PROGRAMACIÓ AVANÇADA
TRIMESTRE 1. CURS 2022/23
PRÀCTICA 4

****

Objectiu 1: Implementació de l'estructura de dades Arbre de Cerca Binària (Acb). Generecitat usant la notació <E>

Objectiu 2: Implementació de mètodes aplicant la tècnica del Divideix i Venç

Objectiu 3: Col·leccions de Java: Cua

Durada: Una sessió

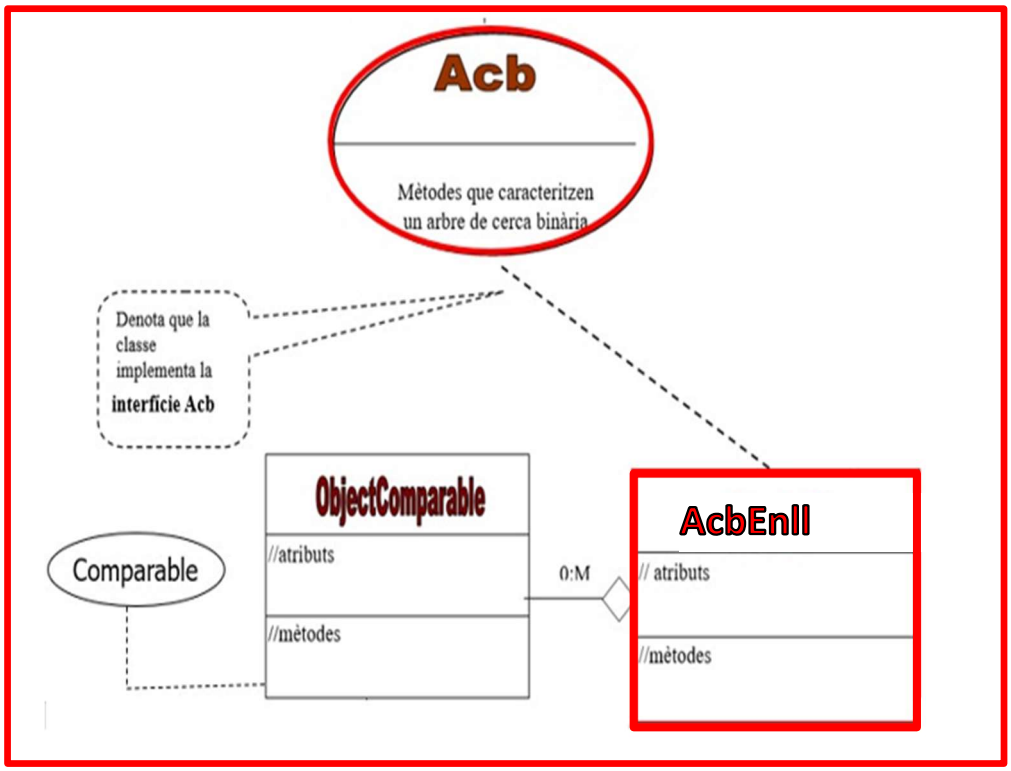
Lliurament: Llistat imprès dels fonts i penjar el projecte al Moodle

Data Lliurament: A l'inici de la pràctica 5

Enunciat

Es vol implementar una estructura de dades que permeti obtenir la informació que emmagatzema de forma ordenada, ascendentment o descendentment segons vulgui l'usuari.

Un arbre de cerca binària emmagatzema les dades ordenadament, per tant l'objectiu és implementar aquesta l'estructura de dades (ACB). Usareu la tècnica enllaçada per representar un arbre, seguint exactament la mateixa representació que la implementada a classe de teoria però no es farà herència, es muntarà directament la classe demanada.



Recordeu que per obtenir les dades que emmagatzema un arbre de cerca binària de forma ordenada s'ha d'aplicar un recorregut en inordre. L'ordenació de les dades obtingudes és ascendent.
Es vol que la nostra estructura sigui **recorrible**, obtenint ordenadament, ascendentment o descendentment segons indiqui l'usuari, **un a un** els diferents elements que emmagatzema l'estructura.
Concretament es vol que l'usuari pugui obtenir les dades ordenades seguint la següent programació:

```
Acb<E> arbre;
arbre = new AcbEnll();
Comparable<E> c;
arbre.inserir(...);
...
arbre.iniRecorregut(true);
while (!arbre.finalRecorregut()) {
      c = arbre.segRecorregut();
      // fer el que sigui amb c

}
```

Caldrà indicar si es volen obtenir les dades ordenades en un sentit u altre

No es vol un mètode que retorni TOT el recorregut de cop, es volen mètodes que permetin obtenir els elements un a un.

PART 1. Implementació de l'estructura de dades: Arbre de Cerca Binària

Cal definir la interfície i implementar-la en una classe AcbEnll usant la tècnica enllaçada. Les operacions que ubicareu dins d'aquesta interfície són les que caracteritzen l'estructura. Aquesta col·lecció serà genèrica usant la notació <E>.

Implementacions:

Interfície: Acb<E extends Comparable<E>>

És una classe derivada de Exception

```
public E arrel() throws ArbreException;

//Cal llençar una excepció si es demana l'arrel d'un arbre buit
public Acb<E> fillEsquerre();

// No llença excepcions. Si no té fill esquerre retorna un arbre buit
public Acb<E> fillDret();

// No llença excepcions. Si no té fill dret retorna una arbre buit
public boolean abBuit();

public void buidar();

public void inserir(E e) throws ArbreException;
// Llença una excepció si l'element que s'insereix està repetit

public void esborrar(E e) throws ArbreException;
// Llança una excepció si l'element no hi és i en conseqüència no es
// realitza la eliminació

public boolean membre(E e);
// Retorna true si l'arbre conté un element com el donat com a
// paràmetre
```

Classe AcbEnll<E extends Comparable<E>>

En la implementació d'aquesta classe heu d'usar el **mètode compareTo** que obligatòriament tindran implementat els objectes E continguts a l'arbre (`Comparable` de l'API de Java), i **evitar l'ús del mètode `equals`** per la comparació d'objectes.

La implementació dels mètodes ha de seguir la **versió 2** dels apunts de **teoria però cal adaptar-ho per a usar la interfície Comparable de Java i tenir un paràmetre <E>.**

Classe privada NodeA

```
private class NodeA{
      NodeA esq, dret;
      E inf;
      //mètodes
}
```

Atributs (no es poden afegir atributs)

```
private NodeA arrel; //per referenciar l'arrel de l'arbre
private Queue<E> cua; //Nou atribut!
// Per aconseguir que l'arbre sigui recorrible
```

No podeu afegir atributs

Mètodes (es poden afegir mètodes privats)

```
// Constructor i mètodes de la interfície  que s'ha d'implementar
// Per aconseguir que l'arbre sigui recorrible cal afegir els següents
// mètodes:
```

Obligatòriament s'ha d'implementar la **versió 2 de teoria**

```
public void iniRecorregut (boolean sentit)
/* prepara l'arbre per a ser recorregut en inordre. Després d'invocar
aquest mètode, la invocació del mètode segRecorregut retornarà el
primer element en inordre de l'arbre. Aquest mètode ha d'emplenar l'atribut
cua amb els elements de l'arbre aplicant un recorregut en inordre. Cal tenir
present el paràmetre alhora d'emplenar la cua */
public boolean finalRecorregut( )
/* retorna true si ja s'ha arribat al final del recorregut en inordre de
l'arbre. Això és si:
      - l'arbre és buit
      - la darrera vegada que es va invocar segRecorregut aquest mètode
      ja va retornar el darrer element en inordre de l'arbre.
Tot això és el mateix que dir que retorna true quan no té sentit invocar el
mètode segRecorregut */
public E segRecorregut () throws ArbreException
/*retorna el següent element en inordre, si n'hi ha.
Llença una excepció si:
      - abans d'invocar-lo no s'ha invocat el mètode iniRecorregut
      - la darrera vegada que es va invocar ja va retornar
      el darrer element del recorregut (finalRecorregut retornaria true)
      - s'invoca quan entre la invocació de iniRecorregut i la del mètode
      s'ha produït una modificació de l'arbre, això és, s'ha fet ús del
      mètode inserir, esborrar, buidar*/
```

Important: si la inicialització del recorregut té un paràmetre de true el recorregut a seguir serà l'inordre treballat a classe de teoria, si és false serà un inordre modificat, concretament intercanviant el tractament dels fills esquerra i dret. Si primer es tracta el subarbre esquerra s'obté una ordenació ascendent, descendent en cas contrari. Un recorregut **no es pot iniciar** si prèviament no s'ha invocat el mètode **iniRecorregut**. I si un cop inicialitzat s'invoca algun mètode que modifica el contingut de l'arbre **tampoc**, caldria invocar novament al mètode d'inicialització. Cal que quedi reflectit a la programació!!!!

Perquè us feu una idea de quan el mètode **segRecorregut** llença una excepció, observeu els següents fragments de codi:

```
Acb<E> arbre;
arbre = new AcbEnll();
Comparable<E> c;
arbre.inserir(...);
. . .
c = arbre.segRecorregut();
```

Excepció: no s'ha invocat iniRecorregut

```
Acb<E> arbre;
arbre = new AcbEnll();
Comparable<E> c;
arbre.inserir(...);
...
arbre.iniRecorregut(false);
while (!arbre.finalRecorregut()) {
      c = arbre.segRecorregut();
      // fer el que sigui amb c
}
c = arbre.segRecorregut();
```

Excepció: el recorregut ja ha finalitzat i no s'ha tornat a inicialitzar invocant iniRecorregut

```
Acb<E> arbre;
arbre = new AcbEnll();
Comparable<E> c;
arbre.inserir(...);
...
arbre.iniRecorregut(true);
// suposem que l'arbre té algun element
c = arbre.segRecorregut();
arbre.inserir(...)
c = arbre.segRecorregut();
```

Excepció: l'arbre s'ha modificat abans d'acabar el recorregut

No podeu afegir a la classe AcbEnll ni atributs ni mètodes públics.

PART 2. Afegir mètodes a la classe AcbEnll. La implementació d'aquests s'ha de fer aplicant la tècnica del divideix i venç

Mètodes a afegir:

La implementació dels mètodes s'ha de fer usant la tècnica del divideix i venç. És imprescindible trobar la manera d'expressar la problemàtica de forma recursiva, i evidentment trobar les situacions que aturen la recursivitat.

Recordeu que la tècnica del divideix i venç ha de dividir el problema a resoldre amb problemes del **mateix tipus i de mida més petita**. Si les implementacions no s'ajusten a la tècnica, tot i que la implementació sigui correcta, no s'hi considerarà.

Tots els mètodes que necessitin d'un mètode privat per fer la recursivitat, l'afegiran **obligatòriament** a la classe NodeA i **no pot ser un mètode estàtic**.

- **1.- Redefinició del mètode `clone()`. Aquest mètode arriba via herència, la implementació per defecte és realitzar una copia de referències. El que es demana és que es faci un duplicat. Signatura:**

```
public  Object clone()
```

Adjuntat un document amb informació de com procedir per redefinir el mètode **`clone()`**. No es permet invocar a cap mètode de la classe en la seva implementació. Tampoc la creació de cap objecte mitjançant new

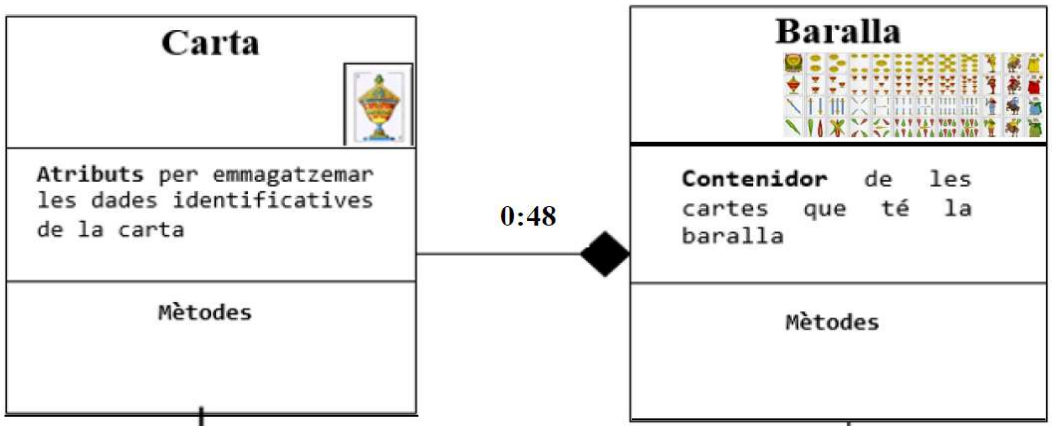
- **2.- Calcula i retorna la cardinalitat, el nombre de nodes que té l'arbre.**

```
public int cardinalitat()
```

PART 3. ÚS DE L'ESTRUCTURA ACB.

Per provar el correcte funcionament de l'arbre es crearà una baralla de cartes i es visualitzarà ordenada usant un ACB recorrible (la nostra col·lecció de la Part 1)

IMPLEMENTEU la classe Carta i la classe Baralla.



La classe Carta

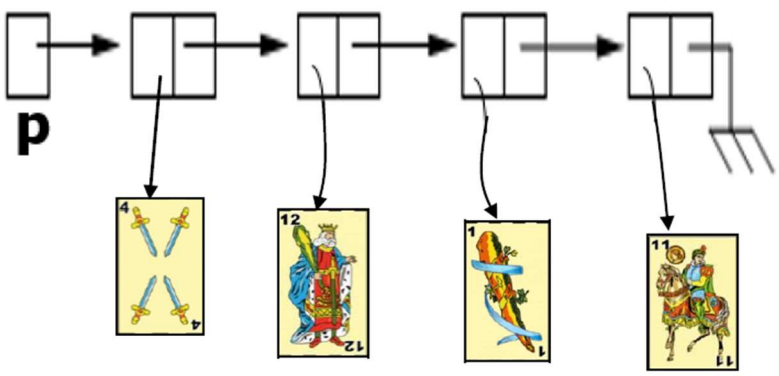
Els objectes d'aquesta classe han de tenir dos atributs privats, un per emmagatzemar la numeració i altre pel pal. Referent als mètodes, a més del constructor afegiu els que considereu més adients. **Imprescindible** que hi hagi la redefinició del mètode `toString`.

La classe Carta **ha d'implementar la interfície Comparable**, les cartes s'han de poder comparar. La relació d'ordre vindrà donada, primer de tot **tenint en compte el pal i a igualtat de pal la relació vindrà donada per la numeració**. Considereu que el pal de COPA < ESPASA < OROS < BASTONS.

La classe Baralla és un contenidor d'objectes de tipus Carta

Heu de determinar vosaltres quins són els atributs i mètodes que calen per representar un objecte Baralla. **Imprescindible** que les cartes que formen la baralla estiguin emmagatzemades en una seqüència enllaçada lineal sense node capçalera. Un node per a cadascuna de les cartes.

El constructor **obligatòriament** ha de crear una baralla sencera desordenada.



En la construcció la seqüència ha de tenir 48 nodes corresponents a les 48 cartes que té la baralla

Sense **cap** criteri d'ordenació

La classe node serà privada dins de la classe Baralla.

Escriu un programa `main` que:

- 1.- Creí una baralla de cartes barrejades (no ordenades) i a continuació les carregi en un Acb invocant al mètode `inserir` (a partir d'aquest moment la baralla ja no s'usarà més).
- 2.- Visualitza **un llistat ordenat** de totes les cartes que conté l'arbre, **en sentit ascendent**.
- 3.- Fes una **copia** de l'arbre creat invocant al mètode **`clone`**, per usar més endavant.
- 4- Repetidament mostra un menú on l'usuari pugui seleccionar cartes que vol eliminar de l'arbre, d'una en una fins que ja no quedin cartes o bé no vulgui eliminar-ne més. Després de cada eliminació s'ha de mostrar novament el llistat ordenat ascendent de cartes que té l'arbre. Per controlar si a l'arbre li queden o no cartes **imprescindible invocar al mètode que calcula la cardinalitat** de l'arbre.

```
Menú Opcions
============
1.- Eliminar carta
2.- Acabar
Tria que vols fer (1 o 2)
```

- 5.- Un cop l'usuari ja no vol o ja no queden cartes, al prémer l'opció Acabar del menú es visualitza l'arbre clonat, visualitzant a pantalla el seu contingut ordenat descendentment. I a continuació cal eliminar cartes del clonat, tal com s'indica en el punt següent.
- 6.- Elimina de l'arbre clonat cartes aleatòries, tantes com el número aleatori que surti (useu la classe Random). Mostreu el contingut de l'arbre després de cada eliminació. Compte, si s'intenta eliminar una carta que no està a l'arbre, no és una eliminació!
- 7.- Finalment indica quin dels dos arbres té més cartes, el clonat o l'arbre inicial del que s'han eliminat tantes cartes com ha volgut l'usuari. **Imprescindible invocar al mètode que calcula la cardinalitat per fer aquest tractament.**

Fes que aquest **`main`** tingui procediments ajudants per resoldre aquelles parts que tenen una funcionalitat clara i/o per aquelles parts repetitives.

Exemple execució

```
S'ha creat la baralla i carregat a l'arbre
1 de COPES  2 de COPES  3 de COPES  4 de COPES  5 de COPES  6 de COPES  7 de
COPES  8 de COPES  9 de COPES  10 de COPES  11 de COPES  12 de COPES  1 de
ESPASES  2 de ESPASES  3 de ESPASES  4 de ESPASES  5 de ESPASES  6 de ESPASES
7 de ESPASES  8 de ESPASES  9 de ESPASES  10 de ESPASES  11 de ESPASES  12 de
ESPASES  1 de OROS  2 de OROS  3 de OROS  4 de OROS  5 de OROS  6 de OROS  7
de OROS  8 de OROS  9 de OROS  10 de OROS  11 de OROS  12 de OROS  1 de
BASTONS  2 de BASTONS  3 de BASTONS  4 de BASTONS  5 de BASTONS  6 de BASTONS
7 de BASTONS  8 de BASTONS  9 de BASTONS  10 de BASTONS  11 de BASTONS  12 de
BASTONS
Menú Opcions
============
1.- Eliminar carta
2.- Acabar
Tria que vols fer (1 o 2)
1
Tria pal de la carta que vols eliminar
1.- Copes
2.- Espases
3.- Oros
4.- Bastons
4
Tria el número de la carta
Indica la numeració de la carta [1,12]
1
S'ha eliminat la carta 1 de BASTONS
1 de COPES  2 de COPES  3 de COPES  4 de COPES  5 de COPES  6 de COPES  7 de
COPES  8 de COPES  9 de COPES  10 de COPES  11 de COPES  12 de COPES  1 de
ESPASES  2 de ESPASES  3 de ESPASES  4 de ESPASES  5 de ESPASES  6 de ESPASES
7 de ESPASES  8 de ESPASES  9 de ESPASES  10 de ESPASES  11 de ESPASES  12 de
ESPASES  1 de OROS  2 de OROS  3 de OROS  4 de OROS  5 de OROS  6 de OROS  7
de OROS  8 de OROS  9 de OROS  10 de OROS  11 de OROS  12 de OROS  2 de
BASTONS  3 de BASTONS  4 de BASTONS  5 de BASTONS  6 de BASTONS  7 de BASTONS
8 de BASTONS  9 de BASTONS  10 de BASTONS  11 de BASTONS  12 de BASTONS
Menú Opcions
============
1.- Eliminar carta
2.- Acabar
Tria que vols fer (1 o 2)
1
Tria pal de la carta que vols eliminar
1.- Copes
2.- Espases
3.- Oros
```

Menú

Visualització ascendent del contingut de l'arbre. Inicialment totes les cartes de la baralla

Sol·licitar dades a l'usuari
Cal controlar la correctesa de les dades entrades, insistir mentre siguin incorrectes

Després de cada eliminació cal mostrar novament el contingut de l'arbre

```
4.- Bastons
1
Tria el número de la carta
Indica la numeració de la carta [1,12]
1
S'ha eliminat la carta 1 de COPES
2 de COPES  3 de COPES  4 de COPES  5 de COPES  6 de COPES  7 de COPES  8 de
COPES  9 de COPES  10 de COPES  11 de COPES  12 de COPES  1 de ESPASES  2 de
ESPASES  3 de ESPASES  4 de ESPASES  5 de ESPASES  6 de ESPASES  7 de ESPASES
8 de ESPASES  9 de ESPASES  10 de ESPASES  11 de ESPASES  12 de ESPASES  1 de
OROS  2 de OROS  3 de OROS  4 de OROS  5 de OROS  6 de OROS  7 de OROS  8 de
OROS  9 de OROS  10 de OROS  11 de OROS  12 de OROS  2 de BASTONS  3 de
BASTONS  4 de BASTONS  5 de BASTONS  6 de BASTONS  7 de BASTONS  8 de BASTONS
9 de BASTONS  10 de BASTONS  11 de BASTONS  12 de BASTONS
Menú Opcions
============
1.- Eliminar carta
2.- Acabar
Tria que vols fer (1 o 2)
1
Tria pal de la carta que vols eliminar
1.- Copes
2.- Espases
3.- Oros
4.- Bastons
4
Tria el número de la carta
Indica la numeració de la carta [1,12]
12
S'ha eliminat la carta 12 de BASTONS
2 de COPES  3 de COPES  4 de COPES  5 de COPES  6 de COPES  7 de COPES  8 de
COPES  9 de COPES  10 de COPES  11 de COPES  12 de COPES  1 de ESPASES  2 de
ESPASES  3 de ESPASES  4 de ESPASES  5 de ESPASES  6 de ESPASES  7 de ESPASES
8 de ESPASES  9 de ESPASES  10 de ESPASES  11 de ESPASES  12 de ESPASES  1 de
OROS  2 de OROS  3 de OROS  4 de OROS  5 de OROS  6 de OROS  7 de OROS  8 de
OROS  9 de OROS  10 de OROS  11 de OROS  12 de OROS  2 de BASTONS  3 de
BASTONS  4 de BASTONS  5 de BASTONS  6 de BASTONS  7 de BASTONS  8 de BASTONS
9 de BASTONS  10 de BASTONS  11 de BASTONS
Menú Opcions
============
1.- Eliminar carta
2.- Acabar
Tria que vols fer (1 o 2)
2
```

L'usuari indica que vol acabar el procés d'eliminació. També ha de finalitzar quan ja no queden cartes a l'arbre.

En aquest moment treballem amb l'arbre clonat. Mostrem en primer lloc el seu contingut en sentit descendent

12 de BASTONS 11 de BASTONS 10 de BASTONS 9 de BASTONS 8 de BASTONS 7 de BASTONS 6 de BASTONS 5 de BASTONS 4 de BASTONS 3 de BASTONS 2 de BASTONS 1 de BASTONS 12 de OROS 11 de OROS 10 de OROS 9 de OROS 8 de OROS 7 de OROS 6 de OROS 5 de OROS 4 de OROS 3 de OROS 2 de OROS 1 de OROS 12 de ESPASES 11 de ESPASES 10 de ESPASES 9 de ESPASES 8 de ESPASES 7 de ESPASES 6 de ESPASES 5 de ESPASES 4 de ESPASES 3 de ESPASES 2 de ESPASES 1 de ESPASES 12 de COPES 11 de COPES 10 de COPES 9 de COPES 8 de COPES

7 de COPES 6 de COPES 5 de COPES 4 de COPES 3 de COPES 2 de COPES 1 de COPES

Ha sortit la carta 6 de BASTONS Carta generada aleatòriament

12 de BASTONS 11 de BASTONS 10 de BASTONS 9 de BASTONS 8 de BASTONS 7 de BASTONS 5 de BASTONS 4 de BASTONS 3 de BASTONS 2 de BASTONS 1 de BASTONS 12 de OROS 11 de OROS 10 de OROS 9 de OROS 8 de OROS 7 de OROS 6 de OROS 5 de OROS 4 de OROS 3 de OROS 2 de OROS 1 de OROS 12 de ESPASES 11 de ESPASES 10 de ESPASES 9 de ESPASES 8 de ESPASES 7 de ESPASES 6 de ESPASES 5 de ESPASES 4 de ESPASES 3 de ESPASES 2 de ESPASES 1 de ESPASES 12 de COPES 11 de COPES 10 de COPES 9 de COPES 8 de COPES 7 de COPES 6 de COPES 5 de COPES 4 de COPES 3 de COPES 2 de COPES 1 de COPES

Ha sortit la carta 5 de ESPASES Carta generada aleatòriament

12 de BASTONS 11 de BASTONS 10 de BASTONS 9 de BASTONS 8 de BASTONS 7 de BASTONS 5 de BASTONS 4 de BASTONS 3 de BASTONS 2 de BASTONS 1 de BASTONS 12 de OROS 11 de OROS 10 de OROS 9 de OROS 8 de OROS 7 de OROS 6 de OROS 5 de OROS 4 de OROS 3 de OROS 2 de OROS 1 de OROS 12 de ESPASES 11 de ESPASES 10 de ESPASES 9 de ESPASES 8 de ESPASES 7 de ESPASES 6 de ESPASES 4 de ESPASES 3 de ESPASES 2 de ESPASES 1 de ESPASES 12 de COPES 11 de COPES 10 de COPES 9 de COPES 8 de COPES 7 de COPES 6 de COPES 5 de COPES 4 de COPES 3 de COPES 2 de COPES 1 de COPES

Ha sortit la carta 4 de BASTONS Carta generada aleatòriament

12 de BASTONS 11 de BASTONS 10 de BASTONS 9 de BASTONS 8 de BASTONS 7 de BASTONS 5 de BASTONS 3 de BASTONS 2 de BASTONS 1 de BASTONS 12 de OROS 11 de OROS 10 de OROS 9 de OROS 8 de OROS 7 de OROS 6 de OROS 5 de OROS 4 de OROS 3 de OROS 2 de OROS 1 de OROS 12 de ESPASES 11 de ESPASES 10 de ESPASES 9 de ESPASES 8 de ESPASES 7 de ESPASES 6 de ESPASES 4 de ESPASES 3 de ESPASES 2 de ESPASES 1 de ESPASES 12 de COPES 11 de COPES 10 de COPES 9 de COPES 8 de COPES 7 de COPES 6 de COPES 5 de COPES 4 de COPES 3 de COPES 2 de COPES 1 de COPES

El procés s'acaba quan s'han eliminat tantes cartes com el número sortit aleatòriament, en l'exemple anterior tres

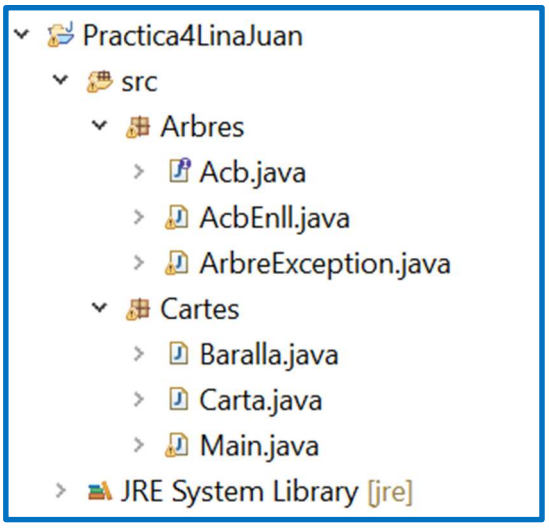
Resultat: Ambdós tenen el mateix nombre de cartes

Organització

Creeu un nou projecte per aquesta pràctica.

Els noms dels vostres projectes, en les pràctiques d'aquesta assignatura han de seguir el següent patró: PràcticaXCognom1Nom1&&Cognom2Nom2. En aquesta pràctica X és 4.

Aquest projecte ha de tenir els següents paquets. I cada paquet les classes indicades en la següent imatge:



Què se us subministra?

Fitxer amb l'enunciat de la pràctica.

Document: Redefinició del mètode **`clone()`.**

Què s'ha de lliurar i com?

S'ha de lliurar la carpeta que conté el **projecte** Eclipse amb el vostre desenvolupament de la pràctica. La carpeta s'ha de lliurar amb tot el seu contingut i comprimida amb ZIP o RAR.

També s'ha de lliurar el **llistat en paper** del codi desenvolupat. El format de lliurament d'aquest codi ha de seguir el patró indicat en la presentació de l'assignatura: amb portada, índex, número de pàgina, tabulació ... En **aquest llistat** cal que indiqueu quina ha estat **la distribució de la feina entre els dos estudiants. És a dir, el grau de participació de cada membre del grup en la realització d'aquesta activitat. Explica les principals dificultats en el seu desenvolupament, valoració del codi lliurat i funcionament de la pràctica.**

On s'ha de lliurar?

El lliurament del projecte es farà a través de la plataforma Moodle i no s'acceptarà cap altra via. Feu atenció a la data i hora límit.

El lliurament en paper es farà directament a la professora a l' inici de la pràctica 5.

Quan s'ha de lliurar?

El lliurament es podrà fer fins el **dia i hora indicats**. Tingueu present que a partir d'aquesta hora el sistema bloquejarà, de manera automàtica, la possibilitat de lliurament.

Tots els grups → 30 de Novembre a les 8:00h

Pautes de correcció

- **PART 1: 6.5 punts**
- **PART 2: 2 punts (1.5 clone())**
- **PART 3: 1.5 punts**

Es valorarà l'eficiència de les implementacions i l'anàlisi descendent aplicat.