Eagle's Wing is a single-deck solitaire card game.

See:
Description

Packages	
impl	
playerAPI	

Eagle's Wing is a single-deck solitaire card game. The objective is to sort the deck by suit into four piles, each pile ordered by increasing rank, kings wrapping around to aces.

The game layout has fifteen piles: four **foundation piles** (refered to in the previous paragraph), eight **wing piles**, and **trunk**, **stock**, and **waste** piles. Cards in a foundation pile match in suit and increase in rank, kings wrapping around to aces. Cards in a wing pile match in suit and decrease in rank, aces wrapping to kings. A wing pile has at most three cards.

At the start of the game one foundation pile contains one face-up card, known as the **foundation card**. The stock contains thirteen cards, all face-down except for the top card, which is face up. Each wing pile has a single card. The waste pile is empty and the stock contains the rest of the cards (thirty of them), all but the top card face down.

A card can be moved from any pile to the top of a foundation pile as long as the card matchs in suit and increases sequentially in rank with top card already in the foundation pile. The first card in an empty foundation pile must match in rank with the foundation card. The game ends in a win when the foundation piles contain all the cards. Once a card is moved to a foundation pile, it is not moved again.

In addition to the foundation piles, the top card on the trunk can be moved to a wing pile as long as the card matches in suit with cards already in the wing pile, the cards rand descends sequentially from the wing-pile top card, and the wing pile has less than three cards. The top trunk card can always be moved to an empty wing pile. No other moves from the trunk are legal.

Cards in a wing pile can be moved to a foundation pile as long as the suits match and the wing cards sequentially increase in rank from the top foundation-pile card or the foundation pile's empty and the top wing-pile card matches the foundation rank. Wing card are moved to a foundation pile in top-to-bottom order; for example the wing pile `5h 4h 3h` (top card to the right) moves to the wing pile `2h` to give 2h 3h 4h 5h. One wing pile can be moved to another wing pile as long as cards in the source wing pile match in suit the cards in the destination pile, the source wing-pile cards sequentially descend from the destination wing-pile cards, and sum of the number of cards in both pile is at most three. No other moves from wing piles are legal.

The top card on the stock is in play, and can be moved to the top of the waste pile at any time, as well as the foundation subject to the constraints on those piles. Once the trunk is empty, moves to the wing piles are allowed too. Moves to the trunk are not allowed.

The top card on the waste pile can be moved to foundation piles subject to the constraints on those piles; moves to the wing piles are also allowed once the trunk is empty. Moves to the trunk are not allowed. If the stock is empty, the cards in the waste pile can be moved to the stock in a **redeal**, emptying the waste pile. There are at most two redeals per game.

The game ends when no more moves are possible. If all the cards are in foundation piles, the game's a win, otherwise it's a loss. Eagle's Wing requires more recognition than skill, and it should be easy to win 80% or more of the games played.

impl
Class EaglesWing

```
java.lang.Object
  └ playerAPI.EaglesWing
      └ impl.EaglesWing
```

```
public class EaglesWingextends EaglesWing
```

Constructor Summary

EaglesWing(long seed) Construct a new instance of an Eagle's Wing game.	

Method Summary

Card	**bottomCard**(CardPiles pile) Determine a card pile's bottom card.
Card[]	**cardsInPile**(CardPiles pile) Get the cards in a pile.
boolean	**gameWon**() Determine if this game's been won.
EaglesWingStatus	**move**(CardPiles from, CardPiles to) Move cards between two piles in the this game.
boolean	**movesPossible**() Determine if there are moves possible in this game's current state.
int	**redealsLeft**() Determine the number of redeals (waste-to-stock moves) left in this game.
int	**size**(CardPiles pile) Determine the number of cards in a card pile.
Card	**topCard**(CardPiles pile) Determine a card pile's top (face-up) card.

Methods inherited from class playerAPI.EaglesWing
`newGame`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

EaglesWing

`public` **`EaglesWing`**`(long seed)`

Construct a new instance of an Eagle's Wing game. The cards have been dealt but no moves have been made.

Parameters:
`seed` - A random-number-generator seed used when creating a shuffed deck.

Method Detail

bottomCard

`public Card` **`bottomCard`**`(CardPiles pile)`

Determine a card pile's bottom card.

Specified by:
`bottomCard` in class `EaglesWing`
Parameters:
`pile` - The card pile of interest.
Returns:
The bottom (base) card in the given pile or null if the given pile's empty.

---

cardsInPile

```
public Card[] cardsInPile(CardPiles pile)
```

Get the cards in a pile.

Specified by:
`cardsInPile` in class `EaglesWing`
Parameters:
`pile` - The card pile of interest.
Returns:
an array of cards from the pile; index 0 is the bottom card. The array is sized to the number of cards actually in the pile.

---

gameWon

```
public boolean gameWon()
```

Determine if this game's been won.

Specified by:
`gameWon` in class `EaglesWing`
Returns:
True iff this game's current configuration is a winner.

---

move

```
public EaglesWingStatus move(CardPiles from,
                             CardPiles to)
```

Move cards between two piles in the this game.

Specified by:
`move` in class `EaglesWing`
Parameters:
`from` - The pile from which cards will be moved.
`to` - The pile to which cards will be moved.
Returns:
The move's status.

---

movesPossible

```
public boolean movesPossible()
```

Determine if there are moves possible in this game's current state.

Specified by:
`movesPossible` in class `EaglesWing`
Returns:
True iff there are moves possible in this game's current state.

---

redealsLeft

```
public int redealsLeft()
```

Determine the number of redeals (waste-to-stock moves) left in this game.

Specified by:
`redealsLeft` in class `EaglesWing`
Returns:
The number of redeals left in this game.

---

size

```
public int size(CardPiles pile)
```

Determine the number of cards in a card pile.

Specified by:
`size` in class `EaglesWing`
Parameters:
`pile` - The card pile of interest.
Returns:
The number of cards in the given pile.

---

topCard

```
public Card topCard(CardPiles pile)
```

Determine a card pile's top (face-up) card.

Specified by:
`topCard` in class `EaglesWing`
Parameters:
`pile` - The card pile of interest.
Returns:
The top card in the given pile or null if the given pile's empty.

playerAPI
Class EaglesWing

java.lang.Object
  └ **playerAPI.EaglesWing**

Direct Known Subclasses:
EaglesWing

---

public abstract class **EaglesWing**extends java.lang.Object

Constructor Summary	
EaglesWing()	

Method Summary

abstract Card	**bottomCard**(CardPiles pile) Get a card pile's bottom (base) card.
abstract Card[]	**cardsInPile**(CardPiles pile) Get a pile's cards.
abstract boolean	**gameWon**() Determine if this game's been won.
abstract EaglesWingStatus	**move**(CardPiles from, CardPiles to) Perform a move between two card piles in this game.
abstract boolean	**movesPossible**() Determine if it's possible to make a move in this game's current configuration.
static EaglesWing	**newGame**(long seed) Create a new instance of the Eagle's Wing game.
abstract int	**redealsLeft**() Determine the number of redeals (waste-to-stock moves) left in this game.
abstract int	**size**(CardPiles pile) Determie the number of cards in a card pile.
abstract Card	**topCard**(CardPiles pile) Get a card pile's top-most card.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

EaglesWing

public **EaglesWing**()

Method Detail

bottomCard

```
public abstract Card bottomCard(CardPiles pile)
```

Get a card pile's bottom (base) card.

Parameters:
`pile` - The card pile of interest.
Returns:
The bottom card in the given pile or `null` if the given pile's empty.

---

cardsInPile

```
public abstract Card[] cardsInPile(CardPiles pile)
```

Get a pile's cards.

Parameters:
`pile` - The card pile of interest.
Returns:
The cards in the pile, bottom card at index 0. The array is sized to the number of cards actually in the pile, not the number of cards that could be in the pile.

---

gameWon

```
public abstract boolean gameWon()
```

Determine if this game's been won.

Returns:
True iff this game's current configuration is a winning game.

---

move

```
public abstract EaglesWingStatus move(CardPiles from,
                                      CardPiles to)
```

Perform a move between two card piles in this game. The move to be made depends on the type of the piles involved in the move.

Parameters:
`from` - The card pile from which the move is being made.
`to` - The card pile to which the move is being made.
Returns:
The move's status.

movesPossible

```
public abstract boolean movesPossible()
```

Determine if it's possible to make a move in this game's current configuration.

Returns:
True iff it's possible to make a move in this game's current configuration.

---

newGame

```
public static EaglesWing newGame(long seed)
```

Create a new instance of the Eagle's Wing game. The cards have been dealt appropriately for the start of a game, but no moves have been made.

Parameters:
`seed` - A random-number-generator seed used when creating a shuffled deck.
Returns:
A new instance of the Eagle's Wing game.

---

redealsLeft

```
public abstract int redealsLeft()
```

Determine the number of redeals (waste-to-stock moves) left in this game.

Returns:
The number of redeals left in this game.

---

size

```
public abstract int size(CardPiles pile)
```

Determie the number of cards in a card pile.

Parameters:
`pile` - The card pile of interest.
Returns:
The number of cards in the given card pile.

---

topCard

```
public abstract Card topCard(CardPiles pile)
```

Get a card pile's top-most card.

Parameters:

`pile` - A card pile of interest.

Returns:

The top-most card in the given pile or `null` if the given pile's empty.

playerAPI
Enum EaglesWingStatus

```
java.lang.Object
  └ java.lang.Enum<EaglesWingStatus>
      └ playerAPI.EaglesWingStatus
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable<EaglesWingStatus>

```
public enum EaglesWingStatusextends java.lang.Enum<EaglesWingStatus>
```

The results of move commands in an Eagle's Wing game. All status values but `MOVE_SUCCESSFUL` indicate the move failed with no cards moved.

Enum Constant Summary

DECREASING_RANK Move failed because cards in the to pile should be in strictly increasing rank.
FOUNDATION_BASE_MISMATCH Move failed because the first card moved to an empty foundation pile should match the rank of the foundation card.
FROM_PILE_EMPTY Move failed because the from pile is empty.
INCREASING_RANK Move failed because cards in the to pile should be in strictly decreasing rank.
INVALID_MOVE Move failed because the move isn't allowed under any circumstances (for example, from wing pile to waste pile).
MISMATCHED_SUIT Move failed because cards in to pile must match in suit.
MOVE_SUCCESSFUL The move succeeded.
NO_REDEALS_LEFT The move failed because each Eagle's Wing game allows at most two redeals.
STOCK_NOT_EMPTY The move failed because the stock must be empty before a redeal can occur.
TRUNK_NOT_EMPTY The move failed because moves to the wing pile from other non-wing piles can occur only after the trunk is empty.
WING_PILE_FULL The move failed because the to wing pile would contain more than three cards.

Method Summary

static EaglesWingStatus	**valueOf**(java.lang.String name) Returns the enum constant of this type with the specified name.
static EaglesWingStatus[]	**values**() Returns an array containing the constants of this enum type, in the order they are declared.

Methods inherited from class java.lang.Enum
clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object
getClass, notify, notifyAll, wait, wait, wait

Enum Constant Detail

DECREASING_RANK

```
public static final EaglesWingStatus DECREASING_RANK
```

Move failed because cards in the to pile should be in strictly increasing rank. This error applies to moves to a wing pile.

FROM_PILE_EMPTY

```
public static final EaglesWingStatus FROM_PILE_EMPTY
```

Move failed because the from pile is empty.

FOUNDATION_BASE_MISMATCH

```
public static final EaglesWingStatus FOUNDATION_BASE_MISMATCH
```

Move failed because the first card moved to an empty foundation pile should match the rank of the foundation card.

INCREASING_RANK

```
public static final EaglesWingStatus INCREASING_RANK
```

Move failed because cards in the to pile should be in strictly decreasing rank. This error appies to moves to a foundation pile.

INVALID_MOVE

```
public static final EaglesWingStatus INVALID_MOVE
```

Move failed because the move isn't allowed under any circumstances (for example, from wing pile to waste pile).

MISMATCHED_SUIT

```
public static final EaglesWingStatus MISMATCHED_SUIT
```

Move failed because cards in to pile must match in suit.

MOVE_SUCCESSFUL

```
public static final EaglesWingStatus MOVE_SUCCESSFUL
```

The move succeeded.

NO_REDEALS_LEFT

```
public static final EaglesWingStatus NO_REDEALS_LEFT
```

The move failed because each Eagle's Wing game allows at most two redeals.

STOCK_NOT_EMPTY

```
public static final EaglesWingStatus STOCK_NOT_EMPTY
```

The move failed because the stock must be empty before a redeal can occur.

TRUNK_NOT_EMPTY

```
public static final EaglesWingStatus TRUNK_NOT_EMPTY
```

The move failed because moves to the wing pile from other non-wing piles can occur only after the trunk is empty.

WING_PILE_FULL

```
public static final EaglesWingStatus WING_PILE_FULL
```

The move failed because the to wing pile would contain more than three cards.

Method Detail

values

```
public static EaglesWingStatus[] values()
```

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```
for (EaglesWingStatus c : EaglesWingStatus.values())
    System.out.println(c);
```

Returns:
an array containing the constants of this enum type, in the order they are declared

---

valueOf

```
public static EaglesWingStatus valueOf(java.lang.String name)
```

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:
`name` - the name of the enum constant to be returned.
Returns:
the enum constant with the specified name
Throws:
`java.lang.IllegalArgumentException` - if this enum type has no constant with the specified name
`java.lang.NullPointerException` - if the argument is null

playerAPI
Enum CardPiles

```
java.lang.Object
  └ java.lang.Enum<CardPiles>
      └ playerAPI.CardPiles
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable<CardPiles>

```
public enum CardPilesextends java.lang.Enum<CardPiles>
```

The card piles used in an Eagle's Wing game. No order among piles is impled by the names of the foundation and wing piles.

Nested Class Summary

static class	**CardPiles.Group** A particular card pile is part of one of five groups of card piles.
static interface	**CardPiles.PairFun** The method to call when iterating over card-pile pairs.

Enum Constant Summary

foundation0 The foundation-pile names are given in no particular order.	
foundation1	
foundation2	
foundation3	
stock	
trunk	
waste	
wing0 The wing-pile names are given in no particular order.	
wing1	
wing2	
wing3	
wing4	
wing5	
wing6	
wing7	

Method Summary

static void	**pairIterate**(CardPiles.PairFun pairFun) Iterate through all pairs of card piles, calling a method on each pair.
static CardPiles.Group	**pileGroup**(CardPiles pile) Find a card pile's group.
static CardPiles	**valueOf**(java.lang.String name) Returns the enum constant of this type with the specified name.
static CardPiles[]	**values**() Returns an array containing the constants of this enum type, in the order they are declared.

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

Enum Constant Detail

foundation0

public static final CardPiles **foundation0**

The foundation-pile names are given in no particular order. At the start of a game, foundation0 contains the foundation-base card.

---

foundation1

public static final CardPiles **foundation1**

---

foundation2

```
public static final CardPiles foundation2
```

---

foundation3

```
public static final CardPiles foundation3
```

---

stock

```
public static final CardPiles stock
```

---

trunk

```
public static final CardPiles trunk
```

---

waste

```
public static final CardPiles waste
```

---

wing0

```
public static final CardPiles wing0
```

The wing-pile names are given in no particular order.

---

wing1

```
public static final CardPiles wing1
```

---

wing2

```
public static final CardPiles wing2
```

---

wing3

```
public static final CardPiles wing3
```

---

wing4

```
public static final CardPiles wing4
```

---

wing5

```
public static final CardPiles wing5
```

---

wing6

```
public static final CardPiles wing6
```

---

wing7

```
public static final CardPiles wing7
```

Method Detail

values

```
public static CardPiles[] values()
```

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```
for (CardPiles c : CardPiles.values())
    System.out.println(c);
```

Returns:
an array containing the constants of this enum type, in the order they are declared

---

valueOf

```
public static CardPiles valueOf(java.lang.String name)
```

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:
`name` - the name of the enum constant to be returned.
Returns:
the enum constant with the specified name

Throws:

`java.lang.IllegalArgumentException` - if this enum type has no constant with the specified name

`java.lang.NullPointerException` - if the argument is null

---

pairIterate

```
public static void pairIterate(CardPiles.PairFun pairFun)
```

Iterate through all pairs of card piles, calling a method on each pair. The pairs are generated in no particular order, but all pairs are generated if the iteration is not interrupted. Some card-pile pairs, such as `(foundation0, foundation0)`, may not make sense with respect to Eagle's Wing game play.

Parameters:

`pairFun` - A class instance containing the method to call for each pair.

---

pileGroup

```
public static CardPiles.Group pileGroup(CardPiles pile)
```

Find a card pile's group.

Parameters:

`pile` - The card pile of interest.

Returns:

The given card pile's group.

playerAPI
Enum CardPiles.Group

```
java.lang.Object
  └ java.lang.Enum<CardPiles.Group>
      └ playerAPI.CardPiles.Group
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable<CardPiles.Group>

Enclosing class:
CardPiles

```
public static enum CardPiles.Groupextends java.lang.Enum<CardPiles.Group>
```

A particular card pile is part of one of five groups of card piles. Three of the five card-pile groups are singletons and aren't too useful, but it is useful to be able to refer to a wing or foundation pile without caring about the particular pile involved.

Enum Constant Summary

foundationPile Any of the four piles in the foundation.	
stockPile	
trunkPile	
wastePile	
wingPile Any of the eight wing piles.	

Method Summary

static CardPiles.Group	**valueOf**(java.lang.String name) Returns the enum constant of this type with the specified name.
static CardPiles.Group[]	**values**() Returns an array containing the constants of this enum type, in the order they are declared.

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

Enum Constant Detail

foundationPile

public static final CardPiles.Group **foundationPile**

Any of the four piles in the foundation.

---

stockPile

public static final CardPiles.Group **stockPile**

---

trunkPile

public static final CardPiles.Group **trunkPile**

---

wastePile

```
public static final CardPiles.Group wastePile
```

---

wingPile

```
public static final CardPiles.Group wingPile
```

Any of the eight wing piles.

Method Detail

values

```
public static CardPiles.Group[] values()
```

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```
for (CardPiles.Group c : CardPiles.Group.values())
    System.out.println(c);
```

Returns:
an array containing the constants of this enum type, in the order they are declared

---

valueOf

```
public static CardPiles.Group valueOf(java.lang.String name)
```

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:
`name` - the name of the enum constant to be returned.
Returns:
the enum constant with the specified name
Throws:
`java.lang.IllegalArgumentException` - if this enum type has no constant with the specified name
`java.lang.NullPointerException` - if the argument is null

playerAPI
Interface CardPiles.PairFun

Enclosing class:
CardPiles

```
public static interface CardPiles.PairFun
```

The method to call when iterating over card-pile pairs.

Method Summary

int	**f**(CardPiles pile1, CardPiles pile2) Perform some operation on the given pair of card piles.

Method Detail

f

```
int f(CardPiles pile1,
      CardPiles pile2)
```

Perform some operation on the given pair of card piles.

Parameters:
`pile1` - A card pile.
`pile2` - Another card pile.

Returns:
The return value controls the iteration:

- `0`: continue the iteration with the next pair.
- `1`: break the inner iteration (generating the second piles in the pair) and continue with the next outer iteration.
- `2`: break the outer iteration (generating the first piles in the pair), ending the pair iteration.

Any other return values cause undefined results.

playerAPI
Interface Card

`public interface` **`Card`**

The card behaivor used by Eagle's Wing. Given two cards,

1. Do the suits match? and
2. How do the ranks differ in which direction?

Method Summary

`boolean`	**`matchingSuit`**`(Card c)` Determine if this card has the same suit as the given card.
`int`	**`smallestRankIncrement`**`(Card c)` Find the smallest difference (in absolute value) in rank between this card and the given card.

Method Detail

matchingSuit

`boolean` **`matchingSuit`**`(Card c)`

Determine if this card has the same suit as the given card.

Parameters:
`c` - The given card.
Returns:
True iff the given card's suit matches this card's suit.

smallestRankIncrement

`int` **`smallestRankIncrement`**`(Card c)`

Find the smallest difference (in absolute value) in rank between this card and the given card.

Parameters:

`c` - The given card.

Returns:

The smallest (in absolute value) increment that takes this card's rank to the given card's rank.

Constant Field Values

Contents