Kenken

Grup 41.5

Manual d'usuari

Ermias Valls Mayor : ermias.valls@estudiantat.upc.edu
David Giribet : david.giribet@estudiantat.upc.edu
Nil Beascoechea : nil.beascoechea@estudiantat.upc.edu

ÍNDEX

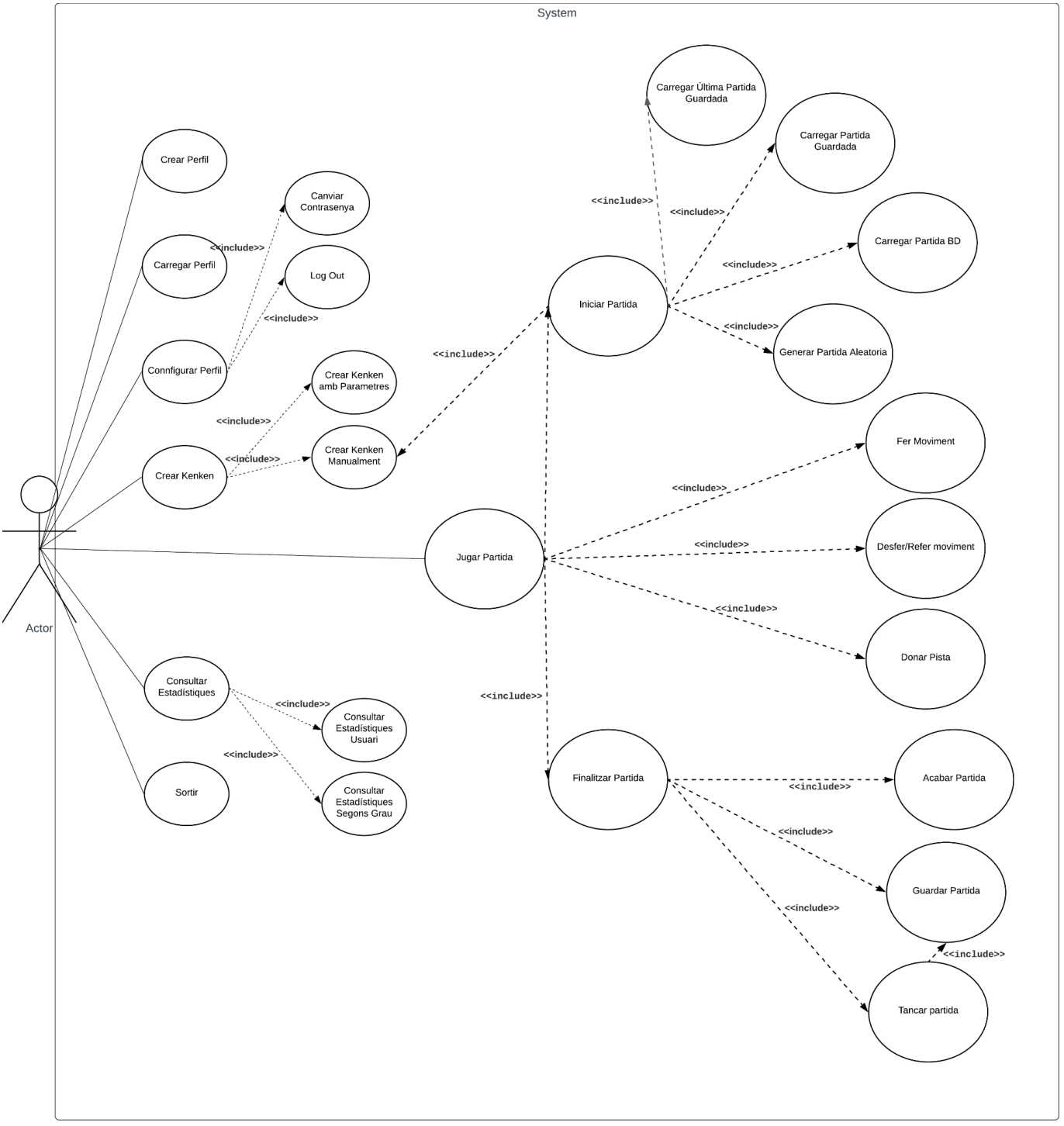
1. Casos d'ús **5**
Diagrama de casos d'ús 5
Descripció de casos d'ús 6
Crear Perfil 6
● Crear perfil: 6
Carregar Perfil 6
● Carregar Perfil: 6
Configuració Perfil 6
● Canviar Contrasenya: 6
● Tancar Sessió: 7
Crear kenken 7
● Crear kenken amb paràmetres: 7
● Crear kenken manualment: 8
Jugar Partida 8
● Fer Moviment: 8
● Desfer/Refer un moviment 8
● Donar Pista: 9
Iniciar Partida: 9
● Carregar Última Partida Guardada: 9
● Carregar Partida Guardada: 9
● Carregar Partida BD: 10
● Generar partida aleatòria: 10
● Generar Partida Introduint Tauler: 10
Finalitzar Partida: 11
● Acabar Partida: 11
● Guardar Partida: 11
● Tancar Partida: 11
Consultar estadístiques: 11
● Consultar Estadístiques Usuari: 11
● Consultar Estadístiques segons Grau: 12
Sortir: 12
● Sortir: 12
Diagrames UML **13**
Descripció de classes **16**
1. Usuari 16
Mètodes: 16
2. Partida 16
3. Tauler 18
4. Regió 20
Atributs: 20

5. Casella 21
6. Ranking 21
7. PartidaAcabada 22
8. Operacions 22
8.1. Suma (1) 23
8.2. Resta (2) 25
8.3. Multiplicació (3) 26
8.4. Divisió (4) 28
8.5. Mòdul (5) 29
8.6. Exponenciació (6) 29
9. CreadoraKenkenParàmetres 30
9.1. CreadorKenkenParam 30
9.2. Board 31
9.3. Cell 31
9.4. Candidate 32
10. SolucionadorKenken 32
Descripció dels controladors **32**
1. CtrlUsuari 32
2. ControladorPartida 33
3. CtrlKenkens 35
4. CtrlDomini 36
5. ControladorRanking 36
Descripció de la presentació **37**
1. CrearKenkenManual 37
1.1. Casella Constructora 37
1.2. CrearKenkenManual 38
1.3. TaulerConstructor 39
2. Partida 41
2.1. ComponentCasella 41
2.2. ComponentLlistaPartides 42
2.3. ComponentTauler 43
2.4. ComponentTimer 43
2.5. ControladorPresentacioPartida 43
2.6. MenuPartida 45
2.7. ObservadorBoto 45
2.8. ObservadorCasella 46
2.9. ObservadorLlista 46
2.10. VistaMenuJugarPartida 46
2.11. VistaPartida 46
2.12. VistaPartidesGuardades 47
3. Ranking 47
3.1. BuscadorUsuari 47
3.2. ComponentSelectorMida 48
3.3. ControladorPresentacioRanking 48

3.4. ObservadorBuscador 49
3.5. ObservadorSelectorMida 49
3.6. PartidesRanking 49
3.7. VistaRankings 49
4. ConfigUsuari 50
5. CreakKenKenParametres 50
6. CtrlPresentacio 51
7. IniciarSessio 52
8. MenuPrincipal 52
9. Registrarse 53
10. Utils 54
Descripció de la persistència **54**
1. ControladorPersistencia 54
2. ControladorPersistenciaPartida 55
3. ControladorPersistenciaTauler 57
4. CtrlUsuariData 58
Mètodes: 58
Estructura de les dades: **59**
1. Usuaris 59
2. Taulers 59
3. Partides 59
a. Partides acabades 59
b. Partides guardades 60
c. Partida en joc/Estat 60

1. Casos d’ús

Diagrama de casos d’ús



Descripció de casos d'ús

Crear Perfil

- Crear perfil:
    - Actor principal: Usuari
    - Precondició: Cap
    - Detonant: L'usuari entra a l'aplicació per primera vegada.
    - Escenari principal/Comportament:
        1. L'usuari informa del nom d'usuari, la contrasenya i confirma la contrasenya.
        2. El sistema valida els valors i crea el nou perfil amb el nom indicat
    - Extensions/possibles errors:
        - Si ja existeix un perfil amb el nom indicat, el sistema informa de l'error
        - Si la contrasenya i la seva confirmació no són iguals, el sistema informa de l'error

Carregar Perfil

- Carregar Perfil:
    - Actor principal: Usuari
    - Precondició: Cap
    - Detonant: L'usuari selecciona l'opció per carregar el seu perfil
    - Escenari principal/Comportament:
        1. El sistema sol·licita al usuari que introdueixi el nom d'usuari i la contrasenya
        2. L'usuari introdueix el nom d'usuari i la contrasenya
        3. El sistema valida les credencials i carrega el perfil de l'usuari si són correctes
    - Extensions/possibles errors:
        - Si el nom d'usuari no es troba a la base de dades, el sistema informa a l'usuari que el perfil no existeix.
        - Si el password proporcionat no coincideix amb el de l'usuari, el sistema informa a l'usuari que les credencials són incorrectes.

Configuració Perfil

- Canviar Contrasenya:
    - Actor principal: Usuari
    - Precondició: L'usuari esta loggejat
    - Detonant: L'usuari selecciona l'opció per canviar la contrasenya al seu perfil.
    - Escenari principal/Comportament:

1. El sistema sol·licita a l'usuari que introdueix la contrasenya actual i la nova contrasenya.
2. L'usuari introdueix la contrasenya actual i la nova contrasenya.
3. El sistema valida la contrasenya actual i canvia la contrasenya del perfil de l'usuari per la nova contrasenya si són correctes.

- Extensions/possibles errors:
  - Si la contrasenya actual proporcionada no coincideix amb la contrasenya del perfil de l'usuari, el sistema informa a l'usuari que la contrasenya actual és incorrecta.

- Tancar Sessió:
  - Actor principal: Usuari
  - Precondició: L'usuari ha iniciat sessió o s'ha registrat prèviament
  - Detonant: L'usuari selecciona l'opció per sortir de la seva sessió actual.
  - Escenari principal/Comportament:
    1. El sistema confirma si l'usuari vol sortir.
    2. El sistema tanca la sessió de l'usuari i redirigeix a la pàgina d'inici de sessió.
  - Extensions/possibles errors:
    - No hi ha errors possibles per al cas d'ús "Log out". Si l'usuari confirma la sortida, el sistema simplement tanca la sessió i redirigeix a l'inici de sessió. Si l'usuari decideix cancel·lar, es manté a la pàgina actual sense tancar la sessió

Crear kenken

- Crear kenken amb paràmetres:
  - Actor principal: Usuari
  - Precondició: L'usuari està registrat, ha iniciat sessióI ha seleccionat creació de kenken
  - Detonant: L'usuari vol crear un tauler de kenken indicant els mínims paràmetres necessaris
  - Escenari principal/Comportament:
    1. L'usuari selecciona la creació de tauler de kenken amb paràmetres
    2. L'usuari indica els paràmetres: mida del tauler, operacions del tauler
    3. El sistema verifica la correctesa dels paràmetres
    4. El sistema crea un tauler correcte que compleixi les restriccions introduïdes
    5. El sistema afegeix a les seves dades el tauler generant-li un identificador
    6. El sistema afegeix a partides creades de l'usuari el tauler
  - Extensions/possibles errors:
    - Mida massa gran: la mida del tauler que es demana és més gran que el màxim

        - Operació inexistent: no existeix l'operació en el sistema

- Crear kenken manualment:
    - Actor principal: Usuari
    - Precondició: L'usuari està registrat i ha iniciat sessió i ha seleccionat creació de kenken.
    - Detonant: L'usuari vol crear un tauler de kenken indicant tots els paràmetres necessaris
    - Escenari principal/Comportament:
        1. L'usuari selecciona la creació de tauler de kenken manual
        2. L'usuari dona tots els paràmetres en el format d'entrada i sortida dels taulers: mida, operacions, peces amb la seva operació i localització
        3. El sistema verifica la correctesa dels paràmetres
        4. El sistema afegeix a les seves dades el tauler generant-li un identificador
        5. El sistema afegeix a partides creades de l'usuari el tauler
    - Extensions/possibles errors:
        - Mida massa gran: la mida del tauler que es demana és més gran que el màxim
        - Operació inexistent: no existeix l'operació en el sistema
        - Peça fora del tauler: les coordenades d'una de les caselles d'una peça estan fora dels marges del tauler
        - Solapament de peces: dues peces tenen la mateixa casella
        - Casella buida: hi ha una casella que no pertany a cap peça
        - Tauler no resoluble: El tauler donat no té cap solució

Jugar Partida

- Fer Moviment:
    - Actor principal: Usuari
    - Precondició: L'usuari està jugant una partida
    - Detonant: L'usuari vol fer un moviment a la partida, és a dir, posar un valor a una casella
    - Escenari principal/Comportament:
        i. L'usuari introdueix a on vol afegir el valor i quin
        ii. Es canvia el valor d'aquella casella
    - Extensions/possibles errors:
        i. La posició no és vàlida.
        ii. El valor no és vàlid.
- Desfer/Refer un moviment
    - Actor principal: Usuari
    - Precondició: L'usuari està jugant una partida, s'ha fet algun moviment.

    - Detonant: L'usuari vol desfer un moviment a la partida, és a dir, tornar el valor de l'última casella modificada al seu estat anterior, o un cop desfet, refer el moviment.
    - Escenari principal/Comportament:
        i. L'usuari tria desfer moviment
        ii. Es canvia el valor de l'última casella modificada a l'anterior
        iii. L'usuari tria si refer el moviment, continuar desfent o una de les altres opcions de Jugar
    - Extensions/possibles errors:
        i. No es pot desfer més.
- Donar Pista:
    - Actor principal: Usuari
    - Precondició: L'usuari està jugant una partida
    - Detonant: L'usuari vol que se li doni una pista sobre la partida actual
    - Escenari principal/Comportament:
        i. Es soluciona el Kenken a partir de l'estat actual i se li emplena aleatòriament una casella que no tingui valor encara.
        ii. Si el Kenken està malament la pista és que ho està i on té un error.
    - Extensions/possibles errors:
        i. Cap

Iniciar Partida:

- Carregar Última Partida Guardada:
    - Actor principal: Usuari
    - Precondició: L'usuari està registrat i ha iniciat sessió; L'usuari té mínim una partida començada però no acabada.
    - Detonant: L'usuari vol continuar l'última partida que va deixar a mitges.
    - Escenari principal/Comportament:
        i. L'usuari selecciona l'opció "Jugar" al menú principal de l'aplicació.
        ii. Després, selecciona l'opció "Carregar Última Partida Guardada".
        iii. Es comença a jugar l'última partida guardada de l'usuari.
    - Extensions/possibles errors:
      L'usuari torna enrere en qualsevol punt del comportament cancel·lant el procés.

- Carregar Partida Guardada:
    - Actor principal: Usuari
    - Precondició: L'usuari està registrat i ha iniciat sessió; L'usuari té mínim una partida començada però no acabada.
    - Detonant: L'usuari vol continuar una partida a mitges.
    - Escenari principal/Comportament:
        i. L'usuari selecciona l'opció "Jugar" al menú principal de l'aplicació.
        ii. Després, selecciona l'opció "Carregar Partida Guardada".
        iii. Mostra a l'usuari una llista de les seves partides guardades.
        iv. L'usuari selecciona una de les seves partides.

        v. Es comença a jugar la partida seleccionada.
    - Extensions/possibles errors:
        i. L'usuari torna enrere en qualsevol punt del comportament cancelant el procés.

- Carregar Partida BD:
    - Actor principal: Usuari
    - Precondició: L'usuari està registrat i ha iniciat sessió; Existeix mínim 1 tauler a la BD.
    - Detonant: L'usuari vol començar una partida nova agafant un tauler ja existent a la BD.
    - Escenari principal/Comportament:
        i. L'usuari selecciona l'opció "Jugar" al menú principal de l'aplicació.
        ii. Després, selecciona l'opció "Carregar Partida BD".
        iii. Mostra a l'usuari una llista de taulers que existeixen a la BD. Pot filtrar perquè només surtin els seus.
        iv. L'usuari selecciona un dels taulers i comença a jugar.
    - Extensions/possibles errors:
        i. L'usuari torna enrere en qualsevol punt del comportament cancel·lant el procés.

- Generar partida aleatòria:
    - Actor principal: Usuari
    - Precondició: L'usuari està registrat i ha iniciat sessió.
    - Detonant: L'usuari vol començar una partida nova sense haver de triar tauler.
    - Escenari principal/Comportament:
        i. L'usuari selecciona l'opció "Jugar" al menú principal de l'aplicació.
        ii. Després, selecciona l'opció "Generar Partida Aleatòria".
        iii. Es selecciona la mida de la partida que vol jugar.
        iv. Es busca a la base de dades un tauler que no hagi jugat, si no en troba en crea un de nou.
        v. L'usuari comença a jugar el tauler.
    - Extensions/possibles errors:
        i. L'usuari torna enrere en qualsevol punt del comportament cancel·lant el procés.
- Generar Partida Introduint Tauler:
    - Actor principal: Usuari
    - Precondició: L'usuari està registrat i ha iniciat sessió
    - Detonant: L'usuari vol començar una partida i introduït el tauler a jugar.
    - Escenari principal/Comportament:
        i. L'usuari selecciona l'opció "Jugar" al menú principal de l'aplicació.
        ii. Després, selecciona l'opció "Generar Partida Introduint Tauler".
        iii. Es va a Crear Kenken Manualment.
        iv. L'usuari comença a jugar el tauler.
    - Extensions/possibles errors:
        i. L'usuari torna enrere en qualsevol punt del comportament cancel·lant el procés.
        ii. L'usuari ja ha jugat un kenken exactament igual.

iii. Es podria extendre a que si el pugui jugar altre cop però no pugui entrar als rankings

Finalitzar Partida:

- Acabar Partida:
    - Actor principal: Usuari
    - Precondició: L'usuari està jugant una partida
    - Detonant: L'usuari creu que l'ha resolt
    - Escenari principal/Comportament:
        i. La partida es corregeix.
        ii. Si està bé s'informa a l'usuari i es guarda i es tanca.
        iii. Si està malament s'informa a l'usuari i es continua jugant.
    - Extensions/possibles errors:
        i. Cap

- Guardar Partida:
    - Actor principal: Usuari
    - Precondició: L'usuari està jugant una partida
    - Detonant: L'usuari vol guardar la partida actual
    - Escenari principal/Comportament:
        i. La partida es guarda
    - Extensions/possibles errors:
        i. Cap

- Tancar Partida:
    - Actor principal: Usuari
    - Precondició: L'usuari està jugant una partida
    - Detonant: L'usuari vol tancar i guardar la partida actual
    - Escenari principal/Comportament:
        i. La partida es guarda
        ii. La partida es tanca
    - Extensions/possibles errors:
        i. Cap

Consultar estadístiques:

- Consultar Estadístiques Usuari:
    - Actor principal: Usuari
    - Precondició: Cap
    - Detonant: L'usuari selecciona l'apartat d'estadístiques
    - Escenari principal/Comportament:
        1. L'usuari insereix el nom d'un usuari

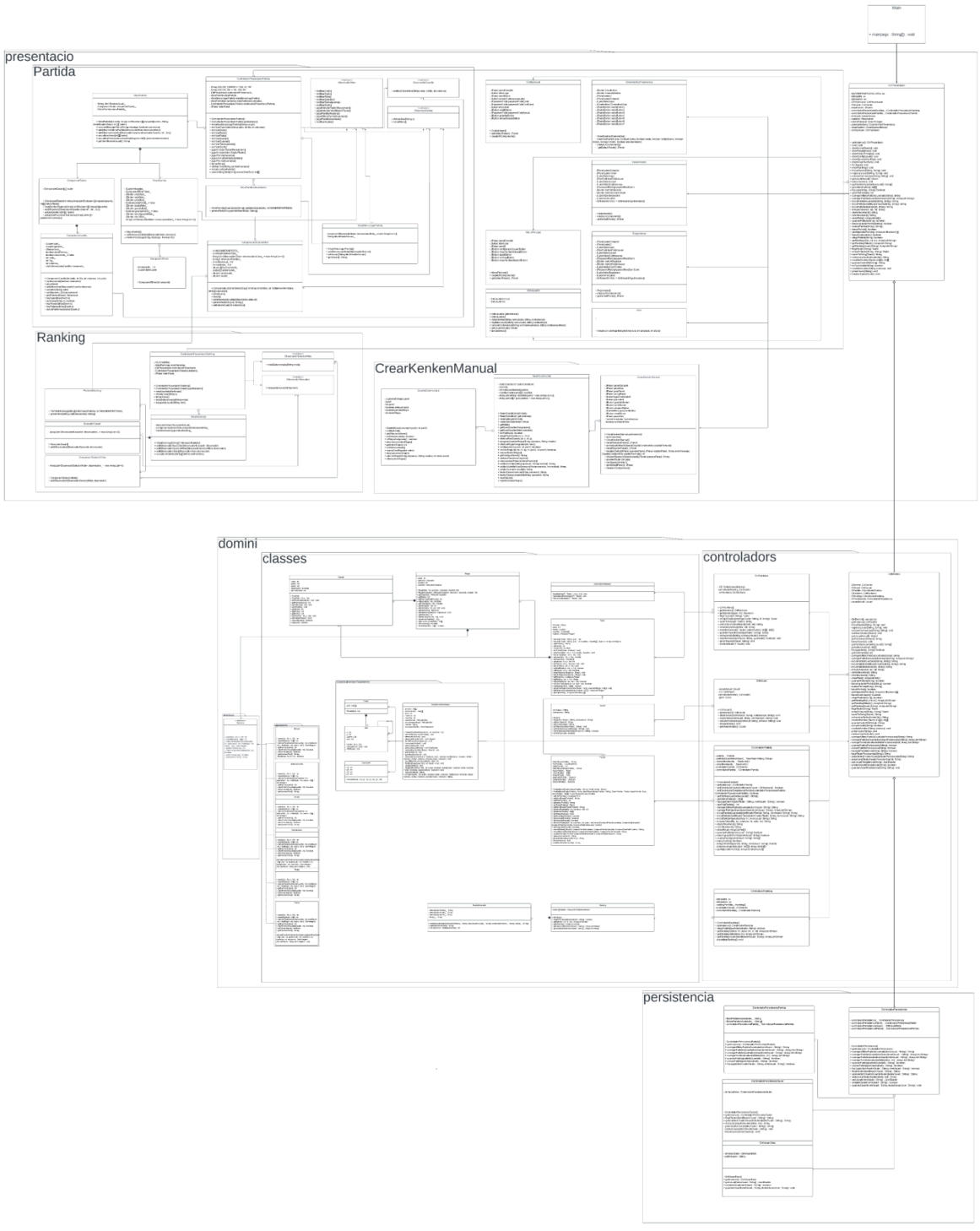
        2. Mostra tots els millors temps de l'usuari seleccionat
    - Extensions/possibles errors:
        - El nom d'usuari inserit no existeix

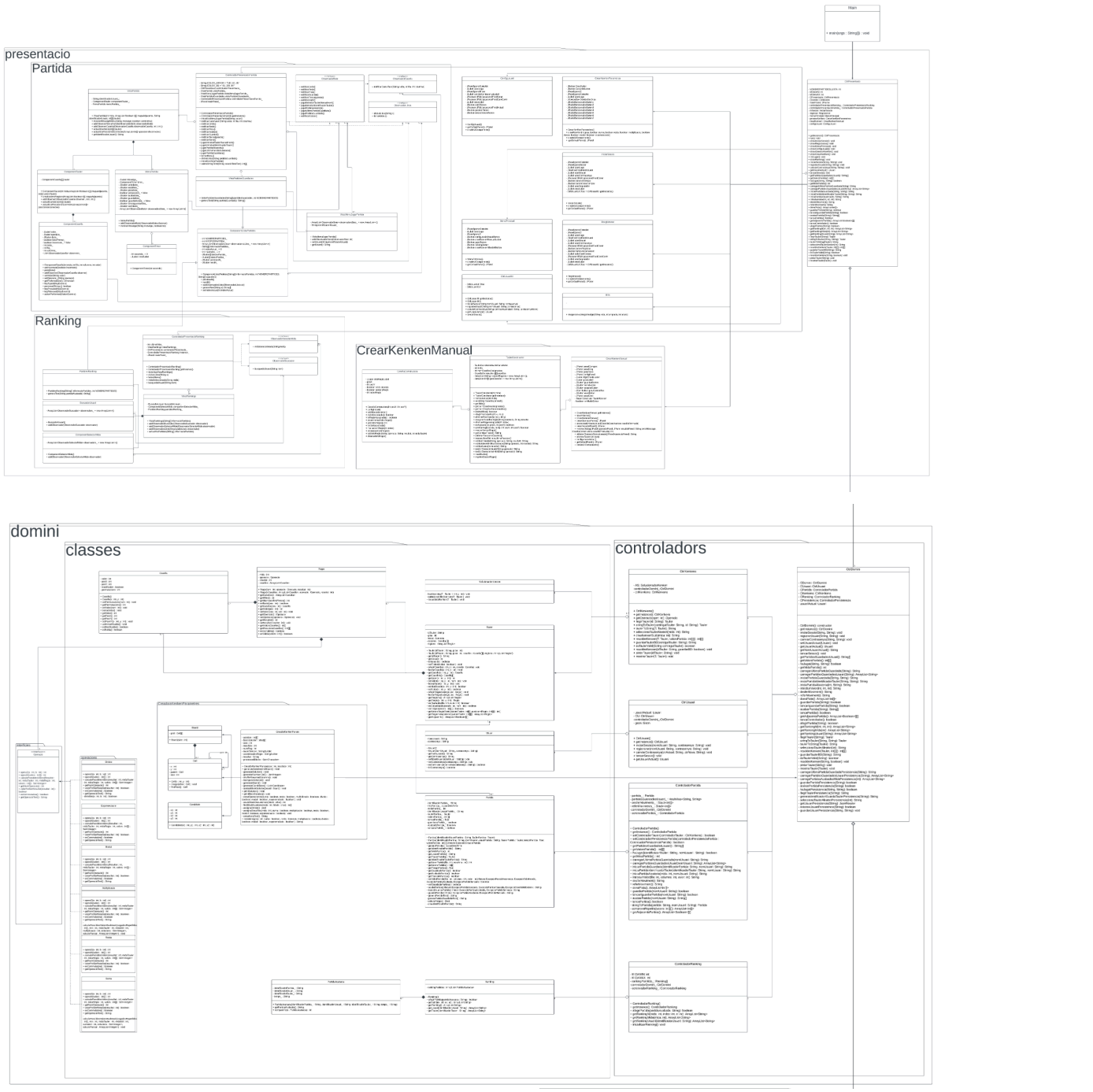
- Consultar Estadístiques segons Grau:
    - Actor principal: Usuari
    - Precondició: Cap
    - Detonant: L'usuari selecciona l'apartat d'estadístiques
    - Escenari principal/Comportament:
        1. L'usuari insereix un possible grau de tauler
        2. Mostra un ranking dels temps per a aquell grau
    - Extensions/possibles errors:
        - El grau inserit no és real

Sortir:

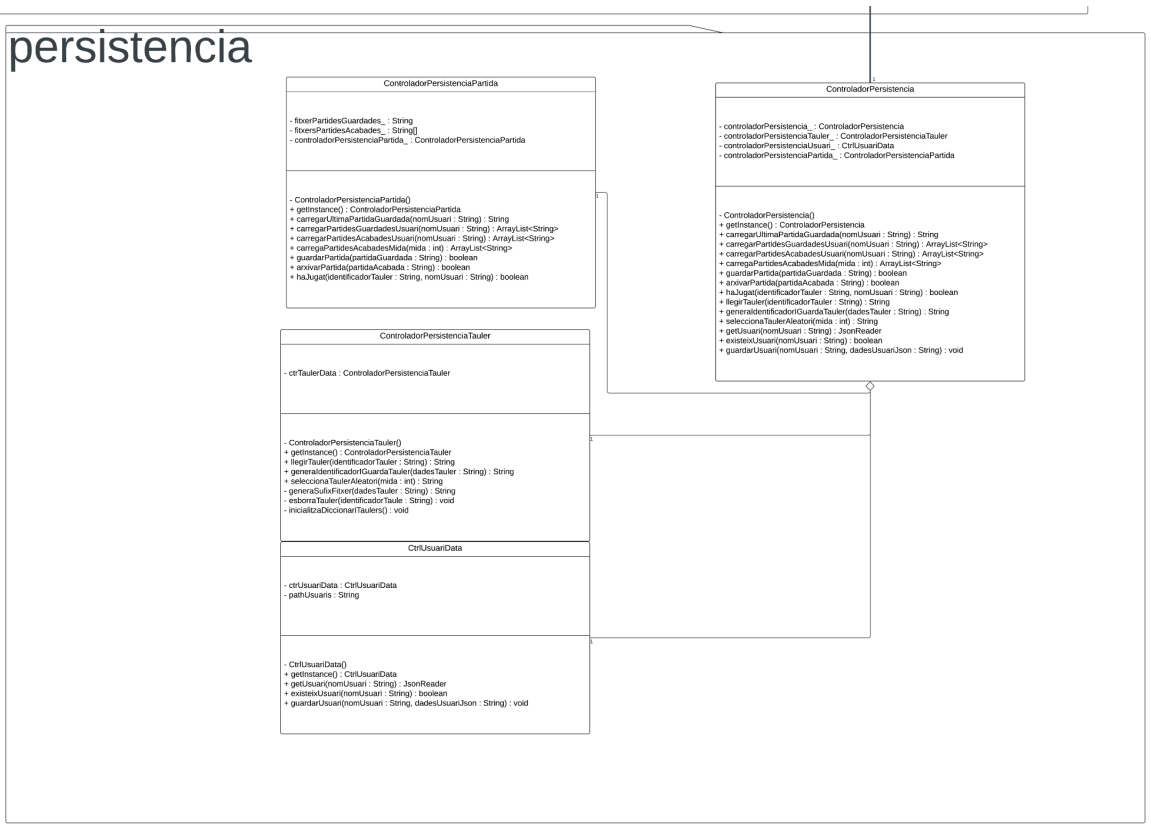
- Sortir:
    - Actor principal: Usuari
    - Precondició: L'aplicació està oberta
    - Detonant: L'usuari vol tancar l'aplicació
    - Escenari principal/Comportament:
        1. L'usuari selecciona l'opció sortir
        2. L'aplicació es tanca sense guardar l'estat
    - Extensions/possibles errors:
        - En cas d'estar en partida el sistema avisa a l'usuari si està segur, que perdrà tot el procediment no guardat

Diagrames UML




Main
+ main(args : String[]) : void
presentacio
Partida
Ranking
CrearKenkenManual
domini
classes
controladors

persistencia



Descripció de classes

1. Usuari

La classe Usuari és la que conté els jugadors de l'aplicació. S'encarrega de totes les gestions dels jugadors, és a dir, crear usuaris nous, fer l'inici de sessió i canviar contrasenya.
Atributs:

- **String nomUsuari:** Nom d'usuari de l'usuari.
- **String contrasenya:** Contrasenya de l'usuari.

Mètodes:

- **Usuari():** Constructora dela classe per defecte.
- **Usuari(String nomUsuari, String contrasenya):** Constructora de la classe amb paràmetres.
- **getNomUsuari(): String:** Retorna el nom d'usuari de l'usuari.
- **getContrasenya(): String:** Retorna la contrasenya de l'usuari.
- **setNomUsuari(String nomUsuari):** Estableix el nom d'usuari de l'usuari.
- **setContrasenya(String contrasenya):** Estableix la contrasenya de l'usuari.
- **esContrasenyaCorrecta(String contrasenya): boolean:** Comprova si la contrasenya donada és igual a la contrasenya de l'usuari.
- **teContrasenya(): boolean:** Comprova si l'usuari té una contrasenya.

2. Partida

La classe Partida representa una partida de Kenken d'un usuari amb un cert tauler. Ve determinada per un identificador únic creat a partir de l'usuari i el moment de creació de la partida.
Conté la data d'inici de la instanciació actual, el tauler de la partida, la seva mida per a tenir accés ràpid a ella, l'identificador de l'usuari que l'ha creada, els valors de les caselles de la partida en un instant, el temps que s'ha estat jugant, i si la partida està guardada, acabada o tancada.
Atributs:

- **-String identificadorPartida_**: És l'identificador de la partida, creat a partir de l'usuari i el moment de creació de la partida. El format és "identificadorUsuariPartida:yyyy-mm-ddThh:mm:ss".
- **-LocalDateTime iniciPartida_**: Data d'inici de la instanciació actual. S'ha optat per el format LocalDateTime ja que permet obtenir dates de forma

exacta i junt amb la biblioteca de duration, calcular temps també de forma exacte.

- **-int grauPartida_**: Grau del tauler de la partida, és a dir la N d'un tauler NxN.
- **-String identificadorUsuariPartida_**:Identificador de l'usuari que ha creat la partida.
- **-TaulerJoc taulerPartida_**: Tauler de la partida.
- **-int[][] valorsPartida_**: Valors de les caselles de la partida en un instant com a una matriu d'enters. S'ha optat per una matriu d'ints ja que la mida de la partida no ha de variar en cap moment i aleshores té accés constant als valors.
- **-float tempsPartida_**: Temps que s'ha estat jugant. Només calculat en punts concrets per a estalviar actualitzacions periòdiques. És un float ja que té més exactitud i millor parsing amb el format LocalDateTime de iniciPartida_.
- **-boolean guardadaPartida_**: Indica si la partida està guardada. Es guardarà a l'arxiu pertinent de partides guardades.
- **-boolean acabadaPartida_**:Indica si la partida està acabada. Només es pot acabar una partida si està correctament resolta. No es poden fer més modificacions a la partida i es guardarà a l'arxiu pertinent de partides acabades.
- **-boolean tancadaPartida_**: Indica si la partida està tancada. Només es pot tancar una partida si està acabada. No es poden fer més modificacions a la partida.

Mètodes:

- **+Partida(String, TaulerJoc)**: Crea una nova partida amb un identificador d'usuari i un tauler donats. Utilitzada en crear una partida totalment nova.
- **+Partida(String, TaulerJoc, int, int[][])**: Crea una nova partida amb un identificador d'usuari, un tauler, un temps i uns valors donats. Utilitzat per carregar una partida guardada.
- **+getiniciPartida(): LocalDateTime**: Retorna la data d'inici de la instanciació actual.
- **+getIdentificadorPartida(): String**: Retorna l'identificador de la partida.
- **+getGrauPartida(): int**: Retorna el grau del tauler de la partida.
- **+getUsuariPartida(): String**: Retorna l'identificador de l'usuari que ha creat la partida.
- **+getTaulerPartida(): Tauler**: Retorna el tauler de la partida.
- **+getIdentificadorTaulerPartida(): String**: Retorna l'identificador del tauler de la partida.
- **+getValorPartida(int, int): int**: Retorna el valor d'una casella de la partida.
- **+getValorsPartida(): int[][]**: Retorna els valors de les caselles de la partida en un instant.
- **+getTempsPartida(): int**: Retorna el temps que s'ha estat jugant. Per no haver d'anar actualitzant el temps periòdicament, es calcula cada vegada que es necessita, és a dir, quan es vol guardar, acabar o tancar la partida.
- **+getGuardadaPartida(): boolean**: Retorna si la partida està guardada.
- **+getAcabadaPartida(): boolean**: Retorna si la partida està acabada.
- **+getTancadaPartida(): boolean**: Retorna si la partida està tancada.

- **+setValorPartida(int, int, int): boolean**: Estableix un valor a una casella de la partida.
- **+setGuardadaPartida(): boolean**: Posa la partida com a guardada
- **+acabaPartida(): String**: Acaba la partida si està correctament resolta. El format de la informació de la partida acabada és:
  Identificador de la partida
  Identificador de l'usuari
  Identificador del tauler
  Temps total de la partida
  Grau del tauler
  Si ha estat guardada o no.
  Per tant, cada partida acabada ocupa 6 línies.
- **+tancalGuardaPartida(): String**: Tanca i guarda la partida. Genera un text amb el format descrit a guardaPartida().
- **+guardaPartida(): String**: Guarda la partida. Per a encapsular una partida guardada, el format és:
  Identificador de la partida
  Identificador del tauler
  Temps total de la partida
  Grau del tauler
  Valors de les caselles de la partida separats per espais en files i per salts de línia en columnes.
  És a dir ocuparà 4 + Grau línies.
- **+generaPartidaText(): String**: Genera un text amb la informació de l'estat dels valors de la partida. El format és:
  'x" 'x" 'x" ... 'x"\n', ..., 'x" 'x" 'x" ... 'x"\n'
  On hi haurà tantes x a cada fila com el grau de la partida i tantas files com el grau de la partida.
- **-generaPartidaGuardadaText(): String**: Funció privada auxiliar per a generar un text amb la informació de la partida. Genera un text amb el format descrit a guardaPartida().
- **-calculaTemps(): int**: Funció privada auxiliar per a calcular el temps total de la partida. Ho calcula a partir de la data d'inici de la sessió de la partida actual i el temps de la crida, sumant el temps acumulat.
- **-crealdentificadorPartida(): String**: Funció privada auxiliar per a crear l'identificador de la partida. L'identificador de la partida és únic i es crea a partir de l'usuari i el moment de creació de la partida. El format és "identificadorUsuariPartida:yyyy-mm-ddThh:mm:ss".

3. Tauler

La classe Tauler conté tots els taulers de KenKen de la base de dades. Conte un id que l'identifica i un grau, que es la dimensió del mateix. La seva funció és gestionar totes les dades dels taulers.
Atributs:

- **String idTauler:** Identificador únic del tauler.
- **int grau:** Grau del tauler, que indica la seva mida.
- **boolean trobat:** Indica si s'ha trobat una solució per al tauler.
- **Casella[][] caselles:** Matriu de caselles que formen el tauler.
- **ArrayList<Regio> regions:** Llista de regions que formen el tauler.

Mètodes:

- **Tauler(String idTauler, int grau):** Constructor de la classe Tauler. Crea un nou tauler amb un identificador i un grau especificats.
- **Tauler(String idTauler, int grau, Casella [][] caselles, ArrayList<Regio> regions):** Constructor de la classe Tauler. Crea un nou tauler amb un identificador, un grau, una matriu de caselles i una llista de regions especificats.
- **getIdTauler(): String:** Obté l'identificador del tauler.
- **getGrau(): int:** Obté el grau del tauler.
- **teSolucio(): boolean:** Comprova si el tauler té una solució.
- **setTrobat(boolean trobat):** Estableix si s'ha trobat una solució per al tauler.
- **afegirCasella(int x, int y, Casella casella):** Afegeix una casella al tauler en una posició especificada.
- **getCasella(int x, int y): Casella:** Retorna la casella situada a la posició (x, y) del tauler.
- **getCaselles(): Casella [][]:** Obté la llista de caselles del tauler.
- **getValor(int x, int y): int:** Obté el valor de la casella a la posició (x, y) del tauler.
- **setValor(int x, int y, int num):** Estableix el valor de la casella a la posició (x, y) del tauler.
- **borrarValor(int x, int y):** Borra el valor de la casella a la posició (x, y) del tauler.
- **esModificable(int x, int y): boolean:** Verifica si la casella a la posició (x, y) del tauler és modificable.
- **esBuida(int x, int y): boolean:** Verifica si la casella a la posició (x, y) del tauler està buida.
- **afegirRegioJoc(Regio regioJoc):** Afegeix una regió de joc a la llista de regions.
- **borrarRegioJoc(Regio regioJoc):** Esborra una regió de joc de la llista de regions.
- **getRegions(): ArrayList<Regio>:** Retorna la llista de regions de joc.
- **getRegio(int x, int y): Regio:** Retorna la regió de joc que conté la casella amb les coordenades x, y.
- **esFilaValida(int fila, int num): boolean:** Comprova si un número és vàlid per a una fila donada.
- **esColumValida(int colum, int num): boolean:** Comprova si un número és vàlid per a una columna donada.
- **corretgeix(int[][] valors): boolean:** Comprova si els valors donats són vàlids per a cada fila i columna del tauler.
- **getValorsRegioMatriu(int [][] valorsTauler, int [][] posicionsRegio): int[]:**
- **getRegionsIncorrectes(int [][] valorsTauler): ArrayList<Regio>:**

- **getAdjacents(): ArrayList<Boolean>[][]:** Per a cada posicio del tauler retorna un vector de Booleans amb 4 valors, top left bottom i right que és true si la casella de la posicio que indica pertany a la regió.

4. Regió

Proporciona una estructura per gestionar una regió en un joc, amb funcions per accedir i manipular les caselles i els seus valors associats.

Atributs:

- **int mida:** Mida de la regió. Indica el nombre de caselles que la formen.
- **Operacio operacio:** Operació que s'ha de realitzar amb els valors de les caselles de la regió per obtenir el resultat. La operacio pot der de : Suma, Resta, Multiplicacio, Divisio, Modul, Exponenciacio.
- **int resultat:** Resultat que s'ha d'obtenir en realitzar l'operació especificada amb els valors de les caselles de la regió.
- **ArrayList<Casella> caselles:** Llista de caselles que formen la regió.

Mètodes:

- **Regio(int tam, Operacio operacio, int resultat):** Constructor de la classe Regio. Crea una nova regió amb una mida, operació i resultat especificats.
- **Regio(ArrayList<Casella> vCaselles, Operacio operacio, int resultat):** Constructor de la classe Regio. Crea una nova regió amb un conjunt de caselles, operació i resultat especificats.
- **getCaselles(): ArrayList<Casella>:** Retorna les caselles de la regió.
- **getMida(): int:** Retorna la mida de la regió.
- **getCasella(int pos): Casella:** Comprova si una casella de la regió està buida.
- **getValor(int pos): int:** Retorna el valor d'una casella de la regió.
- **setValor(int pos, int val):** Estableix el valor d'una casella de la regió.
- **getOperacio(): Operacio:** Retorna l'operació de la regió.
- **setOperacio(Operacio operacio):** Estableix l'operació de la regió.
- **getResultat(): int:** Retorna el resultat de la regió.
- **seResultat(int resultat):** Estableix el resultat de la regió.
- **getValorsCaselles(): int[]:** Retorna els valors de les caselles de la regió.
- **getPosicionsCaselles(): int[][]:** Retorna les posicions de les caselles de la regió.
- **esCompleta(): boolean:** Comprova si la regió està completa.
- **esValida(int[] valors): boolean:** Comprova si la regió és vàlida segons els valors proporcionats.

5. Casella

La classe Casella conte un valor, el número que s'introdueix a la casella i dos valors de posició (posicio_X i posicio_Y).
Atributs:

- **int valor:** Valor de la casella. Aquest valor pot ser qualsevol enter entre 0 i el grau del tauler. Un valor de 0 indica que la casella està buida.
- **int posX:** Coordenada x de la casella en el tauler. Aquest valor és un enter entre 1 i el grau del tauler.
- **int posY:** Coordenada y de la casella en el tauler. Aquest valor és un enter entre 1 i el grau del tauler.
- **boolean modificable:** Indica si la casella és modificable o no. Si es true, el valor de la casella pot ser canviat. Si és false, el valor de la casella no pot ser canviat.

Mètodes:

- **Casella():** Constructor per defecte. Inicialitza la cel·la amb la posició (-1, -1).
- **Casella(int x, int y):** Constructor amb posició. Inicialitza la cel·la amb la posició donada i estableix el seu valor a 0 i modificable a true.
- **setValor(int num):** Estableix el valor de la cel·la.
- **borrarValor():** Esborra el valor de la cel·la (l'estableix a 0).
- **getValor(): int:** Retorna el valor de la cel·la.
- **getPosX(): int:** Retorna la posició x de la cel·la.
- **getPosY(): int:** Retorna la posició y de la cel·la.
- **setPosXY(int x, int y):** Estableix la posició de la cel·la.
- **setInmodificable ():** Estableix la cel·la com a no modificable.
- **esModificable(): boolean:** Retorna si la cel·la és modificable o no.
- **esBuida(): boolean:** Retorna si la cel·la està buida o no.

6. Ranking

Classe que representa el ranking de les partides acabades i avaluables
Una partida és avaluable si s'ha solucionat correctament sense guardar-la ni utilitzar cap pista.
Un ranking és una llista de les partides acabades i avaluables ordenades per temps.

Atributs:

- **ArrayList<PartidaAcabada> rankingPartides:** Llista de les partides acabades, utilitzant el format de la classe PartidaAcabada, a l'apartat 7 del document. S'ha optat per utilitzar una ArrayList ja que el ranking és de mida variable, amb aquesta estructura de dades es poden afegir fàcilment elements amb l'add().

Mètodes:

- **Ranking():** Constructora de la classe.

- **afegirPartida(String partidaAcabada): boolean:** Afegeix una partida acabada al ranking. El ranking es manté ordenat per temps.
- **getN(int index, int n): ArrayList<String>:** Retorna les N partides amb millor temps a partir de l'índex donat.
- **getRanking(): ArrayList<String>:** Retorna el ranking complet de les partides.
- **getUsuari(String identificadorUsuari): ArrayList<String>:** Retorna les partides d'un usuari donat.
- **getTauler(String identificadorTauler): ArrayList<String>:** Retorna les partides d'un tauler donat.

7. PartidaAcabada

Classe que representa una partida acabada i avaluable.
Atributs:

- **-String IdentificadorPartida_** : String que conté l'identificador de la partida acabada.
- **-String identificadorUsuari_ :** String que conté l'identificador de l'usuari de la partida acabada.
- **-String identificadorTauler_** : String que conté l'identificador del tauler de la partida acabada
- **-String temps_** : String que conté el temps de resolució de la partida acabada. S'ha optat per utilitzar-ho en format String i no float ja que facilitava la comunicació entre capes ja que les partides acabades es llegeixen com String per la capa de persistencia.

Mètodes:

- **+ PartidaAcabada(identificadorPartida_ : String, identificadorUsuari_ : String, identificadorTauler_ : String, temps_ : String) :** Constructora de la classe, posa a cada atribut el valor introduït com a paràmetre.
- **+ getPartidaAcabada() : String** : Retorna la representació d'una partida acabada com un text amb el format:
  identificadorPartida_ + " " + identificadorUsuari_ + " " + identificadorTauler_ + " " + temps_
- **+ compareTo(o : PartidaAcabada) : int** : Compara dues partides acabades pel temps de resolució. Retorna un valor negatiu si aquesta partida té un temps menor, un valor positiu si aquesta partida té un temps major, 0 si tenen el mateix temps.

8. Operacions

Com el seu propi nom indica, aquestes són les classes que s'encarreguen de cada una de les operacions possibles en un KenKen.

8.1. Suma (1)

La classe Suma és la implementació de la interfície Operació que realitza els càlculs mitjançant la suma d'enters.
Mètodes:

- **+opera2(a : int, b : int) : int**: realitza la suma de dos enters i retorna el resultat.
- **+operaN(valors : int[]) : int**: realitza la suma de n enters donats en un vector i retorna la suma total.
- **+calculaPossiblesValors(resultat : int, midaTauler : int, midaRegio : int, valors : int[]) : Set<Integer>**: calcula totes les possibles solucions de dos enters d'1 a la mida del tauler que sumats donin el resultat introduït. Retorna els valors únics d'1 a midaTauler que apareixen en alguna de les solucions calculades.
- **+getNumOperacio() : int**: retorna el número de l'operació, que en aquest cas és 1.
- **-valorPotSerResultat(int resultat): boolean:** Retorna si el valor pot ser resultat de la suma.
- **-esCommutativa()): boolean:** Retorna si l'operació de suma és commutativa.
- **-String getOperacioText():** Retorna el símbol de suma.
- **-calculaPossiblesValorsBacktrack(vegadesRepetibles : int[], min : int, midaTauler : int, midaUtil : int, sumatori : int, solucions : Set<Integer>, solucioParcial : ArrayList<Integer>) : void**: és una funció auxiliar per a calcular les possibles solucions de la suma mitjançant backtracking. Va afegint els valors trobats al seu paràmetre solucions
- **Explicació de l'algorisme principal calculaPossiblesValors():**
  Variables:
  resultat: El resultat del qual es volen trobar les possibles solucions que sumades donin aquell valor.
  midaTauler: indica el nombre de valors possibles que poden tenir les caselles així com el nombre de files i columnes. És a dir els valors poden ser de 1 a midaTauler, i el tauler te midaTauler x midaTauler caselles.
  midaRegió: mida de la regió de la qual es vol calcular els possibles valors. Les caselles buides d'aquesta regió indiquen la mida de la solució que estem buscant. Per exemple, en una regió de mida 3 buida es buscarien 3 possibles valors que sumats donin el *resultat.*
  valors: Un vector de ints que indica caselles no buides de la regió, és a dir que el resultat serà aquests valors sumats amb els possibles nombres trobats.
  Funcionament:
  Primer de tot, si el nombre de valors inicials introduït és superior o igual a la mida de la regió es llançarà una excepció, ja que no té sentit calcular-ho, es tracta d'un error.
  Seguidament calcula el màxim de possibles repeticions d'un número en una regió, que seria si aquesta tingués forma d'escala, així no calcularà

possibilitats que mai es podrien donar en el context del joc, on no es poden repetir valors en columnes i files. Es crea un vector de la mida del tauler, amb el nombre de repeticions possibles per a cada valor. Per exemple el nombre de repeticions possible del 1 estaria a vegadesRepetible[0] (n-1).
Després es redueix el resultat a trobar i la mida de la solució restant la solució amb els valors no buits de la regió. Si no són valors entre 1 i midaTauler llança excepció, si ho són, redueix la mida de la solució (*midaUtil*), el nombre de repeticions d'aquell valor, i el resultat a trobar, restant-lo pel valor (variable *sumatori*). Finalment s'entra al backtrack per a trobar les possibles solucions.

Backtrack:

Variables*:*

vegadesRepetible:int[]: La variable vegadesRepetible de la funció principal

min:int: El valor mínim que es pot posar en aquesta iteració del backtracking, la seva funció és reduïr l'espai de cerca, i no repetir valors ja posats.
Per exemple, que sumatori = 4 en una midaTauler de 3, i midaUtil de 3, hi ha les possibles solucions 1+1+2, 2+1+1, 1+2+1. Posant aquesta variable mínim, s'aconsegueix que primer es tinguin en compte les solucions on apareix l'1 (1+1+2), i després ja no es pugui tornar a posar, evitant solucions repetides. És clar que això només es pot tenir en compte en operacions commutatives.

midaTauler:int: La variable midaTauler de la funció principal

midaUtil:int: La mida de la solució a buscar després d'haver tingut en compte les caselles no buides de la regió.

sumatori:int: El resultat que es busca

solucions:Set<Integer>: Un HashSet amb els valors trobats fins el moment que satisfan la premisa. S'ha triat un HashSet perquè, al buscar valors únics, proporciona l'accés més ràpid (O(1)).

solucioParcial:ArrayList<Integer>: Vector amb la solució trobada fins el moment, fins al cas base no se sabrà si és vàlida.

El cas base d'aquest algorisme de backtracking és que la solucioParcial tingui la mida que es busca (*midaUtil*), i que el sumatori dels seus elements sigui la de *sumatori*, això és que *sumatori* sigui igual a 0, ja que en cada backtrack es resta per l'últim valor afegit a la solució parcial. Si es verifica que la *solucioParcial* és correcta, s'afegeixen els seus valors a *solucio*.
El cas recursiu es fa en un for, que va de min a midaTauler (inclosos els dos), i per a cada i de la iteració prova si encara es pot posar (vegadesRepetible > 0), i si pot restar a *sumatori* sense baixar de 0 (i <= sumatori). Si així és, s'afegeix a la solucioParcial, es resta 1 a vegadesRepetible[i-1], i es fa backtracking amb aquella *solucioParcial*, la i com a min, les noves *vegadesRepetible*, la *midaUtil* i *sumatori-i* com a nou resultat a buscar.
Finalment es restableix l'estat previ al backtrack eliminant el valor introduit a la *solucioParcial* i sumant 1 a les *vegadesRepetibles.*

8.2. Resta (2)

És la implementació de la interfície Operacio que realitza els càlculs mitjançant l'operació de resta d'enters.
Mètodes:

- **+opera2(int a, int b) : int**: realitza la resta de dos enters independentment de l'ordre i retorna el valor absolut de la resta.
- **+operaN(int[] valors) : int**: realitza la resta de dos enters independentment de l'ordre donats com a vector. Retorna el valor absolut de la resta de tots els enters del vector.
- **+calculaPossiblesValors(int resultat, int midaTauler, int midaRegio, int[] valors) : Set<Integer>**: calcula totes les possibles solucions de dos enters d'1 a la mida del tauler que restats donin el resultat introduït. Retorna els valors d'1 a midaTauler que apareixen en alguna de les solucions calculades. Llança una excepció si el vector té igual o més de dos valors o la regió té més de dues caselles, o si algun dels valors és 0.
- **-getNumOperacio() : int**: retorna el número de l'operació, que en aquest cas és 2.
- **-valorPotSerResultat(int resultat): boolean:** Retorna si el valor pot ser resultat de l'operació de resta.
- **-esCommutativa(): boolean:** Retorna si l'operació és commutativa.
- **-getOperacioText(): String:** Retorna el símbol de resta.
- **Explicació de l'algorisme principal calculaPossiblesValors():**
  Variables:
  resultat: El resultat del qual es volen trobar les possibles solucions que restades donin aquell valor.
  midaTauler: indica el nombre de valors possibles que poden tenir les caselles així com el nombre de files i columnes. És a dir els valors poden ser de 1 a midaTauler, i el tauler te midaTauler x midaTauler caselles.
  midaRegió: mida de la regió de la qual es vol calcular els possibles valors. Les caselles buides d'aquesta regió indiquen la mida de la solució que estem buscant. Per exemple, en una regió de mida 2 buida es buscarien 2 possibles valors que restat donin el *resultat.*
  valors: Un vector de ints que indica caselles no buides de la regió, és a dir que el resultat serà aquests valors restats amb els possibles nombres trobats.
  Funcionament:
  Per la definició de l'operació de resta en el kenken, que només permet dos valors, aquest algorisme no precisa de backtracking, ja que els resultats són deterministes des d'un primer moment.
  Primer de tot, si la regió té mida diferent de 2, no pot ser una resta, i si el nombre de valors inicials introduït és superior o igual a la mida de la regió es llançarà una excepció, ja que no té sentit calcular-ho, es tracta d'un error.
  Si el resultat no és 0 es retorna el Set buit (ja que la solució aleshores seria el mateix valor que valors[0] i per la no repetició en fila i columna en una resta és impossible que ho sigui)

Això deixa dos casos possibles, o valors és buit, o valors té 1 element.
Cas 1 valor: Es comprova que l'element estigui entre 1 i midaTauler, aleshores els possibles valors que pot haver-hi (després de comprovar que estan dins els permesos) serien valors[0]-resultat i valors[0] + resultat. S'afegeixen aquests valors a la solució. Per exemple si el resultat fos 3, i el valor que hi ha a valors[0] fos 6, els nombres serien 9 i 3, ja que 9-6=3, i 6-3=3.
Cas 0 valors: En aquest cas es fa un for de i=1 a midaTauler i 'safegeixen els dos valors *i*, *i*+resultat si es compleix que *i*+resultat està dins els valors permesos (*i* sempre ho estarà).

8.3. Multiplicació (3)

És la implementació de la interfície Operació que realitza els càlculs mitjançant l'operació de multiplicació d'enters.
Mètodes:

- **+opera2(int a, int b) : int**: realitza la multiplicació de dos enters i la retorna.
- **+operaN(int[] valors) : int**: realitza la multiplicació de n enters donats en un vector. Retorna la multiplicació de tots els enters del vector.
- **+calculaPossiblesValors(int resultat, int midaTauler, int midaRegio, int[] valors) : Set<Integer>**: calcula totes les possibles solucions de dos enters d'1 a la mida del tauler que multiplicats donin el resultat introduït. Retorna els valors únics d'1 a midaTauler que apareixen en alguna de les solucions calculades
- **+getNumOperacio() : int**: retorna el número de l'operació, que en aquest cas és 3.
- **-valorPotSerResultat(int resultat): boolean:** Retorna si el valor pot ser resultat de la multiplicació.
- **-esCommutativa(): boolean:** Retorna si l'operació és commutativa.
- **-getOperacioText(): String:** Retorna el símbol de multiplicació.
- **-calculaPossiblesValorsBacktrack(int[] vegadesRepetibles,int min, int midaTauler, int midaUtil, int multiplicacio, Set<Integer> solucions, ArrayList<Integer> solucioParcial) : void**: és una funció auxiliar per a calcular les possibles solucions de la multiplicació mitjançant backtracking. Va afegint els valors trobats al seu paràmetre solucions.
- **Explicació de l'algorisme principal calculaPossiblesValors():**
  Variables:
  resultat: El resultat del qual es volen trobar les possibles solucions que multiplicades donin aquell valor.
  midaTauler: indica el nombre de valors possibles que poden tenir les caselles així com el nombre de files i columnes. És a dir els valors poden ser de 1 a midaTauler, i el tauler te midaTauler x midaTauler caselles.
  midaRegió: mida de la regió de la qual es vol calcular els possibles valors. Les caselles buides d'aquesta regió indiquen la mida de la solució que estem

buscant. Per exemple, en una regió de mida 3 buida es buscarien 3 possibles valors que multiplicats donin el *resultat.*
valors: Un vector de ints que indica caselles no buides de la regió, és a dir que el resultat serà aquests valors multiplicats amb els possibles nombres trobats.
Funcionament:
Primer de tot, si el nombre de valors inicials introduït és superior o igual a la mida de la regió es llançarà una excepció, ja que no té sentit calcular-ho, es tracta d'un error.
Seguidament calcula el màxim de possibles repeticions d'un número en una regió, que seria si aquesta tingués forma d'escala, així no calcularà possibilitats que mai es podrien donar en el context del joc, on no es poden repetir valors en columnes i files. Es crea un vector de la mida del tauler, amb el nombre de repeticions possibles per a cada valor. Per exemple el nombre de repeticions possible del 1 estaria a vegadesRepetible[0] (n-1).
Després es redueix el resultat a trobar i la mida de la solució dividint si es pot la solució amb els valors no buits de la regió, si no ho són es que la regió no té solució i llença excepció, si ho és, redueix la mida de la solució (*midaUtil*), el nombre de repeticions d'aquell valor, i el resultat a trobar, dividint-lo pel valor (variable *multiplicacio*). Finalment s'entra al backtrack per a trobar les possibles solucions.
Backtrack:
Variables*:*
vegadesRepetible:int[]: La variable vegadesRepetible de la funció principal
min:int: El valor mínim que es pot posar en aquesta iteració del backtracking, la seva funció és reduïr l'espai de cerca, i no repetir valors ja posats.
Per exemple, que multiplicacio = 4 en una midaTauler de 3, i midaUtil de 3, hi ha les possibles solucions 1*2*2, 2*1*2, 2*2*1. Posant aquesta variable mínim, s'aconsegueix que primer es tinguin en compte les solucions on apareix l'1, i després ja no es pugui tornar a posar, evitant solucions repetides. És clar que això només es pot tenir en compte en operacions commutatives.
midaTauler:int: La variable midaTauler de la funció principal
midaUtil:int: La mida de la solució a buscar després d'haver tingut en compte les caselles no buides de la regió.
multiplicació:int: El resultat que es busca
solucions:Set<Integer>: Un HashSet amb els valors trobats fins el moment que satisfan la premisa. S'ha triat un HashSet perquè, al buscar valors únics, proporciona l'accés més ràpid (O(1)).
solucioParcial:ArrayList<Integer>: Vector amb la solució trobada fins el moment, fins al cas base no se sabrà si és vàlida.

El cas base d'aquest algorisme de backtracking és que la solucioParcial tingui la mida que es busca (*midaUtil*), i que la multiplicació dels seus elements sigui la de *multiplicacio*, això és que multiplicació sigui igual a 1, ja que en cada backtrack es divideix per l'últim valor afegit a la solució parcial. Si es verifica que la *solucioParcial* és correcta, s'afegeixen els seus valors a *solucio*.

El cas recursiu es fa en un for, que va de min a midaTauler (inclosos els dos), i per a cada i de la iteració prova si encara es pot posar (vegadesRepetible > 0), i si es divisor de *multiplicacio*. Si així és, s'afegeix a la solucioParcial, es resta 1 a vegadesRepetible[i-1], i es fa backtracking amb aquella *solucioParcial*, la i com a min, les noves *vegadesRepetible*, la *midaUtil* i *multiplicacio/i* com a nou resultat a buscar.
Finalment es restableix l'estat previ al backtrack eliminant el valor introduit a la *solucioParcial* i sumant 1 a les *vegadesRepetibles.*

8.4. Divisió (4)

És la implementació de la interfície Operacio que realitza els càlculs mitjançant l'operació de divisió d'enters.
Mètodes:

- **+opera2(int a, int b) : int**: realitza la divisió de dos enters independentment de l'ordre i retorna la divisió de l'enter més gran amb el més petit.
- **+operaN(int[] valors) : int**: realitza la divisió de dos enters independentment de l'ordre donats com a vector. Retorna la divisió de l'enter més gran del vector amb el més petit.
- **+calculaPossiblesValors(int resultat, int midaTauler, int midaRegio, int[] valors) : Set<Integer>**: calcula totes les possibles solucions de dos enters d'1 a la mida del tauler que dividits donin el resultat introduït. Retorna els valors d'1 a midaTauler que apareixen en alguna de les solucions calculades.
- **+getNumOperacio() : int**: retorna el número de l'operació, que en aquest cas és 4.
- **-valorPotSerResultat(int resultat): boolean:** Retorna si el valor pot ser resultat de l'operació de divisió.
- **-esCommutativa(): boolean:** Retorna si l'operació és commutativa.
- **-getOperacioText(): String:** Retorna el símbol de divisió.
- **-divisible(int a, int b) : boolean**: petita funció auxiliar que retorna true si a és divisible per b, false en cas contrari.
- **Explicació de l'algorisme principal calculaPossiblesValors():**
  Variables:
  resultat: El resultat del qual es volen trobar les possibles solucions que dividides donin aquell valor.
  midaTauler: indica el nombre de valors possibles que poden tenir les caselles així com el nombre de files i columnes. És a dir els valors poden ser de 1 a midaTauler, i el tauler te midaTauler x midaTauler caselles.
  midaRegió: mida de la regió de la qual es vol calcular els possibles valors. Les caselles buides d'aquesta regió indiquen la mida de la solució que estem buscant. Per exemple, en una regió de mida 2 buida es buscarien 2 possibles valors que dividits donin el *resultat.*
  valors: Un vector de ints que indica caselles no buides de la regió, és a dir que el resultat serà aquests valors dividits amb els possibles nombres trobats.

Funcionament:
Per la definició de l'operació de divisió en el kenken, que només permet dos valors, aquest algorisme no precisa de backtracking, ja que els resultats són deterministes des d'un primer moment.
Primer de tot, si la regió té mida diferent de 2, no pot ser una divisio, i si el nombre de valors inicials introduït és superior o igual a la mida de la regió es llançarà una excepció, ja que no té sentit calcular-ho, es tracta d'un error.
Si el resultat és 1 retorna un set buit (ja que la solució aleshores seria el mateix valor que valors[0] i per la no repetició en fila i columna en una resta és impossible que ho sigui).
Això deixa dos casos possibles, o valors és buit, o valors té 1 element.
Cas 1 valor: Es comprova que l'element estigui entre 1 i midaTauler, aleshores els possibles valors que pot haver-hi (després de comprovar cada un que està dins els permesos i sigui divisible) serien valors[0]*resultat i valors[0]/resultat. S'afegeixen aquests valors a la solució. Per exemple si el resultat fos 3, i el valor que hi ha a valors[0] fos 3, els nombres serien 1 (3/3) i 9 (3*3).
Cas 0 valors: En aquest cas es fa un for de i=1 a midaTauler i 'safegeixen els dos valors *i*, *i**resultat si es compleix que *i**resultat està dins els valors permesos (*i* sempre ho estarà).

8.5. Mòdul (5)

Realitza la divisió entre 2 valors provant totes les possibles combinacions. Es queda amb el residu de la divisió en comptes de amb el resultat.
Mètodes:

- **+opera2(int a, int b) : int**: Realitza l'operació de mòdul entre dos enters.
- **+operaN(int[] valors) : int**: Realitza l'operació de mòdul en un array d'enters.
- **+calculaPossiblesValors(int resultat, int midaTauler, int midaRegio, int[] valors) : Set<Integer>**: Calcula els possibles valors que podrien resultar en el resultat donat quan s'aplica l'operació de mòdul a ells.
- **+getNumOperacio() : int**: retorna el número de l'operació, que en aquest cas és 5.
- **-valorPotSerResultat(int resultat): boolean:** Retorna si el valor pot ser resultat de l'operació de mòdul.
- **-esCommutativa(): boolean:** Retorna si l'operació és commutativa.
- **-getOperacioText(): String:** Retorna el símbol de mòdul.

8.6. Exponenciació (6)

Realitza una potència entre 2 valors provant totes les possibles combinacions fins .
Mètodes:

- **+opera2(int a, int b) : int**: Realitza l'operació d'exponenciació entre dos enters.

- **+operaN(int[] valors) : int**: Realitza l'operació d'exponenciació en un array d'enters.
- **+calculaPossiblesValors(int resultat, int midaTauler, int midaRegio, int[] valors) : Set<Integer>**: Calcula els possibles valors que podrien resultar en el resultat donat quan s'aplica l'operació d'exponenciació a ells.
- **+getNumOperacio() : int**: retorna el número de l'operació, que en aquest cas és 6.
- **-valorPotSerResultat(int resultat): boolean:** Retorna si el valor pot ser resultat de l'operació d'exponenciació.
- **-esCommutativa(): boolean:** Retorna si l'operació és commutativa.
- **-getOperacioText(): String:** Retorna el símbol d'exponenciació.

9. CreadoraKenkenParàmetres

9.1. CreadorKenkenParam

És un algoritme que crea un tauler KenKen resoluble a partir de només 2 paràmetres, els quals són la mida del tauler (és a dir, el grau) i les operacions que ha de contenir.
Atributs:

- **int[][] solution:** Matriu que representa la solució del Kenken
- **char[][] blockSolution:** Matriu que representa la solució dels blocs
- **int size:** Mida del tauler
- **int maxSize:** Mida màxima dels blocs
- **int numReg:** Contador de regions
- **StringBuilder taulerSencer:** String on es guardaran les dades
- **StringBuilder coordenadesRegio:** String que guarda les coordenades d'una sola regio
- **String resultat:** Tauler passat a text sencer que es posara dins del .txt
- **Set<Character> processedBlocks = new HashSet<>():** Set de blocs (regions) visitades quan està creant operacions

Mètodes:

- **CreadorKenkenParam(int size, int maxSize):** Constructor de la classe CreadorKenkenParam.
- **generateSolutionAndBoard():** Genera la solució del Kenken i el tauler de blocs fusionats.
- **generateSolution():** Genera la solució del Kenken (és a dir, els números del resultat).
- **generateNumberList(): List<Integer>:** Genera una llista de nombres consecutius.
- **shuffleRowsAndColumns():** Barreja les files i columnes de la solució.
- **transposeSolution():** Transposa la matriu de solució.
- **generateBoard():** Genera el tauler de blocs fusionats.
- **generateCandidates(): List<Candidate>:** Genera una llista de candidates per fusionar blocs.

- **updateBlockSolution(Board board):** Li posa la lletra a cada block/regio.
- **printSolution():** Imprimeix la solució del Kenken per la terminal.
- **printBlockSolution():** Imprimeix la solució dels blocs per la terminal.
- **crearOperaciones(boolean suma, boolean resta, boolean multiplicació, boolean divisió, boolean mòdul, boolean exponenciacio):** Crea les operacions segons quines estaven seleccionades.
- **countBlockOccurrences(char block): int:** Conta el nombre de caselles que té el bloc d'una lletra en especific.
- **findBlockNumbers(int mida, char block): int[]:** Conta el nombre de caselles que té el bloc d'una lletra en especific.
- **assignaDivisio():** Assigna les divisions als blocs vàlids.
- **assignaBlocsRes(int mida, boolean suma, boolean multiplicació, boolean resta, boolean mòdul, boolean exponenciacio):** Assigna totes les operacions que no son divisions (i que siguin true) als blocs vàlids.
- **crearArxiuText():** Funció que recull les dades generades i ho posa en un .txt a la base de dades. Retorna l'identificador del tauler creat
- **creadora(int grau, boolean suma, boolean resta, boolean multiplicació, boolean divisió, boolean mòdul, boolean exponenciacio):** Funció invocada des de CrearKenKenParametres.java quan aquest vol crear un tauler per paràmetres (és a dir, li dono al botó de crear). La funció concretament posa en marxa tota la generació del tauler. Retorna l'identificador del tauler creat.

9.2. Board

És un Tauler format per Cells que s'utilitza durant la construcció d'un Kenken per paràmetres.
Atributs:

- **public Cell[][] grid:** Matriu formada per Cel·les.

Mètodes:

- **Board(int size):** Constructor de la classe Board.

9.3. Cell

Una Cell és una posició específica del tauler (com si fos una casella), aquesta classe és necessària per a la generació del regions úniques durant la creació de d'un Kenken per paràmetres.
Atributs:

- **public int x:** Coordenada x de la cel·la.
- **public int y:** Coordenada y de la cel·la.
- **public Cell parent:** Cell arrel.
- **public int size:** Mida de la regió de la que forma part la Cell.

Mètodes:

- **Cell(int x, int y):** Constructor de la classe Cell.
- **merge(Cell other):** Fusiona 2 cel·les, es el procés de posar-ho a la mateixa regió.
- **findRoot(): Cell:** Troba la cel·la parent.

9.4. Candidate

És una classe que simplement indica les possibles posicions per fusionar una cel·la amb una altre.
Atributs:

- **public int x1:** Coordenada x de la primera cel·la.
- **public int y1:** Coordenada y de la primera cel·la.
- **public int x2:** Coordenada x de la segona cel·la.
- **public int y2:** Coordenada y de la segona cel·la.

Mètodes:

- **Candidate(int x1, int y1, int x2, int y2):** Constructor de la classe Candidate.

10. SolucionadorKenken

El solucionador és una classe que s'encarrega de rebre un tauler i resoldre'l correctament o d'identificar si no es resoluble.
Atributs:

- **SolucionadorKenken Solucionador:** Instància singleton de SolucionadorKenken.

Mètodes:

- **SolucionadorKenken():** Constructor privat de la classe SolucionadorKenken. Evita que es pugui instanciar la classe directament.
- **SolucionadorKenken getInstance():** Retorna la instància singleton de SolucionadorKenken. Si encara no s'ha creat, la crea.
- **backtracking(Tauler T, int i, int j):** Mètode recursiu per a la resolució del tauler de joc.
- **optimitzacioNoOperacio(Tauler T):** Optimització que estableix el valor de les caselles que pertanyen a una regió amb una única operació.
- **solucionarKenken(Tauler T):** Mètode que soluciona un tauler de Kenken.

Descripció dels controladors

1. CtrlUsuari

El Controlador d'usuari s'encarrega de la gestió de mètodes de classe referents a la classe Usuari. Es comunica amb Usuari.java per tal de realitzar totes les funcionalitats referents a aquest.
Atributs:

- **CtrlUsuari CU:** Instància singleton de CtrlUsuari.

- **Gson gson:** Instància de Gson per a la serialització i deserialització de dades d'usuari.

Mètodes:

- **CtrlUsuari():** Constructor privat de la classe CtrlUsuari.
- **getInstance(): CtrlUsuari:** Retorna la instància singleton de CtrlUsuari.
- **iniciarSessio(String nomUsuari, String contrasenya):** Inicia sessió per a un usuari amb el nom d'usuari i contrasenya proporcionats.
- **registrarUsuari(String nomUsuari, String contrasenya):** Registra un nou usuari amb el nom d'usuari i contrasenya proporcionats.
- **canviarContrasenya(String ctrActual, String ctrNova):** Canvia la contrasenya de l'usuari actual.
- **getUsuariActual(): Usuari:** Retorna l'usuari actual.

2. ControladorPartida

Controlador de la capa de domini que representa una partida i, per tant, efectua les operacions convenients per a mantenir un correcte flux del joc. Quan hi ha un joc en curs el seu atribut partida no és null, i quan no n'hi ha cap és null. També conté una pila de moviments per desfer i una per refer, on els moviments son un int[] (fila,columna,valor). El format d'encapsulament de dades de les partides està descrit la classe Partida, a guardarPartida() per a dades d'una partida guardada, a acabarPartida() per a dades d'una partida acabada, i a generarPartidaText() per a l'estat d'una partida iniciada.

Atributs:

- **-Partida partida_:** La partida en curs. Null si no hi ha cap partida en curs.
- **-HashMap<String, String> partidesGuardadesUsuari_:** Un mapa que relaciona tots els identificadors de les partides d'un usuari amb el seu contingut. La seva funció és el ràpid accés a les partides guardades de l'usuari per si decideix carregar-ne una, per això s'ha utilitzat un HashMap.
- **-Stack<int[]> desferMoviments_:** Pila que guarda els moviments fets a la partida per a poder desfer-los. Un moviment es representa com un vector d'enters de 3 posicions: fila, columna i el valor anterior al moviment. Per la seva característica de només voler desfer l'últim moviment, l'estructura que s'ha trobat més adient per implementar-la és la d'un stack.
- **-Stack<int[]> referMoviments_:** Pila que guarda els moviments desfets a la partida per a poder refer-los. Un moviment es representa com un vector d'enters de 3 posicions: fila, columna i el valor anterior al moviment a refer. La justificació de l'estructura de dades utilitzada és la mateixa que per a desferMoviments_.
- **-CtrlDomini controladorDomini_:** Instància del controlador de domini.
- **-static ControladorPartida controladorPartida_:** Instància del controlador de partida.

Mètodes:

- **+ControladorPartida() : void**: Constructor per defecte: Aquest mètode és el constructor de la classe ControladorPartida. Inicialitza la partida a null i crea un nou HashMap per a les partides guardades de l'usuari.
- **+getInstance(): ControladorPartida:** Getter de la instància del controlador de partida.
- **+getPartidesGuardadesUsuari() : String[]**: Getter dels identificadors de les partides d'un usuari: Aquest mètode retorna un array de Strings amb els identificadors de les partides de l'usuari. Si l'usuari no té cap partida guardada, retorna un array buit.
- **+haJugat(identificadorTauler : String, nomUsuari : String) : boolean**: Indica si hi ha cap existència en memòria d'una partida guardada amb l'identificador del tauler donat per aquell usuari: Aquest mètode retorna true si l'usuari ha jugat aquest tauler, false si no.
- **+carregarUltimaPartidaGuardada(nomUsuari : String) : String[]**: Carrega l'última partida guardada de l'usuari: Aquest mètode carrega l'última partida guardada de l'usuari i comença la partida amb les dades guardades. Retorna un array de Strings amb l'estat de la partida a l'índex [0] i les dades del tauler a l'índex [1].
- **+carregarPartidesGuardadesUsuari(nomUsuari : String) : ArrayList<String>**: Carrega totes les partides guardades de l'usuari per a un ràpid accés: Aquest mètode carrega totes les partides guardades de l'usuari per a un ràpid accés. Retorna un array de Strings amb els identificadors de les partides de l'usuari.
- **+iniciarPartidaGuardada(identificadorPartida : String, nomUsuari : String) : String[]**: Carrega una partida guardada de l'usuari segons un identificador: Aquest mètode carrega una partida guardada de l'usuari segons un identificador i comença la partida amb les dades guardades. Retorna un array de Strings amb l'estat de la partida a l'índex [0] i les dades del tauler a l'índex [1].
- **+iniciaPartidaIdentificadorTauler(identificadorTauler : String, nomUsuari : String) : String[]**: Comença una partida amb el tauler identificat per identificadorTauler: Aquest mètode comença una partida amb el tauler identificat per identificadorTauler. Retorna un array de Strings amb l'estat de la partida a l'índex [0] i les dades del tauler a l'índex [1].
- **+iniciaPartidaAleatoria(int mida, String nomUsuari): String:** Comença una partida amb un tauler aleatori de mida mida. Primer busca a memòria si hi ha taulers que l'usuari no hagi jugat. Si no n'hi ha en crea un automàticament.
- **+introduirValor(fila : int, columna : int, valor : int) : String[]**: Canvia l'estat de la partida, és a dir un dels seus valors: Aquest mètode canvia l'estat de la partida, és a dir, un dels seus valors. Retorna l'estat de la partida després de canviar la casella.
- **+desferMoviment(): String[]**: Desfà l'últim moviment fet a la partida. Retorna l'estat de la partida després de desfer el moviment.
- **+referMoviment(): String[]**: Refà un moviment desfet a la partida. Retorna l'estat de la partida després de refer el moviment.

- **+donaPista(): String[]**: Canvia l'estat de la partida afegint un valor que portaria a una solució, o indica si el que hi ha fins ara no pot portar a una solució. Retorna l'estat de la partida després de canviar la casella.
- **+guardarPartida(nomUsuari: String): boolean**: Guarda la partida en curs del controlador. L'afegeix també al mapa de partides guardades de l'usuari. Retorna true si s'ha guardat la partida. false si no s'ha guardat la partida.
- **+tancarIguardarPartida(): boolean**: Tanca i guarda la partida en curs del controlador. L'afegeix també al mapa de partides guardades de l'usuari. Retorna true si s'ha tancat i guardat la partida. false si no s'ha guardat la partida, i per tant no es tanca.
- **+acabarPartida(): String[]**: Acaba la partida en curs del controlador. Una partida acabada és aquella que està ben resolta. Retorna un vector de 3 strings. La primera indica si s'ha guardat la partida a memòria. La segona si havia estat guardada prèviament. La tercera el temps que ha durat la partida.
- **+tancaPartida(): boolean**: Tanca la partida en curs del controlador.
- **-stringToPartida(partida: String, nomUsuari: String): Partida** : Funció privada que transforma una string de dades d'una partida en una instància de Partida. Retorna una instància de Partida amb les dades de la string.
- **-comprovaRepetits(int[][] valors): ArrayList<int[]>:** Comprova si hi ha números repetits en la mateixa fila o columna. Retorna un vector d'enters amb les posicions de les caselles amb valors repetits
- **-getAdjacentsPartida(): ArrayList<Boolean>[][]:** Genera el mapa d'adjacencies del tauler de la partida i el retorna.

3. CtrlKenkens

El Controlador de Kenkens s'encarrega de la gestió de mètodes de classe referents a la classe Tauler. Es comunica amb Tauler.java per tal de realitzar totes les funcionalitats referents a aquest.
Atributs:

- **ControladorPersistenciaTauler ctrlTaulerData:** Instància de ControladorPersistenciaTauler per a la persistència de dades de tauler.
- **SolucionadorKenken Solucionador:** Instància de SolucionadorKenken per a la resolució de taulers de Kenken.
- **CtrlKenkens ctrlKenkens:** Instància singleton de CtrlKenkens.

Mètodes:

- **CtrlKenkens():** Constructor privat de la classe CtrlKenkens. Inicialitza l'instància de ControladorPersistenciaTauler.
- **getInstance(): CtrlKenkens:** Retorna la instància singleton de CtrlKenkens.
- **getOperacio(int oper): Operacio:** Retorna l'operació corresponent al número d'operació proporcionat.
- **llegirTauler(String id): Tauler:** Llegeix un tauler de la base de dades amb l'identificador proporcionat.

- **Tauler stringToTauler(String contingutTauler, String id):** Converteix una cadena de text a un Tauler.
- **taulerToString(Tauler T): String:** Converteix un tauler a una cadena de text.
- **seleccionaTaulerAleatori(int mida): String:** Selecciona un tauler aleatori de la mida proporcionada.
- **resoldreKenken(Tauler T, int[][] valorsPartida): int[][]:** Resol un tauler de Kenken amb els valors de partida proporcionats i retorna la solució.
- **guardarTaulerBD(String contigutTauler): String:** Guarda un tauler a la base de dades.
- **esTaulerValid(String contingutTauler): boolean:** Comprova si un tauler és vàlid.
- **resoldreKenken(String idTauler, boolean guardarBD)**
- **pintarTauler(String idTauler)**
- **mostrarTauler(Tauler T)**

4. CtrlDomini

La classe CtrlDomini és el controlador principal del domini de l'aplicació Kenken.
Aquesta classe és responsable de gestionar les operacions principals del joc, incloent la gestió d'usuaris, partides, rankings i Kenkens.
Atributs:

- **CtrlDomini CDomini**
- **CtrlUsuari CUsuari**
- **ControladorPartida CPartida**
- **CtrlKenkens CKenkens**
- **ControladorPersistencia CPersistencia**

Mètodes:

- **CtrlDomini():** Creadora de CtrlDomini
- **CtrlDomini getInstance():** Retorna la instància del controlador Domini.

Seguidament conté tots els mètodes per a cridar a tots els mètodes de domini, garantitzant que únicament passen del controlador principal de domini.

5. ControladorRanking

És el controlador de domini encarregat de gestionar el ranking de les partides acabades.
Atributs:

- **-Ranking[] rankingPerMida_:** Rankings de les partides acabades per mida. Cada posició i representa el ranking de les partides acabades de mida i + MIDAMIN. S'ha optat per un array normal ja que la quantitat de rankings està determinada des d'un inici com a la MIDAMAX-MIDAMIN +1. En el nostre cas, 9 - 3 + 1 = 7.

- **-CtrlDomini controladorDomini_:** Instància del controlador de domini.

Mètodes:

- **ControladorRanking():** Constructora de la classe.
- **ControladorRanking getInstance():** Retorna la instància del controlador de ranking.
- **afegirPartida(String partidaAcabada): boolean:** Afegeix una partida acabada al ranking de la mida corresponent. Utilitza el format de acabaPartida() a l'apartat 2 per a les dades de la partidaAcabada.
- **getRankingN(int mida, int index, int n): ArrayList<String>:** Retorna les N partides amb millor temps a partir de l'índex donat de la mida corresponent.
- **getRankingMida(int mida): ArrayList<String>:** Retorna el ranking complet de les partides de la mida corresponent.
- **getRankingUsuari(String identificadorUsuari):** ArrayList<String>: Retorna les partides d'un usuari donat de la mida corresponent.
- **inicialitzarRanking():** Inicialitza el ranking de les partides acabades.

Descripció de la presentació

1. CrearKenkenManual

1.1. Casella Constructora

Atributs:

- **Color COLOR_SELECCIONADA:** Color utilitzat quan la casella està seleccionada.
- **Color COLOR_REGIO:** Color utilitzat per la regió de la casella.
- **Color COLOR_DEFAULT:** Color per defecte de la casella.
- **Color COLOR_VORA_REGIO:** Color de la vora de la regió.
- **Color COLOR_VORA_DEFAULT:** Color de la vora per defecte.
- **int GRUIX_VORA_REGIO:** Gruix de la vora de la regió.
- **int GRUIX_VORA_DEFAULT:** Gruix de la vora per defecte.
- **JLabel infoRegioLabel:** Etiqueta que mostra informació sobre la regió a la que pertany la casella.
- **int posX:** Posició X de la casella.
- **int posY:** Posició Y de la casella.
- **boolean seleccionada:** Indica si la casella està seleccionada.
- **boolean pertanyRegio:** Indica si la casella pertany a una regió**.**
- **int indexRegio:** Índex de la regió a la que pertany la casella.

Mètodes:

- **CasellaConstructora(int posX, int posY):** Constructor de la classe CasellaConstructora. Inicialitza la casella amb la posició donada i configura l'estat inicial de la casella. També afegeix un MouseListener a la casella per a gestionar els events de ratolí.

- **configInicial():** Configura l'estat inicial de la casella. L'estat inicial és no seleccionada, no pertany a cap regió, i té l'índex de regió -1. També estableix el color de fons i la vora de la casella als valors per defecte.
- **addMouseListener():** Afegeix un MouseListener a la casella. El MouseListener gestiona els clics del ratolí a la casella. Si es fa clic amb el botó esquerre del ratolí, la casella es selecciona o desselecciona. Si es fa clic amb el botó dret del ratolí i la casella encara no ha sigut assignada a cap regio, es processa l'entrada de l'usuari. Si ja ha sigut assignada es reseteja la configuracio de les caselles de la regio.
- **esSeleccionada(): boolean:** Retorna si la casella està seleccionada.
- **teRegioAssignada(): boolean:** Retorna si la casella pertany a una regió.
- **decrementaIndexRegio():** Decrementa l'índex de la regió de la casella.
- **getIndexRegio(): int:** Retorna l'índex de la regió de la casella.
- **setSeleccionada():** Estableix l'estat de selecció de la casella. Si la casella està seleccionada, es desselecciona. Si la casella no està seleccionada i és adjacent a alguna de les caselles selecionades, es selecciona.
- **marcaComRegio(int index):** Marca la casella com a part d'una regió. Estableix l'índex de la regió de la casella i canvia el color de fons de la casella al color de la regio.
- **desmarcaComRegio():** Desmarca la casella com a part d'una regió. Restableix l'estat inicial de la casella.
- **addInfoRegio(String operacio, String resultat, int midaTauler)**: Afegeix informació sobre la regió a la casella. Crea un JLabel amb l'operació i el resultat de la regió, i l'afegeix a la casella.
- **pintarVores(boolean[] esVoraRegio):** Estableix el color i gruix de les vores de la casella. Crea un MatteBorder per a cada vora de la casella, amb un gruix i un color que depenen de si la vora és una "vora regio". Després, estableix el borde de la casella com a un CompoundBorder dels quatre bordes creats.
- **eliminarInfoRegio():** Elimina la informació de la regió de la casella. Si la casella té un JLabel amb informació de la regió, l'elimina de la casella.

1.2. CrearKenkenManual

Atributs:

- **CrearKenkenManual creardoraKenken:** Instància de CrearKenkenManual.
- **JPanel panelComplet:** Panel principal de la pantalla de creació de Kenken.
- **JPanel panelEsq:** Panel esquerra de la pantalla de creació de Kenken.
- **JPanel grauPanel:** Panel per seleccionar el grau del tauler.
- **JPanel configPanel:** Panel de configuració.
- **JLabel logoCreateLabel:** Etiqueta del logo de la pantalla de creació.
- **JLabel grauLabel:** Etiqueta per mostrar el grau del tauler.
- **JButton guardarButton:** Botó per guardar el tauler creat.
- **JButton sortirButton:** Botó per sortir de la pantalla de creació de Kenken.
- **JButton aceptarButton:** Botó per acceptar la configuració inicial.
- **JComboBox grauComboBox:** ComboBox per seleccionar el grau del tauler.

- **JButton resetBoton:** Botó per reiniciar la creació del tauler.
- **JPanel panelDret:** Panel dret de la pantalla de creació de Kenken.
- **TaulerConstrutor TaulerKenken:** Instància de TaulerConstrutor per crear el tauler Kenken.
- **boolean enModeEditor:** Boolean per indicar si està en mode editor.

Mètodes:

- **CrearKenkenManual getInstance():** Retorna una instància de CrearKenkenManual. Si no existeix, la crea.
- **newInstance():** Crea una nova instància de CrearKenkenManual.
- **CrearKenkenManual():** Constructor de la classe CrearKenkenManual. Inicialitza els components de la interfície d'usuari i configura els listeners dels botons.
- **processarEntradaUsuari(CasellaConstructora casellaPremuda):** Processa l'entrada de l'usuari quan es prem una casella. Mostra un diàleg per a que l'usuari introdueixi l'operació i el resultat per a la regió seleccionada.
- **mostrarDialog(JPanel operacioPanel, JPanel resultatPanel, String errorMessage, CasellaConstructora casellaPremuda): int:** Mostra un diàleg per a que l'usuari introdueixi l'operació i el resultat. Si hi ha un missatge d'error, es mostra un diàleg d'error.
- **crearOperacioPanel(): JPanel:** Crea un panel amb els botons de les operacions. Tots els botons estan agrupats en un ButtonGroup per a que només es pugui seleccionar un.
- **crearResultatPanel(): JPanel:** Crea un panel amb el camp de text per al resultat. Limita la longitud del text a 9 caràcters i només permet introduir dígits.
- **obtenerOperacioSeleccionada(JPanel operacioPanel): String:** Obté l'operació seleccionada en el panel d'operacions.
- **previewTauler(int size):** Mostra una previsualització del tauler amb la mida especificada. Cada casella del tauler es representa com un JPanel cuadrat amb vores grises.
- **configInicial():** Configura l'estat inicial de la pantalla de creació de Kenken. Inicialment previsualitza un tauler de mida 3.
- **getDefaultPanel():** JPanel: Retorna el panel principal de la pantalla de creació de Kenken.
- **createUIComponents():** Crea els components de la interfície d'usuari que no es poden crear en el dissenyador de formularis. Aquest metode es crida automàticament durant la inicialització de la interfície d'usuari.

1.3. TaulerConstructor

Atributs:

- **TaulerConstrutor taulerConstrutor:** Instància única de TaulerConstrutor (Singleton)
- **int mida:** Mida del tauler
- **int numCasellesAssignades:** Número de caselles assignades

- **CasellaConstructora[][] caselles:** Matriu de caselles
- **ArrayList<String> dadesRegions = new ArrayList<>():** Llista de dades de les regions
- **ArrayList<int[]> posCaselles = new ArrayList<>():** Llista de posicions de les caselles seleccionades
- **boolean[][] valorsUsatsFila:** Matriu de booleans que indica si un valor ha estat usat en una fila

Mètodes:

- **TaulerConstrutor(int mida):** Constructor privat de TaulerConstrutor.
- **TaulerConstrutor getInstance():** Retorna la instància única de TaulerConstrutor.
- **newInstance(int mida):** Crea una nova instància de TaulerConstrutor amb una mida específica.
- **inicialitzarCaselles(int mida):** Inicialitza les caselles del tauler.
- **getMida():** Retorna la mida del tauler.
- **getNumCasellesAssignades():** Retorna el número de caselles seleccionades.
- **getNumCasellesSelecionades():** Retorna el número de caselles seleccionades.
- **getNumFilesSelecionades(): int:** Retorna el nombre de files selecionades.
- **esModificat(): boolean:** Indica si el tauler ha estat modificat.
- **afegirPosCasellaSelecionada (int x, int y):** Afegeix la posició d'una casella seleccionada a la llista de posicions de caselles selecionades.
- **eliminarPosCasellaSelecionada (int x, int y):** Elimina la posició d'una casella selecionada de la llista de posicions de caselles selecionades.
- **assignarCasellesRegio(String operacio, String resultat):** Assigna les caselles seleccionades a una regió.
- **eliminarRegioAssignada(int index):** Elimina una regió assignada.
- **esAdjacent(int posX, int posY): boolean:** Comprova si una posició és adjacent a alguna de les caselles seleccionades.
- **esVoraRegio(int dx, int dy, int posX, int posY): boolean:** Comprova si una posició és una "vora regió".
- **marcarBordesRegio():** Marca les vores de la regió.
- **getContigutTauler(): String:** Retorna el contingut del tauler com a cadena String.
- **ordenarPosicionsCaselles():** Ordena creixentment les posicions de les caselles seleccionades.
- **intercambiarPrimeraUltimaPosicion():** Si la llista de caselles seleccionades no està buida, intercanvia la primera i l'última posició.
- **validarEntrada(String operacio, String resultat): String:** Valida l'entrada de l'operació i el resultat.
- **validarValorMinMaxOperacio(String operacio, int resultat): String:** Valida si el resultat esta dins el valor mínim i màxim que es pot obtenir amb l'operació indicada.
- boolean resultatEsForaInterval(int valor, int min, int max)
- **validarSuma(int resultat): String:** Valida el resultat de l'operació de suma.
- **validarResta(int resultat): String:** Valida el resultat de l'operació de resta.

- **validarDivisio(int resultat): String:** Valida el resultat de l'operació de divisio.
- **validarExponenciacio(int resultat): String:** Valida el resultat de l'operació de exponenciacio.
- **validarModul(int resultat): String:** Valida el resultat de l'operació de modul.
- **validarMultiplicacio(int resultat): String:** Valida el resultat de l'operació de multiplicacio.
- **traduirOperacioAnum(String operacio): String:** Tradueix l'operació especificada a un número.
- **traduirOperacioAsimbol(String operacio): String:** Tradueix l'operació especificada a un símbol.
- **resetTauler():** Restableix l'estat del tauler.

2. Partida

2.1. ComponentCasella

ComponentCasella és la classe que representa una casella del taulell del kenken en la interfície gràfica. Conté un valor, una operació i un botó per a poder introduir un valor a la casella, així com les seves coordenades.
Atributs:

- **JLabel valor_ :** és el JLabel que conté el valor de la casella en text.
- **JLabel operacio_ :** és el JLabel que conté l'operació de la casella en text.
- **JButton boto_ :** és el JButton que permet introduir un valor a la casella.
- **boolean botoPremut_ :** és un booleà que indica si s'ha premut el botó de la casella.
- **boolean incorrecte_ = false :** és un booleà que indica si la casella és incorrecte.
- **int mida_ :** és la mida del kenken.
- **int fila_ :** és la fila de la casella.
- **int columna_ :** és la columna de la casella.
- **List<ObservadorCasella> observers_ :** és la llista d'observadors de la casella.

Mètodes:

- **ComponentCasella(int mida, int fila, int columna, int valor) :** Creadora de la classe ComponentCasella.
- **setIncorrecte(boolean incorrecte) :** Posa la casella com a incorrecte o correcte en funció del paràmetre incorrecte. Pinta el botó de la casella amb un color vermell si és incorrecte, i sense color si és correcte.
- **pintaBoto() :** Pinta el botó de la casella en funció de si és incorrecte o no.
- **addObserver(ObservadorCasella observer) :** Afegeix un observador a la casella.
- **setValor(String valor) :** Posa el valor de la casella al paràmetre valor.
- **setOperacio_(String operacio) :** Posa l'operació de la casella al paràmerte operació.
- **getPreferredSize(): Dimension :** Retorna la dimensió default de la casella.

- **keyTyped(KeyEvent e) :** Quan s'ha premut el botó, es posa el botoPremut_ a true. Aleshores, quan es premi una tecla, si el botoPremut_ és true, es posa el valor de la tecla a la casella, si és permès. Es notifica als observadors que s'ha canviat el valor de la casella.
- **permesa(String c): boolean :** Comprova que el caràcter c estigui permès en el context del joc.
- **keyPressed(KeyEvent e) :** No fa res.
- **keyReleased(KeyEvent e) :** No fa res
- **actionPerformed(ActionEvent e) :** Quan es prem el botó, es posa el botoPremut_ a true.

2.2. ComponentLlistaPartides

És la classe abstracta que representa una llista de NOMBREPARTIDES_ d'elements on cada element és clickable i conté una informació concreta.

Atributs:

- **int NOMBREPARTIDES_ :** Nombre de partides que es mostren a la vegada.
- **int INFOPERPARTIDA_ :** Nombre d'informacions que es mostren per cada partida. Per exemple, si es vol mostrar la mida, data i temps de cada partida, INFOPERPARTIDA_ = 3.
- **ArrayList<ObservadorLlista> observadorsLLista_ :** Llista d'observadors de la llista de partides.
- **String[] informacioPartides_ :** Conté la informació de cada partida.
- **int indexActual_ = 0 :** Índex de la pàgina actual.
- **int maxIndex_ = 0 :** Índex màxim de la pàgina.
- **JButton[] botonsPartida_ :** Botons de les partides.
- **JLabel[] labelsPartida_ :** Labels de les partides.
- **JButton previousN_ :** Botó per anar a la pàgina anterior.
- **JButton nextN_ :** Botó per anar a la pàgina següent.

Mètodes:

- **ComponentLlistaPartides(String[]informacioPartides, int NOMBREPARTIDES, String[] capcalera)**: Crea una llista de partides amb la informació donada i el nombre de partides a mostrar per pàgina.
- **previousN():** Tira enrere si pot i ensenya les NOMBREPARTIDES_ anteriors.
- **nextN():** Tira endavant si pot i ensenya les NOMBREPARTIDES_ següents.
- **addObservadorLlista(ObservadorLlista ob):** Afegeix un observador de la llista de partides.
- **generaText(String s):** Genera el text per posar a cada label en funció de la informació de la partida.
- **setIndexActual(int indexActual):** Posa l'índex actual de la pàgina.

2.3. ComponentTauler

Classe que representa el tauler de la partida en la GUI, exté JPanel.
Atributs:

- **ComponentCasella[][] tauler:** És una matriu de ComponentCasella que formen el tauler.

Mètodes:

- **ComponentTauler(int mida, ArrayList<Boolean>[][] mapaAdjacents, int[][] valorsTauler):** Creadora de la classe ComponentTauler.
- **creaBordersRegions(ArrayList<Boolean>[][] mapaAdjacents):** Crea els borders de les regions del tauler d'acord amb el mapa d'adjacencies.
- **addObserver(ObservadorCasella observer, int i, int j):** Afegeix un observador a una casella del tauler.
- **actualitzaValors(int[][] tauler):** Actualitza els valors de les caselles del tauler.
- **actualitzaPosicionsIncorrectes(ArrayList<int[]> posicionsIncorrectes):** Actualitza les posicions incorrectes del tauler.

2.4. ComponentTimer

Classe que representa el timer de la partida en la GUI.
Atributs:

- **int seconds_:** És el temps en segons del temporitzador.
- **JLabel timerLabel:** És el JLabel que conté el temps del temporitzador.

Mètodes:

- **ComponentTimer(int seconds):** Creadora de la classe ComponentTimer.

2.5. ControladorPresentacioPartida

Atributs:

- **String COLOR_ERROR = "243, 67, 65":** Constant que conté el color d'alguna cosa incorrecta.
- **String COLOR_BE = "40, 250, 85":** Constant que conté el color d'alguna cosa correcta.
- **ControladorPartida controladorPartida_:** Controlador de la partida.
- **VistaPartida vistaPartida_:** Vista de la partida.
- **VistaMenuJugarPartida vistaMenuJugarPartida_:** Vista del menú de jugar partida.
- **VistaPartidesGuardades vistaPartidesGuardades_:** Vista de les partides guardades.
- **ControladorPresentacioPartida controladorPresentacioPartida_:** Controlador de la presentació de la partida.

- **JPanel mainPanel_:** Panell principal.

Mètodes:

- **ControladorPresentacioPartida():** Constructora per defecte.
- **ControladorPresentacioPartida getInstance():** Retorna la instància del controlador de la presentació de la partida.
- **inicialitzaMenuJugarPartida(String usuari):** Inicialitza la vista del menú de jugar partida.
- **notificarCanviValor(String valor, int fila, int columna):** Reacciona quan es notifica que s'ha fet un click a una casella del joc.
- **notificarUndo():** Mètode que s'executa quan es notifica que s'ha fet click al botó de undo de la partida.
- **notificarRedo():** Mètode que s'executa quan es notifica que s'ha fet click al botó de redo de la partida.
- **notificarPista():** Mètode que s'executa quan es notifica que s'ha fet click al botó de pista de la partida.
- **notificarAcabar():** Mètode que s'executa quan es notifica que s'ha fet click al botó d'acabar de la partida.
- **notificarGuardar():** Mètode que s'executa quan es notifica que s'ha fet click al botó de guardar de la partida.
- **notificarTancalguarda():** Mètode que s'executa quan es notifica que s'ha fet click al botó de tancar i guardar de la partida.
- **notificarSortir():** Mètode que s'executa quan es notifica que s'ha fet click al botó de sortir de la partida.
- **jugarIntroduirTaulerManualment():** Mètode que s'executa quan es notifica que s'ha fet click al botó de jugar partida introduint el tauler manualment.
- **jugarIntroduirIdentificadorTauler():** Mètode que s'executa quan es notifica que s'ha fet click al botó de jugar partida introduint l'identificador del tauler.
- **jugarPartidaAleatoria():** Mètode que s'executa quan es notifica que s'ha fet click al botó de jugar partida aleatòria.
- **jugarUltimaPartidaGuardada():** Mètode que s'executa quan es notifica que s'ha fet click al botó de jugar última partida guardada.
- **jugarPartidaGuardada():** Mètode que s'executa quan es notifica que s'ha fet click al botó de jugar partida guardada.
- **tornarMenu():** Mètode que s'executa quan es notifica que s'ha fet click al botó de tornar per tornar al menú de JugarPartida.
- **clickatLlista(String partidaGuardada)**: Mètode que s'executa quan es notifica que s'ha fet click a una partida guardada de la llista.
- **inicialitzaVistaPartida()**: Inicialitza la vista de la partida amb les dades de la partida actual.
- **valorsStringToInt(String solucioTotalText):** Converteix la string que representa l'estat d'una partida a una matriu d'ints.

2.6. MenuPartida

Atributs:

- **JLabel missatge_:** és un missatge que es mostra a l'usuari en situacions concretes.
- **ComponentTimer timer_:** és el rellotge de la partida.
- **JButton undoBoto_:** és el botó per desfer l'última jugada.
- **JButton redoBoto_:** és el botó per refer l'última jugada.
- **JButton pistaBoto_:** és el botó per demanar una pista.
- **boolean pistaAvisat_:** és un booleà que indica si s'ha avisat a l'usuari que si demana una pista no podrà participar en els rankings.
- **JButton acabaBoto_:** és el botó per acabar la partida.
- **JButton guardaBoto_:** és el botó per guardar la partida.
- **boolean guardatAvisat_:** és un booleà que indica si s'ha avisat a l'usuari que si guarda la partida no podrà participar en els rankings.
- **JButton tancalguardaBoto_:** és el botó per tancar la partida i guardar-la.
- **JButton sortirBoto_:** és el botó per sortir de la partida en qualsevol moment.
- **ArrayList<ObservadorBoto> observadorsBoto_:** és una llista d'observadors que estan pendents dels esdeveniments dels botons.

Mètodes:

- **MenuPartida():** Creadora de la classe. Crea cada component del panell i els afegeix al panell.
- **addObservadorBoto(ObservadorBoto observer):** Afegeix un observador a la llista d'observadors.
- **mostrarMissatge(String missatge, boolean bo):** Mostra un missatge a l'usuari. Si el missatge és "dolent" es mostrarà en vermell, si és "bo" es mostrarà en verd. Després d'1,5 segons el missatge desapareixerà.

2.7. ObservadorBoto

Mètodes:

- **notificarUndo():** Notifica als observadors que s'ha premut el botó d'undo.
- **notificarRedo():** Notifica als observadors que s'ha premut el botó de redo.
- **notificarPista():** Notifica als observadors que s'ha premut el botó de pista.
- **notificarGuardar():** Notifica als observadors que s'ha premut el botó de guardar.
- **notificarTancalguarda():** Notifica als observadors que s'ha premut el botó de tancar i guardar.
- **notificarSortir():** Notifica als observadors que s'ha premut el botó de sortir.
- **jugarIntroduirTaulerManualment():** Notifica als observadors que l'usuari vol jugar introduint el tauler manualment.
- **jugarIntroduirIdentificadorTauler():** Notifica als observadors que l'usuari vol jugar introduint l'identificador del tauler.
- **jugarPartidaAleatoria():** Notifica als observadors que l'usuari vol jugar una partida aleatòria.
- **jugarUltimaPartidaGuardada():** Notifica als observadors que l'usuari vol jugar l'última partida guardada.

- **jugarPartidaGuardada()**: Notifica als observadors que l'usuari vol jugar una partida guardada.
- **notificarAcabar():** Notifica als observadors que l'usuari vol acabar.

2.8. ObservadorCasella

Mètodes:

- **notificarCanviValor(String valor, int fila, int columna):** Notifica als observadors que s'ha fet un canvi de valor a la casella.

2.9. ObservadorLlista

Mètodes:

- **clickatLlista(String s):** Notifica que s'ha clicat sobre un element de la llista.
- **tornarMenu():** Notifica que s'ha clicat sobre el botó de tornar.

2.10. VistaMenuJugarPartida

Atributs:

- **ArrayList<ObservadorBoto> observadorsBoto_:** és una llista d'observadors que notificaran quan es premi un botó i quin mètode per tractar-ho.
- **String identificadorUsuari_:** és l'identificador de l'usuari que ha iniciat sessió.

Mètodes:

- **addObservadorBoto(ObservadorBoto ob):** Afegeix un observador de botó.
- **getUsuari(String identificadorUsuari):** Actualitza l'identificador de l'usuari.
- **getUsuari():** String: Retorna l'identificador de l'usuari.

2.11. VistaPartida

Atributs:

- **String identificadorUsuari_:** és l'identificador de l'usuari que ha iniciat sessió.
- **ComponentTauler componentTauler_:** és el component del tauler de la partida.
- **MenuPartida menuPartida_:** és el menú de la partida.

Mètodes:

- **mostrarMissatgeMenu(String missatge, boolean correctesa):** Mostra un missatge en el menú de la partida.
- **addObserverMenuPartida(ObservadorBoto observadorBoto):** Afegeix un observador al menú de la partida.
- **addObserverCasella(ObservadorCasella observadorCasella, int i, int j):** Afegeix un observador a una casella del tauler.
- **actualitzaValors(int[][] tauler):** Actualitza els valors del tauler.
- **actualitzaPosicionsIncorrectes(ArrayList<int[]> posicionsIncorrectes):** Actualitza les posicions incorrectes del tauler.
- **getIdentificadorUsuari(): String**: Retorna l'identificador de l'usuari que ha iniciat la partida.

2.12. VistaPartidesGuardades

Mètodes:

- **VistaPartidesGuardades(String[] partidesGuardades, int NOMBREPARTIDES):** Crea una vista de les partides guardades amb les partides guardades de l'usuari.
- **generaText(String partidaGuardada):** String[]: Genera la mida, data i temps d'una partida guardada.

3. Ranking

3.1. BuscadorUsuari

Atributs:

- **ArrayList<ObservadorBuscador> observadors_:** es una lista de observadores del buscador de usuario.

Mètodes:

- **BuscadorUsuari():** es el constructor de la clase. Crea un panel con un diseño horizontal que contiene una etiqueta de texto "Usuari:", un campo de texto para ingresar el nombre de usuario y un botón "Buscar" para iniciar la búsqueda.
- **addObservador(ObservadorBuscador observador):** agrega un observador al buscador de usuario.

3.2. ComponentSelectorMida

Atributs:

- **ArrayList<ObservadorSelectorMida> observadors_:** es una lista de observadores del selector de tamaño.

Mètodes:

- **ComponentSelectorMida():** es el constructor de la clase. Crea un panel con un diseño horizontal que contiene una etiqueta de texto "Ranking dels taulers de mida: ", un selector de tamaño (JSpinner) que permite seleccionar el tamaño del tablero para el ranking.
- **addObservador(ObservadorSelectorMida observador):** agrega un observador al selector de tamaño.

3.3. ControladorPresentacioRanking

Atributs:

- **int ultimaMida_:** es el tamaño de la última selección de tamaño.
- **VistaRankings vistaRankings_:** es la vista de los rankings.
- **ControladorRanking controladorRanking_:** es el controlador de ranking.
- **ControladorPresentacioRanking instance_:** es una instancia única del controlador de presentación de ranking.
- **JPanel mainPanel_:** es el panel principal que contiene la vista de rankings.

Mètodes:

- **ControladorPresentacioRanking():** es el constructor de la clase. Inicializa el panel principal y el controlador de ranking.
- **ControladorPresentacioRanking getInstance():** devuelve la instancia única del controlador de presentación de ranking.
- inicialitzaVistaRankings(): inicializa la vista de los rankings. Obtiene la información de las partidas según el tamaño mínimo (MIDAMIN), agrega observadores a la vista de rankings y actualiza el panel principal.
- **clickatLlista(String s):** método de la interfaz ObservadorLlista que no tiene implementación en esta clase.
- **tornarMenu():** método de la interfaz ObservadorLlista que vuelve al menú principal.
- **midaSeleccionada(String mida):** método de la interfaz ObservadorSelectorMida que se activa cuando se selecciona un tamaño. Obtiene la información de las partidas según el tamaño seleccionado, actualiza la vista de rankings y repinta el panel principal.
- **busquedaUsuari(String nom):** método de la interfaz ObservadorBuscador que se activa cuando se realiza una búsqueda por usuario. Obtiene la información de las partidas según el nombre de usuario ingresado, muestra un mensaje de error si no se encuentra ninguna partida y actualiza la vista de rankings.

3.4. ObservadorBuscador

Mètodes:

- **busquedaUsuari(String nom):** Este método se utiliza para notificar a los observadores cuando se realiza una búsqueda por usuario en el ranking. Recibe como parámetro el nombre del usuario que se está buscando.

3.5. ObservadorSelectorMida

Mètodes:

- **midaSeleccionada(String mida):** Este método se utiliza para notificar a los observadores cuando se selecciona una determinada medida en el selector de tamaño del ranking. Recibe como parámetro la medida seleccionada en formato de cadena de texto.

3.6. PartidesRanking

Mètodes:

- **PartidesRanking(String[] informacioPartides, int NOMBREPARTIDES):** Crea una lista de partidas con la información dada y el número de partidas a mostrar por página.
- **generaText(String partidaAcabada): String[]:** A partir de la información de una partida acabada en un formato específico, genera el usuario, la fecha, el identificador del tablero y el tiempo de la partida.

3.7. VistaRankings

Atributs:

- **BuscadorUsuari buscadorUsuari_:** Referencia al buscador de usuario.
- **ComponentSelectorMida componentSelectorMida_:** Referencia al componente selector de tamaño.
- **PartidesRanking partidesRanking_:** Referencia a la lista de partidas del ranking.

Mètodes:

- **VistaRankings(String[] informacioPartides):** Crea una vista de rankings con la información de las partidas dada.
- **addObservadorBuscador(ObservadorBuscador observador):** Añade un observador al buscador de usuario.
- **addObservadorSelectorMida(ObservadorSelectorMida observador):** Añade un observador al selector de tamaño.
- **addObservadorLlista(ObservadorLlista observador):** Añade un observador a la lista de partidas.
- **actualitzaPartides(String[] informacioPartides):** Actualiza las partidas con la información dada.

4. ConfigUsuari

Atributs:

- **JPanel panelComplet:** El panell complet de configuració d'usuari
- **JLabel labelLogo:** Etiqueta per a mostrar el logotip
- **JPanel panelDreta:** Panell per a organitzar els components a la dreta
- **JLabel canviarContrasenyaLabel:** Etiqueta per a indicar el camp de canvi de contrasenya
- **JPasswordField passwordFieldContr:** Camp de text per a introduir la nova contrasenya
- **JPasswordField passwordFieldConfContr:** Camp de text per a confirmar la nova contrasenya
- **JLabel errorLabel:** Etiqueta per a mostrar missatges d'error
- **JButton sortirButton:** Botó per a sortir de la pantalla de configuració
- **JPasswordField passwordFieldActual:** Camp de text per a introduir la contrasenya actual
- **JButton guardarButton:** Botó per a desar els canvis de configuració
- **JButton tancarSessioButton:** Botó per a tancar la sessió actual

Mètodes:

- **ConfigUsuari():** Crea una nova instància de ConfigUsuari.
- **JPanel getDefaultPanel():** Obté el panell predeterminat.
- **createUIComponents():** Crea els components de la interfície d'usuari.

5. CreakKenKenParametres

Atributs:

- **JButton CreaButton:** Botó per crear els paràmetres del KenKen
- **JButton CancelaButton:** Botó per cancel·lar la creació del KenKen
- **JLabel panel2:** Panel secundari
- **JLabel panelComplet:** Panel principal
- **JPanel labelLogo:** Etiqueta per a mostrar el logo
- **JComboBox ComboBoxGrau:** Llista desplegable per seleccionar el grau del KenKen
- **JRadioButton radioButton1:** Botó d'opció per a la suma
- **JRadioButton radioButton2:** Botó d'opció per a la resta
- **JRadioButton radioButton3:** Botó d'opció per a la multiplicació
- **JRadioButton radioButton4:** Botó d'opció per a la divisió
- **JRadioButton radioButton5:** Botó d'opció per al mòdul
- **JRadioButton radioButton6:** Botó d'opció per a l'exponenciació
- **JRadioButton radioButton7:** Botó d'opció per a la combinació de totes les operacions

Mètodes:

- **CrearKenKenParametres():** Constructor de la classe CrearKenKenParametres.
- **crearKenKen(int grau, boolean suma, boolean resta, boolean multiplicacio, boolean divisio, boolean modul, boolean exponenciacio)**: Mètode per a crear el KenKen amb els paràmetres seleccionats.
- **createUIComponents():** Mètode per a crear components personalitzats
- **getDefaultPanel():** Mètode per a obtenir el panell per defecte.

6. CtrlPresentacio

La classe CtrlPresentacio és el controlador de presentació principal de l'aplicació. Aquesta classe és responsable de gestionar la interacció entre l'usuari i l'aplicació, incloent la visualització de les vistes i la comunicació amb el controlador de domini.

Atributs:

- **int NOMBREPARTIDESLLISTA:** Nombre de partides a mostrar per pàgina en les llistes
- **int MIDAMAX**: Constant que indica la mida màxima de partida.
- **int MIDAMIN**: Constant que indica la mida mínima de partida.
- **CtrlPresentacio CPresentacio:** Instància del controlador de presentació
- **CtrlDomini CDomini:** Instància del controlador del domini
- JFrame mainFrame: Finestra principal de l'aplicació
- I**niciarSessio iniSessio:** Diàleg d'inici de sessió
- Registrarse registrar: Diàleg de registre d'usuari
- **MenuPrincipal menuPrincipal:** Pantalla del menú principal
- **CrearKenKenParametres generarKenken:** Diàleg per generar paràmetres del KenKen
- **CrearKenkenManual crearKenken:** Diàleg per crear un KenKen manualment
- **ConfigUsuari configUsuari:** Diàleg de configuració de l'usuari
- **controladorPresentacioRanking_: ControladorPresentacioRanking :** Instància del controlador de presentació del ranking.
- **controladorPresentacioPartida_: ControladorPresentacioPartida :** Instància del controlador de presentació de la partida.

Mètodes:

- **getInstance():** Retorna la instància actual del controlador.
- **run():** Mètode per iniciar l'aplicació i mostrar la finestra principal.
- **showIniciarSessio():** Mètode per mostrar la pantalla d'inici de sessió.
- **showRegistrarse():** Mètode per mostrar la pantalla de registre d'usuari.
- **showMenuPrincipal():** Mètode per mostrar el menú principal de l'aplicació.
- **showConfigUsuari():** Mètode per mostrar la pantalla de configuració de l'usuari.
- **showGenerarKenKen():** Mètode per mostrar la pantalla de generació de paràmetres del KenKen.
- **showCrearKenKen():** Mètode per mostrar la pantalla de creació manual del KenKen.
- **initJugar():** Mètode per iniciar una partida de KenKen.

- **showRanking()** : Mètode per mostrar la vista de ranking

Seguidament conté tots els mètodes per a cridar a tots els mètodes de domini, garantitzant que únicament passen del controlador principal de presentació al principal de domini.

7. IniciarSessio

Atributs:

- **JPanel panelComplet:** Panell complet de la pantalla d'iniciar sessió
- **JPanel panelIniciar:** Panell per a iniciar la sessió
- **JLabel labelLogo:** Etiqueta amb el logotip de la pantalla d'iniciar sessió
- **JTextField textFieldUsuari:** Camp de text per introduir el nom d'usuari
- **JLabel labelUsuari:** Etiqueta per indicar el camp del nom d'usuari
- **JLabel labelContrasenya:** Etiqueta per indicar el camp de la contrasenya
- **JPasswordField passwordFieldContr:** Camp de contrasenya per introduir la contrasenya
- **JButton buttonIniSessio:** Botó per iniciar la sessió
- **JButton buttonCrearCompte:** Botó per crear un compte nou
- **JLabel labelSeparador:** Etiqueta amb una imatge de separació
- **JLabel errorLabel:** Etiqueta per mostrar missatges d'error durant l'inici de sessió

Mètodes:

- **IniciarSessio():** Constructor de la classe, inicialitza els components de la pantalla d'iniciar sessió i defineix els seus comportaments
- **createUIComponents():** Mètode per crear els components d'interfície de la pantalla d'iniciar sessió
- **getDefaultPanel():** Mètode per obtenir el panell principal de la pantalla d'iniciar sessió

8. MenuPrincipal

Atributs:

- **JPanel panelComplet:** Panel que contiene todos los elementos del menú principal.
- **JLabel labelLogo:** Etiqueta que muestra el logotipo del menú principal.
- **JPanel panel2:** Panel utilizado para organizar los botones del menú principal.
- **JButton configuracioUsuariButton:** Botón para acceder a la configuración del usuario.
- **JButton crearKenkenManualButton:** Botón para acceder al creador manual de KenKen.
- **JButton jugarButton:** Botón para iniciar el juego.

- **JButton rankingButton:** Botón para acceder al ranking de partidas.
- **JButton crearKenkenAleatoriButton:** Botón para acceder al creador aleatorio de KenKen.

Mètodes:

- **MenuPrincipal():** Constructor de la clase que inicializa la interfaz del menú principal y sus acciones.
- **createUIComponents():** Método privado que inicializa los componentes de la interfaz de usuario, como el logotipo.
- **getDefaultPanel():** Método que devuelve el panel principal del menú.

9. Registrarse

Atributs:

- **JPanel panelComplet:** El panell principal que conté tots els elements de la interfície gràfica.
- **JPanel panel2:** Un panell secundari que pot contenir altres elements de la interfície gràfica.
- **JLabel labelLogo:** Una etiqueta que mostra el logotip de l'aplicació.
- **JTextField textFieldUsuari**: Un camp de text on l'usuari pot introduir el seu nom d'usuari.
- **JLabel labelUsuari:** Una etiqueta que indica el propòsit del camp de text del nom d'usuari.
- **JLabel labelContrasenya:** Una etiqueta que indica el propòsit del camp de contrasenya.
- **JPasswordField passwordFieldContr:** Un camp de contrasenya on l'usuari pot introduir la seva contrasenya.
- **JButton buttonRegistrar:** Un botó que permet a l'usuari registrar-se.
- **JButton buttonIniciarSessio:** Un botó que permet a l'usuari iniciar sessió.
- **JLabel lableComfContra:** Una etiqueta que indica el propòsit del camp de confirmació de contrasenya.
- **JPasswordField passwordFieldConfContr:** Un camp de contrasenya on l'usuari pot confirmar la seva contrasenya.
- **JLabel labelSeparador:** Una etiqueta que mostra un separador visual en la interfície.
- **JLabel errorLabel:** Una etiqueta que mostra missatges d'error o informació a l'usuari en cas que hi hagi un problema durant el registre.

Mètodes:

- **Registrarse():** Constructor que inicialitza la interfície gràfica de la pantalla de registre i defineix els comportaments dels botons.
- **void createUIComponents():** Crea i inicialitza els components personalitzats de la interfície gràfica.
- **JPanel getDefaultPanel():** Retorna el panell principal de la interfície gràfica.

10. Utils

Mètodes:

- **carregarImatge(String ruta, int amplada, int altura):** ImageIcon: Carrega una imatge des de la ruta especificada i la redimensiona a les dimensions donades. Retorna un ImageIcon de la imatge redimensionada o null si hi ha un error en carregar la imatge.

Descripció de la persistència

1. ControladorPersistencia

És el controlador principal de la capa i l'únic controlador que es relaciona directament amb la capa de domini. Utilitza el patró singleton.
Atributs:

- **- controladorPersistencia_ : ControladorPersistencia:** Instància del controlador de persistencia principal, és a dir de sí mateix, ja que segueix el patró singleton.
- **- controladorPersistenciaTauler_ : ControladorPersistenciaTauler :** Instància del controlador de persistencia dels taulers.
- **- controladorPersistenciaUsuari_ : CtrlUsuariData:** Instància del controlador de persistencia dels usuaris.
- **- controladorPersistenciaPartida_ : ControladorPersistenciaPartida :** Instància del controlador de persistencia de les partides.

Mètodes:

- **- ControladorPersistencia():** Creadora privada del controlador de persistencia que es crida quan no hi ha ja una instància. Inicialitza la resta de controladors que té com a atributs.
- **+ getInstance() : ControladorPersistencia:** Retorna la instància del controlador de persistencia principal, si no la hi ha, la crea.
- **+ carregarUltimaPartidaGuardada(nomUsuari : String) : String:** Crida la funció de mateix nom del controlador de persistencia de partides.
- **+ carregarPartidesGuardadesUsuari(nomUsuari : String) : ArrayList<String>:** Crida la funció de mateix nom del controlador de persistencia de partides.
- **+ carregarPartidesAcabadesUsuari(nomUsuari : String) : ArrayList<String>:** Crida la funció de mateix nom del controlador de persistencia de partides.
- **+ carregaPartidesAcabadesMida(mida : int) : ArrayList<String>:** Crida la funció de mateix nom del controlador de persistencia de partides.
- **+ guardarPartida(partidaGuardada : String) : boolean:** Crida la funció de mateix nom del controlador de persistencia de partides.
- **+ arxivarPartida(partidaAcabada : String) : boolean:** Crida la funció de mateix nom del controlador de persistencia de partides.
- **+ haJugat(identificadorTauler : String, nomUsuari : String) : boolean:** Crida la funció de mateix nom del controlador de persistencia de partides.

- **+ llegirTauler(identificadorTauler : String) : String:** Crida la funció de mateix nom del controlador de persistencia de taulers.
- **+ generaldentificadorlGuardaTauler(dadesTauler : String) : String :** Crida la funció de mateix nom del controlador de persistencia de taulers.
- **+ seleccionaTaulerAleatori(mida : int) : String:** Crida la funció de mateix nom del controlador de persistencia de taulers.
- **+ getUsuari(nomUsuari : String) : JsonReader:** Crida la funció de mateix nom del controlador de persistencia d'usuaris.
- **+ existeixUsuari(nomUsuari : String) : boolean:** Crida la funció de mateix nom del controlador de persistencia d'usuaris.
- **+ guardarUsuari(nomUsuari : String, dadesUsuariJson : String) : void :** Crida la funció de mateix nom del controlador de persistencia d'usuaris.

2. ControladorPersistenciaPartida

Controlador de persistencia per a Partida que s'encarrega de gestionar la càrrega i el guardat de partides.
Opera amb fitxers de text per a guardar les partides. El fitxer per guardar les partides guardades és: "data/partides/PartidesGuardades.txt".
Els fitxers per guardar les partides acabades són:
"data/partides/PartidesAcabadesGuardades.txt",
"data/partides/PartidesAcabadesMida%d.txt" on %d és la mida de la partida.
Atributs:

- **- fitxerPartidesGuardades_ : String :**Fitxer on es guarden les partides guardades**.**
- **- fitxersPartidesAcabades_ : String[] :** Vector de Strings on cada una representa un fitxer on s'emmagatzemen les partides acabades. Les que han estat guardades es guarden a l'índex [0]. Les que no han estat guardades es guarden a l'índex [n-2] on n és el mida de la partida.
- **- controladorPersistenciaPartida_ : ControladorPersistenciaPartida :** Instància del propi controlador, al seguir un patró singleton.

Mètodes:

- **- ControladorPersistenciaPartida() :** Creadora per defecte de la classe, es crida quan no hi ha ja una instància d'aquesta.
- **+ getInstance() : ControladorPersistenciaPartida :** Retorna la instància del controlador.
- **+ carregarUltimaPartidaGuardada(nomUsuari : String) : String :** Carrega l'última partida guardada per un usuari. L'usuari existeix. La informació de la partida guardada es troba al fitxer "data/partides/PartidesGuardades.txt" i està ordenada per última guardada. Utilitza el format descrit a guardaPartida() de l'apartat 2 per a llegir les dades de la partida de memòria. Utilitza el mateix format per a retornar les dades de la partida. Retorna una cadena de text amb la informació de la partida guardada.
- **+ carregarPartidesGuardadesUsuari(nomUsuari : String) : ArrayList<String> :** Aquesta és una funció que carrega totes les partides guardades per un usuari. L'usuari existeix. La informació de les partides guardades es troba al fitxer "data/partides/PartidesGuardades.txt". En utilitzar un HashSet per identificar les partides, es garanteix que no es repeteixen. Utilitza el format descrit a guardaPartida() de l'apartat 2 per a llegir les dades de la partida de memòria. Utilitza

el mateix format per a retornar les dades de la partida. Retorna una llista de cadenes de text amb la informació de les partides guardades.

- **+ carregarPartidesAcabadesUsuari(nomUsuari : String) : ArrayList<String> :** Aquesta és una funció que carrega totes les partides acabades per un usuari. L'usuari existeix. Busca a tots els fitxers de partides acabades per trobar les partides de l'usuari. Utilitza el format descrit a arxivarPartida d'aquesta mateixa classe per llegir la informació de la partida de memòria. Utilitza el format descrit a acabaPartida() de l'apartat 2 per retornar la informació. Retorna una llista de cadenes de text amb la informació de les partides acabades.
- **+ carregaPartidesAcabadesMida(mida : int) : ArrayList<String> :** Aquesta és una funció que carrega totes les partides acabades d'una certa mida. La mida és vàlida. La informació de les partides acabades es troba al fitxer "data/partides/PartidesAcabadesMida%d.txt" on %d és la mida. Utilitza el format descrit a arxivarPartida d'aquesta mateixa classe per llegir la informació de la partida de memòria. Utilitza el format descrit a acabaPartida() de l'apartat 2 per retornar la informació. Retorna una llista de cadenes de text amb la informació de les partides acabades.
- **+ guardarPartida(partidaGuardada : String) : boolean :** Aquesta és una funció que guarda una partida. La partida es guarda a l'inici del fitxer de partides guardades, assegurant així un ordre cronològic. La informació de la partida es guarda al fitxer "data/partides/PartidesGuardades.txt". Utilitza el format descrit a guardaPartida() de l'apartat 2 per a guardar les dades a memòria. Retorna true si s'ha guardat la partida, false en cas contrari.
- **+ arxivarPartida(partidaAcabada : String) : boolean :** Aquesta és una funció que arxiva una partida acabada. La partida arxivada es guarda al final del fitxer de partides acabades. La informació de la partida es guarda al fitxer "data/partides/PartidesAcabadesGuardades.txt" si havia estat guardada, o al fitxer "data/partides/PartidesAcabadesMida%d.txt" on %d és la mida de la partida. S'elimina la partida arxivada del fitxer de partides guardades. Utilitza el format descrit a acabaPartida() de l'apartat 2 per a llegir la partida. Però la guarda al fitxer com a:
  Identificador de la partida
  Identificador de l'usuari
  Identificador del tauler
  Temps total de la partida
  Mida del tauler
  Total: 5 línies. El si és guardada o no ve implicit en el nom del fitxer. Retorna true si s'ha arxivat la partida, false en cas contrari.

- **+ haJugat(identificadorTauler : String, nomUsuari : String) : boolean :** Funció de cerca per a saber si un usuari ha jugat ja un tauler. Donat un nom d'usuari i un identificador de tauler, retorna true si l'usuari ja l'ha jugat i false si no.

3. ControladorPersistenciaTauler

Controlador de persistència de taulers. S'encarrega de llegir i guardar taulers al disc. Opera sobre la carpeta "data/taulers".
Atributs:

- **- ctrTaulerData : ControladorPersistenciaTauler :** Instància de sí mateix al seguir un patró singleton.
  Mètodes:
- **- ControladorPersistenciaTauler() :** Creadora per defecte de la classe, es crida quan no hi ha ja una instància d'aquesta.
- **+ getInstance() : ControladorPersistenciaTauler :** Retorna la instància del controlador.
- **+ llegirTauler(identificadorTauler : String) : String :** Donat un identificador de tauler, el llegeix de memòria i en retorna les seves dades en el format d'entrada. Si no existeix llança excepció.
- **+ generaIdentificadorIGuardaTauler(dadesTauler : String) : String :** Aquesta funció genera un identificador únic per a un tauler basat en les seves dades. Primer, comprova si ja existeix un tauler amb les mateixes dades. Si és així, retorna l'identificador d'aquest tauler existent. Si no, genera un nou identificador, guarda el tauler al disc i retorna aquest nou identificador. Utilitza un diccionari que guarda els identificadors en funció d'un sufix creat a partir de les seves operacions, comprova que per el sufix concret del tauler que es vol guardar no existeixi ja un tauler amb les mateixes dades, si existeix retorna l'identificador d'aquest, si no en crea un de nou i el retorna.
- **+ seleccionaTaulerAleatori(mida : int) : String :** Funció que selecciona un tauler aleatori d'una mida concreta, si no existeix, retorna String buida, si existeix, retorna el seu identificador.
- **- generaSufixFitxer(dadesTauler : String) : String :** Genera un sufix per a l'identificador del fitxer del tauler a partir de les dades del tauler.
  Format:
  "-mida-operacio1_operacio2_..._operacioN"
  Retorna el sufix.
- **- esborraTauler(identificadorTauler : String) : void :** Esborra del disc el tauler amb aquell identificador.
- **- inicialitzaDiccionariTaulers() : void :** Inicialitza el diccionari de taulers a partir dels fitxers del disc. També actualitza l'últim identificador dels taulers.

4. CtrlUsuariData

Atributs:

- **- ctrUsuariData : CtrlUsuariData :** Instància singleton de CtrlUsuariData.
- **- pathUsuaris : String :** Ruta del directori on es guarden les dades dels usuaris.

Mètodes:

- **- CtrlUsuariData() :** Constructor privat de la classe CtrlUsuariData.
- **+ getInstance() : CtrlUsuariData :** Retorna la instància singleton de CtrlUsuariData. Si no existeix, la crea.
- **+ getUsuari(nomUsuari : String) : JsonReader :** Retorna un JsonReader per a les dades de l'usuari amb el nom d'usuari proporcionat. Si no el troba llença excepció.

- **+ existeixUsuari(nomUsuari : String) : boolean :** Comprova si existeix un usuari amb el nom d'usuari proporcionat.
- **+ guardarUsuari(nomUsuari : String, dadesUsuariJson : String) : void :** Guarda les dades de l'usuari en format JSON en un fitxer. Llença IOException si hi ha un error d'entrada/sortida.

Estructura de les dades:

1. Usuaris

Les dades estan emmagatzemades en 1 arxiu, el qual conté tots els usuaris. L'arxiu està ordenat amb un usuari per línia seguit de la seva contrasenya.

2. Taulers

Els taulers estan ordenats tal com està indicat al fitxer "formatokenken.pdf". D'aquesta forma tenim 1 fitxer per cada Tauler diferent.

3. Partides

a. Partides acabades

Una partida acabada és aquella que és correcta. Es diferencien dos casos, les que han acabat i podran entrar al ranking, i les que no, ja sigui perquè han estat guardades (no s'han acabat en una sessió de joc) o perquè s'ha utilitzat pistes per a la seva resolució.
El format de les dades que tenen aquestes partides en la comunicació entre classes és la següent:

Identificador de la partida:String
Identificador de l'usuari:String
Identificador del tauler:String
Temps total de la partida:int
Grau del tauler:int
Si ha estat guardada o no:boolean

però concatenat tot com a string ("identificadorPartida\nIdentificadorUsuari\nIdentificadorTauler\nTempsPartida\ngrau\nguardada\n") Per separar-ho només caldria fer un split dels \n. Sempre tindrà 6 línies.
Tot i així, a memòria secundària l'emmagatzematge per a millor eficiència segueix un format diferent.
Primer de tot, les partides acabades es poden emmagatzemar en 8 fitxers diferents.
Les que no són vàlides pel ranking (boolean guardada = true) es guarden al fitxer *"PartidesAcabadesGuardades.txt".*

Les que ho són, es guarden en fitxers en funció del seu grau, on el fitxer sempre té el nom "*PartidesAcabadesGrau%d.txt*" on %d és el grau de la partida.
El format que segueixen però és el mateix:

Identificador de la Partida
Identificador de l'usuari
Identificador del Tauler
Temps total de la partida
Grau del tauler

El boolean que s'inclou en el moment de carregar-les de memòria ve implicita en el fitxer d'on s'extreu, i tot i que no caldria posar el grau del tauler en les partides que poden participar als rankings, s'ha fet per unificar formats.

b. Partides guardades

Una partida guardada és aquella que s'ha guardat en algun punt del joc a memòria secundària, o que s'ha resolt utilitzant pistes en algun punt. Aquestes segueixen el mateix format tant per la comunicació entre classes com per a l'emmagatzematge a memòria, i és el següent:

Identificador de la partida
Identificador del tauler
Temps total de partida
Grau del tauler
Valor de totes les caselles en el moment de guardat, separant files per \n i valors amb espais.

Per tant una partida guardada ocupa sempre 4 + grauTauler línies.
A memòria es guarden a "*PartidesGuardades.txt"* i òbviament s'inicialitzaran amb el boolean de guardat a true. Una característica a tenir molt en compte és que es guarden cronològicament, l'última partida guardada estarà sempre a l'inici.

c. Partida en joc/Estat

El que es denomina en el programa estat de la partida, és l'estructura de dades que representa els valors de les caselles d'una partida en un instant concret d'una partida que s'està jugant, on el format és:

'x'' ''x'' ''x''\n'

…
‘x’’ ‘’x’’ ‘’x’’\n’

on cada x representa el valor de la casella.
Hi ha tantes files com el grau del tauler i tantes ‘x’ per fila (és a dir columnes) com el grau. Tot això concatenat com una string.