トランプゲームを作ってみよう (Ruby/Tk 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 Ruby/Tk 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。

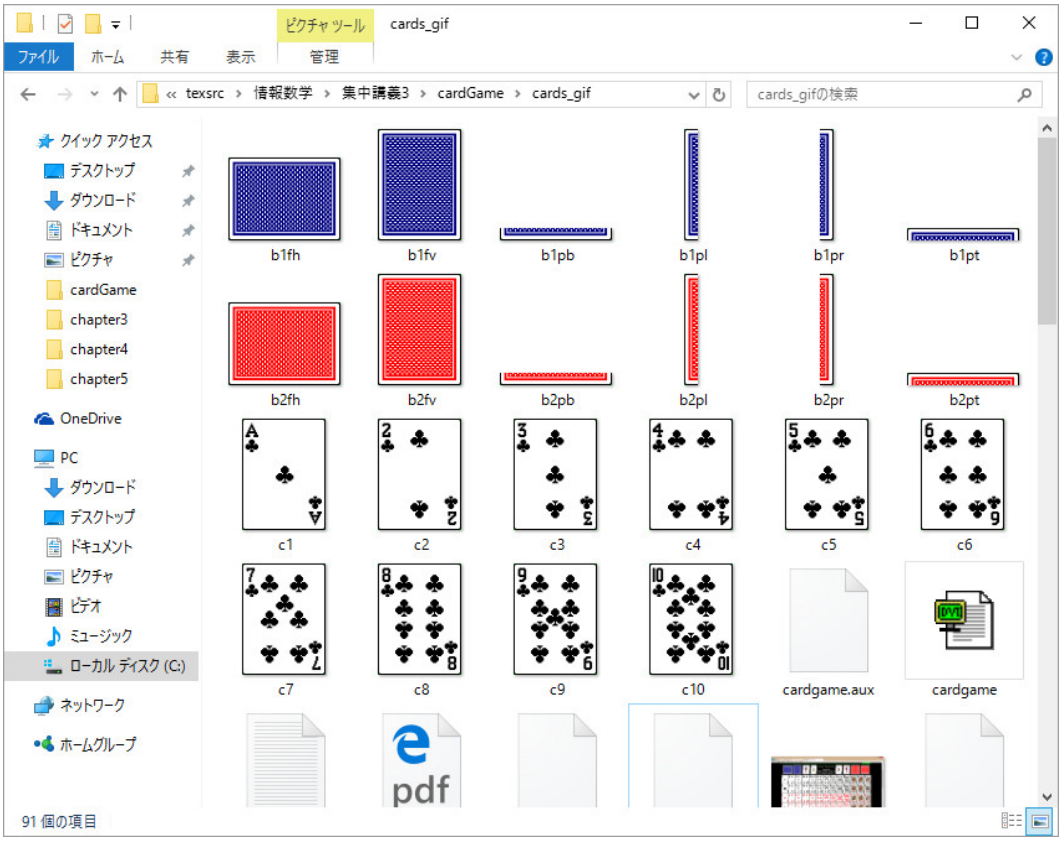
ルールは

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 Ruby 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。



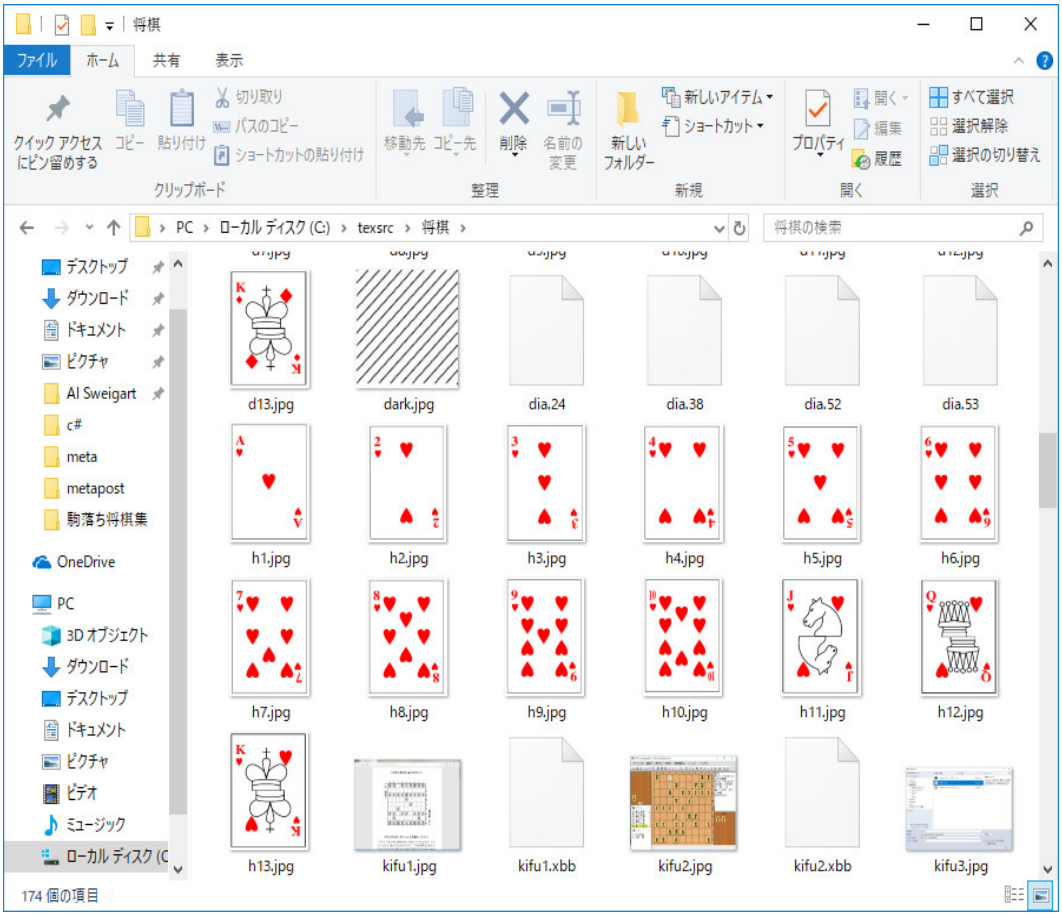
自分で楽しむだけでなければ、GIMP などで、例えばスーツと数字の単純なオリジナルな画像を作れば良いです。

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。

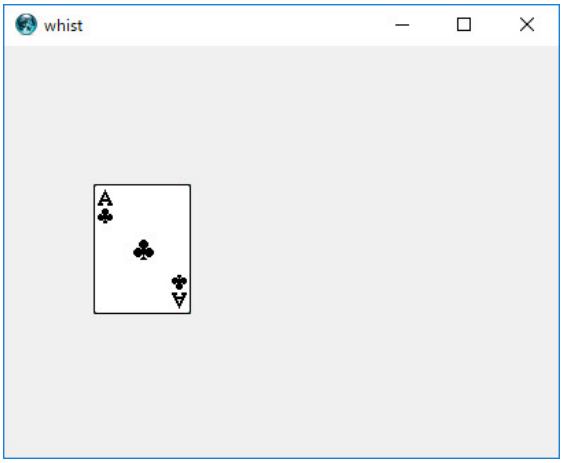
私の作った metapost のプログラムでは次のような画像が作れます。



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。

```
require 'tk'
canvas = TkCanvas.new(width: 400, height: 300).pack
image1 = TkPhotoImage.new(file: 'c1.gif')
TkcImage.new(canvas, 100, 150) do
  image(image1)
end
Tk.mainloop
```

で、カードを表示できます。



となります。VC++ の時のように、拡大・縮小など細かな細工は出来ません。多分。兎も角、カードの画像を Canvas の任意の位置に表示する方法が分かったので、プログラムが作れます。

```
image1 = TkPhotoImage.new(file: 'c1.gif')
TkcImage.new(canvas, 100, 150) do
  image(image1)
end
```

で、c1.gif という名前の画像をキャンバス canvas の座標 (100,150) を中心とする位置に表示してくれます。

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、Ruby 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。従って、次のようなクラスを作ります。

```
class Card
  def initialize(suit, rank, image, bkimage)
    @suit = suit
    @rank = rank
    @image = image
    @bkimage = bkimage
  end
  attr_reader :suit, :rank, :image, :bkimage
end
```

ここで、

```
  def initialize(suit, rank, image, bkimage)
    @suit = suit
    @rank = rank
    @image = image
    @bkimage = bkimage
  end
```

が、C++ では、コンストラクタと呼ばれている関数で、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を与えて、初期化します。

```
  attr_reader :suit, :rank, :image, :bkimage
```

は、Card のオブジェクトに対して、.suit などでアクセスできるようにするためのおまじないです。
次に、カードデックのクラスを作ります。

```
class Carddeck
  def initialize()
    @carddeck = []
    back_image = TkPhotoImage.new(file: 'b1fv.gif')
    for s in ["c", "d", "h", "s"]
      for r in 0...13
        name = s
        if r < 10
          name += (r+1).to_s
        elsif r == 11
          name += "j"
        elsif r == 12
          name += "q"
        else
          name += "k"
        end
        name += ".gif"
        image = TkPhotoImage.new(file: name)
        card = Card.new(s, r, image, back_image)
        @carddeck.push(card)
      end
    end
  end
  def shuffle()
    @carddeck.shuffle!
  end
  def deal()
    @carddeck.shift
  end
  attr_reader  :carddeck
end
```

このゲームではジョーカーは使わないので、52 枚のカードをオブジェクト変数 `@carddeck` にセットしています。スーツは文字列　"c", "d", "h", "s"　で、ランクは 1-13 の数字でセットしています。関数 shuffle() はカードの順番をシャッフルする関数です。関数 deal() はカードを一枚づつ配ります。ここでも

```
  attr_reader  :carddeck
```

で、.carddeck で、カードデックにアクセスできるようにしています。
次に、ゲームのクラスを作ります。

```
class Game
  def initialize()
    @deck = Carddeck.new
  end
  def run()
    canvas = TkCanvas.new(width: 500, height: 300).pack
    card = @deck.carddeck[0]
    TkcImage.new(canvas, 100, 150) do
      image(card.image)
    end
    card = @deck.deal
    TkcImage.new(canvas, 200, 150) do
      image(card.image)
    end
    @deck.shuffle
    card = @deck.carddeck[0]
    TkcImage.new(canvas, 300, 150) do
      image(card.image)
    end
    card = @deck.deal
    TkcImage.new(canvas, 400, 150) do
      image(card.image)
    end

    Tk.mainloop
  end
end
```

今のところ初期設定はカードデッキのオブジェクトを生成し、オブジェクト変数 `@deck` にセットするだけです。関数 run() でゲームを始めます。ここでは、カンバスを生成し、カードを表示しているだけです。関数 shauffle() がちゃんと動くかも確かめています。プログラムの最後に

```
game = Game.new
game.run
```

を追加します。
プログラムの全体は

```
require 'tk'

class Card
  def initialize(suit, rank, image, bkimage)
```

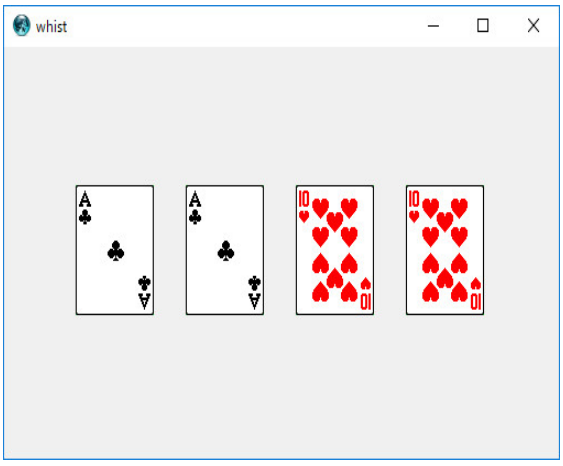
```
    @suit = suit
    @rank = rank
    @image = image
    @bkimage = bkimage
  end
  attr_reader :suit, :rank, :image, :bkimage
end
class Carddeck
  def initialize()
    @carddeck = []
    back_image = TkPhotoImage.new(file: 'b1fv.gif')
    for s in ["c", "d", "h", "s"]
      for r in 0...13
        name = s
        if r < 10
          name += (r+1).to_s
        elsif r == 11
          name += "j"
        elsif r == 12
          name += "q"
        else
          name += "k"
        end
        name += ".gif"
        image = TkPhotoImage.new(file: name)
        card = Card.new(s, r, image, back_image)
        @carddeck.push(card)
      end
    end
  end
  def shuffle()
    @carddeck.shuffle!
  end
  def deal()
    @carddeck.shift
  end
  attr_reader  :carddeck
end

class Game
  def initialize()
    @deck = Carddeck.new
  end
```

```
  def run()
    canvas = TkCanvas.new(width: 500, height: 300).pack
    card = @deck.carddeck[0]
    TkcImage.new(canvas, 100, 150) do
      image(card.image)
    end
    card = @deck.deal
    TkcImage.new(canvas, 200, 150) do
      image(card.image)
    end
    @deck.shuffle
    card = @deck.carddeck[0]
    TkcImage.new(canvas, 300, 150) do
      image(card.image)
    end
    card = @deck.deal
    TkcImage.new(canvas, 400, 150) do
      image(card.image)
    end

    Tk.mainloop
  end
end
game = Game.new
game.run
```

です。実行してみましょう。



左側のクラブのエースはいつも一緒ですが、右側には何が表示されるかはランダムで、実行する度に変化します。

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みます。

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。

```
class Player
  def initialize()
    @cards = []
  end
  def getcard(card)
    @cards.push(card)
  end
  def draw()
    @cards.shift
  end
end
class Dealer < Player
  def initialize()
    super
  end
end
class User < Player
  def initialize()
    super
  end
end
```

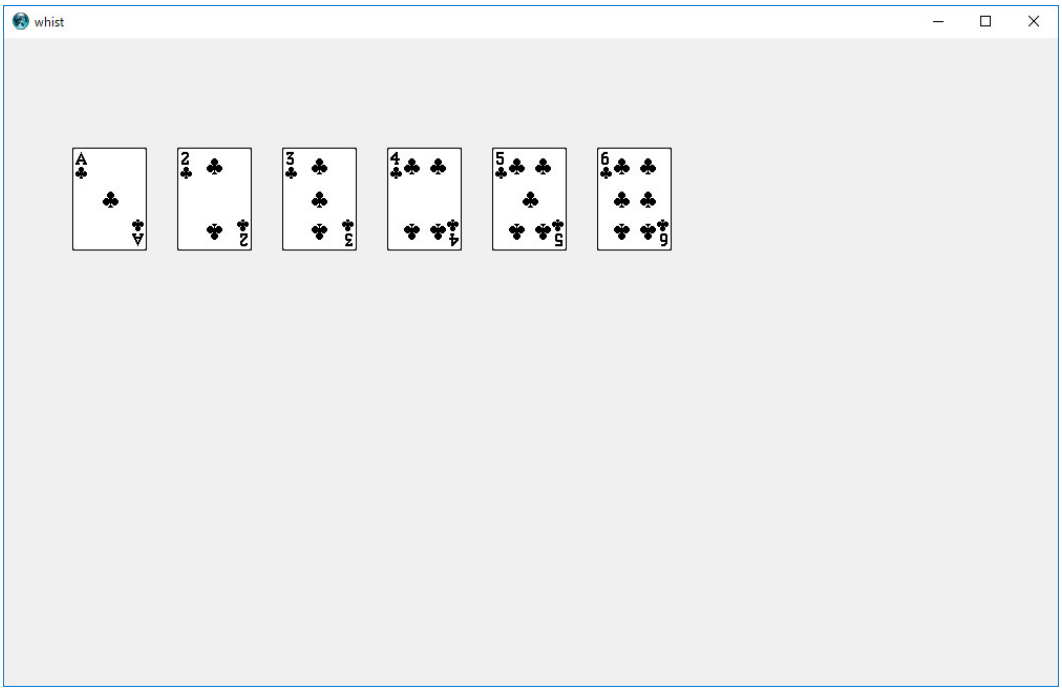
クラス Player はオブジェクト変数 `@cards` に手札を保持し、関数 getcard() でカードを受け取り、手札に追加し、関数 draw() でカードを一枚出します。ここでは、ひとまず最初にあるカードを出すことにしています。クラス Dealer と User はクラス Player を継承したクラスです。関数 draw() をオーバーロードして、適切な札を出すようにする必要があります。

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。コンストラクターで 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。

```
class Game
  def initialize()
    @deck = Carddeck.new
    @dealer = Dealer.new
    @user = User.new
  end
  def run()
    canvas = TkCanvas.new(width: 1000, height: 600).pack
    for i in 1..13
```

```
      @dealer.getcard(@deck.deal)
      @user.getcard(@deck.deal)
    end
    for i in 0..2
      card = @dealer.draw
      TkcImage.new(canvas, 100+200*i, 150) do
        image(card.image)
      end
      card = @user.draw
      TkcImage.new(canvas, 200+200*i, 150) do
        image(card.image)
      end
    end
    Tk.mainloop
  end
end
```

実行すると



です。ここまでは間違っていないみたいです。キャンバスを保持するグローバル変数 `$canvas` を作ります。

```
canvas = nil
```

クラス Dealer と User に関数 display() を定義します。

```
class Dealer < Player
  def initialize()
```

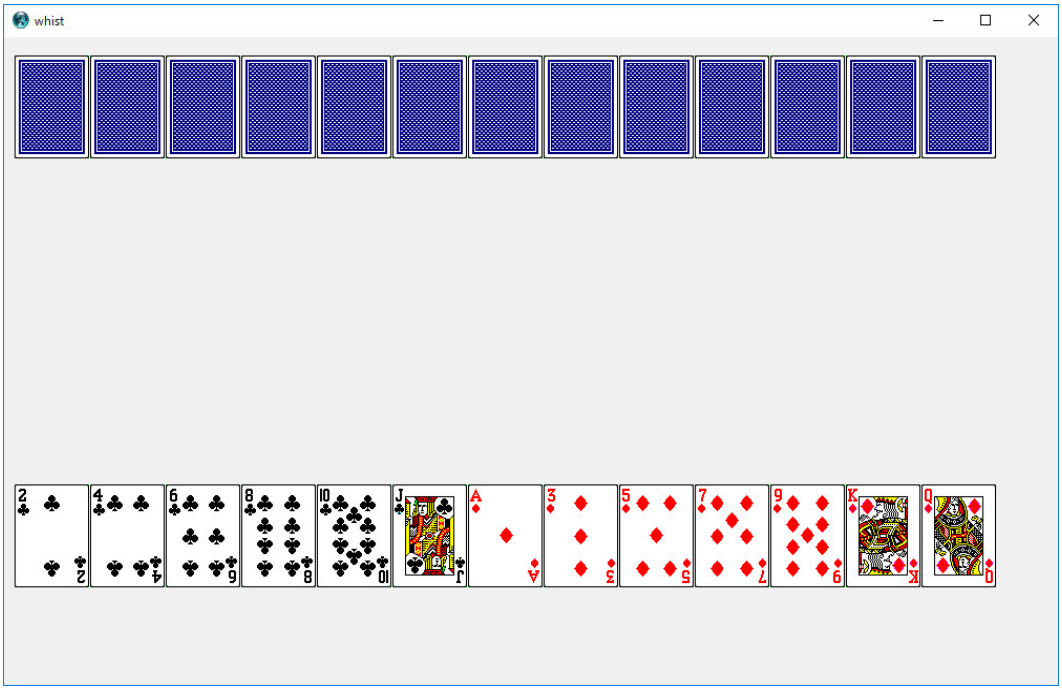
```
    super
  end
  def display()
    @cards.each_with_index do |card, i|
      TkcImage.new($canvas,45+72*i , 65) do
        image(card.bkimage)
      end
    end
  end
end
class User < Player
  def initialize()
    super
  end
  def display()
    @cards.each_with_index do |card, i|
      TkcImage.new($canvas,45+72*i , 465) do
        image(card.image)
      end
    end
  end
end
```

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。

```
class Game
  def initialize()
    @deck = Carddeck.new
    @dealer = Dealer.new
    @user = User.new
  end
  def run()
    $canvas = TkCanvas.new(width: 1000, height: 600).pack
    for i in 1..13
      @dealer.getcard(@deck.deal)
      @user.getcard(@deck.deal)
    end
    @dealer.display
    @user.display

    Tk.mainloop
  end
end
```

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。間違っています。クラス Carddeck を

```
class Carddeck
  def initialize()
    @carddeck = []
    back_image = TkPhotoImage.new(file: 'b1fv.gif')
    for s in ["c", "d", "h", "s"]
      for r in 0...13
        name = s
        if r < 10
          name += (r+1).to_s
        elsif r == 10
          name += "j"
        elsif r == 11
          name += "q"
        else
          name += "k"
        end
        name += ".gif"
        image = TkPhotoImage.new(file: name)
        card = Card.new(s, r, image, back_image)
        @carddeck.push(card)
      end
    end
  end
```

```
  def shuffle()
    @carddeck.shuffle!
  end
  def deal()
    @carddeck.shift
  end
  attr_reader  :carddeck
end
```

と修正します。if 文の数字が間違っていました。
ストックのクラス Stock も作ります。

```
class Stock
  def initialize()
    @cards = []
  end
  def getcard(card)
    @cards.push(card)
  end
  def draw()
    @cards.shift
  end
  def display()
    if @cards.size > 0
      card = @cards[0]
      TkcImage.new($canvas, 45, 265) do
        image(card.image)
      end
    end
  end
end
```

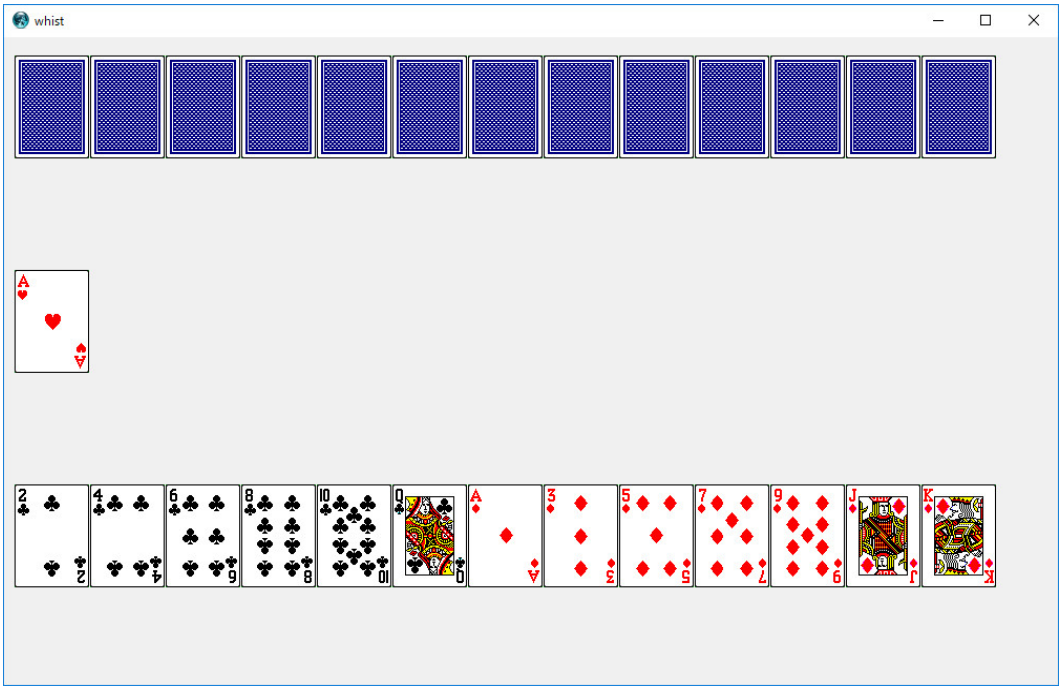
ここまでが間違っていないか確かめるために、クラス Game を修正します。

```
class Game
  def initialize()
    @deck = Carddeck.new
    @dealer = Dealer.new
    @user = User.new
    @stock = Stock.new
  end
  def run()
    $canvas = TkCanvas.new(width: 1000, height: 600).pack
    for i in 1..13
      @dealer.getcard(@deck.deal)
```

```
      @user.getcard(@deck.deal)
    end
    for i in 1..26
      @stock.getcard(@deck.deal)
    end
    @dealer.display
    @user.display
    @stock.display

    Tk.mainloop
  end
end
```

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。

```
class Getcard
  def initialize()
    @cards = []
  end
  def getcard(card)
    @cards.push(card)
  end
  def draw()
    @cards.shift
```

```
  end
end
class DealerGetCard < Getcard
  def initialize()
    super
  end
  def display()
    @cards.each_with_index do |card, i|
      TkcImage.new($canvas,45+18*i , 165) do
        image(card.image)
      end
    end
  end
end
class UserGetCard < Getcard
  def initialize()
    super
  end
  def display()
    @cards.each_with_index do |card, i|
      TkcImage.new($canvas,45+18*i , 365) do
        image(card.image)
      end
    end
  end
end
```

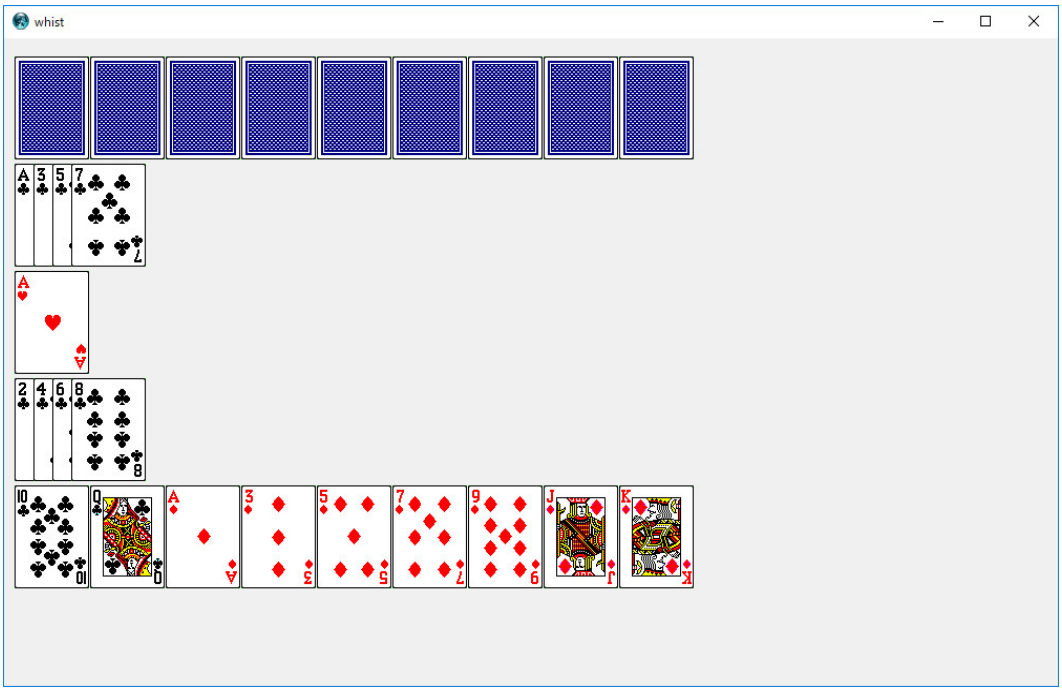
ここまでが間違っていないか確かめるために、クラス Game を修正します。

```
class Game
  def initialize()
    @deck = Carddeck.new
    @dealer = Dealer.new
    @user = User.new
    @stock = Stock.new
    @dealergetcard = DealerGetCard.new
    @usergetcard = UserGetCard.new
  end
  def run()
    $canvas = TkCanvas.new(width: 1000, height: 600).pack
    for i in 1..13
      @dealer.getcard(@deck.deal)
      @user.getcard(@deck.deal)
    end
```

```
    for i in 1..26
      @stock.getcard(@deck.deal)
    end
    for i in 0..3
      @dealergetcard.getcard(@dealer.draw)
      @usergetcard.getcard(@user.draw)
    end
    @dealer.display
    @user.display
    @stock.display
    @dealergetcard.display
    @usergetcard.display

    Tk.mainloop
  end
end
```

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。

```
class LeadCard
  def initialize()
    @cards = []
  end
  def getcard(card)
```

```
    @cards.push(card)
  end
  def draw()
    @cards.shift
  end
  def display()
    for card in @cards
      TkcImage.new($canvas,145, 265) do
        image(card.image)
      end
    end
  end
end
```

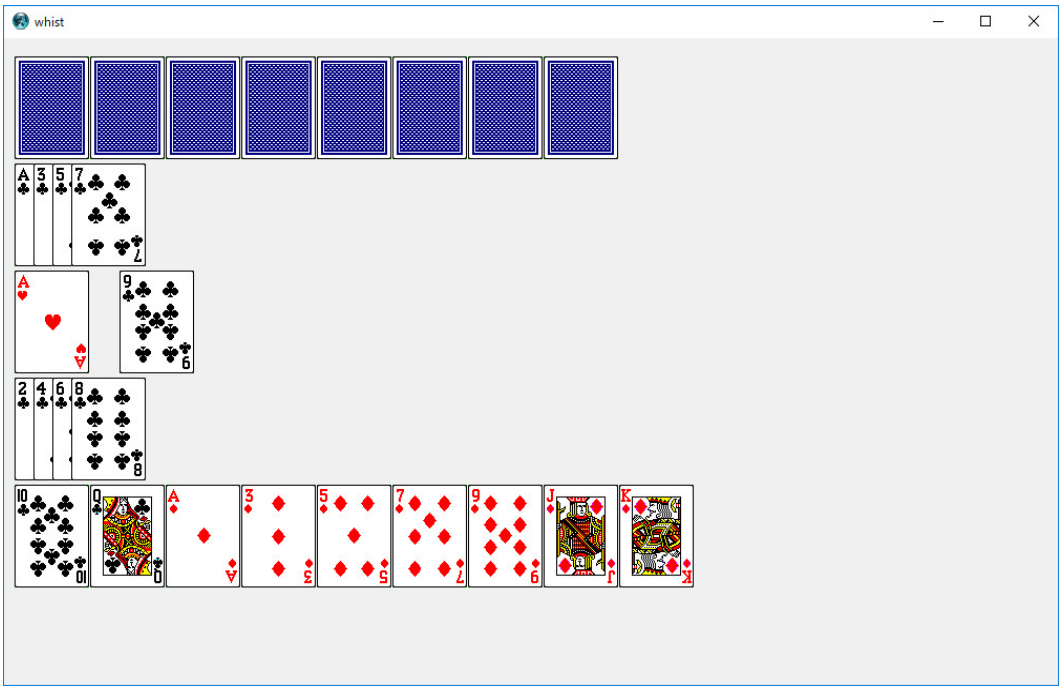
ここまでが間違っていないか確かめるために、クラス Game を修正します。

```
class Game
  def initialize()
    @deck = Carddeck.new
    @dealer = Dealer.new
    @user = User.new
    @stock = Stock.new
    @dealergetcard = DealerGetCard.new
    @usergetcard = UserGetCard.new
    @leadcard = LeadCard.new
  end
  def run()
    $canvas = TkCanvas.new(width: 1000, height: 600).pack
    for i in 1..13
      @dealer.getcard(@deck.deal)
      @user.getcard(@deck.deal)
    end
    for i in 1..26
      @stock.getcard(@deck.deal)
    end
    for i in 0..3
      @dealergetcard.getcard(@dealer.draw)
      @usergetcard.getcard(@user.draw)
    end
    @leadcard.getcard(@dealer.draw)
    @dealer.display
    @user.display
    @stock.display
    @dealergetcard.display
```

```
    @usergetcard.display
    @leadcard.display

    Tk.mainloop
  end
end
```

実行すると



です。ここまでは間違っていないみたいです。

これで登場人物がすべてそろいました。実際にゲームをするようにクラス Game を作り替えていきましょう。

```
class Game
  def initialize()
    @deck = Carddeck.new
    @dealer = Dealer.new
    @user = User.new
    @stock = Stock.new
    @dealergetcard = DealerGetCard.new
    @usergetcard = UserGetCard.new
    @leadcard = LeadCard.new
  end
  def run()
    $canvas = TkCanvas.new(width: 1000, height: 600).pack
    @deck.shuffle
    for i in 1..13
```

```
      @dealer.getcard(@deck.deal)
      @user.getcard(@deck.deal)
    end
    for i in 1..26
      @stock.getcard(@deck.deal)
    end
    @dealer.display
    @user.display
    @stock.display
    TkcText.new($canvas, 400, 265) {
      text 'カードを選んで下さい'
      font 'courier 24'
      fill 'blue'
    }
    Tk.mainloop
  end
end
```

が、ゲームの開始の局面を表示する部分までプログラミングしたところです。実行すると

です。ここまでは間違っていないみたいです。ユーザーがマウスでリードするカードを選べるようにする必要があります。まず、グローバル変数

```
$gameOverP = false
$dealerPlayP = false
$trump = nil
$leadcard = nil
```

を追加し、
クラス Player に

```
  attr_reader :cards
```

を追加し、

```
class Player
  def initialize()
    @cards = []
  end
  def getcard(card)
    @cards.push(card)
  end
  def draw()
    @cards.shift
  end
  attr_reader :cards
end
```

とします。クラス Dealer に

```
  def draw()
    if $dealerPlayP
      index = rand(@cards.size)
      @cards.delete_at(index)
    else
      index = -1
      suit = $leadcard.cards[0].suit
      for i in 0...@cards.size
        if @cards[i].suit == suit
          index = i
          break
        end
      end
      if index < 0
        for i in 0...@cards.size
          if @cards[i].suit == $trump.suit
            index = i
            break
          end
        end
        if index < 0
          index = rand(@cards.size)
        end
```

```
      end
      @cards.delete_at(index)
    end
  end
```

を追加し、

```
class Dealer < Player
  def initialize()
    super
  end
  def display()
    @cards.each_with_index do |card, i|
      TkcImage.new($canvas,45+72*i , 65) do
        image(card.bkimage)
      end
    end
  end
  def draw()
    if $dealerPlayP
      index = rand(@cards.size)
      @cards.delete_at(index)
    else
      index = -1
      suit = $leadcard.cards[0].suit
      for i in 0...@cards.size
        if @cards[i].suit == suit
          index = i
          break
        end
      end
      if index < 0
        for i in 0...@cards.size
          if @cards[i].suit == $trump.suit
            index = i
            break
          end
        end
        if index < 0
          index = rand(@cards.size)
        end
      end
      @cards.delete_at(index)
    end
```

```
  end
end
```

とします。クラス User に

```
  def draw(index)
    @cards.delete_at(index)
  end
```

を追加し、

```
class User < Player
  def initialize()
    super
  end
  def display()
    @cards.each_with_index do |card, i|
      TkcImage.new($canvas,45+72*i , 465) do
        image(card.image)
      end
    end
  end
  def draw(index)
    @cards.delete_at(index)
  end
end
```

とします。クラス Stock に

```
  attr_reader :cards
```

を追加し、

```
class Stock
  def initialize()
    @cards = []
  end
  def getcard(card)
    @cards.push(card)
  end
  def draw()
    @cards.shift
  end
  def display()
    if @cards.size > 0
      card = @cards[0]
      TkcImage.new($canvas, 45, 265) do
```

```
        image(card.image)
      end
    end
  end
  attr_reader :cards
end
```

とします。クラス LeadCard に

```
  attr_reader :cards
```

を追加し、関数 display() を修正し

```
class LeadCard
  def initialize()
    @cards = []
  end
  def getcard(card)
    @cards.push(card)
  end
  def draw()
    @cards.shift
  end
  def display()
    for card in @cards
      if card
        TkcImage.new($canvas,145, 265) do
          image(card.image)
        end
      end
    end
  end
  attr_reader :cards
end
```

とします。クラス GetCard に

```
def getvalue()
    @cards.size
  end
```

を追加し、

```
class Getcard
  def initialize()
    @cards = []
  end
```

```
  def getcard(card)
    @cards.push(card)
  end
  def draw()
    @cards.shift
  end
  def getvalue()
    @cards.size
  end
end
```

とします。最後に、クラス Game を

```
class Game
  def initialize()
    @deck = Carddeck.new
    @dealer = Dealer.new
    @user = User.new
    @stock = Stock.new
    @dealergetcard = DealerGetCard.new
    @usergetcard = UserGetCard.new
    $leadcard = LeadCard.new
    $gameOverP = false
    $dealerPlayP = false
  end
  def run()
    $canvas = TkCanvas.new(width: 1000, height: 600).pack
    $canvas.bind('Button-1', proc {|x, y| button_press(x, y)}, "%x, %y")
    TkcRectangle.new($canvas,0, 0, 1000, 600) do
      fill 'white'
    end
    def judge(usercard, dealercard)
      if usercard.suit == dealercard.suit
        if dealercard.rank == 0
          @dealergetcard.getcard(usercard)
          @dealergetcard.getcard(dealercard)
          $dealerPlayP = true
        elsif usercard.rank == 0
          @usergetcard.getcard(usercard)
          @usergetcard.getcard(dealercard)
          $dealerPlayP = false
        elsif dealercard.rank > usercard.rank
          @dealergetcard.getcard(usercard)
          @dealergetcard.getcard(dealercard)
```

```
      $dealerPlayP = true
    else
      @usergetcard.getcard(usercard)
      @usergetcard.getcard(dealercard)
      $dealerPlayP = false
    end
  elsif dealercard.suit == $trump.suit
    @dealergetcard.getcard(usercard)
    @dealergetcard.getcard(dealercard)
    $dealerPlayP = true
  elsif usercard.suit == $trump.suit
    @usergetcard.getcard(usercard)
    @usergetcard.getcard(dealercard)
    $dealerPlayP = false
  else
    if $dealerPlayP == true
      @dealergetcard.getcard(usercard)
      @dealergetcard.getcard(dealercard)
      $dealerPlayP = true
    else
      @usergetcard.getcard(usercard)
      @usergetcard.getcard(dealercard)
      $dealerPlayP = false
    end
  end
end
def legalP(card)
  suit = $leadcard.cards[0].suit
  if card.suit != suit
    flag = true
    for i in 0...@user.cards.size
      if @user.cards[i].suit == suit
        flag = false
        break
      end
    end
    return flag
  end
  true
end
def button_press(x, y)
  index = -1
  pos = 10
```

```
xx = x.to_i
yy = y.to_i
for i in 0...@user.cards.size
  if xx > pos and xx < pos+72 and yy > 415 and yy < 515
    index = i
    break
  end
  pos += 72
end
if index == -1
  return
end
if $dealerPlayP == false
  $leadcard.getcard(@user.draw(index))
  usercard = $leadcard.cards[0]
  dealercard = @dealer.draw()
  $leadcard.draw
  judge(usercard, dealercard)
else
  if legalP(@user.cards[index])
      usercard = @user.draw(index)
    dealercard = $leadcard.draw
    judge(usercard, dealercard)
  else
    ss = "リードは"
    case $leadcard.cards[0].suit
    when "c"
      ss += "クラブです"
    when "d"
      ss += "ダイヤです"
    when "h"
      ss += "ハートです"
    when "s"
      ss += "スペードです"
    end
    Tk.messageBox(title: '正しいカードを出してください',
      type: 'ok', icon: 'warning', message: ss
    )
    return
  end
end
card = @stock.draw
if card
```

```
  if $dealerPlayP == true
    @dealer.getcard(card)
    @user.getcard(@stock.draw)
  else
    @user.getcard(card)
    @dealer.getcard(@stock.draw)
  end
end
if @dealer.cards.size == 0
  $gameOverP = true
end
if $dealerPlayP == true
  $leadcard.getcard(@dealer.draw)
end
TkcRectangle.new($canvas,0, 0, 1000, 600) do
  fill 'white'
end
@dealer.display
@user.display
@stock.display
@dealergetcard.display
@usergetcard.display
$leadcard.display
if $gameOverP == false
  TkcText.new($canvas, 400, 265) {
    text 'カードを選んで下さい'
    font 'courier 24'
    fill 'blue'
  }
else
  dealervalue = @dealergetcard.getvalue
  uservalue = @usergetcard.getvalue
  ss = "Game Over "
  ss += uservalue.to_s
  ss += " : "
  ss += dealervalue.to_s
  if dealervalue > uservalue
    ss += "コンピュータの勝ちです。頑張ってね！"
  elsif dealervalue < uservalue
    ss += "あなたの勝ちです。強いですね！"
  else
    ss += "引き分けです。もう一度しませんか？"
  end
```

```
        TkcText.new($canvas, 400, 265) {
          text ss
          font 'courier 24'
          fill 'blue'
        }
      end
    end
    @deck.shuffle
    for i in 1..13
      @dealer.getcard(@deck.deal)
      @user.getcard(@deck.deal)
    end
    for i in 1..26
      @stock.getcard(@deck.deal)
    end
    @dealer.display
    @user.display
    @stock.display
    $trump = @stock.cards[0]
    TkcText.new($canvas, 400, 265) {
      text 'カードを選んで下さい'
      font 'courier 24'
      fill 'blue'
    }

    Tk.mainloop
  end
end
```

と修正します。これで、多分、正しいプログラムができました。最後に、再度、ゲームができるように、ダイアログを追加しましょう。

```
module Dialog
  def Dialog.getValue()
    var = TkVariable.new("")
    toplevel = TkToplevel.new do
      grab
    end
    TkLabel.new(toplevel) do
      text "再挑戦しますか？"
      font 'helvetica 18'
      pack
    end
    TkButton.new(toplevel) do
```

```
      text 'Yes'
      font 'helvetica 18'
      command do
        toplevel.grab("release")
        toplevel.destroy
        var.value = 'y'
      end
      pack
    end
    TkButton.new(toplevel) do
      text 'No'
      font 'helvetica 18'
      command do
        toplevel.grab("release")
        toplevel.destroy
        var.value = 'n'
      end
      pack
    end
    toplevel.wait_destroy
    var.value
  end
end
```

を追加し、関数 button_press() の最後に

```
        v = Dialog.getValue()
        if v == 'y'
          initialize()
          @deck.shuffle
          for i in 1..13
            @dealer.getcard(@deck.deal)
            @user.getcard(@deck.deal)
          end
          for i in 1..26
            @stock.getcard(@deck.deal)
          end
          TkcRectangle.new($canvas,0, 0, 1000, 600) do
            fill 'white'
          end
          @dealer.display
          @user.display
          @stock.display
          $trump = @stock.cards[0]
```

```
        TkcText.new($canvas, 400, 265) {
          text 'カードを選んで下さい'
          font 'courier 24'
          fill 'blue'
        }
```

を追加し、関数 button_press() は

```
    def button_press(x, y)
      index = -1
      pos = 10
      xx = x.to_i
      yy = y.to_i
      for i in 0...@user.cards.size
        if xx > pos and xx < pos+72 and yy > 415 and yy < 515
          index = i
          break
        end
        pos += 72
      end
      if index == -1
        return
      end
      if $dealerPlayP == false
        $leadcard.getcard(@user.draw(index))
        usercard = $leadcard.cards[0]
        dealercard = @dealer.draw()
        $leadcard.draw
        judge(usercard, dealercard)
      else
        if legalP(@user.cards[index])
            usercard = @user.draw(index)
          dealercard = $leadcard.draw
          judge(usercard, dealercard)
        else
          ss = "リードは"
          case $leadcard.cards[0].suit
          when "c"
            ss += "クラブです"
          when "d"
            ss += "ダイヤです"
          when "h"
            ss += "ハートです"
          when "s"
```

```
      ss += "スペードです"
    end
    Tk.messageBox(title: ' 正しいカードを出してください',
      type: 'ok', icon: 'warning', message: ss
    )
    return
  end
end
card = @stock.draw
if card
  if $dealerPlayP == true
    @dealer.getcard(card)
    @user.getcard(@stock.draw)
  else
    @user.getcard(card)
    @dealer.getcard(@stock.draw)
  end
end
if @dealer.cards.size == 0
  $gameOverP = true
end
if $dealerPlayP == true
  $leadcard.getcard(@dealer.draw)
end
TkcRectangle.new($canvas,0, 0, 1000, 600) do
  fill 'white'
end
@dealer.display
@user.display
@stock.display
@dealergetcard.display
@usergetcard.display
$leadcard.display
if $gameOverP == false
  TkcText.new($canvas, 400, 265) {
    text ' カードを選んで下さい'
    font 'courier 24'
    fill 'blue'
  }
else
  dealervalue = @dealergetcard.getvalue
  uservalue = @usergetcard.getvalue
  ss = "Game Over "
```

```
      ss += uservalue.to_s
      ss += " : "
      ss += dealervalue.to_s
      if dealervalue > uservalue
        ss += "コンピュータの勝ちです。頑張ってね！"
      elsif dealervalue < uservalue
        ss += "あなたの勝ちです。強いですね！"
      else
        ss += "引き分けです。もう一度しませんか？"
      end
      TkcText.new($canvas, 400, 265) {
        text ss
        font 'courier 24'
        fill 'blue'
      }
      v = Dialog.getValue()
      if v == 'y'
        initialize()
        @deck.shuffle
        for i in 1..13
          @dealer.getcard(@deck.deal)
          @user.getcard(@deck.deal)
        end
        for i in 1..26
          @stock.getcard(@deck.deal)
        end
        TkcRectangle.new($canvas,0, 0, 1000, 600) do
          fill 'white'
        end
        @dealer.display
        @user.display
        @stock.display
        $trump = @stock.cards[0]
        TkcText.new($canvas, 400, 265) {
          text ' カードを選んで下さい'
          font 'courier 24'
          fill 'blue'
        }
      end
    end
  end
```

となります。最終的なプログラムの全体は

```
require 'tk'

class Card
  def initialize(suit, rank, image, bkimage)
    @suit = suit
    @rank = rank
    @image = image
    @bkimage = bkimage
  end
  attr_reader :suit, :rank, :image, :bkimage
end
class Carddeck
  def initialize()
    @carddeck = []
    back_image = TkPhotoImage.new(file: 'b1fv.gif')
    for s in ["c", "d", "h", "s"]
      for r in 0...13
        name = s
        if r < 10
          name += (r+1).to_s
        elsif r == 10
          name += "j"
        elsif r == 11
          name += "q"
        else
          name += "k"
        end
        name += ".gif"
        image = TkPhotoImage.new(file: name)
        card = Card.new(s, r, image, back_image)
        @carddeck.push(card)
      end
    end
  end
  def shuffle()
    @carddeck.shuffle!
  end
  def deal()
    @carddeck.shift
  end
  attr_reader  :carddeck
end
class Player
```

```
  def initialize()
    @cards = []
  end
  def getcard(card)
    @cards.push(card)
  end
  def draw()
    @cards.shift
  end
  attr_reader :cards
end
class Dealer < Player
  def initialize()
    super
  end
  def display()
    @cards.each_with_index do |card, i|
      TkcImage.new($canvas,45+72*i , 65) do
        image(card.bkimage)
      end
    end
  end
  def draw()
    if $dealerPlayP
      index = rand(@cards.size)
      @cards.delete_at(index)
    else
      index = -1
      suit = $leadcard.cards[0].suit
      for i in 0...@cards.size
        if @cards[i].suit == suit
          index = i
          break
        end
      end
      if index < 0
        for i in 0...@cards.size
          if @cards[i].suit == $trump.suit
            index = i
            break
          end
        end
        if index < 0
```

```
          index = rand(@cards.size)
        end
      end
      @cards.delete_at(index)
    end
  end
end
class User < Player
  def initialize()
    super
  end
  def display()
    @cards.each_with_index do |card, i|
      TkcImage.new($canvas,45+72*i , 465) do
        image(card.image)
      end
    end
  end
  def draw(index)
    @cards.delete_at(index)
  end
end

class Stock
  def initialize()
    @cards = []
  end
  def getcard(card)
    @cards.push(card)
  end
  def draw()
    @cards.shift
  end
  def display()
    if @cards.size > 0
      card = @cards[0]
      TkcImage.new($canvas, 45, 265) do
        image(card.image)
      end
    end
  end
  attr_reader :cards
end
```

```
class Getcard
  def initialize()
    @cards = []
  end
  def getcard(card)
    @cards.push(card)
  end
  def draw()
    @cards.shift
  end
  def getvalue()
    @cards.size
  end
end
class DealerGetCard < Getcard
  def initialize()
    super
  end
  def display()
    @cards.each_with_index do |card, i|
      TkcImage.new($canvas,45+18*i , 165) do
        image(card.image)
      end
    end
  end
end
class UserGetCard < Getcard
  def initialize()
    super
  end
  def display()
    @cards.each_with_index do |card, i|
      TkcImage.new($canvas,45+18*i , 365) do
        image(card.image)
      end
    end
  end
end

class LeadCard
  def initialize()
    @cards = []
```

```
  end
  def getcard(card)
    @cards.push(card)
  end
  def draw()
    @cards.shift
  end
  def display()
    for card in @cards
      if card
        TkcImage.new($canvas,145, 265) do
          image(card.image)
        end
      end
    end
  end
  attr_reader :cards
end

module Dialog
  def Dialog.getValue()
    var = TkVariable.new("")
    toplevel = TkToplevel.new do
      grab
    end
    TkLabel.new(toplevel) do
      text "再挑戦しますか？"
      font 'helvetica 18'
      pack
    end
    TkButton.new(toplevel) do
      text 'Yes'
      font 'helvetica 18'
      command do
        toplevel.grab("release")
        toplevel.destroy
        var.value = 'y'
      end
      pack
    end
    TkButton.new(toplevel) do
      text 'No'
      font 'helvetica 18'
```

```
      command do
        toplevel.grab("release")
        toplevel.destroy
        var.value = 'n'
      end
      pack
    end
    toplevel.wait_destroy
    var.value
  end
end

$canvas = nil
$gameOverP = false
$dealerPlayP = false
$trump = nil
$leadcard = nil

class Game
  def initialize()
    @deck = Carddeck.new
    @dealer = Dealer.new
    @user = User.new
    @stock = Stock.new
    @dealergetcard = DealerGetCard.new
    @usergetcard = UserGetCard.new
    $leadcard = LeadCard.new
    $gameOverP = false
    $dealerPlayP = false
  end
  def run()
    $canvas = TkCanvas.new(width: 1000, height: 600).pack
    $canvas.bind('Button-1', proc {|x, y| button_press(x, y)}, "%x, %y")
    TkcRectangle.new($canvas,0, 0, 1000, 600) do
      fill 'white'
    end
    def judge(usercard, dealercard)
      if usercard.suit == dealercard.suit
        if dealercard.rank == 0
          @dealergetcard.getcard(usercard)
          @dealergetcard.getcard(dealercard)
          $dealerPlayP = true
        elsif usercard.rank == 0
```

```
      @usergetcard.getcard(usercard)
      @usergetcard.getcard(dealercard)
      $dealerPlayP = false
    elsif dealercard.rank > usercard.rank
      @dealergetcard.getcard(usercard)
      @dealergetcard.getcard(dealercard)
      $dealerPlayP = true
    else
      @usergetcard.getcard(usercard)
      @usergetcard.getcard(dealercard)
      $dealerPlayP = false
    end
  elsif dealercard.suit == $trump.suit
    @dealergetcard.getcard(usercard)
    @dealergetcard.getcard(dealercard)
    $dealerPlayP = true
  elsif usercard.suit == $trump.suit
    @usergetcard.getcard(usercard)
    @usergetcard.getcard(dealercard)
    $dealerPlayP = false
  else
    if $dealerPlayP == true
      @dealergetcard.getcard(usercard)
      @dealergetcard.getcard(dealercard)
      $dealerPlayP = true
    else
      @usergetcard.getcard(usercard)
      @usergetcard.getcard(dealercard)
      $dealerPlayP = false
    end
  end
end
def legalP(card)
  suit = $leadcard.cards[0].suit
  if card.suit != suit
    flag = true
    for i in 0...@user.cards.size
      if @user.cards[i].suit == suit
        flag = false
        break
      end
    end
    return flag
```

```
  end
  true
end
def button_press(x, y)
  index = -1
  pos = 10
  xx = x.to_i
  yy = y.to_i
  for i in 0...@user.cards.size
    if xx > pos and xx < pos+72 and yy > 415 and yy < 515
      index = i
      break
    end
    pos += 72
  end
  if index == -1
    return
  end
  if $dealerPlayP == false
    $leadcard.getcard(@user.draw(index))
    usercard = $leadcard.cards[0]
    dealercard = @dealer.draw()
    $leadcard.draw
    judge(usercard, dealercard)
  else
    if legalP(@user.cards[index])
        usercard = @user.draw(index)
      dealercard = $leadcard.draw
      judge(usercard, dealercard)
    else
      ss = "リードは"
      case $leadcard.cards[0].suit
      when "c"
        ss += "クラブです"
      when "d"
        ss += "ダイヤです"
      when "h"
        ss += "ハートです"
      when "s"
        ss += "スペードです"
      end
      Tk.messageBox(title: '正しいカードを出してください',
        type: 'ok', icon: 'warning', message: ss
```

```
    )
    return
  end
end
card = @stock.draw
if card
  if $dealerPlayP == true
    @dealer.getcard(card)
    @user.getcard(@stock.draw)
  else
    @user.getcard(card)
    @dealer.getcard(@stock.draw)
  end
end
if @dealer.cards.size == 0
  $gameOverP = true
end
if $dealerPlayP == true
  $leadcard.getcard(@dealer.draw)
end
TkcRectangle.new($canvas,0, 0, 1000, 600) do
  fill 'white'
end
@dealer.display
@user.display
@stock.display
@dealergetcard.display
@usergetcard.display
$leadcard.display
if $gameOverP == false
  TkcText.new($canvas, 400, 265) {
    text 'カードを選んで下さい'
    font 'courier 24'
    fill 'blue'
  }
else
  dealervalue = @dealergetcard.getvalue
  uservalue = @usergetcard.getvalue
  ss = "Game Over "
  ss += uservalue.to_s
  ss += " : "
  ss += dealervalue.to_s
  if dealervalue > uservalue
```

```
      ss += "コンピュータの勝ちです。頑張ってね！"
    elsif dealervalue < uservalue
      ss += "あなたの勝ちです。強いですね！"
    else
      ss += "引き分けです。もう一度しませんか？"
    end
    TkcText.new($canvas, 400, 265) {
      text ss
      font 'courier 24'
      fill 'blue'
    }
    v = Dialog.getValue()
    if v == 'y'
      initialize()
      @deck.shuffle
      for i in 1..13
        @dealer.getcard(@deck.deal)
        @user.getcard(@deck.deal)
      end
      for i in 1..26
        @stock.getcard(@deck.deal)
      end
      TkcRectangle.new($canvas,0, 0, 1000, 600) do
        fill 'white'
      end
      @dealer.display
      @user.display
      @stock.display
      $trump = @stock.cards[0]
      TkcText.new($canvas, 400, 265) {
        text 'カードを選んで下さい'
        font 'courier 24'
        fill 'blue'
      }
    end
  end
end
@deck.shuffle
for i in 1..13
  @dealer.getcard(@deck.deal)
  @user.getcard(@deck.deal)
end
for i in 1..26
```

```
      @stock.getcard(@deck.deal)
    end
    @dealer.display
    @user.display
    @stock.display
    $trump = @stock.cards[0]
    TkcText.new($canvas, 400, 265) {
      text ’カードを選んで下さい’
      font ’courier 24’
      fill ’blue’
    }

    Tk.mainloop
  end
end
game = Game.new
game.run
```

です。

このプログラムは、私の「ジャーマン・ホイスト」の Python のプログラム

```
# -*- coding:utf-8 -*-
from Tkinter import *
import random

def callback():
    print "callback"

class Card:
    def __init__(self, suit, rank, image, image2):
        self.suit = suit
        self.rank = rank
        self.image = image
        self.backimage = image2
    def getSuit(self):
        return self.suit
    def getRank(self):
        return self.rank
    def setSuit(self, s):
        self.suit = s
    def setRank(self, r):
        self.rank = r
carddeck = []
comHand = []
```

```
userHand = []
Yama = []
getCom = []
getUser = []
gameOverP = False
comPlayP = False
dispCom = []

trump = 0

def card_set():
    back_image = PhotoImage(file = 'b1fv.gif')

    global carddeck
    for s in range(4):
        for r in range(13):
            name = ""
            if s == 0:
                name = "c"
            elif s == 1:
                name = "d"
            elif s == 2:
                name = "h"
            else:
                name = "s"
            if r < 10:
                name += str(r+1)
            elif r == 11:
                name += "j"
            elif r == 12:
                name += "q"
            else:
                name += "k"
            name += ".gif"
            image = PhotoImage(file = name)
            card = Card(s, r, image, back_image)
            carddeck.append(card)

def card_initialize():
    for i in range(104):
        s = random.randrange(0, 52)
        t = random.randrange(0, 52)
        temp = carddeck[s]
```

```
        carddeck[s] = carddeck[t]
        carddeck[t] = temp

def gameInitialize():
    global carddeck, comHand, userHand, Yama, getCom, getUser, dispCom
    global gameOverP, comPlayP, trump
    comHand = []
    userHand = []
    Yama = []

    card_initialize()
    index = 0
    for i in range(13):
        comHand.append(carddeck[index])
        index += 1
    for i in range(13):
        userHand.append(carddeck[index])
        index += 1
    for i in range(26):
        Yama.append(carddeck[index])
        index += 1
    getCom = []
    getUser = []
    dispCom = []
    gameOverP = False
    comPlayP = False
    trump = Yama[len(Yama)-1].getSuit()

def ShowBan():
    canvas.create_rectangle(0, 0, 1000, 600, fill="white")
    pos = 45
    for i in range(len(comHand)):
        canvas.create_image(pos, 65, image = comHand[i].backimage)
        pos += 72
    pos = 45
    for i in range(len(getCom)):
        canvas.create_image(pos, 165, image = getCom[i].image)
        pos += 18
    pos = 45
    if len(Yama) > 0:
        canvas.create_image(pos, 265, image = Yama[len(Yama)-1].image)
    pos = 145
    if len(dispCom) > 0:
```

```
            canvas.create_image(pos, 265, image = dispCom[0].image)

        pos = 45
        for i in range(len(getUser)):
            canvas.create_image(pos, 365, image = getUser[i].image)
            pos += 18
        pos = 45
        for i in range(len(userHand)):
            canvas.create_image(pos, 465, image = userHand[i].image)
            pos += 72
        f1, f2 = "courier 24", "courier 12"
        c1, c2, c3 = "blue", "red", "black"
        if gameOverP == False:
            canvas.create_text(400, 265, text=u"カードを選んで下さい", font=f1, fill=c1)
        else:
            comValue = len(getCom)/2
            userValue = len(getUser)/2
            drawString = "Game Over "
            drawString += str(userValue)
            drawString += " : "
            drawString += str(comValue)
            if comValue > userValue:
                drawString += u" コンピュータの勝ちです。頑張ってね！"
                canvas.create_text(500, 265, text=drawString, font=f1, fill=c1)
            elif comValue < userValue:
                drawString += u" あなたの勝ちです。強いですね！"
                canvas.create_text(500, 265, text=drawString, font=f1, fill=c1)
            else:
                drawString += u"  引き分けです。もう一度しませんか？"
                canvas.create_text(500, 265, text=drawString, font=f1, fill=c1)

    def buttonPress(event):
        global carddeck, comHand, userHand, Yama, getCom, getUser, dispCom
        global gameOverP, comPlayP, trump
        mX = event.x
        mY = event.y
        userIndex = -1
        pos = 10
        for i in range(len(userHand)):
            if mX>pos and mX<pos+72 and mY>415 and mY<515:
                userIndex = i
                break
            pos += 72
```

```
if userIndex < 0:
    return
if comPlayP == False:
    suit = userHand[userIndex].suit
    rank = userHand[userIndex].rank
    comIndex = -1
    for i in range(len(comHand)):
        if comHand[i].suit == suit:
            comIndex = i
            break
    if comIndex < 0:
        for i in range(len(comHand)):
            if comHand[i].suit == trump:
                comIndex = i
                break
        if comIndex < 0:
            comIndex = random.randrange(0, len(comHand))
    if comHand[comIndex].suit == userHand[userIndex].suit:
        if comHand[comIndex].rank == 0:
            getCom.append(userHand[userIndex])
            getCom.append(comHand[comIndex])
            comPlayP = True
        elif userHand[userIndex].rank == 0:
            getUser.append(userHand[userIndex])
            getUser.append(comHand[comIndex])
            comPlayP = False
        elif comHand[comIndex].rank > userHand[userIndex].rank:
            getCom.append(userHand[userIndex])
            getCom.append(comHand[comIndex])
            comPlayP = True
        else:
            getUser.append(userHand[userIndex])
            getUser.append(comHand[comIndex])
            comPlayP = False
    elif comHand[comIndex].suit == trump:
        getCom.append(userHand[userIndex])
        getCom.append(comHand[comIndex])
        comPlayP = True
    elif userHand[userIndex].suit == trump:
        getUser.append(userHand[userIndex])
        getUser.append(comHand[comIndex])
        comPlayP = False
    else:
```

```
            getUser.append(userHand[userIndex])
            getUser.append(comHand[comIndex])
            comPlayP = False
        del userHand[userIndex]
        del comHand[comIndex]
    else:
        suit = userHand[userIndex].suit
        rank = userHand[userIndex].rank
        if suit != dispCom[0].suit:
            flag = False
            for i in range(len(userHand)):
                if userHand[i].suit == dispCom[0].suit:
                    flag = True
                    break
            if flag == True:
                s ="リードは"
                if dispCom[0].suit == 0:
                    s += "クラブ"
                elif dispCom[0].suit == 1:
                    s += "ダイヤ"
                elif dispCom[0].suit == 2:
                    s += "ハート"
                else:
                    s += "スペード"
                s += "です"
                sub_win = Toplevel()
                Message(sub_win, text=s).pack()
                return
        if dispCom[0].suit == userHand[userIndex].suit:
            if dispCom[0].rank == 0:
                getCom.append(userHand[userIndex])
                getCom.append(dispCom[0])
                comPlayP = True
            elif userHand[userIndex].rank == 0:
                getUser.append(userHand[userIndex])
                getUser.append(dispCom[0])
                comPlayP = False
            elif dispCom[0].rank > userHand[userIndex].rank:
                getCom.append(userHand[userIndex])
                getCom.append(dispCom[0])
                comPlayP = True
            else:
                getUser.append(userHand[userIndex])
```

```
                getUser.append(dispCom[0])
                comPlayP = False
        elif dispCom[0].suit == trump:
            getCom.append(userHand[userIndex])
            getCom.append(dispCom[0])
            comPlayP = True
        elif userHand[userIndex].suit == trump:
            getUser.append(userHand[userIndex])
            getUser.append(dispCom[0])
            comPlayP = False
        else:
            getCom.append(userHand[userIndex])
            getCom.append(dispCom[0])
            comPlayP = True
        del userHand[userIndex]
    if len(Yama) > 0:
        if comPlayP == True:
            comHand.append(Yama[len(Yama)-1])
            del Yama[len(Yama)-1]
            userHand.append(Yama[len(Yama)-1])
            del Yama[len(Yama)-1]
        else:
            userHand.append(Yama[len(Yama)-1])
            del Yama[len(Yama)-1]
            comHand.append(Yama[len(Yama)-1])
            del Yama[len(Yama)-1]
    if len(comHand) == 0:
        gameOverP = True
        dispCom = []
    elif comPlayP == True:
        comIndex = random.randrange(0, len(comHand))
        dispCom = [comHand[comIndex]]
        del comHand[comIndex]
    else:
        dispCom = []
    ShowBan()

def game_start():
    random.seed()
    card_set()
    gameInitialize()
    ShowBan()
```

```
root = Tk()
canvas = Canvas(root, width=1000, height=600)
canvas.pack()
canvas.bind("<ButtonPress-1>", buttonPress)
button = Button(root, text='Game Start', command=game_start)
button.pack()
root.mainloop()
```

を見ながら作りました。オブジェクト指向プログラミングにすると少し長くなるみたいです。クラス Game の関数が長く複雑です。クラス Referee を作り、簡単にしてみましょう。クラス Referee は

```
class Referee
  def Referee.judge(usercard, dealercard)
    if usercard.suit == dealercard.suit
      if dealercard.rank == 0
        $game.dealergetcard.getcard(usercard)
        $game.dealergetcard.getcard(dealercard)
        $dealerPlayP = true
      elsif usercard.rank == 0
        $game.usergetcard.getcard(usercard)
        $game.usergetcard.getcard(dealercard)
        $dealerPlayP = false
      elsif dealercard.rank > usercard.rank
        $game.dealergetcard.getcard(usercard)
        $game.dealergetcard.getcard(dealercard)
        $dealerPlayP = true
      else
        $game.usergetcard.getcard(usercard)
        $game.usergetcard.getcard(dealercard)
        $dealerPlayP = false
      end
    elsif dealercard.suit == $trump.suit
      $game.dealergetcard.getcard(usercard)
      $game.dealergetcard.getcard(dealercard)
      $dealerPlayP = true
    elsif usercard.suit == $trump.suit
      $game.usergetcard.getcard(usercard)
      $game.usergetcard.getcard(dealercard)
      $dealerPlayP = false
    else
      if $dealerPlayP == true
        $game.dealergetcard.getcard(usercard)
        $game.dealergetcard.getcard(dealercard)
        $dealerPlayP = true
```

```
    else
      $game.usergetcard.getcard(usercard)
      $game.usergetcard.getcard(dealercard)
      $dealerPlayP = false
    end
  end
end
def Referee.legalP(card)
  suit = $leadcard.cards[0].suit
  if card.suit != suit
    flag = true
    for i in 0...$game.user.cards.size
      if $game.user.cards[i].suit == suit
        flag = false
        ss = "リードは"
        case $leadcard.cards[0].suit
        when "c"
          ss += "クラブです"
        when "d"
          ss += "ダイヤです"
        when "h"
          ss += "ハートです"
        when "s"
          ss += "スペードです"
        end
        Tk.messageBox(title: '正しいカードを出してください',
          type: 'ok', icon: 'warning', message: ss
        )
        break
      end
    end
    return flag
  end
  true
end
def Referee.adjudge()
  dealervalue = $game.dealergetcard.getvalue
  uservalue = $game.usergetcard.getvalue
  ss = "Game Over "
  ss += uservalue.to_s
  ss += " : "
  ss += dealervalue.to_s
  if dealervalue > uservalue
```

```
      ss += "コンピュータの勝ちです。頑張ってね！"
    elsif dealervalue < uservalue
      ss += "あなたの勝ちです。強いですね！"
    else
      ss += "引き分けです。もう一度しませんか？"
    end
    TkcText.new($canvas, 400, 265) {
      text ss
      font 'courier 24'
      fill 'blue'
    }
    Dialog.getValue()
  end
end
```

です。クラスメソッドを使っています。クラス Game は次のように修正します。

```
class Game
  def initialize()
    @deck = Carddeck.new
    @dealer = Dealer.new
    @user = User.new
    @stock = Stock.new
    @dealergetcard = DealerGetCard.new
    @usergetcard = UserGetCard.new
    $leadcard = LeadCard.new
    $gameOverP = false
    $dealerPlayP = false
  end
  attr_reader :dealer, :user, :stock, :dealergetcard, :usergetcard

  def run()
    $canvas = TkCanvas.new(width: 1000, height: 600).pack
    $canvas.bind('Button-1', proc {|x, y| button_press(x, y)}, "%x, %y")
    TkcRectangle.new($canvas,0, 0, 1000, 600) do
      fill 'white'
    end
    def button_press(x, y)
      index = -1
      pos = 10
      xx = x.to_i
      yy = y.to_i
      for i in 0...@user.cards.size
        if xx > pos and xx < pos+72 and yy > 415 and yy < 515
```

```
    index = i
    break
  end
  pos += 72
end
if index == -1
  return
end
if $dealerPlayP == false
  $leadcard.getcard(@user.draw(index))
  usercard = $leadcard.cards[0]
  dealercard = @dealer.draw()
  $leadcard.draw
  Referee.judge(usercard, dealercard)
else
  if Referee.legalP(@user.cards[index])
    usercard = @user.draw(index)
    dealercard = $leadcard.draw
    Referee.judge(usercard, dealercard)
  else
    return
  end
end
card = @stock.draw
if card
  if $dealerPlayP == true
    @dealer.getcard(card)
    @user.getcard(@stock.draw)
  else
    @user.getcard(card)
    @dealer.getcard(@stock.draw)
  end
end
if @dealer.cards.size == 0
  $gameOverP = true
end
if $dealerPlayP == true
  $leadcard.getcard(@dealer.draw)
end
TkcRectangle.new($canvas,0, 0, 1000, 600) do
  fill 'white'
end
@dealer.display
```

```
  @user.display
  @stock.display
  @dealergetcard.display
  @usergetcard.display
  $leadcard.display
  if $gameOverP == false
    TkcText.new($canvas, 400, 265) {
      text ' カードを選んで下さい'
      font 'courier 24'
      fill 'blue'
    }
  else
    if Referee.adjudge() == 'y'
      initialize()
      @deck.shuffle
      for i in 1..13
        @dealer.getcard(@deck.deal)
        @user.getcard(@deck.deal)
      end
      for i in 1..26
        @stock.getcard(@deck.deal)
      end
      TkcRectangle.new($canvas,0, 0, 1000, 600) do
        fill 'white'
      end
      @dealer.display
      @user.display
      @stock.display
      $trump = @stock.cards[0]
      TkcText.new($canvas, 400, 265) {
        text ' カードを選んで下さい'
        font 'courier 24'
        fill 'blue'
      }
    end
  end
end
@deck.shuffle
for i in 1..13
  @dealer.getcard(@deck.deal)
  @user.getcard(@deck.deal)
end
for i in 1..26
```

```
      @stock.getcard(@deck.deal)
    end
    @dealer.display
    @user.display
    @stock.display
    $trump = @stock.cards[0]
    TkcText.new($canvas, 400, 265) {
      text ’カードを選んで下さい’
      font ’courier 24’
      fill ’blue’
    }

    Tk.mainloop
  end
end
```

そして、game をグローバル変数とし、

```
$game = Game.new
$game.run
```

で、メソッド run() を呼び出します。

クラスをいっぱい作って、複雑怪奇なプログラムになって、Python 版で与えたプログラムの方が理解しやすいのではないかと思うかもしれませんが、このプログラムを改良して、コンピュータが考えて手札を出すようにするには、、クラス Dealer

```
class Dealer < Player
  def initialize()
    super
  end
  def display()
    @cards.each_with_index do |card, i|
      TkcImage.new($canvas,45+72*i , 65) do
        image(card.bkimage)
      end
    end
  end
  def draw()
    if $dealerPlayP
      index = rand(@cards.size)
      @cards.delete_at(index)
    else
      index = -1
      suit = $leadcard.cards[0].suit
      for i in 0...@cards.size
```

```
        if @cards[i].suit == suit
          index = i
          break
        end
      end
      if index < 0
        for i in 0...@cards.size
          if @cards[i].suit == $trump.suit
            index = i
            break
          end
        end
        if index < 0
          index = rand(@cards.size)
        end
      end
      @cards.delete_at(index)
    end
  end
end
```

の関数　draw()

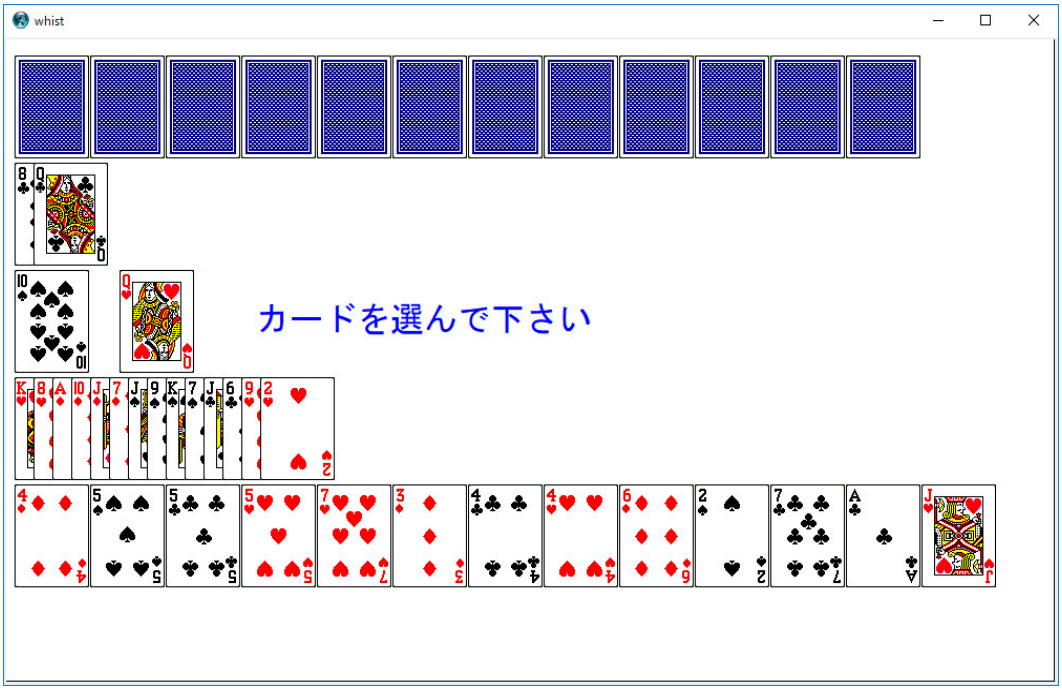
```
  def draw()
    if $dealerPlayP
      index = rand(@cards.size)
      @cards.delete_at(index)
    else
      index = -1
      suit = $leadcard.cards[0].suit
      for i in 0...@cards.size
        if @cards[i].suit == suit
          index = i
          break
        end
      end
      if index < 0
        for i in 0...@cards.size
          if @cards[i].suit == $trump.suit
            index = i
            break
          end
        end
        if index < 0
```

```
          index = rand(@cards.size)
        end
      end
      @cards.delete_at(index)
    end
  end
```

を修正すれば良いので、プログラムが見やすくなります。コンピュータは単にルールに合った手を選んでいるだけです。強くするのは難しいですが、挑戦してみてください。

実行すると

です。



がプレイの途中図です。

がプレイの終了画面です。ダイアログボックスの「Yes」をクリックすると

と、再びゲームをすることが出来ます。

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（強化学習など）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。