Estructura de Datos y de la Información

Práctica Final Curso 2021-2022

El objetivo de la práctica es el desarrollo de un conjunto de juegos basados en “Celdas”, concretamente el juego de las parejas, las 4 en rayas y el buscaminas. Para ello será necesario el uso de una metodología orientada a objetos así como el uso de estructuras de datos lineales y no lineales. El desarrollo de la práctica se realizará en dos/tres fases.

- Nota: En documento adicional se asignará qué juego debe realizar cada alumno (sólo se debe implementar uno de ellos).

Entrega 1

La primera entrega consistirá en el desarrollo del juego usando los contenidos teóricos / prácticos de la asignatura: Programación orientada a objetos, composición y herencia.

Paso 1: Jerarquía de Clase: Celda

Independientemente del juego a realizar se implementarán todas las clases que conforman la jerarquía Celda. Como puede verse en el diagrama UML del último folio la jerarquía está formada:

- Clase Base: Celda: clase abstracta formada por
  - fila: el valor de la fila donde se encuentra situada la celda en la matriz del juego
  - columna: el valor de la columna donde se encuentra situada la celda en la matriz del juego
  - Estado: Tendrá distinto signficado dependiendo de la celda usada
    - En CeldaRaya: 0 vacia 1 asignada a humana 2 asignada a ordenador
    - En CeldaPareja: 0 NoMostrada 1 Mostrada 2 MostradaEmparejada
    - En CeldaMina: 0 agua 1 mina 2 agua marcada como mina 3 mina marcada como mina
- Clase Derivada: CeldaRaya: Estará formada por:
  - color: valor entero que indica el color de la Celda (rojo o azul)
- Clase Derivada: CeldaPareja: Estará formada por:
  - valor: valor entero usado para representar el valor de la Celda para emparejar
- Clase Derivada: CeldaMina: Estará formada por **descubierta** Con valor verdadero o falso. De este modo habría los siguientes estados:
  - Si estado = 0 y descubierta=false –> agua oculta
  - Si estado = 1y descubierta=false –> mina oculta
  - Si estado = 2 y descubierta=false –> agua marcada como mina
  - Si estado = 3 y descubierta=false –> mina marcada como mina
  - Si estado = 0 y descubierta=true–> agua descubierta
  - Si estado = 1 y descubierta=true –> mina descubierta y por tanto FIN

Consejo:

- Estas clases no incorporan ninguna lógica adicional, por tanto es una jerarquía de clase básica

Paso 2: Composición nAria→ Clase Tablero

Una vez realizada la jerarquía de clase de Celda, el siguiente paso será modelar el concepto de tablero de Juego. En este caso un Tablero estará formado por una matriz bidimensional de Celdas. En este caso modelaremos una clase base abstracta denominada **Tablero** e implementaremos **únicamente** una clase derivada (la del juego que debe implementarse). Como puede verse en la imagen del último folio esta jerarquía de clase estará formada:

- Clase base Tablero. Clase abstracta formada por:
  - numFilas: Número de filas (modificable en el constructor)
  - numColumnas: Número de columnas (modificable en el constructor)
  - Celda[][] tablero; Matriz bidimensional de Celdas
  - Tiene los siguientes métodos:
    - void setEstado(int f, int c, int e): Se encarga de cambiar el estado (e) de la celda indicada por la fila (f) y la columna (c)
    - int getEstado(int f, int c): Devuelve el estado de la celda indicada en la fila (f) y la columna (c)
    - Celda getCelda(int f, int c): Devuelve la Celda indicada en la fila (f) y la columna (c)
  - Además esta formado por tres métodos abstractos que deben implementarse en la clase derivada:
    - abstract inicializar(): Se encarga de inicializar el tablero a las celdas (será llamado desde el constructor de la clase derivada)
    - abstract void repartir(): Se encarga de repartir (sólo válido para TableroPareja y TableroMinas). En tablero4Raya se repartirá tres movimientos aleatorios para cada jugador.
    - abstract void mostrar(): Se encarga de mostrar el Tablero en un formato gráfico (puede valer con toString)
- Clase derivada TableroParejas:
  - No tiene ningún atributo adicional
  - Debe implementar los métodos abstractos
- Clase derivada TableroRaya:
  - No tiene ningún atributo adicional
  - Debe implementar los métodos abstractos
- Clase derivada TableroMinas:
  - Tiene como atributo el número máximo de minas.
  - Debe implementar los métodos abstractos

Nota:

- Como puede observarse prácticamente el paso 1 y el paso 2 es indiferente del juego seleccionado.
- El principal cambio reside donde se realiza la lógica del juego → Clase Juego
- Si se quisiera implementar otro juego basado en celdas, únicamente se tendría que añadir una clase derivada a la jerarquía de clase Celda y Tablero.

Paso 3 Composición unaria→ Clase Juego

En este último paso, el objetivo será la implementación del juego en sí. Se deberá implementar únicamente la clase del Juego a implementar

- Clase JuegoPareja esta formado por:
  - Un objeto de la clase TableroParejas t;
  - Tendrá los siguientes métodos:

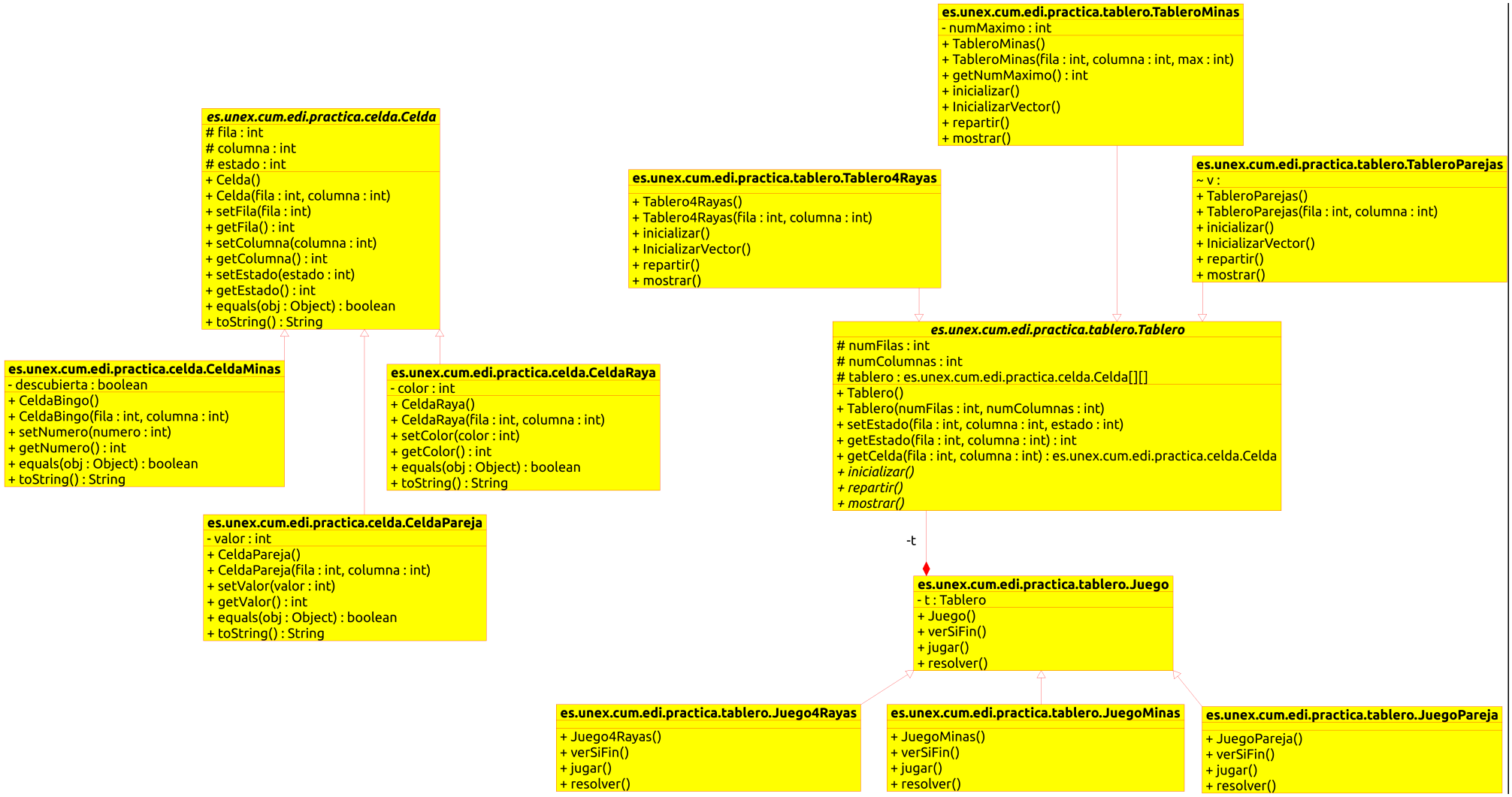
        - Jugar: Bucle que se encarga de pedir una fila-columna y una segunda fila-columna y si son iguales se descubre y sino se vuelve a ocultar
        - Resolver: método que dado 4 posiciones se encarga de ver si son iguales o no
        - VerSiFin: Se encarga de ver si el juego ha acabado
- Clase JuegoRaya: Sólo supondremos que se gana cuando hay 4 piezas seguidas en horizontal o vertical.
    - Un objeto de la clase TableroRaya t;
    - Tendrá los siguientes métodos:
        - Jugar: Bucle que se encarga de pedir movimiento al usuario y al ordenador (será aleatorio)
        - JugarHumano: Pide la columna y recorre el tablero hasta que encuentra una posición sin ocupar
        - JugarOrdenador: Aleatoriamente seleccionado un valor de columna y recorre el tablero hasta que encuentra una posición sin ocupar
        - VerSiFin: Se encarga de ver si el juego ha acabado bien porque no hay más huecos o bien porque uno de los dos haya terminado.
- Clase JuegoMinas:
    - Un objeto de la clase TableroMinas t;
    - Tendrá los siguientes métodos:
        - Jugar: Bucle que se encarga de pedir una posición (x,y) del tablero. Se debe pedir si se quiere descubrir o anotar como mina.
        - Resolver: Se encarga de cambiar el estado de la celda según el paso anterior.
        - VerSiFin: Se encarga de ver si el juego ha acabado (todas las celdas están descubiertas menos las minas)

Nota:

- El diagrama así como el nombre y funcionalidades expuestas previamente son borradores que pueden ser modificadas.
- Es obligatorio la entrega de la autodocumentación (Javadoc)

Reparto:

Minas		Parejas		4Rayas	
JOSE MANUEL	AGUILERA MARTIN	DIEGO	GONZÁLEZ RODRÍGUEZ	JUAN CARLOS	REDONDO PRIOR
MÍRIAM	ÁLVAREZ ASENSIO	DAVID	GRAGERA FERNÁNDEZ	LAURA	REINOSA SÁNCHEZ
CARMEN	ÁLVAREZ MURILLO	JAIME	GUERRERO AMAYA	JAVIER	REY SÁNCHEZ
ANTONIO	CALVO PICÓN	OTHMANE	HERAGUA	ANTONIO JAVIER	RINO CIDONCHA
ALBERTO	CAÑAVERAS SOLÍS	IVÁN	HERCULANO GARCÍA	RAÚL	RODAS MATEOS
JOSÉ ALEJANDRO	CANCHO RODRÍGUEZ	LUIS IGNACIO	HERNÁNDEZ MORANO	PABLO	RODRIGUEZ BERNAL
DANIEL	CARVAJAL FERNÁNDEZ	MARCO	HERRERA IBORRA	HUGO	RODRÍGUEZ GALVÁN
DANIEL	CASADO MORENO	CARMEN MELIBEA	MADIBA NFUMU	CARLOS	RUIZ HIDALGO
CARLOS	CASTAÑÓN CINTADO	ADRIÁN	MONROY SOLÍS	IGNACIO	SÁNCHEZ AGUILERA
MIGUEL	CENDRERO CALDERÓN	JESÚS	MONTAÑO FARRONA	HUGO	SANCHEZ DE LA RODA RIVERA
FRANCISCO JAVIER	CHAVES GARCÍA	PAULA	MONTERO MORENO	JOSÉ ÁNGEL	SÁNCHEZ PECERO
JUAN	DELGADO LÓPEZ	ANA	MONTERO PRECIADO	CHRISTHILD JESSENIA	SÁNCHEZ SANTILLÁN
GUILLERMO	DÍAZ PALACIOS	FRANCISCO	MONTERO SÁNCHEZ	ANTONIO	SANZ-CALCEDO CARRASCO
ANTONIO JESÚS	DOMÍNGUEZ UGUINA	ISMAEL	MORUNO RODRÍGUEZ	ADRIÁN	SEGADOR BARROSO
DIEGO	DURÁN BARROSO	MARIO	NÚÑEZ CIDONCHA	MANUEL	SERRANO PASTOR
GEDEON MAYE	EKUA EDAYONG	DAVID	ORTEGA CONTRERAS	PABLO	SETRAKIAN BEARZOTTI
JOSÉ LUIS OBIANG	ELA NANGUAN	JORGE	ORTEGA RIVERA	CLAUDIO	TERRADOS SÁNCHEZ
ÁNGEL	FLORES SEVILLANO	MARTA	PAREDES MARTÍNEZ	DAVID	URBANO RANCHAL
JUAN JOSÉ	GALINDO COTANO	JUAN	PÉREZ VILLEGAS	ALBERTO	VALERO MLYNARICOVA
VÍCTOR	GALLARDO SÁNCHEZ	VÍCTOR	PINILLA CORRALIZA	PABLO	VAQUERO SEVILLA
DAVID	GIL PAREJO	JUAN	PÍRIZ NAVARRETE	ALBERICO	ZIRONI
JOSÉ	GIMÉNEZ BASELGA	JAVIER	PRADA NAHARROS		


es.unex.cum.edi.practica.tablero.TableroMinas
- numMaximo : int
+ TableroMinas()
+ TableroMinas(fila : int, columna : int, max : int)
+ getNumMaximo() : int
+ inicializar()
+ InicializarVector()
+ repartir()
+ mostrar()
es.unex.cum.edi.practica.celda.Celda
fila : int
columna : int
estado : int
+ Celda()
+ Celda(fila : int, columna : int)
+ setFila(fila : int)
+ getFila() : int
+ setColumna(columna : int)
+ getColumna() : int
+ setEstado(estado : int)
+ getEstado() : int
+ equals(obj : Object) : boolean
+ toString() : String
es.unex.cum.edi.practica.tablero.Tablero4Rayas
+ Tablero4Rayas()
+ Tablero4Rayas(fila : int, columna : int)
+ inicializar()
+ InicializarVector()
+ repartir()
+ mostrar()
es.unex.cum.edi.practica.tablero.TableroParejas
~ v :
+ TableroParejas()
+ TableroParejas(fila : int, columna : int)
+ inicializar()
+ InicializarVector()
+ repartir()
+ mostrar()
es.unex.cum.edi.practica.celda.CeldaMinas
- descubierta : boolean
+ CeldaBingo()
+ CeldaBingo(fila : int, columna : int)
+ setNumero(numero : int)
+ getNumero() : int
+ equals(obj : Object) : boolean
+ toString() : String
es.unex.cum.edi.practica.celda.CeldaRaya
- color : int
+ CeldaRaya()
+ CeldaRaya(fila : int, columna : int)
+ setColor(color : int)
+ getColor() : int
+ equals(obj : Object) : boolean
+ toString() : String
es.unex.cum.edi.practica.tablero.Tablero
numFilas : int
numColumnas : int
tablero : es.unex.cum.edi.practica.celda.Celda[][]
+ Tablero()
+ Tablero(numFilas : int, numColumnas : int)
+ setEstado(fila : int, columna : int, estado : int)
+ getEstado(fila : int, columna : int) : int
+ getCelda(fila : int, columna : int) : es.unex.cum.edi.practica.celda.Celda
+ inicializar()
+ repartir()
+ mostrar()
es.unex.cum.edi.practica.celda.CeldaPareja
- valor : int
+ CeldaPareja()
+ CeldaPareja(fila : int, columna : int)
+ setValor(valor : int)
+ getValor() : int
+ equals(obj : Object) : boolean
+ toString() : String
-t
es.unex.cum.edi.practica.tablero.Juego
- t : Tablero
+ Juego()
+ verSiFin()
+ jugar()
+ resolver()
es.unex.cum.edi.practica.tablero.Juego4Rayas
+ Juego4Rayas()
+ verSiFin()
+ jugar()
+ resolver()
es.unex.cum.edi.practica.tablero.JuegoMinas
+ JuegoMinas()
+ verSiFin()
+ jugar()
+ resolver()
es.unex.cum.edi.practica.tablero.JuegoPareja
+ JuegoPareja()
+ verSiFin()
+ jugar()
+ resolver()