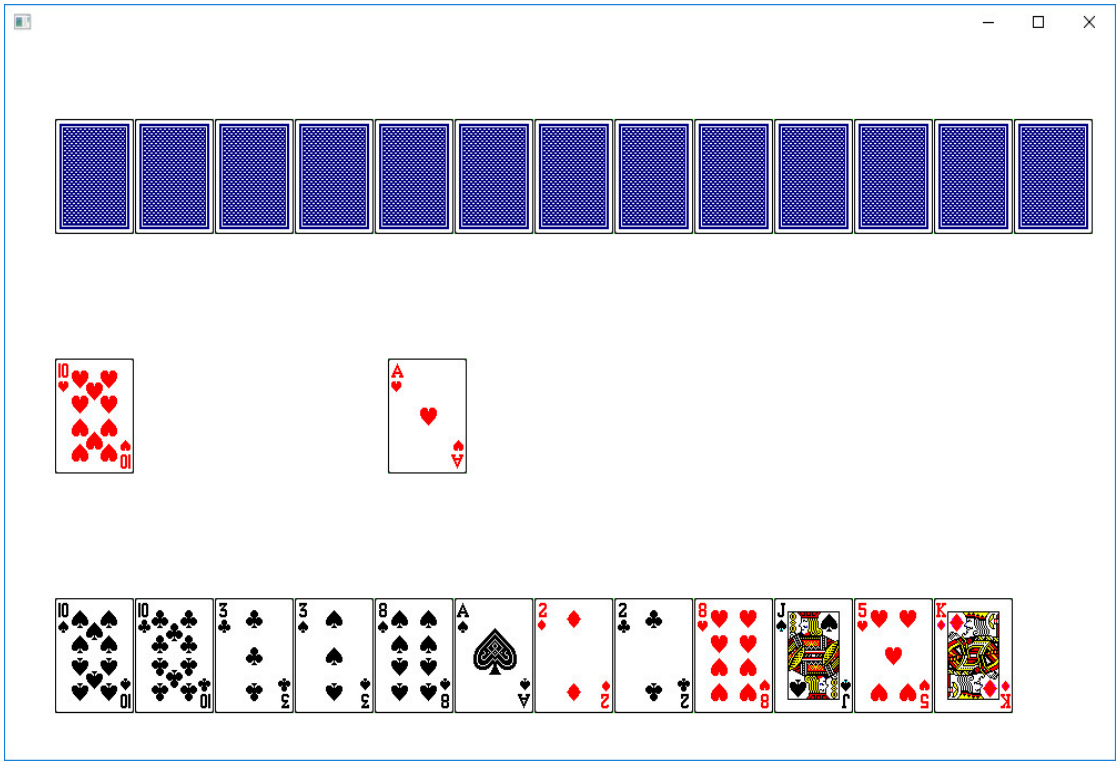
```
public void start(final Stage stage) {
  VBox root = new VBox();
  Canvas cvs = new Canvas(1000, 600);
  root.getChildren().add(cvs);
  this.g = cvs.getGraphicsContext2D();
  cvs.setOnMouseClicked((event)->{
    buttonPressed(event.getX(), event.getY());
  });
  Scene scene = new Scene(root);
  stage.setScene(scene);
  stage.show();
  run();
}
```

と修正し、クラス Game にメソッド buttonPressed(double x, double y)

```
  void buttonPressed(double x, double y) {
    int index = -1;
    int pos = 45;
    for (int i=0; i<user.hand.size(); i++) {
      if (x>pos && x<pos+72 && y>465 && y<565) {
        index = i;
        break;
      }
      pos += 72;
    }
    if (index == -1)
      return;
    Card card = user.hand.remove(index);
    g.clearRect(0, 0, 1000, 600);
    Image img = card.image;
    g.drawImage(img, 345, 265);
    dealer.display();
    user.display();
    stock.display();
    dealergetcard.display();
    usergetcard.display();
    leadcard.display();
  }
```

を追加します。
実行すると

のような画面が現れるので、例えば、ハートのエースをクリックしてみます。



となります。上手くいっているみたいです。次はディーラーがユーザーが出したリードに対して応答する必要があります。手札の一番左のカードを出せば良い訳ではありません。手札にハートがあ

れば、ハートのカードのどれかを出さなければいけません。クラス Dealer でクラス Player のメソッド

```
  public Card draw() {
    Card card = (Card)hand.remove(0);
    return card;
  }
```

をオーバーライドする必要があります。その時、ディーラーがリードする場合とそうでない場合で、カードの選び方が変わります。そこで、どちらがリードするかを保持する変数が必要になります。dealerPlayP が true か false で区別することにします。更に、切り札が何かによっても、手の選び方が変わります。切り札は最初にストックに表示されているカードと同じスーツの札です。最初にストックに表示されたカードを保持している変数 Card trump も宣言します。

クラス Dealer を

```
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.image.Image;

public class Dealer extends Player {
  boolean dealerPlayP;
  Card userlead;
  Card trump;

  public Dealer(GraphicsContext g) {
    super(g);
  }
  public void display() {
    int len = hand.size();
    for (int i=0; i<len; i++) {
      Card card = hand.get(i);
      Image bkimg = card.bkimage;
      gc.drawImage(bkimg, 45+72*i, 65);
    }
  }
  public void getWhoLead(boolean dealerPlayP) {
    this.dealerPlayP = dealerPlayP;
  }
  public void getUserLead(Card card) {
    this.userlead = card;
  }
  public void getTrump(Card card) {
    this.trump = card;
  }
  public Card draw() {
```

```
    if (dealerPlayP) {
      Card card = (Card)hand.remove(0);
      return card;
    } else {
      String suit = userlead.suit;
      int len = hand.size();
      for (int i=0; i<len; i++) {
        if (hand.get(i).suit == suit) {
          Card card = (Card)hand.remove(i);
          return card;
        }
      }
      Card card = (Card)hand.remove(0);
      return card;
    }
  }
}
```

と修正します。単にルール通りの手を出すだけです。強くするのは自分で考えて下さい。クラス Game を次のように修正します。

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.image.Image;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.TextAlignment;
import javafx.stage.Stage;

public class Game extends Application {

  public static void main(final String... args) {
    launch(args);
  }

  GraphicsContext g;
  Dealer dealer;
  User user;
  Stock stock;
  DealerGetCard dealergetcard;
  UserGetCard usergetcard;
```

```
LeadCard leadcard;
boolean dealerPlayP = false;
Card trump;

@Override
public void start(final Stage stage) {
  VBox root = new VBox();
  Canvas cvs = new Canvas(1000, 600);
  root.getChildren().add(cvs);

  this.g = cvs.getGraphicsContext2D();
  cvs.setOnMouseClicked((event)->{
    buttonPressed(event.getX(), event.getY());
 });
  Scene scene = new Scene(root);
  stage.setScene(scene);
  stage.show();
  run();
}
void buttonPressed(double x, double y) {
  int index = -1;
  int pos = 45;
  for (int i=0; i<user.hand.size(); i++) {
    if (x>pos && x<pos+72 && y>465 && y<565) {
      index = i;
      break;
   }
    pos += 72;
 }
  if (index == -1)
    return;
  Card card = user.hand.remove(index);
  dealer.getWhoLead(dealerPlayP);
  dealer.getUserLead(card);
  g.clearRect(0, 0, 1000, 600);
  Image img = card.image;
  g.drawImage(img, 345, 265);
  Card drawcard = dealer.draw();
  Image dealimg = drawcard.image;
  g.drawImage(dealimg, 445, 265);
  dealer.display();
  user.display();
  stock.display();
```

```
    dealergetcard.display();
    usergetcard.display();
    leadcard.display();
  }
  private void run() {
    dealerPlayP = false;
    dealer = new Dealer(g);
    user = new User(g);
    stock = new Stock(g);
    dealergetcard = new DealerGetCard(g);
    usergetcard = new UserGetCard(g);
    leadcard = new LeadCard(g);
    CardDeck carddeck = new CardDeck();
    carddeck.shuffle();
    for (int i=0; i<13; i++) {
      dealer.getCard(carddeck.getCard(2*i));
      user.getCard(carddeck.getCard(2*i+1));
    }
    for (int i=26; i<52; i++) {
      stock.getCard(carddeck.getCard(i));
    }
    trump = stock.cards.get(0);
    dealer.getTrump(trump);
    dealer.display();
    user.display();
    stock.display();
    dealergetcard.display();
    usergetcard.display();
    leadcard.display();
    g.setFont(new Font("courier", 40));
    g.setFill(Color.BLUE);
    g.setTextAlign(TextAlignment.CENTER);
    g.fillText("カードを選んで下さい", 400, 315);
  }
}
```

```
  boolean dealerPlayP = false;
  Card trump;
```

の宣言とメソッド buttonPressed(double x, double y) の後半

```
    Card card = user.hand.remove(index);
    dealer.getWhoLead(dealerPlayP);
    dealer.getUserLead(card);
```

```
        g.clearRect(0, 0, 1000, 600);
        Image img = card.image;
        g.drawImage(img, 345, 265);
        Card drawcard = dealer.draw();
        Image dealimg = drawcard.image;
        g.drawImage(dealimg, 445, 265);
        dealer.display();
        user.display();
        stock.display();
        dealergetcard.display();
        usergetcard.display();
        leadcard.display();
    }
```

の部分とメソッド run() の

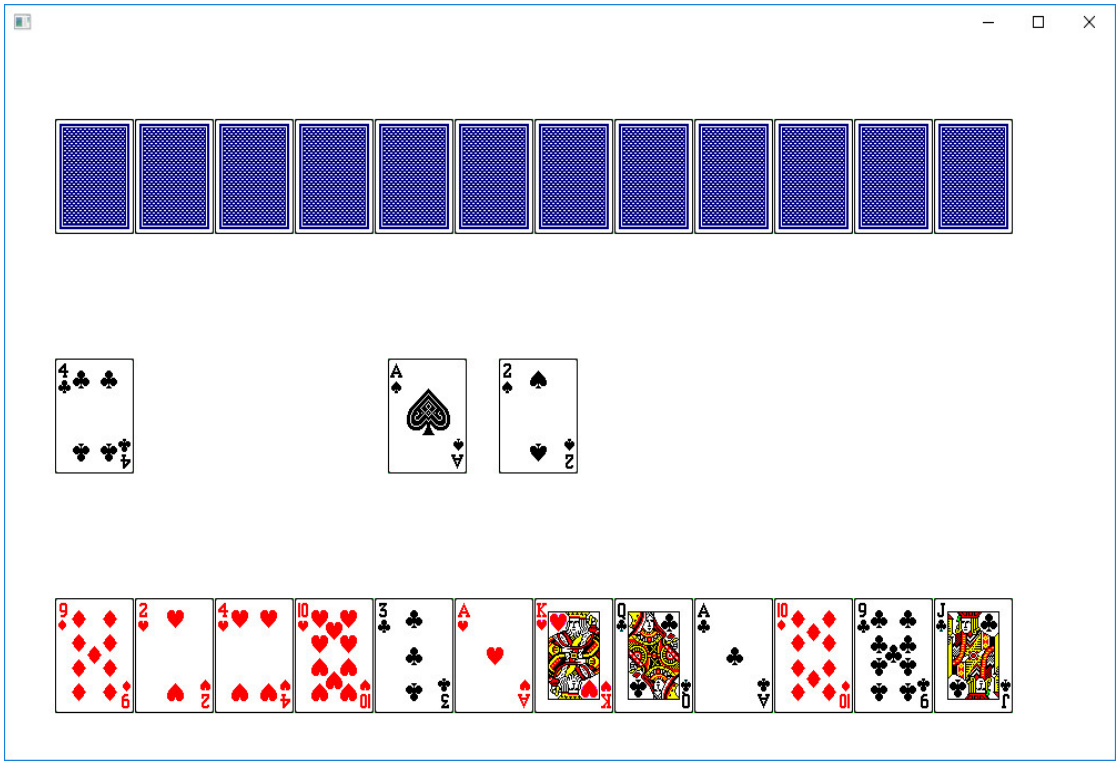
```
        dealerPlayP = false;
```

と

```
        dealer.getTrump(trump);
```

を変えたと思います。実行すると

で、スペードのエースを選ぶと



となります。ここまでは、思い通りに動いているみたいです。

次に、場に出ているカードのどちらが勝っているか判定する必要があります。クラス Referee を作ってみる方法もありますが、ここでは非オブジェクト指向で、Game のメソッド dealerWinP() で対応します。

```
public boolean dealerWinP(Card usercard, Card dealercard) {
  if (usercard.suit == dealercard.suit) {
    if (dealercard.rank == 1) {
      return true;
    } else if (usercard.rank == 1) {
      return false;
    } else if (dealercard.rank > usercard.rank) {
      return true;
    } else {
      return false;
    }
  } else if (dealercard.suit == trump.suit) {
    return true;
  } else if (usercard.suit == trump.suit) {
    return false;
  } else if (dealerPlayP) {
    return true;
  } else {
    return false;
```

```
  }
}
```

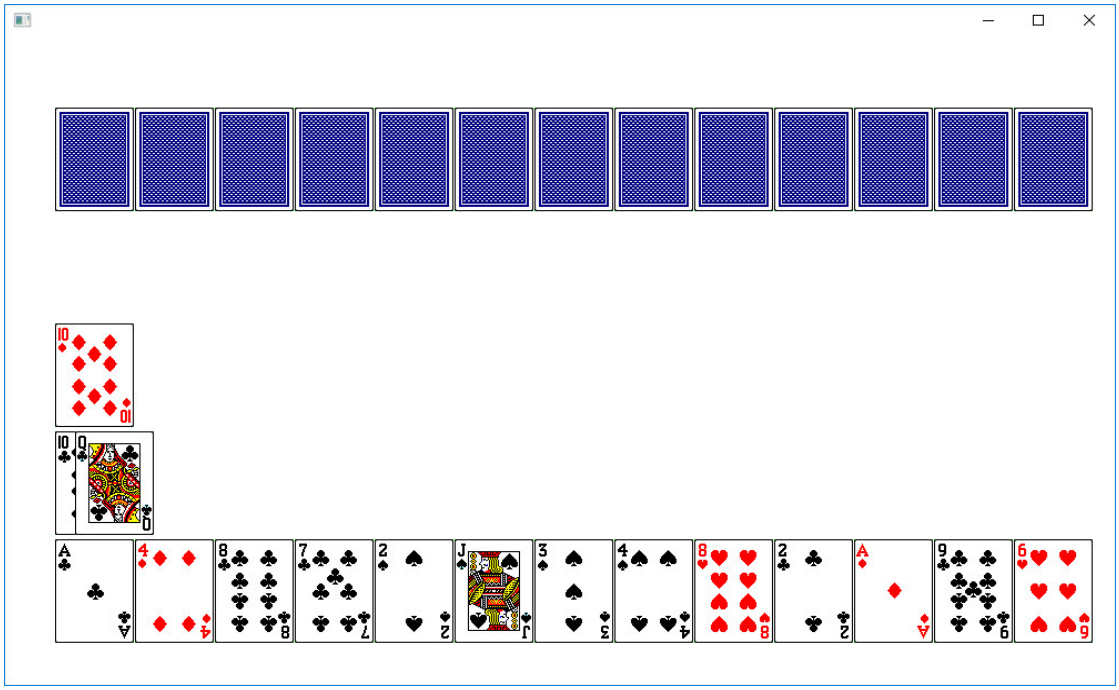
と定義します。ディーラーが勝っていれば、場に出たカードを dealergetcard に追加し、ストックにカードがあれば、上のカードを dealer に、下のカードを user に配ります。ディーラーが負けていれば、場に出たカードを usergetcard に追加し、ストックにカードがあれば、上のカードを user に、下のカードを dealer に配ります。ここまでの部分をプログラミングしてみます。

メソッド buttonPressed(double x, double y) を

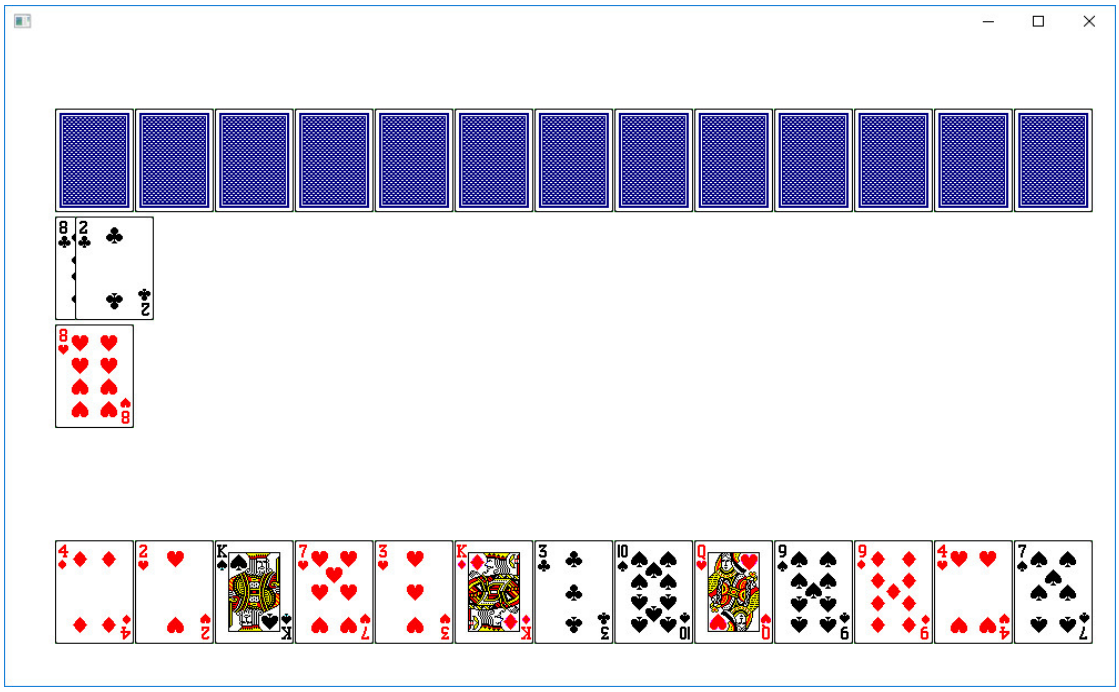
```
void buttonPressed(double x, double y) {
  int index = -1;
  int pos = 45;
  for (int i=0; i<user.hand.size(); i++) {
    if (x>pos && x<pos+72 && y>465 && y<565) {
      index = i;
      break;
    }
    pos += 72;
  }
  if (index == -1)
    return;
  Card card = user.hand.remove(index);
  dealer.getWhoLead(dealerPlayP);
  dealer.getUserLead(card);
  Card drawcard = dealer.draw();
  if (dealerWinP(card, drawcard)) {
    dealergetcard.getCard(drawcard);
    dealergetcard.getCard(card);
    if (stock.cards.size() > 0) {
      dealer.getCard(stock.draw());
      user.getCard(stock.draw());
    }
  } else {
    usergetcard.getCard(drawcard);
    usergetcard.getCard(card);
    if (stock.cards.size() > 0) {
      user.getCard(stock.draw());
      dealer.getCard(stock.draw());
    }
  }
  g.clearRect(0, 0, 1000, 600);
  dealer.display();
  user.display();
  stock.display();
```

```
        dealergetcard.display();
        usergetcard.display();
        leadcard.display();
    }
```

と修正します。実行すると



や



です。間違っていないみたいです。
ユーザーが勝った場合は、ゲームが終わっているか判断し、終わっていなければ、ユーザーがリードする番ですから、

```
g.setFont(new Font("courier", 40));
g.setFill(Color.BLUE);
g.setTextAlign(TextAlignment.CENTER);
g.fillText("カードを選んで下さい", 400, 315);
```

を打ち出して、このメソッドは終わりです。ディーラーが勝った場合は、ゲームが終わっているか判断し、終わっていなければ、ディーラーがリードし、マウスでユーザーが手を選ぶのを待ちます。即ち、この場合もディーラーのリードを表示したら、このメソッドは終わりです。従って、メソッド buttonPressed(double x, double y) はユーザーがリードする場合にも、ディーラーのリードに対応する場合にも使います。そのどちらかは dealerPlayP で判断します。従って、メソッド buttonPressed(double x, double y) は

```
void buttonPressed(double x, double y) {
  int index = -1;
  int pos = 45;
  for (int i=0; i<user.hand.size(); i++) {
    if (x>pos && x<pos+72 && y>465 && y<565) {
      index = i;
      break;
    }
    pos += 72;
  }
  if (index == -1)
    return;
  if (dealerPlayP == false) {
    Card card = user.hand.remove(index);
    dealer.getWhoLead(dealerPlayP);
    dealer.getUserLead(card);
    Card drawcard = dealer.draw();
    dealerwinp = dealerWinP(card, drawcard);
    if (dealerwinp) {
      dealergetcard.getCard(drawcard);
      dealergetcard.getCard(card);
      if (stock.cards.size() > 0) {
        dealer.getCard(stock.draw());
        user.getCard(stock.draw());
      }
      dealerPlayP = true;
    } else {
      usergetcard.getCard(drawcard);
      usergetcard.getCard(card);
```

```
    if (stock.cards.size() > 0) {
      user.getCard(stock.draw());
      dealer.getCard(stock.draw());
    }
    dealerPlayP = false;
  }
}
 if (dealer.hand.size() == 0) {
   g.clearRect(0, 0, 1000, 600);
   dealer.display();
   user.display();
   stock.display();
   dealergetcard.display();
   usergetcard.display();
   leadcard.display();
   g.setFont(new Font("courier", 40));
   g.setFill(Color.BLUE);
   g.setTextAlign(TextAlignment.CENTER);
   if (dealergetcard.cards.size() > usergetcard.cards.size()) {
     StringBuffer s = new StringBuffer();
     s.append(String.valueOf(usergetcard.cards.size()/2));
     s.append(" vs. ");
     s.append(String.valueOf(dealergetcard.cards.size()/2));
     s.append(" あなたの負けです。");
     g.fillText(s.toString(), 400, 315);
   }
   else if (dealergetcard.cards.size() < usergetcard.cards.size()) {
     StringBuffer s = new StringBuffer();
     s.append(String.valueOf(usergetcard.cards.size()/2));
     s.append(" vs. ");
     s.append(String.valueOf(dealergetcard.cards.size()/2));
     s.append(" あなたの勝ちです。");
     g.fillText(s.toString(), 400, 315);
   }
   else {
     StringBuffer s = new StringBuffer();
     s.append(String.valueOf(usergetcard.cards.size()/2));
     s.append(" vs. ");
     s.append(String.valueOf(dealergetcard.cards.size()/2));
     s.append(" 引き分けです。");
     g.fillText(s.toString(), 400, 315);
   }
   return;
```

```
    }
    if (dealerwinp) {
      dealer.getWhoLead(dealerPlayP);
      Card card = dealer.draw();
      leadcard.getCard(card);
      g.clearRect(0, 0, 1000, 600);
      dealer.display();
      user.display();
      stock.display();
      dealergetcard.display();
      usergetcard.display();
      leadcard.display();
      g.setFont(new Font("courier", 40));
      g.setFill(Color.BLUE);
      g.setTextAlign(TextAlignment.CENTER);
      g.fillText("カードを選んで下さい", 400, 315);
    } else {
      g.clearRect(0, 0, 1000, 600);
      dealer.display();
      user.display();
      stock.display();
      dealergetcard.display();
      usergetcard.display();
      leadcard.display();
      g.setFont(new Font("courier", 40));
      g.setFill(Color.BLUE);
      g.setTextAlign(TextAlignment.CENTER);
      g.fillText("カードを選んで下さい", 400, 315);
    }
  }
```

とすべきでした。実行すると

です。間違っていないみたいです。最後に、ユーザーがディーラーのリードに対応するようにします。この場合は、ユーザーの選んだカードがルールに合っているかチェックし、勝敗を判定します。

まず、クラス LeadCard に

```
import java.util.ArrayList;
import java.util.List;

import javafx.scene.canvas.GraphicsContext;
import javafx.scene.image.Image;

public class LeadCard {
  protected List<Card> cards = null;
  GraphicsContext gc;

  public LeadCard(GraphicsContext g) {
    cards = new ArrayList<Card>();
    this.gc = g;
  }
  public void getCard(Card card) {
    cards.add(card);
  }
  public Card draw() {
    Card card = cards.remove(0);
    return card;
  }
  public void display() {
```

```
    int len = cards.size();
    for (int i=0; i<len; i++) {
      Card card = cards.get(i);
      Image img = card.image;
      gc.drawImage(img, 145, 265);
    }
  }
}
```

とメソッド draw() を追加し、
ユーザーの選んだカードがルールに合っているかチェックするメソッド legalP(Card card) を

```
  public boolean legalP(Card card) {
    String suit = leadcard.cards.get(0).suit;
    if (card.suit == suit) {
      return true;
    }
    int len = user.hand.size();
    for (int i=0; i<len; i++) {
      if (user.hand.get(i).suit == suit) {
        return false;
      }
    }
    return true;
  }
```

と定義し、メソッド buttonPressed(double x, double y) に

```
    else {
      Card card = user.hand.get(index);
      if (legalP(card)) {
        Card card1 = user.hand.remove(index);
        Card card2 = leadcard.draw();
        dealerwinp = dealerWinP(card1, card2);
        if (dealerwinp) {
          dealergetcard.getCard(card2);
          dealergetcard.getCard(card1);
          if (stock.cards.size() > 0) {
            dealer.getCard(stock.draw());
            user.getCard(stock.draw());
          }
          dealerPlayP = true;
        } else {
          usergetcard.getCard(card2);
          usergetcard.getCard(card1);
```

```
        if (stock.cards.size() > 0) {
          user.getCard(stock.draw());
          dealer.getCard(stock.draw());
        }
        dealerPlayP = false;
      }
    } else {
      Alert alert = new Alert(AlertType.INFORMATION);
      alert.setTitle("出すカードが間違っています");
      alert.setHeaderText(null);
      StringBuffer string = new StringBuffer();
      string.append("スーツは");
      Card card3 = leadcard.cards.get(0);
      switch(card3.suit) {
      case "c":
        string.append("クラブ"); break;
      case "d":
        string.append("ダイヤ"); break;
      case "h":
        string.append("ハート"); break;
      case "s":
        string.append("スペード"); break;
      }
      string.append("です！");
      alert.setContentText(string.toString());
      alert.showAndWait();
      g.clearRect(0, 0, 1000, 600);
      dealer.display();
      user.display();
      stock.display();
      dealergetcard.display();
      usergetcard.display();
      leadcard.display();
      g.setFont(new Font("courier", 40));
      g.setFill(Color.BLUE);
      g.setTextAlign(TextAlignment.CENTER);
      g.fillText("カードを選んで下さい", 400, 315);
      return;
    }
  }
```

を追加し、全体を

```
  void buttonPressed(double x, double y) {
```

```
int index = -1;
int pos = 45;
for (int i=0; i<user.hand.size(); i++) {
  if (x>pos && x<pos+72 && y>465 && y<565) {
    index = i;
    break;
  }
  pos += 72;
}
if (index == -1)
  return;
if (dealerPlayP == false) {
  Card card = user.hand.remove(index);
  dealer.getWhoLead(dealerPlayP);
  dealer.getUserLead(card);
  Card drawcard = dealer.draw();
  dealerwinp = dealerWinP(card, drawcard);
  if (dealerwinp) {
    dealergetcard.getCard(drawcard);
    dealergetcard.getCard(card);
    if (stock.cards.size() > 0) {
      dealer.getCard(stock.draw());
      user.getCard(stock.draw());
    }
    dealerPlayP = true;
  } else {
    usergetcard.getCard(drawcard);
    usergetcard.getCard(card);
    if (stock.cards.size() > 0) {
      user.getCard(stock.draw());
      dealer.getCard(stock.draw());
    }
    dealerPlayP = false;
  }
}
else {
  Card card = user.hand.get(index);
  if (legalP(card)) {
    Card card1 = user.hand.remove(index);
    Card card2 = leadcard.draw();
    dealerwinp = dealerWinP(card1, card2);
    if (dealerwinp) {
      dealergetcard.getCard(card2);
```

```
      dealergetcard.getCard(card1);
      if (stock.cards.size() > 0) {
        dealer.getCard(stock.draw());
        user.getCard(stock.draw());
      }
      dealerPlayP = true;
    } else {
      usergetcard.getCard(card2);
      usergetcard.getCard(card1);
      if (stock.cards.size() > 0) {
        user.getCard(stock.draw());
        dealer.getCard(stock.draw());
      }
      dealerPlayP = false;
    }
  } else {
    Alert alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("出すカードが間違っています");
    alert.setHeaderText(null);
    StringBuffer string = new StringBuffer();
    string.append("スーツは");
    Card card3 = leadcard.cards.get(0);
    switch(card3.suit) {
    case "c":
      string.append("クラブ"); break;
    case "d":
      string.append("ダイヤ"); break;
    case "h":
      string.append("ハート"); break;
    case "s":
      string.append("スペード"); break;
    }
    string.append("です！");
    alert.setContentText(string.toString());
    alert.showAndWait();
    g.clearRect(0, 0, 1000, 600);
    dealer.display();
    user.display();
    stock.display();
    dealergetcard.display();
    usergetcard.display();
    leadcard.display();
    g.setFont(new Font("courier", 40));
```

```
    g.setFill(Color.BLUE);
    g.setTextAlign(TextAlignment.CENTER);
    g.fillText("カードを選んで下さい", 400, 315);
    return;
  }
}
 if (dealer.hand.size() == 0) {
  g.clearRect(0, 0, 1000, 600);
  dealer.display();
  user.display();
  stock.display();
  dealergetcard.display();
  usergetcard.display();
  leadcard.display();
  g.setFont(new Font("courier", 40));
  g.setFill(Color.BLUE);
  g.setTextAlign(TextAlignment.CENTER);
  if (dealergetcard.cards.size() > usergetcard.cards.size()) {
    StringBuffer s = new StringBuffer();
    s.append(String.valueOf(usergetcard.cards.size()/2));
    s.append(" vs. ");
    s.append(String.valueOf(dealergetcard.cards.size()/2));
    s.append(" あなたの負けです。");
    g.fillText(s.toString(), 400, 315);
  }
  else if (dealergetcard.cards.size() < usergetcard.cards.size()) {
    StringBuffer s = new StringBuffer();
    s.append(String.valueOf(usergetcard.cards.size()/2));
    s.append(" vs. ");
    s.append(String.valueOf(dealergetcard.cards.size()/2));
    s.append(" あなたの勝ちです。");
    g.fillText(s.toString(), 400, 315);
  }
  else {
    StringBuffer s = new StringBuffer();
    s.append(String.valueOf(usergetcard.cards.size()/2));
    s.append(" vs. ");
    s.append(String.valueOf(dealergetcard.cards.size()/2));
    s.append(" 引き分けです。");
    g.fillText(s.toString(), 400, 315);
  }
  return;
}
```

```
    if (dealerwinp) {
      dealer.getWhoLead(dealerPlayP);
      Card card = dealer.draw();
      leadcard.getCard(card);
      g.clearRect(0, 0, 1000, 600);
      dealer.display();
      user.display();
      stock.display();
      dealergetcard.display();
      usergetcard.display();
      leadcard.display();
      g.setFont(new Font("courier", 40));
      g.setFill(Color.BLUE);
      g.setTextAlign(TextAlignment.CENTER);
      g.fillText("カードを選んで下さい", 400, 315);
    } else {
      g.clearRect(0, 0, 1000, 600);
      dealer.display();
      user.display();
      stock.display();
      dealergetcard.display();
      usergetcard.display();
      leadcard.display();
      g.setFont(new Font("courier", 40));
      g.setFill(Color.BLUE);
      g.setTextAlign(TextAlignment.CENTER);
      g.fillText("カードを選んで下さい", 400, 315);
    }
  }
```

と修正します。これでプログラムが出来上がりました。

引き続きゲームができるように修正しましょう。メソッド start(final Stage stage) を

```
  public void start(final Stage stage) {
    VBox root = new VBox();
    Canvas cvs = new Canvas(1000, 600);
    Button b1 = new Button("再ゲーム");
    b1.setOnAction((event) -> {
      run();
    });
    root.getChildren().addAll(cvs, b1);
    this.g = cvs.getGraphicsContext2D();
    cvs.setOnMouseClicked((event)->{
       buttonPressed(event.getX(), event.getY());
```

```
    });
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.show();
    run();
  }
```

と修正します。メソッド run() を

```
  private void run() {
    dealerPlayP = false;
    dealer = new Dealer(g);
    user = new User(g);
    stock = new Stock(g);
    dealergetcard = new DealerGetCard(g);
    usergetcard = new UserGetCard(g);
    leadcard = new LeadCard(g);
    CardDeck carddeck = new CardDeck();
    carddeck.shuffle();
    for (int i=0; i<13; i++) {
      dealer.getCard(carddeck.getCard(2*i));
      user.getCard(carddeck.getCard(2*i+1));
    }
    for (int i=26; i<52; i++) {
      stock.getCard(carddeck.getCard(i));
    }
    trump = stock.cards.get(0);
    dealer.getTrump(trump);
    g.clearRect(0, 0, 1000, 600);
    dealer.display();
    user.display();
    stock.display();
    dealergetcard.display();
    usergetcard.display();
    leadcard.display();
    g.setFont(new Font("courier", 40));
    g.setFill(Color.BLUE);
    g.setTextAlign(TextAlignment.CENTER);
    g.fillText("カードを選んで下さい", 400, 315);
  }
```

と修正します。これで完成です。

だいぶ前に作った Ruby のプログラムを見ながら作ったので、非常に楽でした。新しくコンピュータ言語を学んだら、前に別の言語で作ったことのあるプログラムを学んだことの定着のために書いてみると良いです。

コンピュータはルール通りプレイするだけです。カードをソートし表示を見やすくしたり、ストックに表示されている情報や表示されている獲得したカードの情報を集め強くするのは各自で考えてください。プログラミングの知識というより、カード・ゲーム「ジャーマン・ホイスト」に対する理解や人工知能（強化学習）の理解が必要になります。トランプゲームのプログラムを作るときの参考にしてください。