Especificació comuna

Grup 7

Especificació de les classes de Persistència

Classe CapaPersistència

Classe que guarda i restaura objectes del disc. Un objecte guardat al disc estarà encriptat, i estarà identificat per un parell de strings id,tipus.

CapaPersistència () CapaPersistència

Pre: -
Post: Crea un objecte de tipus CapaPersistència.

obtenirTraducció (id:string, idioma:string) string

Pre: L'idioma *idioma* existeix.
Post: Retorna la frase amb id *id* en l'idioma *idioma*. En cas de no existir la frase retornarà la frase que tingui identificador “untranslated”, i *null* en cas de no trobar-la.

llistarIdiomes () Set(string)

Pre: -
Post: Retorna un llistat amb tots els noms dels idiomes disponibles.

restaurarObjecte (id:string, tipus:string) Object

Pre: Existeix un objecte amb identificador *id* i de tipus *tipus*.
Post: Retorna un nou objecte que va ser guardat prèviament amb identificador i tipus *id,tipus* respectivament fent ús de la funció guardarObjecte.

guardarObjecte (id:string, tipus:string, obj:Objecte) void

Pre: *obj* és una classe Serializable i compleix tots els requisits que l'especificació de Sun Java diu.
Post: Guarda i encripta al disc la classe *obj* i les seves classes associades tal com indica l'especificació de Sun Java identificada mitjançant *id*,*tipus* i en cas d'existir algun objecte amb el mateix identificador i tipus el reemplaça.

esborrarObjecte (id:string, tipus:string) void

Pre: Existeix un objecte amb identificador *id* i de tipus *tipus*.
Post: Esborra l'objecte amb identificador i tipus *id,tipus* respectivament.

llistarIds (tipus:string) Set(string)

Pre: -
Post: Llista els identificadors de tots els objectes guardats que siguin de tipus *tipus*.

Especificació de les classes d'Estadístiques

Classe Estadístiques

Classe que representa el sistema d'estadístiques del joc. Per a cada jugador guarda un conjunt d'atributs els quals el descriuen de forma estadística.

Estadístiques () Estadístiques

Pre: -
Post: Retorna una nova classe d'estadístiques.

seleccionar (jugadors:Set(string), atributs:Set(string)) Set(nom:string, Set(integer))

Pre: *jugadors* és un vector de noms de jugadors existents. a*tributs* és un vector de noms d'atribut.
Post: Retorna un vector de jugadors i cada un dels valors dels atributs especificats. En cas que el vector *jugadors* sigui buit, retorna tots els jugadors.

consultarAtribut (jugador:string, atribut:string) int

Pre: El jugador *jugador* existeix i *atribut* és un dels seus atributs.
Post: Retorna el valor de l'atribut *atribut* del jugador *jugador*.

modificarAtribut (jugador:string, atribut:string, valor:int)

Pre: El jugador *jugador* existeix i *atribut* és un dels seus atributs.
Post: Es modifica el valor de l'atribut *atribut* del jugador *jugador*.

crearJugador (jugador:string, atributs:Set(valorEstadística)) void

Pre: No existeix cap jugador amb nom *jugador*.
Post: S'afegeix el jugador *jugador* i es creen per aquest els atributs definits al vector *atributs* amb els seus corresponents valors inicials.

eliminarJugador (jugador:string) void

Pre: -
Post: Elimina, si existeix, el jugador *jugador* i totes les seves estadístiques.

Classe EstadístiquesJugador

Classe que defineix un jugador dins del sistema d'estadístiques i que té associada un conjunt d'atributs i valors del jugador que representa. Aquesta especificació només és una referència.

Atributs

nom:string Nom del jugador

EstadístiquesJugador (nom:string) EstadístiquesJugador

Pre: -
Post: Crea un jugador amb nom *nom* i retorna el nou objecte de classe creat.

getNom () string

Pre: -.
Post: Retorna el nom del jugador.

getValor (atribut:string) int

Pre: Exiteix un atribut a*tribut*.
Post: Retorna el valor de l'atribut *atribut.*

setValor (atribut:string, valor:int) void

Pre: -
Post: S'actualitza el valor de l'atribut *atribut* a *valor* i en cas que no existeixi l'esmentat atribut es crea.

Classe ValorEstadística

Classe que defineix una parella atribut,valor la qual representa un atribut d'estadístiques amb el seu corresponent valor.

Atributs

atribut:string	Nom de la dada.
valor:integer	Valor de la dada.

ValorEstadística (nom:string,valor:int) ValorEstadística

Pre: -
Post: Crea un objecte ValorEstadística amb nom d'atribut *nom* i valor *valor* i retorna el nou objecte de classe creat.

getValor () int

Pre: -
Post: Retorna el valor de l'atribut.

getAtribut () string

Pre: -
Post: Retorna el nom de l'atribut.

setValor (valor:int) void

Pre: -
Post: S'actualitza el valor de l'atribut a *valor*.

Especificació de les classes Estratègia i Regla

Classe Estratègia

Classe que gestiona una estratègia formada per regles. Cada regla té una prioritat i les prioritats són correlatives, des de 0 (primera en ser avaluada) fins a N-1 on N és el nombre de regles.

Estratègia () Estratègia
Estratègia (e:Estratègia) Estratègia
Pre: -
Post: Crea una estratègia buida o, en cas d'especificar una estratègia e, retorna una nova estratègia que conté les mateixes regles que e (una còpia exacta).

afegirRegla (regla:string, accio:string, variables set(variable:string, valor:double)) void
afegirRegla (regla:string, accio:string, prioritat:int, variables set(variable:string, valor:double)) void
Pre: variable ha de contenir necessàriament totes les variables necessàries per avaluar la regla afegida. Regla ha de ser una regla correcta.
Post: Crea una regla a l'estratègia amb una acció associada i una prioritat. En cas que no especifiquem la prioritat la regla s'afegirà de forma que serà l'última en ésser avaluada.

consultarRegles () Set(prioritat:int, accio:string, regla:string)
Pre: -
Post: Retorna un conjunt de regles. Aquestes estan formades per la seva prioritat, l'acció que prendran i la regla en el format definit.

intercanviarPrioritat (a:integer, b:integer) void
Pre: Existeixen dues regles amb prioritats a i b.
Post: S'intercanvia les prioritats de les regles amb prioritat a i b.

esborrar(prioritat:integer) void
Pre: prioritat és la prioritat d'una regla existent.
Post: S'elimina la regla amb prioritat prioritat.

decidirAcció(variables:Set(string,integer)) string
Pre: Es passa com a paràmetre un conjunt de variables i els seus valors.
Post: Es retorna l'acció de la primera regla que avalua a cert si les ordenem per prioritat.

eliminarAccio(accion:string, variables:set(set(string,double)))
Estratègia
Pre:
Post: retorna una còpia de l'estratègia amb totes les regles excepte aquelles que duen associades l'acció “accion”

Nota: Els sets de variables han de constar de tuples allà on el primer element de la tupla és la variable i el segon el valor de l'esmentada variable.

S'ha afegit la funció “eliminarAccio” per tal de poder treballar amb una còpia d'estratègia on una determinada acció no sigui present. Ens ha semblat necessari per tal de regular cassos on una regla s'avaluï com a certa però l'acció associada no sigui aplicable.

A regla s'ha afegit la funció “esCorrecta” per tal de proveir a qui l'usi de eines per assegurar-se de complir les precondicions i la funció “evaluarReglaNumero” que permet usar la classe regla com una calculadora que no només avaluï true o false.

Classe Regla

Classe que gestiona una regla booleana la qual està formada per operadors, funcions, immediats i variables.

Atributs

regla:string	Representació de la condició que activa la regla.
acció:string	Acció que es durà a terme si es compleix la condició.

Regla (regla:string, accio:string, variables:Set(variable:string, valor:integer)) Regla
Pre: Regla és una regla sintàcticament correcta segons les normes definides pel grup 7. Variable ha de contenir necessàriament totes les variables necessàries per avaluar la regla.
Post: Crea una regla nova amb la corresponent regla i acció.

avaluarRegla (variables:Set(variable:string, valor:integer)) bool
Pre: l'argument és un vector de variables i el seu valor. És estrictament necessari que es proporcionin totes les variables necessàries per avaluar la regla.
Post: Retorna el valor que resulta d'avaluar la regla seguint les normes i la sintaxi definida pel grup 7 de PROP.

getRegla () string
Pre: -
Post: Retorna el string que forma la regla.
getAcció () string
Pre: -
Post: Retorna l'acció associada a la regla

avaluarReglaNumero(variables:Set(variable:string, valor:integer)) Double
Pre: l'argument és un vector de variables i el seu valor. És necessari que es proporcionin totes les variables necessàries per avaluar la regla.
Post: Retorna el valor que resulta d'avaluar matemàticament la regla seguint les normes i la sintaxi definida pel grup 7 de PROP. En cas que no es compleixi la precondició el valor retornat pot ser incorrecte.

esCorrecta (cadena:string, variables:Set(variable:string, valor:integer)) boolean
Pre: variables ha de contenir totes les variables que puguin aparèixer a la regla.
Post: es retorna cert si la cadena és avaluable. És a dir, si és sintàcticament correcta.

Especificació de les classes Carta i ConjuntCartes

Classe Carta

Classe que representa una carta.

Atributs

valor:integer	Valor numèric de la carta.
coll:CollCarta	Tipus de coll de la carta.

Carta (valor:integer, coll:CollCarta) Carta

Pre: *valor* és un enter entre 1 i 13 ambdós inclosos.
Post: Crea una carta amb el seu valor i coll.

getValor() integer

Pre: -
Post: Retorna el valor de la carta.

getColl() CollCarta

Pre: -
Post: Retorna el coll de la carta.

Classe ConjuntCartes

Classe que representa un conjunt de cartes. Un conjunt de cartes pot tenir cartes repetides, és a dir, pot tenir dues cartes amb mateix valor i coll. Això és important ja que cal garantir que la implementació pot tractar les operacions entre conjunts sense errors.

ConjuntCartes (nbaralles:integer) ConjuntCartes

Pre: *n* és un enter positiu.
Post: Crea un nou conjunt de cartes que conté *n* baralles de 52 cartes.

ConjuntCartes (conj:ConjuntCartes) ConjuntCartes

Pre: -
Post: Crea un nou conjunt de cartes còpia de *conj*.

afegirConjunt (conj:ConjuntCartes) void

Pre: -
Post: Afegeix al conjunt actual les cartes contingudes a *conj*. Cal tenir en compte les cartes repetides, és a dir, que es complirà sempre que |a| + |b| = |resultat|.

afegirCarta (carta:Carta) void

Pre: -
Post: Afegeix la carta al conjunt de cartes.

treureCarta (carta:Carta) void

Pre: -
Post: Treu la carta indicada del conjunt de cartes. Novament cal tenir en compte que només es treu una carta.

triarCartaAleatòriament () Carta

Pre: -
Post: Retorna una carta del conjunt triada a l'atzar

obtenirCartes () Set(Carta)

Pre: -
Post: Retorna un vector de cartes que són les cartes que conté el conjunt.

Especificació de la classe d'edició d'estratègies

Classe EditorEstrategia

EditorEstratègia (variables set(set(variable:string, valor:double)),
accions set(accio:string),
accioExtra:bool, cP:CapaPersistencia, idiom:String)
EditorEstratègia
Pre: variables ha de contenir totes les variables que puguin aparèixer a les regles. Accions ha de contenir les accions realitzables. accioExtra ha d'indicar si es desitja editar accions llargues o curtes tal com es defineix a la sintaxi del grup 7. cP ha de retornar la traducció en l'idioma especificat de les frases/paraules presentades a la taula de traduccions.
Post: Crea un nou editor de regles i estratègia.

Id	Mensaje a traducir
ee1	Aceptar
ee2	Cancelar
ee3	Añadir subregla
ee4	Eliminar regla
ee5	- Prioridad
ee6	+ Prioridad
ee7	Nueva regla
ee8	Editor estrategia
ee9	Regla
ee10	Test
ee11	Test acción extra
ee12	Acción
ee13	Prioridad
ee14	Regla incorrecta
ee15	Acción incorrecta
ee16	Regla correcta
ee17	No hay regla
ee18	Acción extra correcta
ee19	Acción extra incorrecta
ee20	Editar regla
ee21	No hay acción extra

Nota: les variables es passen segons el model de regla/estratègia

editaEstratègia (e:Estratègia) Estratègia
Pre: -
Post: Mostra a l'usuari una interfície que li permet editar l'estratègia e i les seves regles. Retorna la nova estratègia (modificada per l'usuari).

Sintaxi de les regles

Les regles han de ser matemàticament correctes. La correctesa d'una regla pot ser comprovada mitjançant la funció esCorrecta de la classe regla. Poden tractar-se tant d'enters com de decimals. Cal tenir en compte que qualsevol operació lògica prendrà qualsevol valor diferent de 0 com TRUE. Les funcions i operadors disponibles sobre la variable 'i' i 'j' són les que segueixen:

- Exponent:	(2 EXP 3)
- Valor Absolut:	VABS(i)
- Negatiu:	-i o -(i)
- Positiu:	+i o +(i)
- Negació:	NOT(i)
- Random entre i i j:	(i RANDOM j)
- And:	i AND j
- Or:	i OR j
- Xor:	i XOR j
- Comparadors	>, <, =, !=, <=, >= i 'comparador' j

Es disposa també de variables i constants.
Variables usades per l'usuari: contindran únicament lletres minúscules i números (com a mínim una lletra).
Constants pròpies: TRUE i FALSE.

Exemple:
(1 +2 / 3 * 2 > 4 AND NOT(VABS(variable1) < variable2)) XOR (variable3 / variable5 -1 != 3)
Limitacions:
No hi ha control de divisió per 0.
Si volem un número negatiu haurem d'expressar-lo entre parèntesis.
L'operador lògic NOT és una funció: la condició a negar haurà d'anar entre parèntesi: NOT(subexpr...) o !(subexpr...)
La regla ha d'estar perfectament jerarquitzada. És a dir, tot i que una construcció del tipus “1 XOR 1 XOR 1” és perfectament correcta matemàticament parlant, haurem d'introduir-la com “(1 XOR 1) XOR 1” o “1 XOR (1 XOR 1)”

Accions:

S'han definit dos tipus d'accions realitzables. Curtes i llargues.
Ambdues consten d'una primera cadena de caràcters sense espais. En cas d'utilitzar accions llargues, l'anterior acció anirà seguida per un espai i a continuació una cadena de caràcters que hauran de ser una regla avaluable

(al aplicar la funció esCorrecta de regla sobre l'esmentada cadena ha de donar com a resultat true).

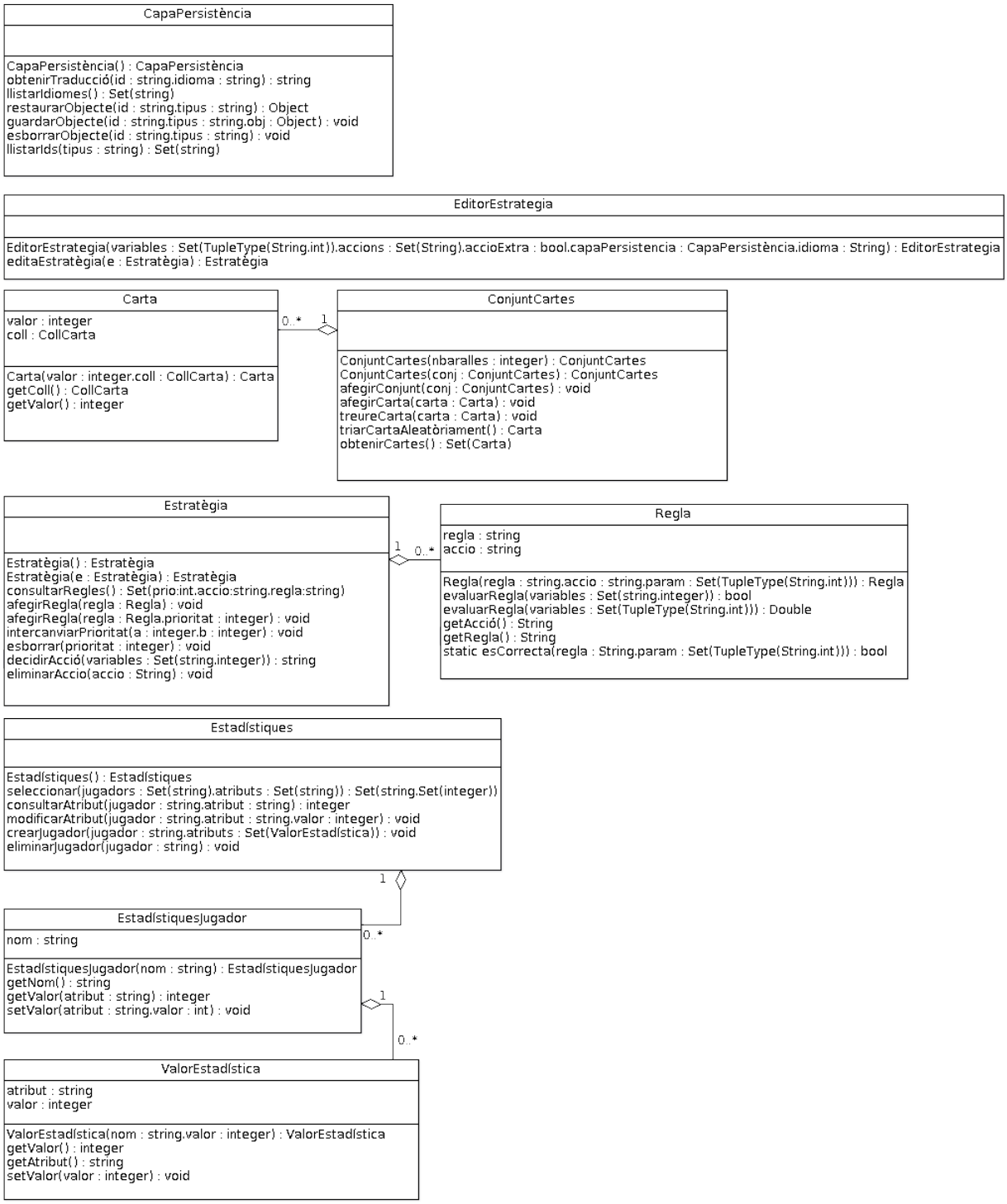
Sintaxi dels idiomes

El sistema d'idiomes de la capa de persistència necessita un espai on escriure les frases que es faran servir així com definir els idiomes.
La sintaxi serà comuna:

Es crearà un fitxer per cada idioma en una carpeta (a definir pel que implementa). Cada fitxer tindrà el següent format.

Nom de l'idioma X
identificador1 Frase 1 en l'idioma X
identificador2 Frase 2 en l'idioma X
untranslated Frase per defecte en cas que no es trobi la buscada!
...

La primera línia del fitxer és reservada per al nom de l'idioma en aquest idioma. A partir d'aquí cada línia és una frase que comença per la paraula que l'identifica i es separa de la frase per un espai. Els identificadors són qualsevol combinació de nombres i lletres que no contenen cap espai en blanc.
És obligatori que existeixi una frase amb identificador “untranslated” la qual serà mostrada en cas de no trobar la frase buscada. Això ajudarà a detectar la falta de traduccions en el sistema i a traduir el programa a nous idiomes.


CapaPersistència
CapaPersistència() : CapaPersistència
obtenirTraducció(id : string.idioma : string) : string
llistarIdiomes() : Set(string)
restaurarObjecte(id : string.tipus : string) : Object
guardarObjecte(id : string.tipus : string.obj : Object) : void
esborrarObjecte(id : string.tipus : string) : void
llistarIds(tipus : string) : Set(string)
EditorEstrategia
EditorEstrategia(variables : Set(TupleType(String.int)).accions : Set(String).accioExtra : bool.capaPersistencia : CapaPersistència.idioma : String) : EditorEstrategia
editaEstratègia(e : Estratègia) : Estratègia
Carta
valor : integer
coll : CollCarta
Carta(valor : integer.coll : CollCarta) : Carta
getColl() : CollCarta
getValor() : integer
0..*
1
ConjuntCartes
ConjuntCartes(nbaralles : integer) : ConjuntCartes
ConjuntCartes(conj : ConjuntCartes) : ConjuntCartes
afegirConjunt(conj : ConjuntCartes) : void
afegirCarta(carta : Carta) : void
treureCarta(carta : Carta) : void
triarCartaAleatòriament() : Carta
obtenirCartes() : Set(Carta)
Estratègia
Estratègia() : Estratègia
Estratègia(e : Estratègia) : Estratègia
consultarRegles() : Set(prio:int.accio:string.regla:string)
afegirRegla(regla : Regla) : void
afegirRegla(regla : Regla.prioritat : integer) : void
intercanviarPrioritat(a : integer.b : integer) : void
esborrar(prioritat : integer) : void
decidirAcció(variables : Set(string.integer)) : string
eliminarAccio(accio : String) : void
1
0..*
Regla
regla : string
accio : string
Regla(regla : string.accio : string.param : Set(TupleType(String.int))) : Regla
evaluarRegla(variables : Set(string.integer)) : bool
evaluarRegla(variables : Set(TupleType(String.int))) : Double
getAcció() : String
getRegla() : String
static esCorrecta(regla : String.param : Set(TupleType(String.int))) : bool
Estadístiques
Estadístiques() : Estadístiques
seleccionar(jugadors : Set(string).atributs : Set(string)) : Set(string.Set(integer))
consultarAtribut(jugador : string.atribut : string) : integer
modificarAtribut(jugador : string.atribut : string.valor : integer) : void
crearJugador(jugador : string.atributs : Set(ValorEstadística)) : void
eliminarJugador(jugador : string) : void
1
0..*
EstadístiquesJugador
nom : string
EstadístiquesJugador(nom : string) : EstadístiquesJugador
getNom() : string
getValor(atribut : string) : integer
setValor(atribut : string.valor : int) : void
1
0..*
ValorEstadística
atribut : string
valor : integer
ValorEstadística(nom : string.valor : integer) : ValorEstadística
getValor() : integer
getAtribut() : string
setValor(valor : integer) : void

Adjudicació de classes als diferents grups

Classe	Usa	Desenvolupa
Regla	Tothom	7.1
Estratègia	Tothom	7.1
EditorEstratègia	Tothom	7.1
CapaPersistència	Tothom	7.2
Estadístiques	Tothom	7.2
ValorEstadística	Tothom	7.2
EstadísticaJugador	Tothom	7.2
Carta	Tothom	7.3
ConjuntCartes	Tothom	7.3