Pràctica 3

Objectius:

- **Estructura de dades no lineals: arbres binaris**
- **Lectura i escriptura de fitxers**

Durada: Dues sessions

Llenguatge: Java utilitzant IntelliJ o Eclipse

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

Data Lliurament: diumenge 3 de novembre.

Enunciat

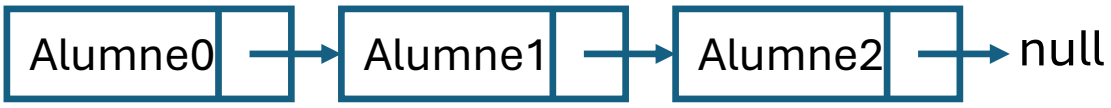
A una escola volen emmagatzemar informació de la família dels seus estudiants, on es podrà fer accions bàsiques:

- Veure un llistat alfabètic d'alumnes
- Afegir un nou alumne
- Modificar la família d'un alumne
- Eliminar un alumne

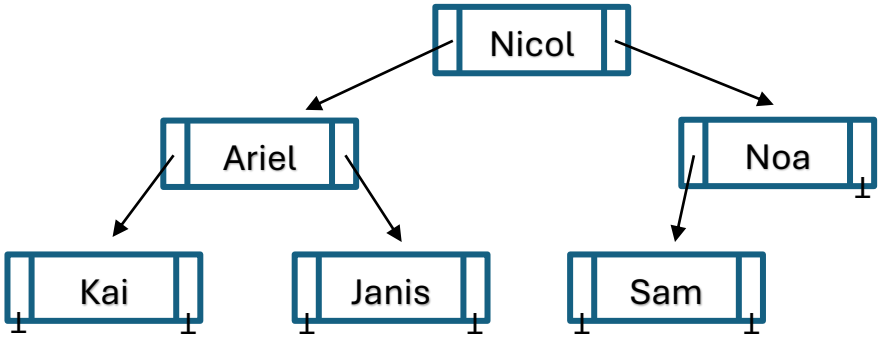
i al mateix temps també vol extreure informació estadística i poder fer preguntes del estil:

- Quants alumnes han nascut a Barcelona?
- Quants alumnes tenen descendència de Girona?
- Quants alumnes tenen un únic progenitor?
- Quants alumnes tenen progenitors divorciats?
- Quants alumnes tenen avis o àvies?
- Quants alumnes tenen més de 2 avis o àvies?

Per això, guardarem a una seqüencia enllaçada la informació de tots els alumnes (guardat alfabèticament), on cada alumne tindrà un arbre binari amb la informació de les seves famílies.



AlumneX conté un arbre binari amb tota la família, per exemple:



Nivell 0: alumne

Nivell 1: progenitors

Nivell 2: avis i àvies

Class Person

Per això, primer creem una classe anomenada Person:

Person
+ WIDOWED : int {readOnly} + DIVORCED : int {readOnly} + MARRIED : int {readOnly} + SINGLE : int {readOnly} - maritalStatus : int - placeOfOrigin : String - name : String
+ Person(name : String, placeOfOrigin : String, maritalStatus : int) + Person(formattedString : String) + getName() : String + getPlaceOfOrigin() : String + getMaritalStatus() : int - getMaritalStatusString() : String + toString() : String

Amb els següents atributs:

- **name**: Emmagatzema el nom de la persona (per exemple, "Rosa").
- **placeOfOrigin**: Emmagatzema el lloc d'origen de la persona (per exemple, "Barcelona").
- **maritalStatus**: Un valor numèric que indica l'estat civil (solter, casat, divorciat, viudo).

La classe té un constructor amb tots els paràmetres d'entrada necessaris, per exemple:

Person personaRosa = new Person("Rosa", "Barcelona", Person.*MARRIED*);

I un segon constructor amb un únic paràmetre d'entrada que conté un string amb el següent format:

"Name: Rosa, place of Origin: Barcelona, marital status: Married"

Aquest segon format s'utilitzarà per instanciar les persones a partir d'un fitxer txt. Per tant, és necessari que el mètode toString segueixi aquest mateix patró. Cal que funcioni la següent sentència:

new Person(personaRosa.toString());

Els constructor donaran una excepció si l'atribut maritalStatus no és correcta.

Class BinaryTree

La classe BinaryTree tindrà una arrel d'una classe interna anomenada NodeA. Cada node té tres atributs, un per emmagatzemar la informació i altres dos per referenciar a un altre objecte node, aquesta implementació ha sigut treballada a classe de teoria.

- **BinaryTree()**

Aquest constructor instancia l'arrel a null.

- **BinaryTree(String filename)**

Aquest constructor carrega un arbre binari des d'un fitxer. Llegeix el fitxer on l'arbre es va desar prèviament en format preordre i recrea l'arbre binari a partir de la informació desada. S'invoca el mètode privat preorderLoad(BufferedReader bur).

- **getName()**

Aquest mètode retorna el nom del alumne que es troba a l'arrel de l'arbre.

- **addNode(Person unaPersona, String Level)**

Aquest mètode públic afegeix un nou node a l'arbre, cridant al mètode recursiu intern addNodeRecursive. El paràmetre level indica la posició del nou node dins l'arbre, seguint el mateix format de "L" per esquerra i "R" per dreta. Per exemple, alguns valors vàlids serien: 'L', 'R', 'LL', 'LR', 'RL', 'RR'.

- **displayTree()**

Aquest mètode mostra l'arbre binari de manera estructurada a la consola, cridant el mètode recursiu displayTreeRecursive. Comença el recorregut des de l'arrel de l'arbre. Us recomano veure l'execució. Visualització del exemple de la pàgina 2:

```
Arbre binari estructurat:
        Nicol
                Ariel
                        Kai
                        Janis
                Noa
                        Sam
```

- **preorderSave()**

Aquest mètode desa l'arbre binari en format preordre a un fitxer de text. Si l'arbre és buit, llança una excepció. El nom del fitxer és derivat del nom de la persona emmagatzemada a l'arrel de l'arbre. Crida al mètode recursiu preorderSaveRecursive per guardar cada node a una línia, on s'afegeix la marca ";" per saber si no hi ha node esquerra o dreta. Us recomano veure el fitxer Nicol.txt.

- **removePerson(String name)**

Aquest mètode permet eliminar un membre de la família amb el nom especificat, excepte l'estudiant (arrel). Crida al mètode privat removePersonRecursive(String name).

- **isFrom(String place)**

Aquest mètode comprova si l'estudiant (arrel de l'arbre) prové d'una localitat específica.

- **isDescentFrom(String place)**

Aquest mètode comprova si algun membre de l'arbre de la família prové de la localitat especificada. Utilitza el mètode recursiu privat isDescentFromRecursive(String place) per verificar si hi ha algun membre a l'arbre amb aquesta localitat d'origen.

- **howManyParents()**

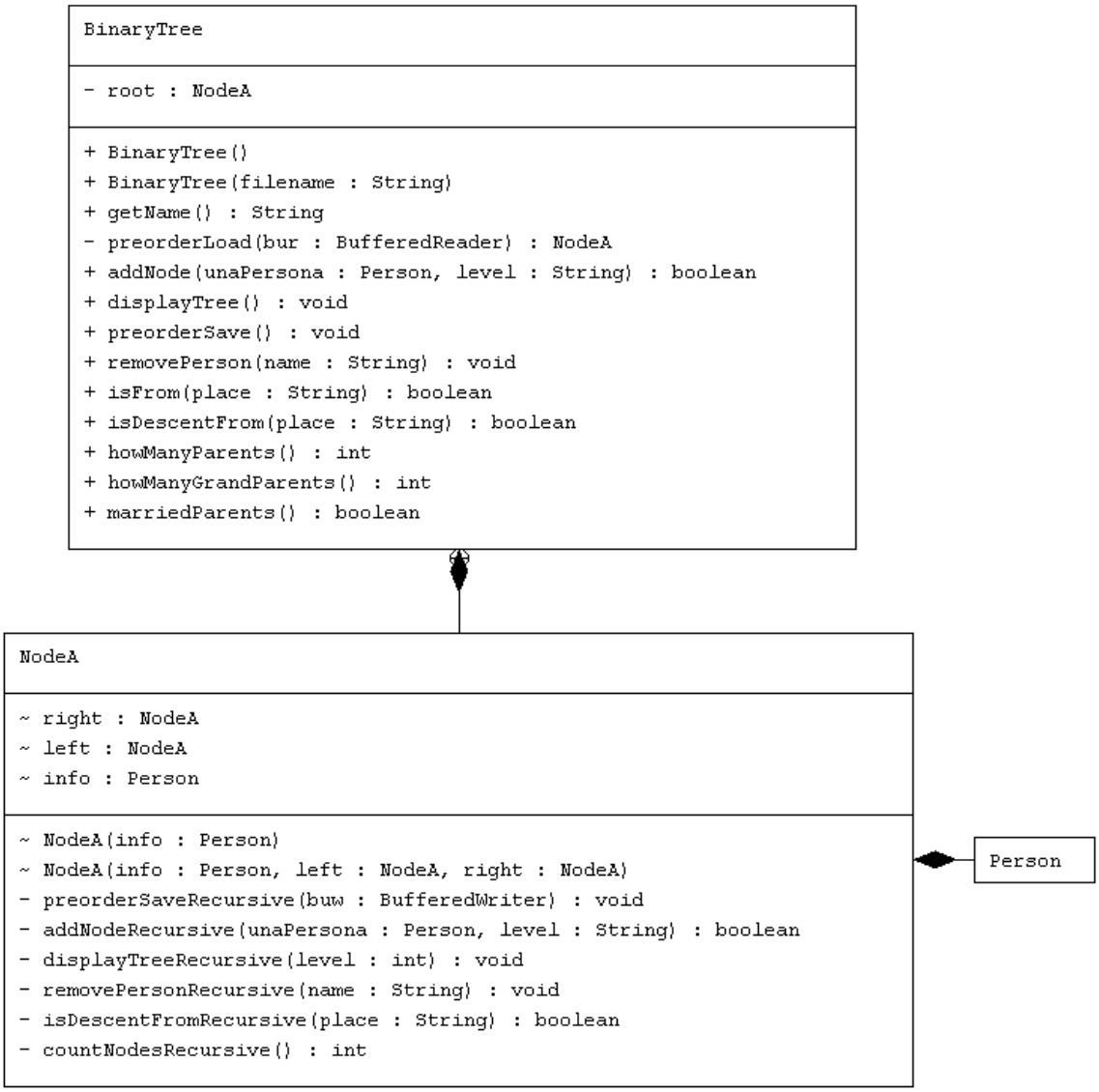
Aquest mètode retorna el nombre de progenitors de l'estudiant. L'arbre binari té dos nodes per als pares: el node esquerre (left) i el node dret (right).

- **howManyGrandParents()**

Aquest mètode compta el nombre de membres de la família que són avis o àvies. Utilitza el mètode recursiu countNodesRecursive() per comptar tots els nodes de l'arbre.

- **marriedParents()**

Aquest mètode comprova si ambdós progenitors de l'estudiant (tant l'esquerra com la dreta) estan casats.



Class Students

La classe Students gestiona una seqüència enllaçada sense capçalera on cada node conté un arbre binari. Aquesta estructura s'utilitza per emmagatzemar una llista d'estudiants alfabèticament.

- **addStudent(BinaryTree nouEstudiant)**:

Afegeix un nou arbre binari a la seqüència de manera ordenada alfabèticament pel nom de l'alumne (persona a l'arrel de l'arbre). Cal fer-ho de manera eficient.

- **removeStudent(String name):**

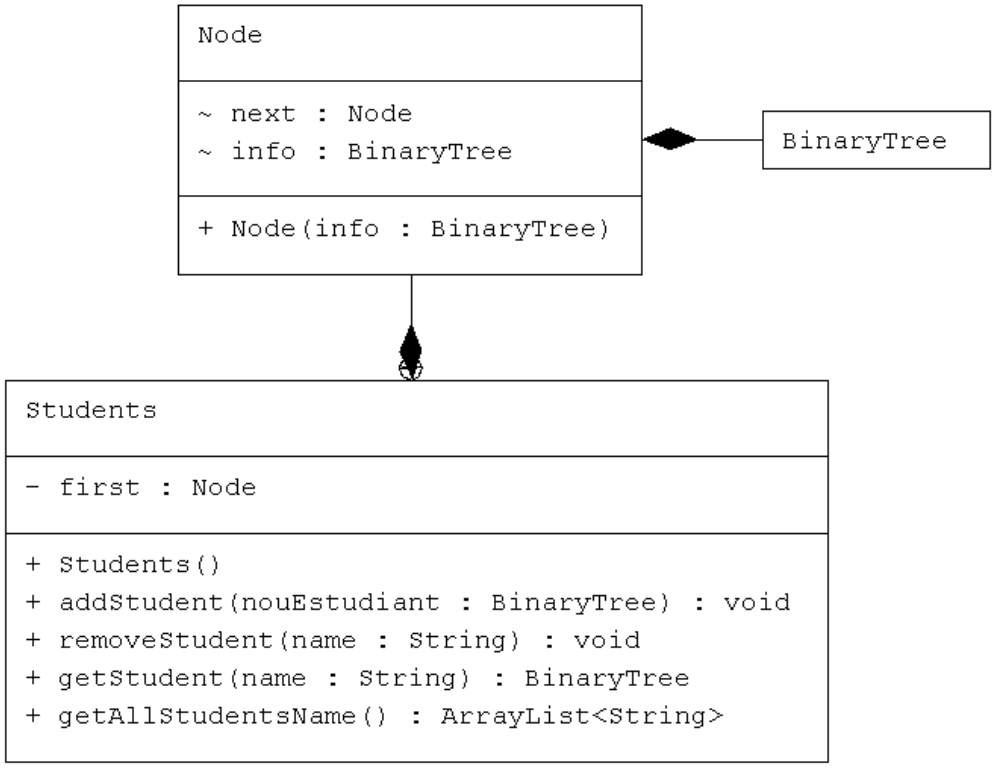
Elimina de la seqüència l'arbre que tingui el nom especificat. Cal fer-ho de manera eficient.

- **getStudent(String name)**:

Cerca un arbre dins de la seqüència pel seu nom. Si el troba, retorna l'arbre, si no, retorna null. També comprova si la llista està buida. Cal fer-ho de manera eficient.

- **getAllStudentsName()**:

Retorna una arrayList amb els noms de tots els estudiants emmagatzemats a la seqüència enllaçada. Si la llista està buida, retorna null.



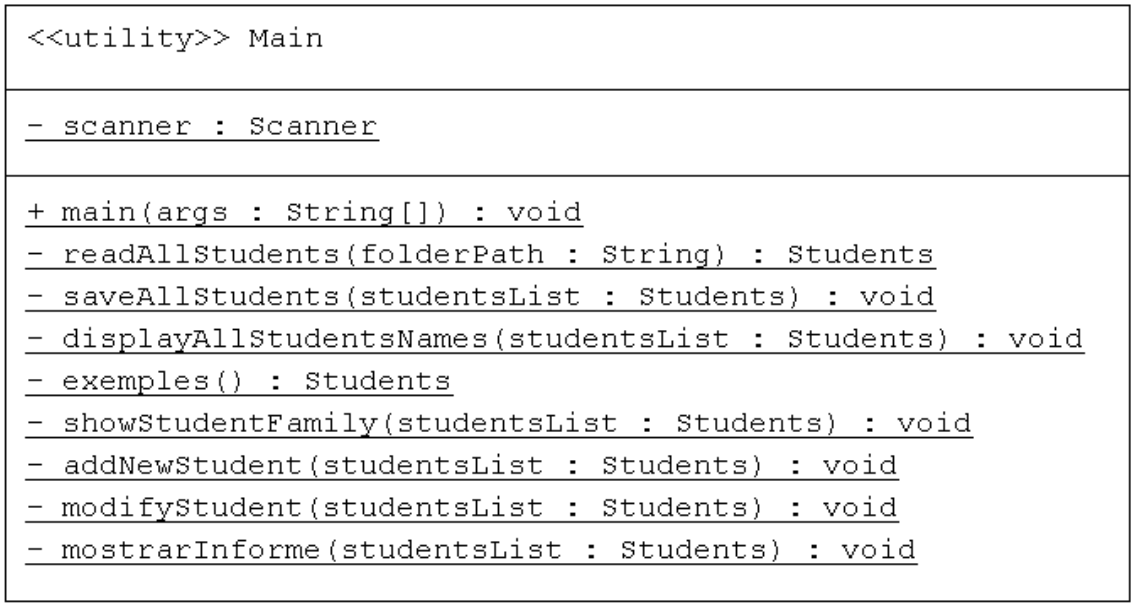
Class Main

A la classe main es mostrarà el següent menú:

Menú Principal:

1. Mostrar llistat d'estudiants
2. Mostrar família d'un estudiant
3. Afegir un estudiant
4. Modificar un estudiant
5. Mostrar el informe
6. Guardar i Sortir

Tria una opció:



- **readAllStudents**

Llegeix tots els fitxers de text dins d'una carpeta especificada (folderPath) que conté la informació dels estudiants. Cada fitxer representa un estudiant, i el contingut és convertit en un arbre binari per representar la seva família. Afegeix cada estudiant a un objecte de la classe Students que emmagatzema aquests arbres.

- **saveAllStudents**

Guarda tota la informació dels estudiants emmagatzemada en arbres binaris. Per cada estudiant, genera un fitxer que representa l'estructura de l'arbre utilitzant un recorregut preordre de l'arbre binari (preorderSave).

- **displayAllStudentsNames**

Mostra tots els noms dels estudiants emmagatzemats a la llista de Students.
Si no hi ha estudiants, mostra un missatge indicant-ho.

- **showStudentFamily**

Mostra la família d'un estudiant en particular. Es demana el nom de l'estudiant a l'usuari, i si l'estudiant existeix, es mostra l'estructura de l'arbre binari que representa la seva família.

- **addNewStudent**

Permet afegir un nou estudiant. Es demana a l'usuari el nom, lloc d'origen i estat civil de l'estudiant. Es crea un arbre binari per l'estudiant i s'afegeix a la llista d'estudiants.

- **modifyStudent**

Permet modificar la informació familiar d'un estudiant existent a partir del seu nom. Hi ha dues opcions afegir o eliminar membres de la família a l'arbre binari de l'estudiant.

- **mostrarInforme**

Genera un informe amb estadístiques sobre els estudiants, primer caldrà preguntar totes dues ciutats (ciutat on ha nascut l'estudiant i ciutat de procedència de la família), veieu l'exemple fet amb Barcelona i Girona.