ESTRUCTURA DE DADES PRÀCTICA 3

Anàlisi de requeriments

Els principals requisits per a dur a terme aquesta pràctica son principalment 3:

-tenir una sèrie de certs coneixements de Java i l'ús d'un programa per a implementar aquest codi.

-dispondre d'un editor de text per a realitzar l'informe posterior tal com el Word i passar-lo després a PDF.

-tenir els recursos suficients per superar tots el problemes que et trobis durant la realització de la practica, com ara molta paciència i ganes de treballar.

Estudi Previ

Per a anar iniciant el desenvolupament de la practica primerament tractarem la implementació dels diferents TADs que empraré, per fer això, crearé una classe que serà el TAD jugador i per altra banda una interfície que hem servirà per donar herència a una TAD de jugadors i a una classe JugadorsHash(per a poder cridar mètodes de la hash amb els mètodes de jugador) i una altra interfície per a la hash table(part opcional).

Per anar finalitzant implementaré dos classes que seran filles del TAD de jugadors, les classes ordenada i desordenada i posteriorment generaré totes les classes necessàries per controlar les excepcions definides. Últimament crearé el main on ajuntaré totes les funcions descrites al llarg de les classes per a generar finalment un programa 100% funcional.

Decisions preses durant el desenvolupament de la pràctica

-Per a les funcions getEquips i getPosicions he creat un mètode apart per comprovar que al incloure un nom a la llista amb els equips i posicions no es repeteixi múltiples vegades el mateix equip o posició dintre de la llista, per mostrar així nomes els equips o posicions diferents.

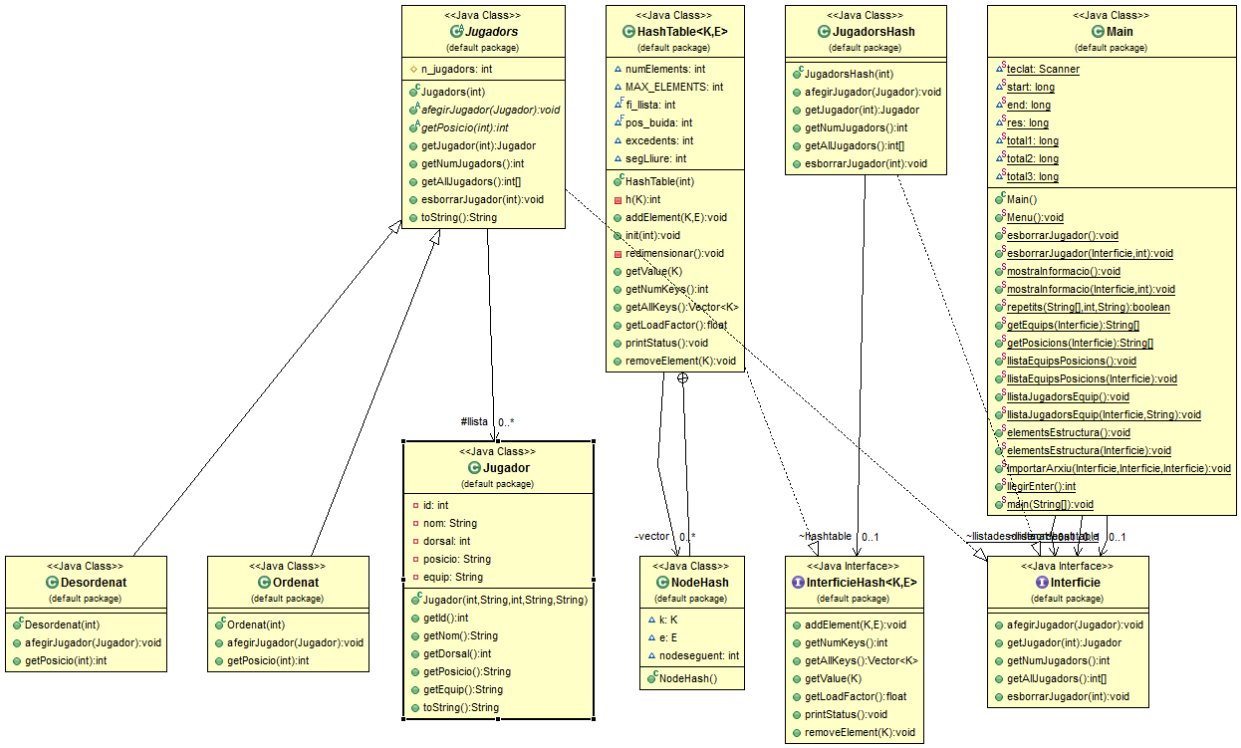
-Per a controlar que el programa no deixi de funcionar durant el seu funcionament, he decidit crear quatre excepcions diferents, una que per controlar que no hi dos jugador a la llista de jugadors amb el mateix identificador, una altra per controlar quan no existeix un jugador a la llista, una altra per comprovar quan la llista esta plena i no hi caben mes jugadors i finalment una altra per comprovar si la llista esta buida. També he decidit crear un mètode anomenat llegirenter() per tal de controlar que el valor introduït sigui un enter on es demana un numero, evitant així molts errors.

-Tal com ens ha especificat el professor a les classes de practiques, per tal de fer el programa el mes eficient possible, a les dos filles del TAD jugadors nomes he definit 2 funcions diferents, ja que son els que canviaran segons si implementem una llista ordenada o desordenada, un mètode per afegir jugadors de forma ordenada o desordenada, i un mètode per trobar la posició a la llista d'un jugador segons l'identificador tant de forma normal com de forma dicotòmica.

-Pel que fa la taula de hash, dintre de la mateixa classe he creat una altra classe que seran els nodes que contindrà la hash, també he afegit una variable de tipus enter per a poder tenir controlada la pròxima posició lliure i una altra per controlar si estic a la part principal o a la zona d'excedents.

-Per a no tenir que repetir els mètodes diverses vegades al main, he creat mètodes que van cridant a un altre mètode tres vegades(un cop per la ordenada, un cop per la desordenada i un altre cop per la taula de hash); per a poder fer això he tingut que implementar una altra classe que hereti de la interfície de jugadors on vaig cridant els mètodes de la de hash.

Diagrama de classes utilitzades



Cost temporal de les operacions

COSTOS TEMPORALS	LLISTA ORDENADA	LLISTA NO ORDENADA
afegirJugador	N	N
getEquips	N^2	N^2
getPosicions	N^2	N^2
getJugador	Log(n)	N
esborrarJugador	N	N
getNumJugador	1	1
getAllJugadors	N	N

COSTOS TEMPORALS	Hash Table
addElement	N
redimensionar	N
getValue	N
getNumKeys	1
getAllKeys	N
getLoadFactor	1
printStatus	N^2
removeElement	N

Aspectes millorables

Els principals aspectes a millorar suposo que seria intentar disminuir el codi que he implementat segons els mètodes, però ara per ara no disposo d'uns coneixements tan desenvolupats de java per reduir l'ocupació del codi mantenint la mateixa eficiència.

Un altre aspecte a millorar a part de disminuir l'ocupació del codi seria intentar fer mètodes encara mes eficients amb uns costos temporals mínims.

Per finalitzar també es podria implementar una interfície gràfica per no tenir que treballar mitjançant la consola tot el temps.