Pràctica 4

Objectius:

- **Implementació de l'estructura de dades Arbre de Cerca Binària (Acb). Generecitat usant la notació <E>.**
- **Implementació de mètodes aplicant la tècnica del Divideix i Venç**
- **Utilització de la col·lecció cua de Java: java.util.Queue**

Durada: Dues sessions

Llenguatge: Java utilitzant IntelliJ o Eclipse

Lliurament: penjar el projecte comprimit en format ZIP a l'aula virtual

Data Lliurament: diumenge 17 de novembre.

Enunciat

A la universitat volem oferir una nova beca, els alumnes interessats s'hauran d'inscriure, i el guanyador serà l'alumne amb millor nota mitjana.

A la normativa de la UPF es descriu com es calcula la nota mitjana d'un alumne utilitzant la següent taula d'equivalències:

Qualificació	Equivalència en punts
5,0-6,9 - Aprovat	1
7,0-8,9 - Notable	2
9,0-10 - Excel·lent	3
9,0-10 - Excel·lent amb menció de Matrícula d'Honor	4

Es calcula la nota mitjana a partir de la següent formula, essent P = la puntuació de cada assignatura d'acord amb la taula d'equivalència, NCa = nombre de crèdits que integren l'assignatura, i NCs = nombre total de crèdits superats.

$$Nota\ Mitjana = \frac{\sum P \times NCa}{NCs}$$

Per exemple, anem a calcular la nota mitjana de l'alumne Rosa on el seu expedient és el següent:

Assignatura	Crèdits	Nota	Punts	P x NCa
Fonaments de la Programació	6	7 Notable	2	2x6=12
Programació Orientada a l'objecte	6	5 Aprovat	1	1x6=6
Estructura de Dades i Algorismes	4	9 Excel·lent	3	4x3=12
Programació Avançada	4	5 Aprovat	1	1x4=4
Suma	20		Suma	34

La nota mitjana de Rosa serà $\frac{34}{20} = 1{,}7$

Un segon exemple, amb l'expedient de l'Enric:

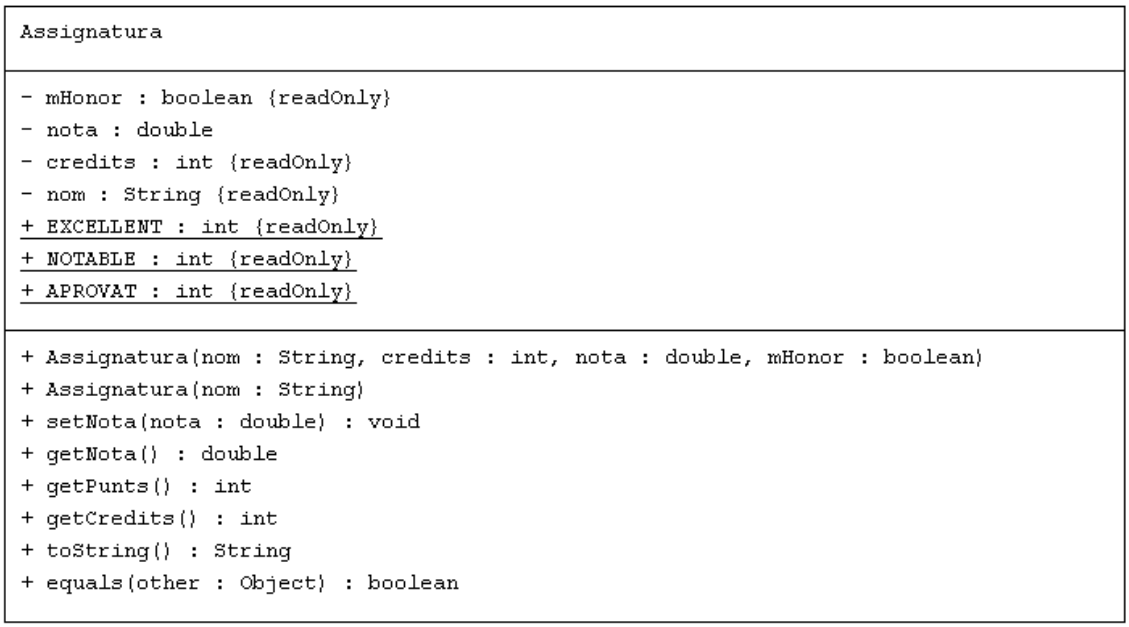
Assignatura	Crèdits	Nota	Punts	P x NCa
Fonaments de la Programació	6	8 Notable	2	2x6=12
Programació Orientada a l'objecte	6	6 Aprovat	1	1x6=6
Estructura de Dades i Algorismes	4	9 Matrícula d'honor	4	4x4=16
Programació Avançada	4	3 Suspès	0	0
Suma	16		Suma	34

La nota mitjana d'Enric serà $\frac{34}{16} = 2{,}125$

Observa que les assignatures suspeses no afecten a la nota mitjana, es calcula només a partir de les assignatures aprovades.

Classe Assignatura

Començarem la pràctica implementant una classe per les assignatures, aquesta classe pertany al package Alumnes:



Atributs de classe:

Els atributs de classe són estàtics i els valors són comuns per tots els objectes.

- Defineix les constants APROVAT (5), NOTABLE (7) i EXCELLENT (9) que són les notes mínimes per obtenir els punts corresponents segons la taula d'equivalència.

Atributs d'instància:

Respecte als atributs d'instància cada objecte té el seu propi valor.

- **nom** (String): Emmagatzema el nom de l'assignatura. És un atribut final i no pot ser modificat després de la inicialització.
- **credits** (int): Indica el nombre de crèdits de l'assignatura. També és final.
- **nota** (double): Guarda la nota mitjana de l'alumne en aquesta assignatura. Pot ser actualitzada.
- **mHonor** (boolean): Indica si l'alumne ha obtingut matrícula d'honor en l'assignatura. L'atribut és final i només pot ser true si la nota és igual o superior a EXCELLENT.

Constructors:

- **Assignatura(String nom, int credits, double nota, boolean mHonor)**: Inicialitza una instància amb el nom, els crèdits, la nota i l'estat de matrícula d'honor. Si els crèdits o la nota són negatius, llença una excepció IllegalArgumentException. A més, si la nota és EXCELLENT o superior i mHonor és true, llavors l'atribut mHonor es fixa a true; en qualsevol altre cas, mHonor es fixa a false.
- **Assignatura(String nom)**: Constructor alternatiu que utilitzarem per inicialitza només l'atribut nom, fixant els altres atributs crèdits, la nota i l'estat de matrícula d'honor amb valors 0, 0.0, i false respectivament.

Mètodes:

- **setNota(double nota)**: Actualitza la nota mitjana de l'alumne, la nota no pot ser inferior a 0, llença una excepció IllegalArgumentException.
- **getNota()**: Retorna la nota mitjana de l'alumne.
- **getPunts()**: Retorna els punts obtinguts per l'alumne segons la taula d'equivalències.
- **getCredits()**: Retorna el nombre de crèdits de l'assignatura.

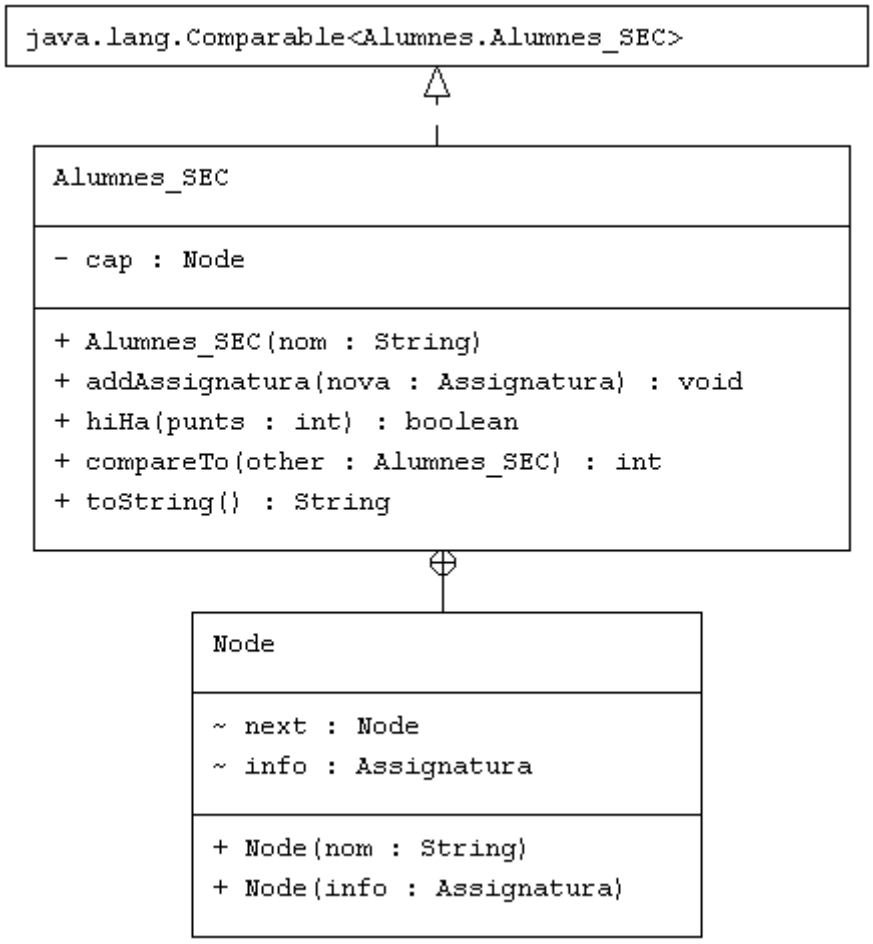
Sobreescriu els mètodes de la classe Objecte: No cal posar cap implements

- **toString()**: Retorna una representació en format de text de la instància Assignatura, mostrant el nom de l'alumne i la seva nota mitjana.
- **Equals():** Compara el nom de les dues assignatures.

Classe Alumnes_SEC

Ara implementarem una seqüència enllaçada amb capçalera, per això crearem una classe pública anomenada Alumnes_SEC, que contindrà una

classe privada anomenada Node, on el tipus de informació que guardarem serà del tipus Assignatura. Aquesta classe pertany al package Alumnes.



Classe Interna **Node**:
Representa cada node de la seqüència enllaçada, guarda informació sobre una assignatura i un enllaç al següent node.

- Atributs:
  - **info** (Assignatura): Conté l'assignatura amb el nom, els crèdits, la nota i l'estat de matrícula d'honor.
  - **next** (Node): Apunta al següent node de la seqüència. És null si el node és l'últim.
- Constructors:
  - **Node(Assignatura info):** inicialitza un node amb l'assignatura, i l'atribut next igual a null.

- **Node(String nom):** aquest constructor s'utilitzarà només per inicialitzar el node capçalera de la seqüència, el paràmetre que rep és el nom de l'alumne, inicialitzarà l'atribut info amb el segon constructor de la classe Assignatura.

Atributs d'instància:

- **cap** (Node): és el node capçalera de la seqüència enllaçada, guarda el nom de l'alumne i la nota mitjana calculada en funció de les assignatures que hi ha a la seqüència enllaçada, inicialment serà 0.

Constructors:

- **Alumnes_SEC(String nom)**: Inicialitza la seqüència enllaçada amb un únic node de capçalera (atribut cap) que conté el nom de l'alumne i una nota inicial de 0, utilitzant el segon constructor de la classe interna Node.

Mètodes:

- **addAssignatura(Assignatura nova)**: Afegeix una nova assignatura a la seqüència enllaçada.
  - Si l'assignatura ja existeix en la seqüència (determinat pel mètode equals de la classe Assignatura), es sobreescriu amb la nova informació.
  - Si és una assignatura nova, s'afegeix al final de la seqüència.
  - Caldrà recorre tota la seqüència enllaçada per recalcular la nota mitjana de l'alumne i actualitza la capçalera amb la nova mitjana.
  - El càlcul de la nota mitjana es realitza ponderant les assignatures segons els seus crèdits i punts obtinguts, observa els dos exemples anteriors.
- **hiHa(int punts)**: Verifica si existeix alguna assignatura dins la seqüència amb un valor de punts concret. Aquest mètode servirà per comprovar si l'alumne té assignatures amb matrícula d'honor o suspeses.

Mètode de la interfície Comparable:

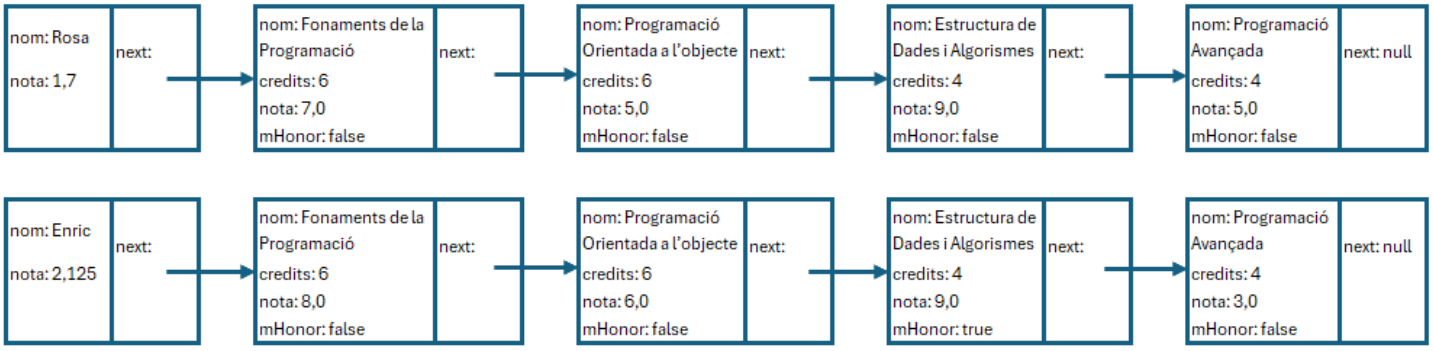
Cal posar implement a la capçalera de la classe

- **compareTo(Alumnes_SEC other)**: La interfície Comparable permet ordenar objectes de Alumnes_SEC segons la nota mitjana. A la classe de la classe cal indicar que implementa la interfície Comparable<Alumnes_SEC>, i cal implementar el mètode compareTo que compara la nota mitjana de l'alumne (emmagatzemada en el node capçalera) amb la d'un altre objecte Alumnes_SEC. Retorna un valor negatiu, zero o positiu segons si la nota mitjana és menor, igual o superior a la de l'altre objecte.

Sobreescriu els mètodes de la classe Objecte:

- **toString()**: Invoca el mètode toString de la capçalera per tal de mostrar el nom de l'alumne i la seva nota mitjana actual.

Exemples:



Classe ArbreException

La classe ArbreException és una excepció personalitzada que hereta de la classe Exception de Java. Aquesta excepció està dissenyada per manejar errors específics relacionats amb operacions en estructures de dades d'arbre. Ubica aquesta classe al package EstructuraArbre.

Constructor:

- **ArbreException(String msg)**: Constructor que rep un missatge de tipus String com a paràmetre i el passa al constructor de la superclasse (Exception). Aquest missatge permet descriure la causa

de l'excepció i facilita el diagnòstic de l'error quan aquesta excepció és llençada i capturada. El missatge és imprescindible!

Interfície Acb

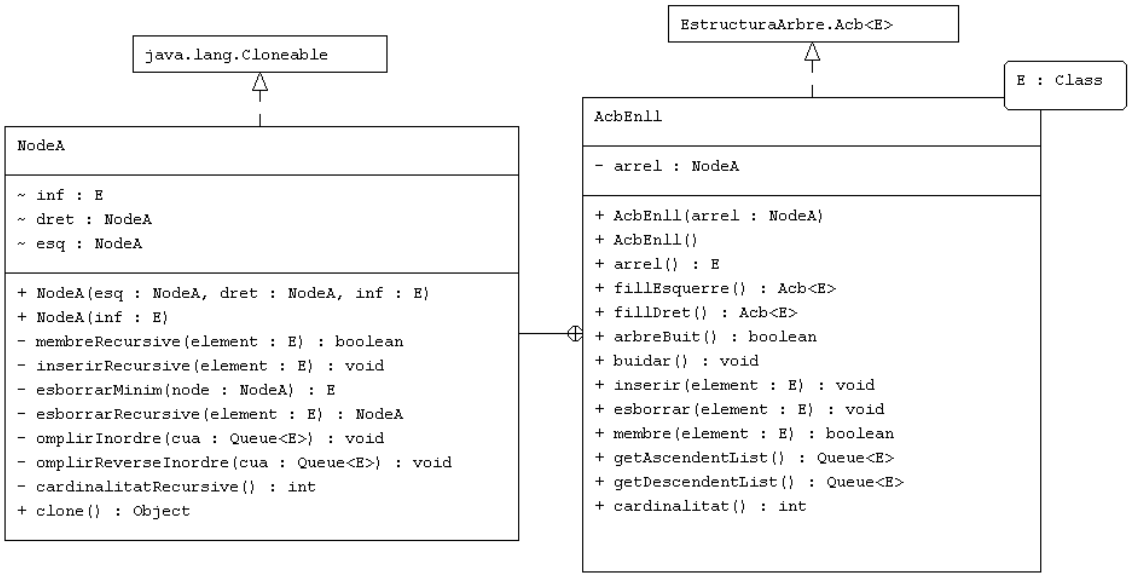
Copia la següent interfície dins del package EstructuraArbre. És la mateixa interfície que utilitza la professora Lina, a excepció dels mètode fillEsquerre i fillDret que és imprescindible que retorneu un subarbre amb els nodes clonats.

```
package EstructuraArbre;

public interface Acb<E extends Comparable<E>> {

    void inserir(E element) throws ArbreException;
    // Insereix un element a l'arbre. Si l'element ja
existeix, llança una excepció ArbreException.

    void esborrar(E element) throws ArbreException;
    // Esborra un element de l'arbre. Llança una excepció si
l'arbre és buit o si l'element no es troba a l'arbre.

    boolean membre(E element);
    // true si l'element està a l'arbre, fals en cas contrari

    E arrel() throws ArbreException;
    // Si no és buit, retorna el contingut de l'arrel, en cas
contrari llança una excepció ArbreException

    Acb<E> fillEsquerre()throws CloneNotSupportedException;
    // retorna una còpia del subarbre esquerre.
    // en cas que l'arbre estigui buit o si no té fill
esquerre retorna un arbre buit.

    Acb<E> fillDret() throws CloneNotSupportedException;
    // retorna una còpia del subarbre dret.
    // en cas que l'arbre estigui buit o si no té fill dret
retorna un arbre buit.

    boolean arbreBuit();
```

```
    // Retorna true si l'arbre és buit.

    void buidar();
    // Neteja l'arbre deixant-lo buit.
}
```

Classe AcbEnll

La classe AcbEnll<E> implementa un arbre binari de cerca (interfície Acb) generat amb nodes enllaçats. Aquesta classe utilitza generics (<E extends Comparable<E>) perquè els elements siguin comparables, i així puguin ser inserits en l'arbre de manera ordenada. A més, s'estructura amb una classe interna privada NodeA, que representa els nodes individuals de l'arbre. Observa que la classe AcbEnll no és clonable, només és clonaran els nodes quan ens demanin el fill dret o esquerre, però no podrem clonar un arbre.



La capçalera de la classe és:

```
public class AcbEnll <E extends Comparable<E>> implements
Acb<E>{

    private class NodeA implements Cloneable {
```

Atributs:

- **arrel:** Referència al node arrel de l'arbre, de tipus NodeA.

Constructors:

- **AcbEnll(NodeA arrel):** Inicialitza l'arbre amb un node arrel donat.
- **AcbEnll():** Constructor per defecte, que inicialitza un arbre buit.

Mètodes de la interfície Acb:

Cal implementar segons el detall de la interfície, a continuació detallo la relació entre els mètodes de la classe AcbEnll i el mètodes de la classe interna NodeA.

- Inserir – inserirRecursive
- Esborrar – esborrarRecursive i esborrarMinim
- Membre – membreRecursive
- FillEsquerre – clone
- FillDret – clone

Mètodes:

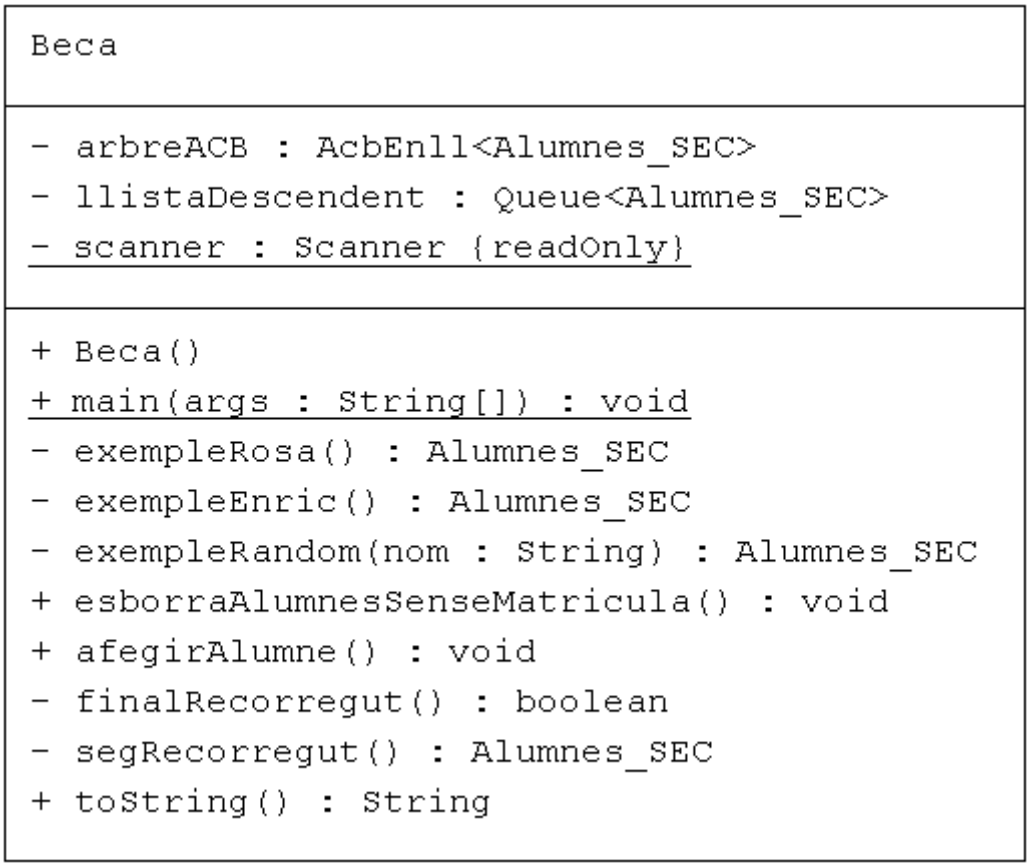
- **getAscendentList():** Retorna una cua amb els elements de l'arbre en ordre ascendent (inordre), invoca al mètode omplirInOrdre.
- **getDescendentList():** Retorna una cua amb els elements de l'arbre en ordre descendent (inordre invers), invoca al mètode omplirReverseInOrdre.
- **cardinalitat():** Retorna el nombre total de nodes en l'arbre, invoca al mètode cardinalitatRecursive.

Classe Beca + Main

La classe Beca gestiona una estructura de dades basada en un arbre binari de cerca (AcbEnll) per mantenir un llistat d'alumnes que estan inscrits per optar a una beca, cadascun amb assignatures i qualificacions associades (Alumnes_SEC).

És molt important que enteneu com funciona Acb i que s'ordenaran els alumnes per nota mitjana, per tant, no podem tenir dos alumnes amb la mateixa nota mitjana, però si podem tenir dos alumnes amb el mateix nom i diferents notes mitjanes.

Aquesta classe incorpora el main amb un menú amb operacions com afegir i esborrar alumnes, mostrar la llista completa d'alumnes en ordre descendent i eliminar alumnes que no tinguin matrícula d'honor en cap assignatura.



Atributs de classe:

- scanner: un objecte Scanner estàtic per gestionar les entrades de l'usuari.

Atributs d'instància:

- arbreACB: una instància d'un arbre binari de cerca (AcbEnll<Alumnes_SEC>) que emmagatzema els alumnes.
- llistaDescendent: una cua que conté la llista d'alumnes en ordre descendent per facilitar la visualització o iteració.

Constructor per defecte de la classe Beca:

- Inicia arbreACB i insereix exemples d'alumnes (Rosa, Enric i tres altres amb noms aleatoris) amb qualificacions predeterminades o

aleatòries a l'arbre. Després inicialitza llistaDescendent amb els alumnes en ordre descendent utilitzant el mètode getDescendentList.

Mètodes privats de la classe Beca:

- **exempleRosa() i exempleEnric()**: Retorna una instància Alumne_SEC amb els exemple esmentat anteriorment.
- **exempleRandom(String nom)**: Genera un alumne amb les quatre mateixes assignatures i amb les notes aleatòries.
- **finalRecorregut():** Retorna true si la llista Descendent ha arribat al final.
- **segRecorregut()**: Gestiona el recorregut de la llistaDescendent retornant els elements un a un i indicant si la llista ha arribat al final.

Sobreescriu els mètodes de la classe Objecte:

- **toString()**: Retorna una representació en format text de la llista completa d'alumnes en ordre descendent. Cal utilitzar els mètodes finalRecorregut i segRecorregut per gestionar el bucle.

Mètodes públics:

- **esborraAlumnesSenseMatricula()**: Recorre la llista d'alumnes i elimina del arbre aquells que no tinguin cap assignatura amb matrícula d'honor.
- **afegirAlumne()**: Permet a l'usuari introduir les dades d'un nou alumne amb múltiples assignatures i l'afegeix a l'arbre. Primer caldrà demanar el nom de l'alumne, i després afegir una a una les assignatura segons indiqui l'usuari.

Mètode main:

- Crea una instància de Beca i ofereix un menú d'opcions perquè l'usuari pugui interactuar amb les funcionalitats:
  - Opció 1: Afegir un nou alumne.
  - Opció 2: Esborrar un alumne a partir del seu nom.
  - Opció 3: Mostrar tots els alumnes en ordre descendent.
  - Opció 4: Esborrar alumnes sense matrícula d'honor.
  - Opció 5: Sortir del programa.