Ejercicios de Colecciones

Programación en Java para la asignatura Algoritmos y estructuras de datos (AyEdD)

Gorka Guardiola Múzquiz

1 de Febrero de 2022

Ejercicio de colecciones.

Ejercicio 5

CardPile El ejercicio implementará un montón de cartas. Para ello es necesario programar tres ADTs, uno de las cuales delegará en el otro. El primero es una carta, `Card`.

	`Card(int value, char suit)`	*Construye una carta. El value será del 1 al 10, el suit de la A a la D.*
`public boolean`	`isValid()`	*Dice si una carta es válida.*
`public String`	`toString()`	*Devuelve representación como string:* `"1 D"`
`public boolean`	`equals(Object other)`	*Dice si dos cartas son iguales (si su valor y palo son iguales).*

Es segundo es una pila de cartas, que se llamará `CardStack`. Este ADT debes implementarlo totalmente de cero, mediante una lista enlazada sin utilizar colecciones de java y tiene que tener el API descrito a continuación.

	`CardStack()`	*construye una pila vacía*
`public void`	`push(Card card)`	*añade un elemento*
`public Card`	`pop()`	*extrae y devuelve la carta* ***más recientemente añadida****,* `null` *si está vacía*
`public int`	`length()`	*devuelve el número de elementos en la pila*
`public String`	`toString()`	*devuelve representación como string de las cartas de la pila en el orden de la pila*

Un ejemplo de lo que devuelve `toString` podría ser la string: `"[6 B, 2 C, 3 D]"`

El tercero es un montón de cartas, que se llamará `CardPile`. En cada momento, sólo puede contener pilas (`CardStack`) y un número constante de cartas en su implementación, incluyendo las variables locales de los métodos.

	`CardPile()`	*Construye un montón vacío*
`public void`	`push(Card card)`	*añade un elemento*
`public Card`	`pop()`	*extrae y devuelve la carta* ***más recientemente añadida****,* `null` *si está vacía*
`public void`	`cut(int n)`	*corta el montón por la carta n-ésima. Si hay menos de n (o igual), o si n es menor o igual que cero deja el montón sin cambios.*
`public void`	`invert()`	*invierte el orden de las cartas en la pila*
`public void`	`mix(int n, int seed)`	*Corta y mezcla la pila de cartas n veces de forma pseudoaleatoria utilizando la semilla. La mezcla se hace uno sí, uno no de forma alterna.*
`public String`	`toString()`	*devuelve representación como string de las cartas de la pila en orden de la pila*

Un ejemplo de lo que devuelve `toString` podría ser la string:
`"[6 B, 2 C, 3 D]"`

Un ejemplo de corte de cartas (`cut`), para la baraja `"[1 B, 2 C, 3 D, 4 A, 5 B, 6 C]"` cortada por 3 sería- `"[4 A, 5 B, 6 C, 1 B, 2 C, 3 D]"`

Una mezcla de cartas sería, para la baraja `"[1 B, 2 C, 3 D, 4 A, 5 B, 6 C]"` cortada por el 2, resultaría en `"[1 B, 3 D, 2 C, 4 A, 5 B, 6 C]"`.
El método `mix` corta la baraja `n` veces por m_i (con m_i pseudoaleatorio y diferente cada vez) y para cada corte realiza una mezcla.

La clase `CardPile` debe tener un programa principal que reciba como único parámetro obligatorio un fichero con operaciones como el que sigue:

```
drop 1 A
drop 1 C
drop 2 D
drop 1 B
drop 1 B
drop 10 B
take
take
mix 4 1
print
drop 2 A
drop 1 C
print
take 1 B
```

Las operaciónes se interpretarán de la siguiente manera:

`drop n a`	*Añade la carta de valor `n` y palo `a` al montón*
`take`	*Saca la carta de arriba del montón*
`mix n m`	*Mezcla las cartas, cortando `n` veces con semilla `m`*
`print`	*Imprime la pila por la salida estándar*

Entrega:

Recuerda que hay que entregar mediante el sistema de entregas y en el aula virtual. Si no está en ambos, no se considera entregado. Recuerda ejecutar `rprueba.sh` periódicamente y para realizar la entrega final y `prueba.sh` para probar tu práctica. Esta práctica la desarrollarás sin tests y más adelante te daremos tests. Recuerda también compilar con el parámetro `-encoding utf-8`.

Pautas de implementación:

Un ejemplo para la generación de números pseudoaleatorios en java:

```
import edu.princeton.cs.algs4.*;
import java.util.Random;

class PruebaGenerarNums
{
    public static void main(String[] args)
    {
      int i;
      Random rnum = new Random();
      rnum.setSeed(20);
      i = 1+rnum.nextInt(5); //deja en i un número entre 1 y 4
      StdOut.println("Num: " + i);
      i = 1+rnum.nextInt(5);
      StdOut.println("Num: " + i);
    }
}
```