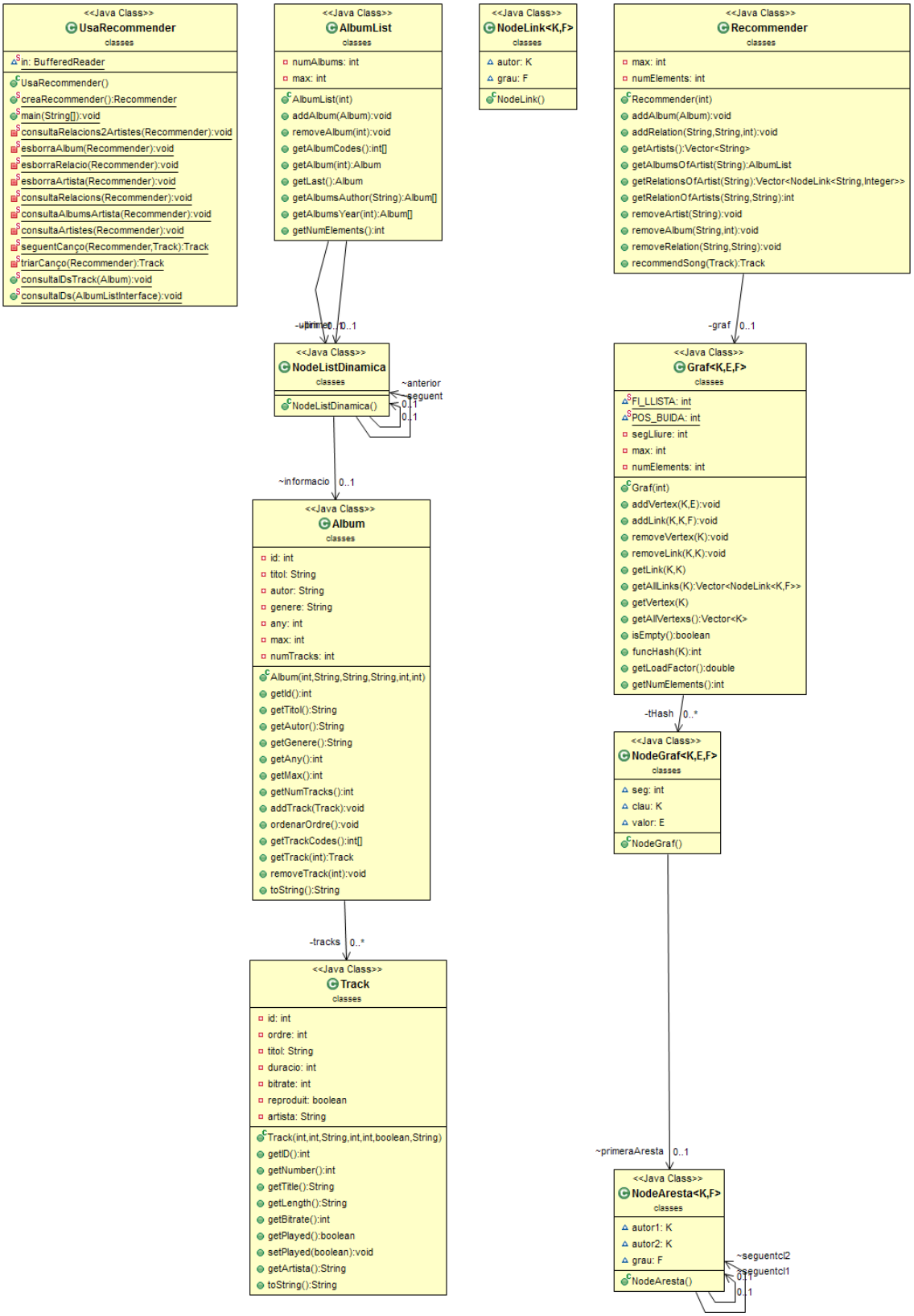
UNIVERSITAT ROVIRA I VIRGILI

ESCOLA TÈCNICA SUPERIOR ENGINYERIA

Estructures de Dades
Pràctica 4

ALUMNE/S: Marc Elias Del Pozzo
PROFESSORS: Benet Campderrich Falgueras
ASSIGNATURA: Estructures de Dades
ENSENYAMENT: Grau d'Enginyeria Informática
DATA: 21/ Maig/ 2013

Diagrama de classes



Anàlisi de requeriments i estudi previ

En primer lloc, aquesta quarta pràctica es basa en les anteriors. Per tant he aprofitat les classes Track, Album, AlbumListPunters i TaulaHash. A partir d'aquí he creat una nova classe Graf que implementa una taula de hash genèrica per a guardar els vèrtexs del graf, i una llista d'adjacència per a cada vèrtex aquesta última utilitzant punters. A més a més he creat l'altre classe Recommender que utilitzarà un graf per a guardar els artistes i les seves relacions.

Classe Graf

En aquest punt, abans de implementar la taula de hash he hagut de crear una classe auxiliar NodeGraf per a guardar la clau, el valor, l'element següent de les col·lisions de les taules de hash i un punter a la primera aresta de tipus NodeAresta.

Aquesta nova classe implementa una llista com ja havíem vist a la pràctica 2, amb la diferencia que té 2 punters de següent element i la etiqueta del graf.

A la classe graf he creat els següents atributs per a guardar la informació:

1. `static int FI_LLISTA=-1`, constant que em servirà per a controlar si s'ha arribat al final de la llista.
2. `static int POS_BUIDA=-2`, constant que utilitzo per a controlar si la posició actual està buida.
3. `private Vector<NodeGraf<K,E,F>> tHash`, un vector de les col·leccions de java de NodeGraf genèrica i que serà la nostra taula de hash en sí.
4. `int segLliure`, enter que serveix per a guardar la següent posició lliure a la llista.
5. `int numElements`, enter que ens indica el número de claus i valors que hi ha a la llista.
6. `int max`, enter que ens indica el nombre màxim de claus i valors a guardar a la llista.

A continuació he implementat tots els mètodes que son necessaris per a crear i manipular el graf amb els seus vèrtexs i arestes.

Classe Recommender

A la classe AlbumLibrary, he creat els següents atributs per a guardar la informació:

1. `Graf<String, AlbumList, Integer> graf`, utilitzem la classe creada anteriorment per a guardar el nostre graf d'artistes i la seva llista d'Àlbums i el grau de similitud entre ells.
2. `int numElements`, enter que ens permet saber el nombre d'elements que tenim al graf.
3. `int max`, enter que ens indica el nombre màxim d'artistes a guardar a la llista.

I a partir d'aquí he creat els mètodes que son necessaris per a crear i manipular el graf i a més a més el mètode *recommendSong()* que permet recomanar una cançó segons el grau de similitud entre artistes.

Per acabar, he creat la classe UsaAlbum que conté el mètode main() i permet executar i guardar la informació als tipus abstracte de dades, manipular-la i mostrar-la a l'usuari per pantalla.

Quadre resum cost temporal a cada implementació

Els costos temporals de la classe Àlbum els he resumit en la taula que hi ha a continuació:

	Mètode addTrack()	**Mètode getTrackCodes()**	**Mètode getTrack()**	**Mètode removeTrack()**
Album	Quasi Lineal	Lineal	Lineal	Lineal

El cost del mètode addTrack és quasi lineal degut a l'utilització d'un algorisme d'ordenació.

Ara analitzarem els costos de la implementació dinàmica de la llista d'Àlbums a la taula que tenim a continuació:

	Add Album	**Remove Album**	**get Album Codes**	**get Album**	**get Last**	**get Albums Author**	**get Albums Year**	**get Num Elements**
Punters	Lineal	Lineal	Lineal	Lineal	Constant	Lineal	Lineal	Constant

I per acabar, els costos temporals de la Taula de Hash genèrica que hem implementat a aquesta última pràctica:

	add Vertex	**addLink**	**remove Vertex**	**remove Link**	**get All Links**	**get Link**	**get Vertex**	**get All Vertexs**
Graf	Constant	Constant	Constant	Lineal (Arestes)	Lineal (Arestes)	Lineal (Arestes)	Constant	Lineal

Les funcions que treballen amb la taula de hash tenen la majoria un cost constant excepte getAllVertexs que és estrictament lineal ja que ha de retornar tots els elements de la taula. Les altres operacions que treballen amb la llista d'adjacències tenen un cost lineal respecte al nombre d'arestes que s'hagin guardar per cada artista.

He omès els costos de la classe Recommender, ja que són tots constants ja que treballem directament amb una taula de hash per a guardar els artistes i una llista d'adjacències per a cada artista.