UNIVERSIDADE DE ÉVORA

Departamento de Informática

licenciatura em Engenharia Informática

- SOLITÁRIO-

(Cadeira de Estruturas de Dados e Algoritmos I)

2ºAno

Daniel Gonçalo Jesus Ramos, 29423

Évora, 2012

Introdução

Para este trabalho prático, tem-se como objectivo implementar uma versão não gráfica do jogo solitário.

O jogo está dividido em três zonas em que na zona A é composta por duas pilhas, a pilha das cartas viradas para baixo e a pilha das cartas viradas para cima. Apenas é permitido virar cartas e retirar uma carta do baralho das cartas viradas para cima. A carta retirada da zona A pode ser movida para a zona B ou para a zona B caso a jogada seja permitida.

Para implementar as pilhas da zona A foram usadas Stacks e implementados os respectivos métodos para que seja possível mover uma carta da respectiva zona.

A zona B é composta por quatro pilhas em que cada pilha só poderá conter cartas do mesmo naipe e na ordem correcta, sendo o ÁS a primeira e o REI a ultima carta a ser posta. O jogo só estará completo quando todas as cartas tiveram nas quatro pilhas da zona B.

Para implementar as pilhas da zona B foram usadas Stacks tal como na zona A e implementados os respectivos métodos para que seja possível mover cartas da zona A e da Zona C para a Zona B.

A zona C contem 7 pilhas onde existe pilhas com cartas voltadas para cima e com cartas voltadas para baixo. O jogador deverá tentar retirar cartas da zona A e move-las para a Zona C de modo a tentar virar as cartas que estão voltadas para baixo. Apenas é permitido junção de cartas se o naipe da carta a mover for de cor diferente do naipe da ultima carta que esta na pilha de destino e a carta a ser movida for uma carta inferior a que está na pilha. A ordem neste caso começerá pelo REI e acabará no ÁS(o oposto das cartas da zona B).

Para implementar as pilhas da zona C foram usadas Listas duplamente ligadas para facilitar a remoção de cartas e as várias verificações que são feitas quando se tenta mover uma carta de uma zona para outra ou para pilhas diferentes na mesma zona.

O programa apresenta tratamento de excepções quer para o jogo em si quer para as estruturas implementadas(stacks e DoubleLinkedLists).

Estrutura do programa

Classes implementadas

Classe Baralho

Modificador e tipo	Metodo e descrição
Carta	get() Retorna uma carta aleatoria do baralho.
StackArray<Carta>	getPilhaCartas(int num) Retorna uma pilha de n cartas.
java.lang.String	toString()
boolean	verifica(Carta c) verifica se o baralho contem a carta c.

Classe Carta

Modificador e tipo	Metodo e descrição
Boolean	equals(java.lang.Object x)
boolean	isVoltadaParaCima()
Carta.Naipe	naipe()
void	setVoltadaParaCima(boolean voltadaParaCima)
java.lang.String	toString()
Carta.Valor	valor()

Classe GameException

Implementa a excepção GameException que é usado no jogo inteiro.

Classe Solitaire

Modificador e tipo	Metodo e descrição
boolean	canMoveFromAToB(int pilhaIndex) Verifica se pode mover uma carta da zona A para uma pilha especifica da Zona B.
boolean	canMoveFromAToC(int pilhaIndex) Verifica se pode mover uma carta da zona A para uma pilha especifica da zona C.
boolean	canMoveFromBToC(int indexSource, int IndexDest) Verifica se pode mover uma carta de uma pilha especifica da zona B para uma pilha especifica da zona C.
boolean	canMoveFromCToB(int indexSource, int indexDest) Verifica se pode mover uma carta de uma pilha especifica da zona C para uma pilha especifica da zona B.
boolean	canMoveFromCToC(int indexCarta, int indexSource, int indexDest) Verifica se pode mover uma ou um conjunto de cartas de uma pilha para outra pilha na zona C.
void	moveAB(int pilhaIndex) Move uma carta da zona A para uma pilha da zona B.
void	moveAC(int pilhaIndex) Move uma carta da zona A para uma pilha da zona C.
void	moveBC(int indexSource, int indexDest) Move uma carta de uma pilha da zona B para uma pilha da zona C.
void	moveCB(int indexSource, int indexDest) Move uma carta de uma pilha da zona C para uma pilha da zona B.
void	moveCC(int indexCarta, int indexSource, int indexDest) Move uma ou um conjunto de cartas de uma pilha da zona C para outra pilha da zona C.
Carta	retiraZonaA() Retira uma carta da zona A.
Carta	retiraZonaB(int index) Retira uma carta de uma pilha da zona B.
Carta	retiraZonaC(int index) Retira uma carta de uma pilha da zona C.
DoubleLinkedList<Carta>	retiraZonaC(int IndexCarta, int index) Retira um conjunto de cartas da zona C de uma pilha.
void	testarZonaB()
void	testarZonaC()
java.lang.String	toString()
void	virar() Vira uma carta da zona A.

Classe DoubleLinkedList

Modificador e tipo	Metodo e descrição
void	add(int index, T e) Adiciona um elemento a um indice especifico.
void	add(T element) Adiciona um elemento.
void	clear() Faz um reset a lista.
boolean	contains(T e) Verifica se o elemento existe na lista.
T	get(int index) Obtem o elemento de um indice especifico caso exista.
T	getFirst() Obtem o primeiro elemento se existir.
T	getLast() Obtem o ultimo elemento se existir.
DoubleNode<T>	getTail() Retorna a cauda da lista.
java.util.Iterator	iterator() Retorna o iterator da lista.
T	remove(int index) Remove o elemento que esta num determinado indice e retorna-o.
boolean	remove(T element) Remove um elemento se existir.
T	removeLast()
void	removeNode(DoubleNode<T> node) Remove um node, desligando-o do node anterior e do node seguinte.
void	set(int index, T e) Substitui o elemento num indice especifico caso este exista.
int	size() Retorna o tamanho actual da lista.
java.lang.String	toString()

Classe StackArray

Modificador e tipo	Metodo e descrição
int	getSize()
boolean	isEmpty()
T	pop()
void	push(T e)
T	top()

Classe DoubleLinkedListIterator

Modificador e tipo	Metodo e descrição
boolean	hasNext() Retorna verdadeiro de ouver um elemento seguinte.
T	next() Retorna o elemento actual.
void	remove() Remove o elemento actual.

Classe DoubleNode

Modificador e tipo	Metodo e descrição
T	getElement() Retorna o elemento.
DoubleNode<T>	getNext() Retorna o node seguinte.
DoubleNode<T>	getPrev() Obtem o node anterior.
void	setElement(T element) Atribui o elemento.
void	setNext(DoubleNode<T> next) Atribui o node seguinte.
void	setPrev(DoubleNode<T> prev) Atribui o node anterior.

Classe IndexOutOfBoundsException

Implementa a excepção IndexOutOfBounds que é usado no jogo e nas TADs.

Classe NoSuchElementException

Implementa a excepção NoSuchElementException que é usado no jogo e nas TADs.

Classe StackIsEmptyException

Implementa a excepção StackIsEmptyException que é usado no jogo e nas TADs.

Classe StackIsFullException

Implementa a excepção StackIsFullException que é usado no jogo e nas TADs.

Classe ZonaA

Modificador e tipo	Metodo e descrição
Carta	getCartaVirada() Retorna a carta que esta no topo da pilha das cartas viradas.
Carta	retirar() Retira uma carta do topo da pilha das cartas viradas para cima
java.lang.String	toString()
void	virar() Vira uma carta e adiciona-a na pilha das cartas viradas.

Classe ZonaB

Modificador e tipo	Metodo e descrição
boolean	canPut(Carta x, int pilhaIndex) Verifica se pode por uma carta numa pilha especifica.
Carta	get(int index) Obtem uma referencia da carta que estÃ¡ no topo de uma pilha na ZonaB caso esta exista.
void	put(Carta x) Tenta por automaticamente a carta desejada numa das pilhas disponiveis
void	put(Carta x, int i) Adiciona uma carta a uma pilha especifica da zona B.
Carta	retirar(int index) Retira uma carta de uma pilha da zona B.
java.lang.String	toString()

Classe ZonaC

Modificador e tipo	Metodo e descrição
void	add(Carta x, int index) Adiciona uma carta a uma pilha especifica.
void	add(DoubleLinkedList<Carta> s, int index) Adiciona uma sequencia de cartas a uma pilha especifica.
boolean	canPut(Carta x, int index) Verifica se pode adicionar uma carta a uma pilha especifica.
boolean	canPut(DoubleLinkedList<Carta> s, int index) Verifica se pode adicionar uma sequencia de cartas a uma pilha.
Carta	getCarta(int index) Obtem a carta que estÃ¡ no topo de uma pilha especifica.
DoubleLinkedList<Carta>	getListSnapshot(int indexCarta, int index) Obtem um conjunto de cartas de uma pilha especifica.
Carta	retira(int index) Retita uma carta de uma pilha especifica
DoubleLinkedList<Carta>	retira(int indexCarta, int index) Retira um conjunto de cartas de uma pilha especifica.
java.lang.String	toString()

Interfaces do programa

Interface linkedList

Modificador e tipo	Metodo e descrição
void	add(int index, T e)
void	add(T e)
void	clear()
T	get(int index)
T	remove(int index)
boolean	remove(T e)
void	set(int index, T e)
int	size()

Interface Stack

Modificador e tipo	Metodo e descrição
`int`	`getSize()`
`boolean`	`isEmpty()`
`T`	`pop()`
`void`	`push(T e)`
`T`	`top()`

Conclusão

O trabalho não apresentou grandes dificuldades em termos de execução. A única dificuldade em si foi a implementação das várias TADS e os métodos que seriam precisos para que a movimentação de cartas fosse possível.