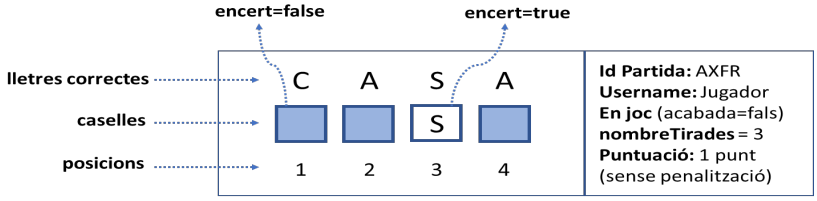
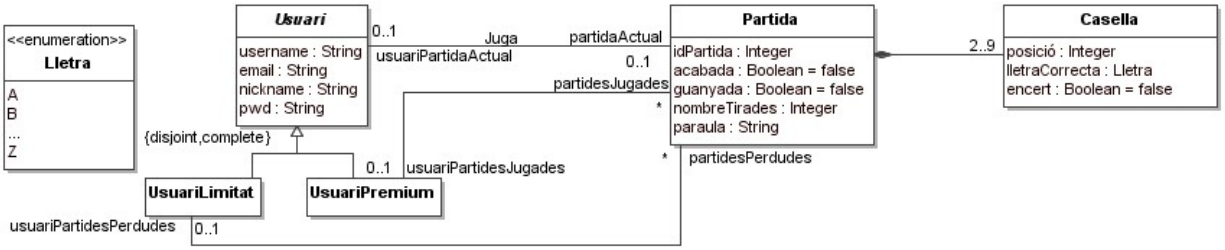
**Primer Control AS – Grup 10** **19-04-21**

**1. [4 punts]** Volem desenvolupar un joc per endevinar paraules. El joc enregistra usuaris amb el seu username, email, nickname, password i la partida actual en joc. Els usuaris poden ser de dos tipus: els premium (que també enregistren totes les partides jugades) i el limitat (que enregistra només les partides perdudes). Cada partida permet a un usuari endevinar una paraula i enregistra el seu identificador, si està acabada, si s'ha guanyat, el nombre de tirades i la paraula a endevinar. El sistema genera automàticament una paraula entre 2 i 9 lletres (no ens preocupem com). El sistema mostrarà a l'usuari un tauler amb caselles tapades, una per a cada lletra de la paraula a endevinar. L'usuari a cada tirada indica una casella (posició) i una lletra. Si la lletra és correcta en aquella posició, s'enregistra l'encert. Si la lletra introduïda és correcta i s'ha endevinat la paraula, s'enregistrarà la partida com acabada i guanyada. Si encara no s'ha endevinat la paraula, i el nombre de tirades és igual a 2 vegades el nombre de lletres de la paraula, s'enregistrarà la partida com acabada i perduda. En cas contrari, l'usuari podrà seguir fent tirades. Inicialment el joc té dues formes per calcular la puntuació d'una partida: sense i amb penalització. La primera dona un punt per cada encert i no aplica penalització per error. La segona dona un punt per cada encert però resta un punt per error (els errors són el nombre de tirades menys el nombre d'encerts). La forma de calcular la puntuació s'assigna quan es crea la partida i es pot canviar en qualsevol moment de la partida. Més endavant afegirem noves formes de calcular la puntuació sempre en funció de valors relacionats amb la partida. A continuació disposeu de la imatge d'una partida en un moment del joc i de l'esquema conceptual del sistema:




R.I. Textuals:
- Claus: (Usuari, username); (Partida, idPartida); (Casella, Partida::idPartida+posició).
- Una partida d'un usuari no pot ser alhora actual i jugada o perduda.
- L'atribut acabada de la classe Partida té com a valor cert si totes les caselles de la partida tenen l'atribut encert igual a cert o si el nombre de tirades és igual a 2 vegades el nombre de lletres de la paraula.
- L'atribut guanyada de la classe Partida té com a valor cert si totes les caselles de la partida tenen l'atribut encert igual a cert i el nombre de tirades és inferior o igual a 2 vegades el nombre de lletres de la paraula.
- L'atribut posició de la classe Casella té un valor entre 1 i el nombre de lletres de la paraula corresponent a la partida. Les posicions de les caselles són consecutius.
- L'atribut lletraCorrecta de la classe Casella té com a valor la lletra de la paraula que està a la posició que indica la casella.
- Altres restriccions no rellevants pel problema.

Suposant navegabilitats dobles, es demana el diagrama de seqüència de l'operació *FerJugada* (mireu les notes avall). Indiqueu en el diagrama de seqüència els singleton, les interfaces i les operacions que són abstractes.

**context** CapaDomini :: FerJugada (idPartida: String, pos: Integer, lletra: Lletra): Integer
**pre** *partidaExisteix*: la partida *idPartida* existeix.
**pre** *posicióCorrecta*: la posició *pos* està entre 1 i el nombre de lletres de la paraula de la partida.
**pre** *posicióAmbLletraNoCorrecta*: l'atribut encert de la posició *pos* és fals (no s'ha encertat la lletra).
**exc** *partidaAcabada*: la partida està acabada.
**post** *comprovaEncertLletra*: si la lletra *lletra* coincideix amb la *lletraCorrecta* de la casella identificada per la posició *pos* llavors *encert* pren per valor cert. Incrementar el nombre de tirades de la partida.
**post** *partidaAcabada*: si amb la lletra *lletra* introduïda a la posició *pos* s'ha encertat la paraula o el nombre de tirades és igual a 2 vegades el nombre de lletres de la paraula llavors l'atribut *acabada* pren per valor cert.
**post** *partidaGuanyada*: si amb la lletra *lletra* introduïda a la posició *pos* s'ha encertat la paraula i el nombre de tirades és inferior o igual a 2 vegades el nombre de lletres de la paraula llavors l'atribut *guanyada* pren per valor cert.
**post** *result*= puntuació de la partida en funció de la forma de puntuació assignada.

**Notes:**
- Podeu suposar l'existència de les operacions *Partida::paraulaEncertada?():Boolean* (retorna cert si la paraula de la partida s'ha encertat i fals en cas contrari), *Partida::nombreEncerts():Integer* (retorna el nombre de lletres encertades de la paraula de la partida) i *String::size(): Integer* (retorna el nombre de caràcters del string).
- Assumim que enregistrar les partides jugades dels usuaris premiums i les partides perdudes dels usuaris limitats quan s'acaba una partida ho fa una altra operació.

**2. [0,5 punts] (Competència transversal)** Explica de forma raonada i breu (màxim 5 línies) com la Capa de Presentació o la pantalla associada pot garantir les precondicions de l'operació anterior.

**3. [5,5 punts]** Hem identificat dos serveis que podem utilitzar per millorar el disseny del nostre joc, un servei d'usuaris (SvUsu) i un servei de missatges (SvMail). Els dos serveis són compartits per **altres** aplicacions. A continuació disposeu d'algunes de les operacions de cada servei:

**context** SvUsu::getTotsUsuaris ():Set (TupleType(username: String, email: String, nom:String, nickname:String))
**post** es retorna l'*username*, l'*email*, *nom* i *nickname* de tots els usuaris que hi ha al servei.

**context** SvUsu::getUsuaris (usus:Set(username:String)): Set (TupleType(username:String, email: String, nom: String, nickname:String))
**exc** usuariNoExisteix: algun dels usuaris indicats al conjunt *usus* no existeix.
**post** es retorna l'*email*, *nom* i *nickname* de tots els usuaris indicats al conjunt *usus*.

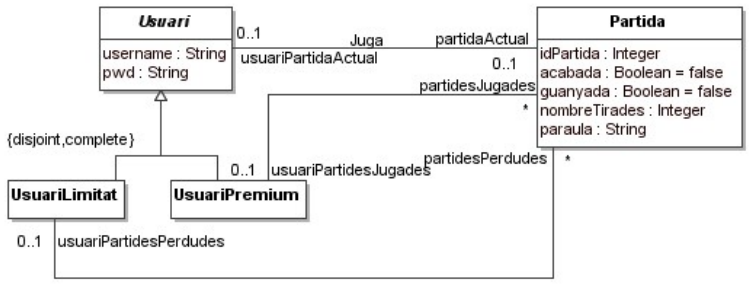
**context** SvUsu :: getUsuari (username: String): TupleType(email: String, nom: String, nickname:String)
**exc** usuariNoExisteix: l'usuari amb l'username *username* no existeix al servei.
**post** es retorna l'*email*, *nom* i *nickname* de l'usuari del paràmetre.

**context** SvMail :: sendMails (emails:Set(String), text:String)
**post** s'envia el *text* indicat al paràmetre a tots els *emails* del conjunt indicat al paràmetre.

**context** SvMail :: sendMail (email: String, text:String)
**post** s'envia el *text* indicat al paràmetre a l'*email* indicat al paràmetre.

**context** SvMail :: sendMailsTuples (d:Set(TupleTyple(email:String, text:String)))
**post** per a cada tupla del conjunt *d*, s'envia el text indicat al camp *text* de la tupla a l'email indicat al camp *email*.

Després d'utilitzar els dos serveis en el nostre sistema, el diagrama de classes resultant (incloent els atributs però no operacions) del fragment del paquet Domain Model de la capa de domini que te a veure amb els usuaris i tota la seva informació ha quedat de la següent manera:




Es demana el diagrama de seqüència de l'operació *puntuacionsJoc* de la capa de domini. Indiqueu clarament en el diagrama de seqüència els singleton, les interfaces i les operacions que són abstractes. **Com a criteris de disseny volem minimitzar el nombre d'invocacions remotes mantenint els criteris habituals de reusabilitat, canviabilitat i minimitzant la redundància de les dades**. A continuació teniu el contracte de la operació:

**context** CapaDomini :: puntuacionsJoc ()
**exc** *noHiHaUsuaris*: el joc no té usuaris.
**post** *enviaPuntuacions:* per a cada usuari del joc s'envia al seu email el text: "L'username " + username de l'usuari + " té una puntuació acumulada de " + puntuació acumulada de l'usuari. La puntuació acumulada d'un usuari és la puntuació de la partida actual, si en té, més: 1) la puntuació de les partides jugades si l'usuari és Premium o bé 2) la puntuació de les partides perdudes si l'usuari és limitat.