Especificació

Oscar Galera Alfaro
Moises Saus Ten

Moduls de dades

Crono

```
/**
 * @author Moises
 * @Class Crono
 * @brief Cronometre amb representacio gràfica amb 00:00:000 minuts:segons:milesimes
 */

    /**
     * @brief Constructor
     * @post Crono preperat per mostrar per pantalla i iniciar.
     */
    public Crono()

    /**
     * @brief El crono inicia el seu compte
     * @post Cronometre iniciat
     */
    public void iniciarCrono()

    /**
     * @brief Crono para el seu compte
     * @post Cronometre parat.
     */
    public void pararCrono()
```

MARCADOR

```
/**
 * @author Moises
 * @Class Marcador
 * @brief Encarregada de mostrar la imatge i la puntuacio de un Personatge durant una partida.
 */

    /**
     * @brief Constructor de Marcador
     * @post Marcador preparat per mostrar per pantalla amb imatge img i puntuacio 0000
     */
    public Marcador(ImageIcon img)

    /**
     * @brief Modifica el marcador de punts de Personatge.
     * @post Es mostra per pantalla el valor de novaPuntuacio.
     */
    public void modificaPuntuacio(int novaPuntuacio)

    /**
     * @brief Mostra per pantalla la puntuacio p.
```

```
     * @post p es mostrat per pantalla.
     */
    public void canviarPuntuacio(int p)
```

FAlta

```
/**
 * @author oscar
 * @brief
 * Formulari per donar d'alta a un nou usuari.
 * Cada usuari haura de especificar:
 *      -Nom d'usuari
 *      -Password
 *      -Imatge de perfil (opcional)
 *
 * @invariant
 * just abans de guardar l'usuari l'atribut imatgeRedimensionada ha de contenir
 * l'imatge de perfil de l'usuari;
 */

    /**
     * @pre --
     * @post s'ha creat el formulari d'alta;
     */
    public FAlta()

    /**
     * @pre el frame no s'esta mostrant;
     * @post s'ha mostrat el frame;
     */
    public void mostrarFrame()
```

FeditorLaberint

```
/**
 * @author oscar
 * @brief
 *  Pantalla que ens permet crear i dissenyar nous mapes
 *
 *  En la part esquerra hi ha el conjunt de elemnts que es poden afegir al laberint
 *  i nomes cal seleccionar l'element i fer click en la posicio on es vol afegir
 *  Un cop creada la pantalla aquesta es validada i exportada en format .txt.)
 *
 *  @invariant
 *  laberint != null, el costat sempre ha de ser > Utils.Constants.MINIM_COSTAT_LABERINT i
 *  la matriu ha de ser cuadrada de costat x costat
 */

    /**
     * @pre costat > Utils.Constants.MINIM_COSTAT_LABERINT;
     * @post em creat l'editor de laberints per un laberint de mida costat
```

```
     */
    public FEditorLaberint(int costat)

    /**
     * @pre el frame no s'ha mostrat
     * @post s'ha mostrat el frame;
     */
    public void mostrarFrame()
```

FframeAmbLog

```
/**
 * @author oscar
 *
 * @brief
 * Frame amb un menuBar que conté un menuItem per obre la pantalla de log.
 */

    /**
     * @pre --;
     * @post s'ha el frame amb log.
     */
    public FFrameAmbLog()
```

FhistoricUsuari

```
/**
 * @author oscar
 *
 * @brief
 * pantalla que utilitzem per mostrar l'historic d'un usuari
 * que està format per els nivells que ha superat i de cada un d'aquests
 * el nombre de punts que ha fet.
 */

    /**
     * @pre puntuacions conté les puntuacions de cada nivell on puntuacions[1]
     * conté el nombre de punts que s'han fet en aquest nivell.
     * @post em creat la pantalla de historic.
     */
    public FHistoricUsuari(String nomUsuari, Icon imatge, int []puntuacions)

    /**
     * @pre el dialeg no s'ha mostrat.
     * @post em mostrat el dialeg.
     */
    public void mostrarFrame()
```

Flog

```
/**
 * @author oscar
 *
 * @brief
 * Pantalla per mostrar el log de l'aplicació;
 *
 * Aquesta pantalla permet:
 *      - Mostrar tot el log.
 *
 *      - Exportar el log en format text (.txt)
 *
 *      - Filtrar els missatges del log de un tipus concret
 *          - DEBUG.
 *          - WARNING.
 *          - ERROR.
 *
 *      - Mostrar l'últim missatge afegit al log.
 */

    /**
     * @pre --
     * @post em creat la pantalla de log.
     */
    public FLog()

    /**
     * @pre --
     * @post em mostrar la pantalla de log.
     */
    public void mostrarFrame()
```

Flogin

```
/**
 * @author oscar
 * @brief
 * Pantalla inicial de l'aplicacio, operacions que es poden fer:
 *      -ENTRAR :
 *          s'ha de entrar usuari i contrasenya, la contrasenya es xifra i es mira que dita
 *          combinació existeixi en la B.D, en cas que sigui correcte llavors es carrega l'usuari
 *          (de forma estàtica ja que només podrà entrar un usuari alhora) i del contrari
 *          es mostra un missatge diguent "usuari o contrasenya incorrecte"
 *
 *      -ALTA :
 *          permet donar d'alta a un nou usuari.
 *
 *      -RANKING :
 *          mostra tots els usuaris registrats per ordre de punts (on el primer
 *          es el que ha fet mes punts)
 *
```

```
*      -SORTIR :
*          tanquem l'aplicacio.
*
* @invarinat
* usuari valdrà null fins que s'accedeixi al menú.
*/

   /**
    * @pre --
    * @post s'ha creat el dialeg per entrar.
    */
   public FLogin()

   /**
    * @pre el dialeg no s'esta mostrant.
    * @post em mostrat el dialeg.
    */
   public void mostrarFrame()

   /**
    * @pre --
    * @post si encara no s'ha entrat a la aplicació es retorna null
    * altrament es retorna l'usuari.
    */
   public static Usuari obtenirUsuari()
```

Fmenu

```
/**
 * @author oscar
 * @brief
 * menu principal del joc on es mostren les diferents possibilitats per jugar, on:
 *      Aventura (AVANÇA EN EL JOC) :
 *              Continuem la partida des de alla on es va deixar.
 *
 *      Provar mapa (NO AVANÇA EN EL JOC):
 *              Permet importar un fitxer en format text que conte un mapa, un cop
 *              importat es valida i si tot es correcte s'inicia una partida amb
 *              el mapa importat.
 *
 *      Crear mapa
 *              Ens mostra un editor per crear els nostres propis mapes. Aquests
 *              mapes mes tart es podran importar per provar-los.
 *
 *      Sortir
 *              Adeu.
 */

   /**
    * @pre --;
    * @post em creat el dialeg de menú.
    */
   public Fmenu()
```

```
    /**
     * @pre el dialeg no està visible.
     * @post em mostrat el dialeg.
     */
    public void mostrarFrame()
```

Fpartida

```
/**
 *
 * @author Moises
 * @brief Encarregada de mostrar per pantalla el desenvolupament de una partida.
 */
```

Franking

```
/**
 * @author oscar
 *
 * @brief
 * Llistat amb els usuaris que tenen més punts ordenats pel nombre total de punts
 * obtinguts en el mode aventura.
 *
 * Al seleccionar un usuari es mostra el seu historic.
 */

    /**
     * @pre --
     * @post em creat el ranking per els Utils.Constants.TOP_N_DEL_RANKING usuaris
     * amb més punts.
     */
    public Franking()

    /**
     * @pre el dialeg no s'ha mostrat.
     * @post el dialeg s'esta mostrant.
     */
    public void mostrarFrame()
```

IpintadorLaberint

```
/**
 *
 * @author Moises
 * @brief Descriu els metodes que necessiten els objectes Laberint
 * per a mostrar-se per pantalla.
```

```
 * @details Laberint es representa gràficament per cel·les, on cada cel·la representa un posció
 * de la matriu bidimensional que conté el Objecte Laberint.
 */
public interface IPintadorLaberint extends KeyListener


    /**
     * @brief Assignacio de controlador de teclat.
     * @pre controlador != null.
     * @post La representacio Grafica te assignat un controlador de teclat.
     */
    public void assignarControladorTeclat(KeyListener controlador)

    /**
     * @brief Mostra per pantalla el desplaçament de un objecte Personatge.
     * @param pOrigen Punt on es troba el Personatge
     * @param direccio Direcció a on es mou el Personatge
     * @param imatge Representació grafica del personatge
     * @pre pOrigen != null.
     * pOrigen es un Punt vàlid.
     * El Punt generat per pOrigen + desplaçament és vàlid.
     * direcció != null && direcció != EDireccio.QUIET.
     * imatge != null.
     * @post pOrigen no conté cap imatge.
     *       La cel·la corresponent a la direccio desde pOrigen conté imatge
     */
    public void pintarMovimentPersonatge(Punt pOrigen, EDireccio direccio, ImageIcon imatge)


    /**
     * @brief Mostra per pantalla el desplaçament de un objecte Item.
     * @details En el moviment del Item s'ha de tindre en compte el EElement que contenia
     * la cel·la abans de que estigues el Item. Aques EElement és el element a restaurar.
     * @param pOrigen Punt desde on s'ha efectuat el moviment
     * @param direccio Direcció a on es mou el Item
     * @param imatge Representació grafica de EElement a restaurar
     * @pre pOrigen != null.
     * pOrigen es un Punt vàlid.
     * El Punt generat per pOrigen + desplaçament és vàlid.
     * direcció != null && direcció != EDireccio.QUIET.
     * imatge != null.
     * @post La cel·la corresponent a la direccio desde pOrigen conté la imatge de pOrigen.
     * pOrigen conté la imatge del EElement a restaurar
     *
     */
    public void pintarMovimentItem(Punt pOrigen, EDireccio direccio, ImageIcon imatge)


    /**
     *@brief Mostra per pantalla un item dintre del Laberint
     *@post La cel·la amb coordenades pNouItem conte la imatge de item
     */
```

```
    public void pintarNouItem(Punt pNouItem, EElement item)

    /**
     * @brief Construcció de la interficie grafica que representa graficament laberint
     * @param laberint Laberint a representar
     * @pre laberint != null
     * @post Representacio grafica preperada per mostrar per pantalla
     */
    public void pintarLaberint(Laberint laberint)

    /**
     * @brief retorna la mida amb la que es pinten les imatges que representen els EElements d'un
Laberint.
     * @return Mida de les representacions grafiques de EElements.
     */
    public Dimension obtenirMidaImatge()

    /**
     * @brief Mostra la representacio de EElement.SORTIDA en les coordenades corresponents a
pSortida
     * @post Representació de EElement.SORTIDA es veu per pantalla.
     */
    public void pintarSortida(Punt pSortida)
```

IpintadorPartida

```
/**
 *
 * @author Moises
 * @brief Descriu els metodes necessaris per a que objecte Partida pugui
 * mostrar el desenvolupament d'una partida per pantalla.
 */
public interface IPintadorPartida

    /**
     * @brief Mostra per pantalla la puntuacio d'en Pacman.
     * @post La puntuacio d'en Pacman que es mostra per pantalla esta actualitzada.
     */
    public void pintarPuntsPacman(int punts)


    /**
     * @brief Mostra per pantalla la puntuacio de l'enemic.
     * @post La puntuacio de l'enemic que es mostra per pantalla esta actualitzada.
     */
    public void pintarPuntsEnemic(int punts)

    /**
     * @brief Mostra per pantalla quin tipus de Item hi ha en el Laberint.
     * @post imatge es mostra per panatalla.
     */
    public void pintarItemPartida(ImageIcon imatge)
```

```
    /**
     * @brief Mostra per pantalla el Item que te en Pacman.
     * @post imatge es mostra per pantalla.
     */
    public void pintarItemPacman(ImageIcon imatge)

    /**
     * @brief Mostra per pantalla el Item que l'enemic.
     * @post imatge es mostra per pantalla.
     */
    public void pintarItemEnemic(ImageIcon imatge)


    /**
     * @brief Construcció de tots els elements necessaris per mostrar partida per pantalla.
     * @post La partida es mostrada per pantalla.
     */
    public void pintarPartida(Partida partida)

    /**
     * @brief Mostra per pantalla el resultat de la partida al jugador.
     * @param guanyat Diu si jugador ha guanyat o no la partida
     * @post
     * Es mostra un dialeg per comunicar al jugador si ha guanyat o no.
     * Partida deixa de ser visible.
     */
    public void pintarFinalPartida(boolean guanyat)

    /**
     * @brief Inicialitza els components de una partida
     */
    public void pintarIniciPartida()

    /**
     * @brief Comunicacio unidireccional d'events de IPintadorPartida cap a partida.
     */
    public void assignarPartida(Partida partida)
```

Plaberint

```
/**
 *
 * @author Moises
 * @brief Encarregada de mostrar per pantalla objectes del tipus Laberint
 */
```

BuscadorCami

```
/**
 *
 author Moises
```

```
 brief Classe que implementa l'algorisme de Backtracking per a trobar un
 * camí que maximitzi la distancia desde un punt inici respecte a un punt enemic
 */

    /**
     brief Retorna el cami amb maxima distancia desde personatge respecte enemic.
     param personatge Punt inicial.
     param enemic Punt del qual ens volem distanciar.
     pre personatge != null, enemic != null
     return Retorna els moviments necessaris per a maximitzar la distancia desde personatge respecte
a enemic
     */
    public HistoricMoviments buscaCamiMaxim(Punt personatge, Punt enemic)
```

Solucio

```
/**
 *
 author Moises
 brief Classe encarregada de guardar els valors de la solució composta per a
 * l'algorisme de Backtracking
 */

    /**
     *
     brief Retorna cert si la solució es completa
     */
    public boolean esSolucioCompleta()

    /**
     brief Assignacio del punt del cual volem maximitzar la distancia
     pre enemic != null
     param enemic Punt del qual volem fugir.
     */
    public void assignaPuntAFugir(Punt enemic)

    /**
     *
     brief Diu si solucio actual es millor solucio que s.
     param s Solucio a comparar amb objecte actual
     return Retorna cert si Solucio actual es millor solucio que s.
     */
     public boolean esMillorSolucio(Solucio s)

    /**
     *
     brief Diu si el candidat es acceptable
     param c Candidat a la solucio parcial
     pre c != null.
     return Retorna cert si c encara no ha estat processada
     */
    public boolean acceptable(Casella c)
```

```
    /**
     brief Anota el candidat c a la solucio actual.
     pre c != null
     post Afegeix el candidat c a la estructura de dades del objecte actual. Candidat c forma part de la solució. nivell++.
     param c Candidat actual a la solucio parcial
     */
    public void anotar (Casella c)

    /**
     brief Desanota el candidat c de la solució actual.
     pre c != null
     post Elimina el candidat c de la estructura de dades del objecte actual. Candidat c no forma part de la solució. nivell--.
     param c Candidat c es eliminat de la solucio parcial.
     */
    public void desanotar(Casella c)

    /**
     brief Retorna cert si afegim el candidat c a la solucio actual i aquesta pot ser millor solucio que opt.
     param opt Solucio optima fins al moment
     param c Candidat actual
     return Retorna cert si la distancia entre c i enemic mes els passos restants, es mes gran que la distancia obtinguda en opt.
     */
    public boolean potSerMillor(Solucio opt, Casella c)

    /**
     brief Retorna les possibles Caselles a les que podem accedir desde la Casella amb coordenades p.
     pre p != null. Anteriorment s'ha d'haver assignat un enemic.
     param p Punt actual en la construccio del camí màxim.
     return Retorna una llista de 0..4 candidats que estan dins del tauler i no son paret, ordenats ascendentment per distancia al enemic.
     */
    public LlistaOrdenadaCandidats generarCandidats(Punt p)

    /**
     brief Retorna els n Moviments corresponents al cami calculat per l'algorisme de Backtracking. On n = PROFUNDITAT -1 (n caselles, n-1 moviments)
     pre cami no esta buit.
     */
    public HistoricMoviments generaRuta()

    /**
     brief Deixa al objecte actual preperat per a tornar a fer l'algorisme de Backtracking
     post Objecte actual reiniciat a estat inicial.
     */
    public void reset()
```

Casella

```
/**
 *
 author Moises
 brief Conté la informació heuristica necessaria per a poder implementar els algoritmes
 * de backTracking i AStar.  La Heuristica utilitzada en el algorisme de AStar es:
 * F = profunditat + distanciaAlObjectiu (on distanciaAlObjectiu, es la distancia Manhattan
 * entre dos punts dintre de un Laberint format per cel·les)
 */

   /**
    *
    brief Assigna un pare al objecte actual.
    pre _pare != null
    post Objecte actual te un punter a _pare.
    */
   public void assignarParent(Casella _pare)

   /**
    *
    brief Retorna el pare del objecte actual
    */
   public Casella obtenirParent()

   /**
    brief Assigna la distancia al objecte actual.
    */
   public void assignarDistanciaAlObjectiu(int n)

   /**
    *
    brief Retorna la distancia del objecte actual
    */
   public int obtenirDistanciaAlObjectiu()

   /**
    *
    brief Retorna la profunditat del objecte acutal.
    */
   public int obtenirProfunditat()

   /**
    *
    brief Retorna el Punt del objecte actual.
    */
   public Punt obtenirPunt()

   /**
    brief Marca el objecte actual com a processat
    post Objecte Casella processat
    */
   public void processat()
```

```
    /**
     brief Marca el objecte actual com a no processat
     post Objecte Casella no processat
     */
    public void noProcessat()

    /**
     brief Retorna cert si l'objecte actual ha estat processat, altrament fals.
     */
    public boolean haEstatProcessat()

    /**
     brief Es reinicien els valors heuristics.
     post Els valors heurisitics reiniciats als valors inicials.
     */
    public void reset()
```

GestorCamins

```
/**
 *
 author Moises
 brief Encarregada de gestionar les cerques de camins sobre un objecte de tipus Laberint.
 */

    /**
     brief Retorna un camí mínim entre dos punts.
     pre origen != null, desti != null.
     return Retorna el minim de moviments que cal fer per anar de origen a desti, en un objecte de
tipus Laberint.
     */
    public HistoricMoviments trobarCamiMinim(Punt origen, Punt desti)

     /**
      brief Retorna un cami que maximitza la distancia respecte a puntAFugir.
      pre inici != null, puntAFugir != null.
      param inici Punt de partida
      param puntAFugir Punt del qual volem maximitzar la distància
      return Retorna n-1 moviments que maximitzan la distancia desde inici respecte puntAFugir. On
n = profunditat del backtracking.
      */
    public HistoricMoviments trobarCamiMaximitzarDist(Punt inici, Punt puntAFugir)
```

LlistaordenadaCandidats

```
/**
 *
 author Moises
 brief Estructura ordenada de Caselles (candidats) per a als algoritmes de Backtracking i AStar.
 */
```

```
/**
 brief Retorna la Casella de menys pes.
 pre LlistaOrdenadaCandidats no buida.
 return Retorna la primera Casella de LlistaOrdenadaCandidats.
 */
public Casella obtenirPrimer()

/**
 brief Retorna la Casella de mes pes.
 pre LlistaOrdenadaCandidats no buida.
 return Retorna la ultima Casella de LlistaOrdenadaCandidats
 */
public Casella obtenirUltim()

/**
 brief Elimina totes les Caselles de LlistaOrdenadaCandidats.
 post LlistaOrdenadaCandidats buida.
 */
public void clear()

/**
 brief Insercio de c de forma ordenada a LlistaOrdenadaCandidats.
 post LlistaOrdenadaCandidats conté la Casella c.
 */
public void afegir(Casella c)
/**
 * @brief Elimina c de LlistaOrdenadaCandidats.
 * @post LlistaOrdenadaCandidats no conté c.
 */
public void eliminar(Casella c)

/**
 * @brief Diu si LlistaOrdenadaCandidats no te cap element.
 * @return Retorna cert si LListaOrdenadaCandidats no conté cap Casella. Altrament retorna fals.
 */
public boolean esBuida()

/**
 * @brief Diu si LlistaOrdenadaCandidats conté la Casella c.
 * @return Retorna cert si c esta continguda en LlistaOrdenadaCandidats. Altrament retorna fals.
 */
public boolean conteElement(Casella c)

/**
 * @brief Diu el numero de objectes Casella continguts a LlistaOrdenadaCandidats.
 * @return Retorna el numero de objectes Casella continguts a LlistaOrdenadaCandidats.
 */
public int mida()
```

CONTROLADORS PACMAN

ControladorTeclat

```
/**
 * @author oscar
 *
 * @brief
 * Controlador per dirigir en pacman amb les fletxes del teclat
 *
 * @invariant
 * en pacman ha de deixar de ser null just abans de començar a controlar-lo.
 */
public class ControladorTeclat implements IControlador, KeyListener
```

Icontrolador

```
/**
 * @author oscar
 * @brief
 * Interficie que ens crea una capa de abstracció de com controlar a en pacman.
 *
 * És indiferent el dispositiu amb el qual el volem controlar ja que només necessitem
 * saber quina direcció representa l'event sobre el periferic.
 */

   /**
    * @pre --
    * @post em assignat en pacman al controlador "això és necessari perquè el controlador
    * cal que es comuniqui amb en pacman per dir-li la nova direcció que ha de pendre".
    */
   public void assignarPacman(Pacman pacman)
```

ENUMERACIONS

Edireccio

```
/**
 *
 * @author oscar
 * @brief Enumeracio de les diferents direccions que pot prendre un ItemMobile
 * per desplaçar-se per el Laberint.
 * @details Laberint es representat per cel·les, on cada cel·la te unes
 * coordenades que son representades per el objecte Punt.
 *
 *                        COLUMNES
 *     ---------------------------------------------------------------------
 *  F  |  Punt(0,0)  |  Punt (0,1)  |  Punt (0,2) | .........| Punt (0,n-1)|
 *  I  ---------------------------------------------------------------------
 *  L  |  Punt(1,0)  |  Punt (1,1)  |  Punt (1,2) | .........| Punt (1,n-1)|
 *  E  ---------------------------------------------------------------------
 *  S  |  Punt(2,0)  |  Punt (2,1)  |  Punt (2,2) | .........| Punt (2,n-1)|
 *     ---------------------------------------------------------------------
 *     |   ......... | .........    | .........   | ........ | ........... |
 *     ---------------------------------------------------------------------
 *     | Punt(n-1,0) | Punt (n-1,1) | Punt (n-1,2)| ........|Punt (n-1,n-1)|
```

```
 *     ---------------------------------------------------------------
 *
 * 0 en les files es la maxima posicio a la esquerra.
 * 0 en les columnes es la maxima posicio amunt.
 * n-1 en les files es la maima posició a la dreta.
 * n-1 en les columnes es la maxima posicio avall.
 *
 *
 * Els moviments només s'efectuen en un eix, eix de les files (esquerra i dreta)
 * o eix de les columnes( amunt i avall), es a dir No existeixen moviments
 * en diagonal, per tant els moviments són amunt, avall, esquerra, dreta, quiet.
 */

    /**
     * @brief Retorna el increment que s'ha d'aplicar a la coordenada
     * columna de un Punt per efectuar un moviemnt en una direccio EDireccio
     * actual, dins de Laberint .
     */
    public int obtenirIncrementColumna()

    /**
     * @brief Retorna el increment que s'ha d'aplicar a la coordenada
     * fila de un Punt per efectuar un moviemnt en una direccio EDireccio
     * actual, dins de Laberint .
     */
    public int obtenirIncrementFila()

    /**
     * @brief Retorna el moviemt oposat al actual.
     * @return EDireccio oposat al actual.
     */
    public EDireccio obtenirMovimentInvers()

    /**
     * @brief Donada una EDireccio retorna totes les EDireccions possibles menys
EDireccio.QUIET i _direccio.
     * @return Vector que conté totes les EDireccions possibles menys EDireccio.QUIET i _direccio.
     */
    public static EDireccio[] obtenirRestaDireccions(EDireccio _direccio)

    /**
     * @brief Donats dos punts retorna la direccio correcte per anar de origen a desti.
     * @pre origen i desti son adjacents
     * @return EDireccio corresponent al desplaçament de origen a desti
     * @exception ExceptionPunt si origen i desti no son adjacents.
     */
    public static EDireccio obtenirDireccio(Punt origen, Punt desti )
```

Eelement

```
/**
 *
```

```
* @author oscar
* @brief Enumeracio dels diferents elements que conte un objecte del tipus
* Laberint.
*/

   /**
    * @brief Retorna el caracter que representa a EElement
    * @details S'utilitza per a mostrar el EElement en consola.
    * @return Carater que representa a EElement.
    */
   public char obtenirLletraRepresentacio()

   /**
    * @brief Retorna la imatge que representa a EElement.
    */
   public ImageIcon obtenirImatge()

   /**
    * @brief Retorna el identificador de EElement.
    * @details S'utilitza per identificar els EElements a l'hora de tractar
    * fitxers.
    */
   public final int obtenirId()

   /**
    * @brief Diu si Objecte actual es un Fantasma.
    * @return Cert si Objecte actual es del tipus FANTASMA1, FANTASMA2, o FANTASMA3
    */
   public boolean esEnemic()

   /**
    * @brief Redimensiona la imatge que te associada cada EElement
    * @pre px > 0.
    * @post imatges de EElements redimensionades a mida (px x px).
    */
   public static void redimensionarImatges(int px)

   /**
    * @brief Retorna el EElement amb identificador = id.
    * @return EElement amb identificador = id.
    */
   public static EElement buscarElementPerId(int id)
```

ElaberintPredefinits

```
/**
 *
 * @author oscar
 * @brief laberints que tenim predefinits en l'aplicació.
 */
```

EfinalitzarPartida

```
/**
 * @author oscar
 *
 * @brief
 * Excepció per quan hi ha algun problema al finalitzar la partida.
 */
```

EformatLaberint

```
/**
 * @author oscar
 *
 * @brief
 * Excepció per quan el laberint no té un format correcte.
 */
```

EgeneradorLaberint

```
/**
 * @author oscar
 * Excepció per quan hi ha hagut algun problema en el procés de generació de laberints.
 */
```

Ehistoric

```
/**
 * @author oscar
 * Excepció que indica que hi ha hagut un problema amb l'historic d'algun personatge.
 */
```

EhistoricBuit

```
/**
 * @author oscar
 *
 * @brief
 * Excepció que es llença quan es vol accedir al historic i aquest està buit.
 */
```

EiniciarPartida

```
/**
 * @author oscar
 *
 * @brief
 * Excepció per quan hi ha algun problema al iniciar la partida.
 */
```

EitemMovibleIniciat

```
/**
 * @author oscar
 * @brief
```

```
 * Excepció per quan s'ha iniciat un item que ja estava iniciat.
 */
```

Elaberint

```
/**
 * @author oscar
 * Excepció per quan hi ha hagut algun problema amb el laberint.
 */
```

Epartida

```
/**
 * @author oscar
 *
 * @brief
 * Excepció per quan hi ha hagut algun problema en l'evolució de la partida.
 */
```

EbuscadorCamins

```
/**
 *
 * @author Moises
 * @brief Excepcio que es llença en la cerca de Camins
 */
```

Epunt

```
/**
 *
 * @author Moises
 * @brief Excepcio que es llença en les operacions amb objectes Punt
 */
```

Historic_moviments

HistoricMoviments

```
/**
 * @author oscar
 * @brief
 * Historic de moviments de caràcter FILO
 *
 * @invariant
 * nElements >= 0
 */

    /**
     * @pre --;
     * @post s'ha creat un nou historic de moviments;
     */
    public HistoricMoviments()

    /**
```

```
     * @pre --;
     * @post s'ha afegit un nou moviment al historic;
     */
    public void afegirMoviment(EDireccio nouMoviment)

    /**
     * @pre l'historic no està buit;
     * @post s'ha retornat i eliminat l'últim element afegit al historic;
     */
    public EDireccio eliminarMoviment()

    /**
     * @pre l'historic no està buit;
     * @post s'ha retornat l'ultim moviment afegit;
     */
    public EDireccio obtenirUltimMoviment()

    /**
     * @pre --;
     * @post diu si l'historic està buit;
     */
    public boolean esBuida()

    /**
     * @pre --;
     * @post s'ha retornat el nombre d'elements que contè l'historic;
     */
    public int obtenirMida()
```

Pila

```
/**
 * @author oscar
 * @brief
 * Estructura de dades de tipus FILO on
 * afegir té cost O(1)
 * cim té cost O(1)
 * desenpilar té cost O(1)
 * esBuida té cost O(1)
 *
 * @invariant
 * mentre la pila estigui buida cim ha de ser null altrament no pot ser null.
 */

    /**
     * @pre: --;
     * @post: dada està en el cim de la pila, COST O(1);
     */
    public void afegir(T dada)

    /**
     * @pre: --;
     * @post: em retornat el cim de la pila, COST O(1);
```

```
     */
    public T cim()

    /**
     * @pre: la pila no és buida;
     * @post: em desenpilat el cim de la pila, COST O(1);
     */
    public T desenpilar()

    /**
     * @pre: --;
     * @post: diu si la pila està buida, COST O(1);
     */
    public boolean esBuida()
```

Laberint

```
/**
 * @author oscar
 * @brief
 * class que contè un tauler amb elements que es poden desplasar, no coneix
 * en cap moment l'estat el seu estat sino que només coneix la seva representació.
 *
 * El laberint s'ha de poder comunicar amb la partida per noficiar quan no queden
 * monedes o quan s'ha obtingut un item (mongeta, patins o monedes x 2).
 *
 * També caldrà que pugui enviar missatges a un pintador per així poder representar-se de forma gràfica.
 *
 * @invariant
 * nMonedes conté en tot moment el nombre de monedes que hi ha en el laberint,
 * costat >= Utils.Constants.MINIM_COSTAT_LABERINT, tauler és una matriu cuadrada de costat x costat,
 * pintador no pot ser null i nMonedesPerItem > 0
 */

    /**
     * @pre el fitxer existeix i té un format correcte;
     * @post em creat un laberint a partir del fitxer que forma part de la partida i
     * que la seva representació gràfica serà sobre pintadorLaberint;
     */
    public Laberint(String fitxer,
                    Partida partida,
                    IPintadorLaberint pintadorLaberint) throws EFormatLaberint

    /**
     * @pre elements ha de ser una matriu cuadrada;
     * @post em creat un laberint a partir de elements i forma part de partida;
     */
    public Laberint(EElement elements[][], Partida partida)


    /**
```

```
     * @pre: --;
     * @post: retornem si la posició és valida dins del tauler, per valida entenem
     * que no surt del rang del tauler i no és paret;
     */
    public boolean posicioValida(Punt posicio)


    /**
     * @pre: --;
     * @post: retornem l'element ubicat en posició, si la posició no és valida es retorna PARET;
     */
    public EElement obtenirElement(Punt posicio)

    /**
     * @pre en origen hi ha un item, origen desplaçat per direccio produeix una
     * posició valida i direccio != QUIET;
     * @post en cas que origen desplaçat per direcció sigui legal (per legal
     * entenem que no trapitja un enemic ni a en pacman)
     *
     * llavors: em desplasat item de origen al punt generat amb la direcció i en origen
     *          hi em restaurat elementARestaurar, em retornat l'element capturat;
     *
     * altrament, el moviment no és legal i
     *          retornem null;
     */
    public synchronized EElement moureItem(Punt origen, EDireccio direccio, EElement
elementARestaurar)

    /**
     * @pre en origen hi ha un personatge, origen desplaçat per direccio produeix
     * una posició valida i direccio != QUIET;
     * @post en cas que origen desplaçat per direcció sigui legal (per legal
     * entenem que no trapitja un enemic ni a en pacman a no ser que tingui super poders)
     *
     * llavors: em desplasat personatge de origen al punt generat amb la direcció i en origen
     *          hi em deixat RES, em retornat l'element capturat;
     *
     * altrament, el moviment no és legal:
     *          retornem null;
     */
    public synchronized EElement mourePersonatge(Punt origen, EDireccio direccio, ImageIcon
imatge, boolean superPoders)

    /**
     * @pre --
     * @post retornem la posició on hi ha en pacman;
     */
    public Punt obtenirPosicioPacman()


    /**
     * @pre --;
```

```
     * @post em retornat la posició del tauler on hi ha l'enemic;
     */
    public Punt obtenirPosicioInicialEnemic()


    /**
     * @pre --;
     * @post em assignat un controlador de teclat asociat al pintador;
     */
    public void assignarControladorTeclat(KeyListener listener)

    /**
     * @pre --;
     * @post em retornat la mida del costat del laberint;
     */
    public int obtenirMidaCostatTauler()

    /**
     * @pre --;
     * @post em pintat el laberint sobre un surface gràfic;
     */
    public void pintarLaberint()

    /**
     * @pre --;
     * @post em obtingut la mida que ha de fer una imatge dins d'una casella del tauler;
     */
    public Dimension obtenirMidaImatge()


    /**
     * @pre --;
     * @post em retornat el nombre de monedes que hi ha en el laberint;
     */
    public synchronized int obtenirMonedes()
```

LaberintAleatori

```
/**
 * @author oscar
 * @brief
 * Laberint amb camins aleatoris, l'estrategia per generar laberints d'aquest tipus és:
 * Posar en pacman al extrem esquerra superior "casella [0, 0]" i l'enemic especificat
 * en l'extrem inferior dret "casella [N-1, N-1].
 *
 * Llavors sortejar una casella dins del tauler, buscar la posició habilitada
 * més propera al punt sortejat i traçar un camí, en tot el camí es van posant
 * monedes.
 *
 * Aquest procés s'ha de repetir fins arribar a un nombre de monedes proporcional
 * a la mida del tauler i mentre el tauler no sigui valid.
 *
```

```
 * Per validar el tauler s'utilitza la classe ValidadorLaberint.java
 */

    /**
     * @pre costat > 5 i enemic és Fantasma1, Fantasma2 o Fantasma3
     * @post em cret un labrint aleatori (costat x costat) que forma part de partida, es pintarà a traves
     * de pintadorLaberint i on l'extrem superior esquerra [0, 0] hi ha en pacman
     * i en l'extrem inferior dret hi ha l'enemic;
     */
    public LaberintAleatori(Partida partida, int costat, EElement enemic, IPintadorLaberint
pintadorLaberint)
```

LaberintLinealHoritzontal

```
/**
 * @author oscar
 * @brief
 * Laberint amb camins horitzontals, l'estrategia per generar laberints d'aquest tipus és:
 * Posar tota una columna de monedes en la columna nùmero 0 i una en la columna nùmero n-1
 * llavors situar en pacman en la posició [0, 0] i fer posar tot de monedes a totes
 * les files que són un número parell (0, 2, 4 ...)
 *
 * L'únic requisit per aquest tipus de laberint és que el seu costat ha de ser imparell
 * per fer quadrar les files de paret.
 *
 * Aquest algorisme genera sempre un laberint valid.
 */

    /**
     * @pre costat >= Utils.Constants.MINIM_COSTAT_LABERINT i enemic és un dels fantasmes.
     * @post em creat un laberint de caràcter lineal i horitzontal amb partida, costat
     * enemic i un pintador per representar el laberint gràficament.
     */
    public LaberintLinealHoritzontal(Partida partida, int costat, EElement enemic, IPintadorLaberint
pintadorLaberint)
```

LaberintLinealVertical

```
/**
 * @author oscar
 * @brief
 * Laberint amb camins verticals, l'estrategia per generar laberints d'aquest tipus és:
 * Posar tota una fila de monedes en la fila nùmero 0 i una en la fila nùmero n-1.
 *
 * llavors situar en pacman en la posició [0, 0] i fer posar tot de monedes a totes
 * les columnes que són un número parell (0, 2, 4 ...)
 *
 * L'únic requisit per aquest tipus de laberint és que el seu costat ha de ser imparell
 * per fer quadrar les columnes de paret.
 *
 * Aquest algorisme genera sempre un laberint valid.
 */
```

```
    /**
     * @pre costat >= Utils.Constants.MINIM_COSTAT_LABERINT i enemic és un dels fantasmes.
     * @post em creat un laberint de caràcter lineal i vertical amb partida, costat
     * enemic i un pintador per representar el laberint gràficament.
     */
    public LaberintLinealVertical(Partida partida, int costat, EElement enemic, IPintadorLaberint
pintadorLaberint)
```

Fantasma1

```
/**
 * @author oscar
 * @brief
 * Eenemic més bàsic i facil de guanyar contra el que pots jugar en una partida.
 *
 * CARACTERISTIQUES DEL FANTASMA
 * Els seus moviments són totalment aleatoris.
 * Sempre serà igual de dolent (estigui com estigui el tauler)
 */

    /**
     * @pre inici és un punt valid dins de laberint.
     * @post em creat el fantasma 1 i està dins de laberint en la posició inici i jugant partida.
     */
    public Fantasma1(Partida partida, Laberint laberint, Punt inici)
```

Fantasma2

```
/**
 * @author oscar
 *
 * @brief
 * Eenemic úna mica més elaborat que Fantasma 1.
 *
 * CARACTERISTIQUES DEL FANTASMA.
 * La seva visibilitat només és en linia recta per tant per tant només valora
 * les quatre direccions (fins a trobar paret) en las que pot anar i decideix
 * quina és la que més li interesa.
 *
 * Es bo especialment quan hi ha moltes coses en el laberint.
 */

    /**
     * @pre inici és un punt valid dins de laberint.
     * @post em creat el fantasma 1 i està dins de laberint en la posició inici i jugant partida.
     */
    public Fantasma2(Partida partida, Laberint laberint, Punt inici)
```

Fantasma3

```
/**
```

```
 *
 * @author Moises
 * @brief Implementació del comportament de Fantasma3
 */

    /**
     * @brief Constructor de Fantasma3
     * @param partida Partida actual.
     * @param laberint Laberint actual.
     * @param inici Punt on apareixera Item.
     * @post Objecte actual ja interectua amb EElements de Laberint.
     */
    public Fantasma3(Partida partida, Laberint laberint, Punt inici)
```

Item

```
/**
 *
 * @author Moises
 * @brief Implementació del comportament dels items (PATINS, MONEDES_X2, MONGETA).
 */

    /**
     * @brief Constructor de Item
     * @param partida Partida actual.
     * @param tipusElement Tipus de EElement de objecte actual.
     * @param elementTrapitjat EElement contingut a la cel·la inici.
     * @param laberint Laberint actual.
     * @param inici Punt on apareixera Item.
     * @post Objecte actual ja interectua amb EElements de Laberint.
     */
    public Item(Partida partida, EElement tipusElement, EElement elementTrapitjat, Laberint
laberint, Punt inici)

    /**
     * @brief Retorna el element el qual esta a la mateixa cel·la que item.
     * @details Laberint conté un EElements per cel·la. Quan item es mou es necessita
     * guardar el EElement que contenia la cel·la actual on es troba el item.
     * @return Element que hi ha a la mateixa cel·la que item.
     */
    public EElement obtenirElementTrapitgat()
```

ItemMovible

```
/**
 *
 * @author oscar
 * @brief Objecte amb la capacitat de moures per les diferents poscions d'un Laberint
 * durant una Partida.
 */
```

```
    /**
     * @brief Inicialitzacio de un ItemMovible
     * @post
     * temporitzador inicialitzat
     * següent Moviment calculat
     * iniciat = true
     * @exception EItemMovibleIniciat si ItemMovible ja estava incialitzat.
     */
    public void iniciarItemMovible()


    /**
     * @brief Retorna el temps utilitzat nomes en calcul de moviments de un ItemMovible, durant una partida.
     * @return Temps total de càlcul de moviments.
     */
    public long obtenirTempsTotalCalcul()

    /**
     * @brief Descripció del metode que realitza un moviment dins de la estructura de Laberint.
     * @return EELement que contenia la posicio on s'ha desplaçat ItemMovible.
     */
    public abstract EElement realitzarMoviment()

    /**
     * @brief Descripcio del metode on s'implementara les logiques de cada ItemMovible per a calcular el seu proxim moviment.
     * @return EDireccio Calculada.
     */
    public abstract EDireccio calcularMoviment()

    /**
     * @brief Descripcio del metode per obtenir el nom que identifica al subtipus de ItemMovible.
     * @return Nom que identifica al subtipus de ItemMovible.
     */
    public abstract String nomItemMovible()
```

Pacman

```
/**
 * @author oscar
 * @brief
 * personatge principal controlat per l'usuari a traves d'un dispositiu físic.
 */

    /**
     * @pre inici és una posició valida dins de laberint
     * @post em creat en pacman que juga la partida dins de laberint (en la posició inici) i controlat
     * per controlador.
     */
    public Pacman(Partida partida, Laberint laberint, IControlador controlador, Punt inici)

    /**
```

```
     * @pre --
     * @post si direccioAMoure no surt del tauler llavors serà el proxim moviment
     * que intentarà fer en pacman.
     */
    public void nouMoviment(EDireccio direccioAMoure)
```

Partida

```
/**
 * @author oscar
 * @brief
 * pantalla on hi ha un pacman i un enemic que es van desplaçant per un laberint
 * ple de monedes amb l'objectiu de aconseguir-ne el maxim nombre possible.
 *
 * la partida finalitza quan no queden monedes.
 *
 * @invariant
 * laberint != null, pacman != null, enemic != null i pintadorPartida != null
 */

    /**
     * @pre: fitxer existeix i conté un laberint en format correcte;
     * @post: em carregat una partida a partir del laberint especificat per parametre;
     */
    public Partida(String fitxer,
                   IPintadorPartida pintadorPartida,
                   IPintadorLaberint pintadorLaberint,
                   IControlador controlador) throws EFormatLaberint


    /**
     * @pre Utils.Constants.MINIM_COSTAT_LABERINT <= costat i enemic
     * és realment un enemic.
     * @post em creat una partida amb un laberint concret de mida costat
     * amb en pacman a la posició [0, 0] un enemic en la posicio [costat-1, costat-1],
     * amb un pintador per la partida i per el laberint i un controlador per moure
     * an pacman.
     */
    public Partida(ELaberintsPredefinits laberint,
                   int costat,
                   EElement enemic,
                   IPintadorPartida pintadorPartida,
                   IPintadorLaberint pintadorLaberint,
                   IControlador controlador)

    /**
     * @pre la partida no ha sigut iniciada anteriorment.
     * @post em iniciat la partida juntament amb tots els seus elements.
     */
    public void iniciarPartida()

    /**
```

```
 * @pre la partida està iniciada i no finalitzada.
 * @post em finalitzat la partida amb tots els seus elements.
 */
public void finalitzarPartida()


/**
 * @pre la partida no ha sigut finalitzada
 * @post em tancat la partida matant al enemic, an pacman i al item (si ni
 * havia en aquest moment).
 */
public void tancarPartida()


/**
 * @pre el laberint ja no té monedes
 * @post em assignat un guanyador (qui té més punts) i ara toca anar
 * cap la sortida.
 */
public void assignarGuanyador()

/**
 * @pre hi ha un item en la partida
 * @post s'ha finalitzat l'item i em retornat que contenia sota seu.
 */
public EElement itemCapturat()

/**
 * @pre --
 * @post em assignat un nou item a la partida.
 */
public void assignarItemEspecial(Item item)

/**
 * @pre --
 * @post diu si hi ha un item en la partida.
 */
public boolean hiHaItemEspecial()

/**
 * @pre --
 * @post em retornat l'imatge de en pacman.
 */
public ImageIcon obtenirImatgePacman()


/**
 * @pre --
 * @post em retornat l'imatge del fantasma.
 */
public ImageIcon obtenirImatgeFantasma()
```

```
/**
 * @pre --
 * @post em retornat el laberint.
 */
public Laberint obtenirLaberint()

/**
 * @pre --
 * @post em pintat els punts que té en pacman.
 */
public void assignarPuntsPacman(int punts)

/**
 * @pre --
 * @post em pintat els punts que té el fantasma.
 * @param punts
 */
public void assignarPuntsEnemic(int punts)

/**
 * @pre --
 * @post em posat a 0 els punts del fantasma i n'hem retornat la quantitat
 * que tenia
 */
public synchronized int reiniciarPuntsEnemic()

/**
 * @pre --
 * @post em posat a 0 els punts den pacman i n'hem retornat la quantitat
 * que tenia
 */
public int reiniciarPuntsPacman()

/**
 * @pre --
 * @post em pintat el nou item que té en pacman.
 */
public void assignarItemAPacman(EElement item)

/**
 * @pre --
 * @post em pintat el nou item que té l'enemic
 */
public void assignarItemAEnemic(EElement item)

/**
 * @pre --
 * @post em retornat la posició den pacman dins del laberint.
 */
public Punt obtenirPuntPacman()

/**
```

```
     * @pre --
     * @post em retornat la posició del enemic dins del laberint.
     */
    public Punt obtenirPuntEnemic()

    /**
     * @pre --
     * @post em retornat l'estat en que es troba en pacman en aquest moment.
     */
    public EEstatPersonatge obtenirEstatPacman()

    /**
     * @pre --
     * @post em retornat l'item que hi ha a la partida (null si no n'hi ha cap)
     */
    public Item obtenirItem()

    /**
     * @pre --
     * @post em retornat el nombre de punts que porta en pacman.
     */
    public int obtenirPuntuacioPacman()
```

Personatge

```
/**
 * @author oscar
 * @breif
 * Representació de tot item que juga una partida dins d'un laberint, i que lluita per
 * obtenir el maxim de punts, que pot anar canviant d'estat i que es comporta de forma diferent
 * segons el seu estat.
 *
 * @invariant
 * punts >= 0, 0 <= indexDireccioAnterior <= 4 i imatges és una matriu de 4 files
 * i 2 columnes.
 */

    /**
     * @pre inici és una posició valida dins de laberint.
     * @post personatge que es juga una partida dins de un laberint en la posició inici
     * i que té una imatge de perfil
     */
    public Personatge(Partida partida, Laberint laberint, ImageIcon imatge, Punt inici)

    /**
     * @pre --
     * @post em retornat el nombre de punts que té el personatge.
     */
    public int obtenirPunts()

    /**
     * @pre ja no queden monedes en el laberint.
     * @post em canviat si guanya o perd el personatge.
```

```
 */
public void assignarGuanya(boolean guanya)

/**
 * @pre --
 * @post em retornat si el personatge està guanyant.
 * @return
 */
public boolean estaGuanyant()

/**
 * @pre --
 * @post em posat a 0 el nombre de punts del personatge i n'hem retornat la
 * quantitat que tenia.
 */
public int reiniciarPunts()
```

Punt

```
/**
 * @author oscar
 * @brief
 * Representació de un punt 2D format per fila i columna;
 */

/**
 * @pre: --;
 * @post: definim un punt a traves de les coordenades fila i columna;
 */
public Punt(int fila, int columna)

/**
 * @pre: --;
 * @post: em retornat el valor del eix de les absices;
 */
public int obtenirColumna()

/**
 * @pre: --;
 * @post: em retornat el valor del eix de les ordenades;
 */
public int obtenirFila()

/**
 * @pre: --;
 * @post: retornem el punt obtingut de aplicar un moviment sobre el punt actual;
 */
public Punt generarPuntDesplasat(EDireccio moviment)

/**
 * @pre: --;
 * @post: em retornat el QUADRAT de la distancia amb desti;
```

```
     */
    public int distancia(Punt desti)

    /**
     * @pre: --;
     * @post:em donat les 4 posicions del voltant del punt;
     */
    public Punt[] obtenirPosicionsDelVoltant()
```

Usuari

```
/**
 * @author oscar
 * @brief
 * usuari registrat a l'aplicació que consta de
 *      - id -> identificador de la B.D.
 *      - nomUsuari -> el nom del usuari
 *      - ulImatge -> ruta de l'imatge de perfil.
 *      - nivell -> en quin nivell es troba de l'aventura.
 *      - dificultat -> dins del nivell per quina dificultat va.
 *      - punts -> cumul de punts del usuari per el nivell en que està.
 *
 * @invariant
 * id != null, nomUsuari != null, imatgePerfil != null, punts >= 0
 */

    /**
     * @pre l'urlImage és una ruta valida a una imatge.
     * @post em creat un usuari amb id, nomUsuari, nivell i ruta de l'imatge de perfil.
     */
    public Usuari(int id, String nomUsuari, int nivell, String urlImatge)

    /**
     * @pre --
     * @post em retornat la dificultat dins del nivell en que esta l'usuari.
     */
    public EDificultat obtenirDificultat()

    /**
     * @pre --
     * @post em retornat l'id de l'usuari.
     */
    public int obtenirId()

    /**
     * @pre --
     * @post em retornat el nom de l'usuari.
     */
    public String obtenirNomUsuari()

    /**
     * @pre --
     * @post em retornat el nivell en que està l'usuari
```

```
 */
public ENivells obtenirNivell()

/**
 * @pre
 * @return
 */
public Icon obtenirImatge()

/**
 * @pre --
 * @post l'usuari va per la seguent dificultat del nivell, si s'ha superat
 * la màxima dificultat per el nivell es va al seguent nivell.
 */
public void pantallaSuperada(int pnt)
```

Moduls Funcionals

BD

```
/**
 * @author oscar
 * @brief
 * modul funcional amb les operacions per treballar sobre la B.D. "sqlite"
 * utilitzada en el projecte;
 */

    /**
     * @pre --;
     * @post s'ha registrat un nou usuari a la B.D. i retorna si l'operació s'ha realitzat correctament;
     */
    public static boolean afegirUsuari(String user, String password, String rutaImatge)

    /**
     * @pre topN > 0;
     * @post em retornat el conjunt de usuaris que conformen el topN per ordre segons el nombre de
punts;
     */
    public static Usuari[] obtenirRanking(int topN)

    /**
     * @pre --;
     * @post diu si està o no registrat usuari;
     */
    public static boolean existeixUsuari(String usuari)

    /**
     * @pre nivell > 0 usuId valid i punts > 0
     * @post s'ha registrat el nou nivell superat per l'usuari amb usuId;
     */
    public static void nivellSuperat(int nivell, int usuId, int punts)

    /**
     * @pre --;
     * @post si existeix l'usuari i el password és correcte es retorna la seva
     * representació en forma d'objecte altrament es retorna null;
     **/
    public static Usuari obtenirUsuari(String user, String password)

    /**
     * @pre --;
     * @post diu si existeix el fitxer "sqlite" de la Base de dades;
     */
    public static boolean existeixBaseDeDades()

    /**
```

```
     * @pre usuari està registrat en el sistema;
     * @post s'ha retornat un array amb el conjunt de nivells superats on
     * l'index correspon al nivell i el contingut als punts.
     *
     * nivell[5] -> 4000 vol dir que del nivell 5 s'han fet 4000 punts
     *
     * En cas de haver un problema amb la B.D. es retorna null;
     */
    public static int [] obtenirHistoricPuntsUsuari(Usuari usuari)
```

Utils

```
/**
 * @author oscar
 * @brief
 * Modul funcional amb un conjunt d'operacions que s'utilitzen
 * a nivell de tota l'aplicació
 */

    /**
     * @pre max > 0
     * @post es retorna un valor pseudoaleatori dins del rang [0, max)
     */
    public static int obtenirValorAleatori(int max)

    /**
     * @pre cadena no és null;
     * @post em retornat la cadena codificada;
     */
    public static String codificarCadena(String cadena)

    /**
     * @pre --;
     * @post em retornat l'hora del sistema en format HH:MI:ss
     */
    public static String obtenirHoraSistema()

    /**
     * @pre --;
     * @post em retornat moment en format HH:MI:ss
     */
    public static String obtenirMomentEnFormatHoraMinutsSegons(final long moment)

    /**
     * @pre px > 0;
     * @post em retornat l'imatge redimensionada a px x px
     */
    public static Image redimensionarImatge(ImageIcon imatge, int px)

    /**
     * @pre originalImage és un buffer de una imatge valida.
     * @post em retornat un buffer amb l'imatge redimensionada.
     */
```

```
public static BufferedImage redimensionarImatge(BufferedImage originalImage,
                                                int tipus,
                                                int amplada,
                                                int altura)
```