Estructura de Datos y de la Información

Práctica Final Curso 2020-2021

El objetivo de la práctica es el desarrollo de un conjunto de juegos basados en “Cartas”. 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.

Entrega 2

El objetivo de esta segunda entrega consistirá en continuar con el desarrollo del juego aportando un mayor número de funcionalidades así como darle más realismo.

- **Modificar la clase Baraja**
  - Se debe cambiar el array de Cartas por una Cola Queue, que nos permita almacenar el mazo al inicio del juego. De este modo:
    - El método inicializar se encargar de añadir las cartas al mazo (sólo será llamado cuando la mano “crea” la baraja). Se debe cambiar b[i]= new Cartas(…) por la llamada que te permite al método que te permite añadir objetos a la cola.
    - Y el método repartir/barajar sólo tiene que desordenarlas (existe un método Collections.shuffle(p); que se encarga de ello donde p es la Cola
    - Debe existir un método que devuelve la carta que se encuentra en el frente (teníamos un getCarta(int pos) se crea uno nuevo getCarta() que devuelve el frente)
    - Debe existir un método que permita añadir cartas a la Baraja

- **Clase Partida**: Almacena el tiempo que tarda un determinado jugador en resolver el juego
  - Se encuentra formado por tres atributos:
    - boolean ganada: Almacena si la partida la ha ganado o no.
    - long tiempo: Almacena en formato long el tiempo que ha tardado
    - Date fecha: Almacena la fecha en la que realizó dicha jugada
    - Se trata de una clase básica con los métodos habituales

- Jerarquía de clase **Persona** (Clase base y abstracta):
  - Compuesta por un nombre, un identificador de usuario y una baraja. Se deberán implementar los métodos habituales
  - **IMPORTANTE**:
    - El atributo baraja tendrá una función diferente para cada juego que se comentará posteriormente. Debe inicializarse vacía e ir añadiendo cartas según el juego que sea.
    - Debe implementar el interface Serializable para que el proceso de cargar/salvar se haga automáticamente
      - Definición: class Persona implements Serializable {

  - **Clase derivada Jugador**: Tiene una lista (ArrayList) de histórico de las veces que ha jugado (objetos de la clase Partida). Es decir, cada vez que un jugador juega a las cartas

y finaliza la partida, se obtiene la información de como ha ido (indicado en la Partida) y se almacena en la lista.
    - Además debe almacenar el número de partidas jugadas, el número de partidas ganadas y el número de partidas perdidas.
    - Tiene los métodos habituales más uno que permita añadir una partida a la lista de partidas jugadas addPartida(Partida j)
    - IMPORTANTE: Debe implementar el interface Serializable para que cargar/salvar se haga automáticamente
        - class Jugador implements Serializable {

- **Clase derivada Mesa**: No tiene ningún atributo adicional. Contiene el mazo o baraja que deberá ser inicializado con todas las cartas.
    - Esta clase tendrá una función diferente dependiendo del juego.

NOTA: Utilidad del atributo baraja y la clase Mesa en cada juego:
- JuegoCartaMasAlta: En este juego un objeto de la clase Mesa será el jugador oponente contra el que jugaremos la partida. Tanto el jugador como la mesa tendrán una baraja **completa** con la que empezar a jugar.

- JuegoCartaSieteMedia: En este juego un objeto de la clase Mesa hará las funciones del "crupier" que reparte las cartas a los jugadores. En este caso, la baraja completa la tendrá el objeto Mesa, mientras que los jugadores, al inicio, tendrán una baraja vacía. Cuando la mesa reparta una carta a un jugador, ésta se levantará en la baraja de la mesa y se almacenará en la baraja del jugador al que se le ha repartido. Así, cada jugador tendrá un mazo o baraja con el conjunto de cartas que le han repartido.

- JuegoParesSolitario: En este juego un objeto de la clase Mesa hará las mismas funciones de "crupier" que en el juego anterior. La baraja o mano del jugador estará vacía inicialmente y al comenzar el juego el crupier le repartirá 6 cartas. Estas cartas se levantarán en la baraja de la mesa. Cuando el jugador seleccione una pareja, las cartas emparejadas se llevarán a la Cola (baraja) del jugador y el crupier repartirá dos nuevas cartas de su baraja.

- **Clase Juego**:
    - Compuesta por un una lista (LinkedList de Persona) que denominaremos "personas" (donde al menos hay una "persona" (crupier) que actúa de mesa y otros tantos como jugadores)
    - Debe tener las siguientes funciones/menu
        - Cargar lista de Personas (jugadores/mesa) (se proporciona el código de esta funcionalidad)
        - Añadir una nueva persona de tipo jugador o crupier a la lista de personas
        - Jugar al juego
            - Indicar el identificador del jugador
            - Seleccionar el crupier de la mesa. Puede haber dos opciones: se elige la mesa

        - por teclado o bien se eligen de forma aleatoria.
        - Coger el tiempo de comienzo (se proporciona el código más abajo)
        - Jugar (ver funcionalidad más abajo)
        - Coger el tiempo de fin (se proporciona el código más abajo)
        - Salvar los datos de la partida en la lista de partidas del jugador
    - Mostrar las personas (jugadores y mesa) que hay en el sistema
    - Mostrar la información y partidas completadas para una determinar persona (dado su id)
    - Salvar la lista de personas

Importante: Con estas funcionalidades se puede obtener un puntuación máxima de 9. Para conseguir una mayor puntuación se deberá implementar:

- Listar el ranking de jugadores por partidas ganadas y perdidas (los 10 primeros)
- Calcular el jugador que menos ha tardado por cada tipo de juego.

Operación de cargar() en la Clase Juego

```
public void cargar() throws FileNotFoundException, IOException,  ClassNotFoundException {
        ObjectInputStream archivoObjetosEnt = new ObjectInputStream(
                        new FileInputStream("datos.dat"));
        personas = (LinkedList) archivoObjetosEnt.readObject();
        archivoObjetosEnt.close();
}
```

Operación de Salvar() en la Clase Juego

```
public void salvar() throws FileNotFoundException, IOException {
        ObjectOutputStream archivoObjetosSal = new ObjectOutputStream(
                        new FileOutputStream("datos.dat"));
        archivoObjetosSal.writeObject(personas);
        archivoObjetosSal.close();
}
```

Cómo coger el tiempo de inicio y como parar el tiempo

```
long tInicial=new Date().getTime(); //Se coge el tiempo actual
j.jugar();                          //Se juega
long tFinal= new Date().getTime();  //Se coge el tiempo final
```

Funcionalidad de juego

- Clase JuegoCartaMásAlta
    - El objetivo será seleccionar dos jugadores de la lista de jugadores (un jugador y un jugador crupier que será la mesa) de la baraja francesa y repartir todas las cartas. A continuación comenzará el juego realizando la comparación de ambos mazos. El que

gane se lleva las dos cartas y el juego acabará cuando uno se quede sin cartas. Si al llegar a las 100 cartas extraidas, no ha acabado, gana el que más cartas tengas.

- Clase JuegoCartaSieteMedia:
  - El objetivo será seleccionar varios jugadores de la Lista (máximo 6) así como un jugador que será la mesa. A continuación, el crupier (mesa) barajará (repartir las cartas) y comenzará a dar carta una cada uno de los jugadores. El sistema preguntará al usuario si quiere sacar otra carta, sabiendo que no puede pasarse de 7 y media. Las cartas del 1 al 7 valen su valor mientras que las figuras valen medio punto. El jugador que se aproxime más o alcance 7 y media ganará la partida. Puede haber empates.
- Clase JuegoParesSolitario:
  - El objetivo será seleccionar un jugador de lista así como un jugador que será la mesa. A continuación la mesa barajará (repartir) y sacará 6 cartas (las pierde la mesa y las gana el jugador en una estructura interna, NO en la baraja). A continuación comenzará el juego. El sistema preguntará al usuario que seleccione dos cartas de las 6 y el sistema comprobará si realmente suman 20. Si es así elimina esas cartas del jugador y se las lleva a la baraja del jugador y a continuación saca otras dos de la mesa. El juego termina si no hay parejas disponibles o se acaba el mazo.

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: Carta

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

- Clase Base: Carta: clase abstracta formada por
  - imageReverso: Path al nombre de la imagen de la celda cuando está dada la vuelta (No se usará para nada, pero por si se hiciera gráficamente)
  - imagenFrontal; Path al nombre de la imagen de la celda cuando se muestra (No se usará para nada, pero por si se hiciera gráficamente)
  - valor: variable char que indica el valor de la carta. Puede valer:
    - Carta Española: 1,2 ,3 4, 5, 6, 7, Sota (S), Caballo (C), Rey (R)
    - Carta Francesa: 1,2 ,3 4, 5, 6, 7, 8 , 9 , 10 (A), y tres figuras, que se llaman Valet (V), Dame (D) y Roi (R).
    - Carta Alemana: Consiste en un mazo de 36 naipes o cartas, clasificadas en cuatro palos. La numeración es: del 6 al 10 (X), Unter (U), Dame (D), König (K) y Daus (A).
  - oculta: Variable boolean que indica si la carta esta "boca-abajo" o no
- Clase Derivada: CartaFrancesa: Estará formada por:
  - símbolo: valor Cadena (enumerado o no) que indica el símbolo de la carta francesa (Diamante, Trébol, Pica y Corazón)
- Clase Derivada: CartaEspanola: Estará formada por:
  - palo: valor Cadena (enumerado o no) que indica el símbolo de la carta española (Oro, Basto, Espada y Copas)
- Clase Derivada: CartaAlemana: Estará formada por:
  - palos: valor Cadena (enumerado o no) que indica el símbolo de la carta alemana. Los cuatro palos son: Schellen (campanas), Gras o Grün o Laub (hojas, verde o follaje), Herz o Rot (corazones o rojos) y Eichel (bellotas).

Consejo:

- Estas clases no incorporan ninguna lógica adicional, por tanto es una jerarquía de clase básica.
- El método getValor() debe devolver un valor numérico asociada a la carta que será usado posteriormente.

Paso 2: Composición nAria→ Clase Baraja

Una vez realizada la jerarquía de clase de Carta, el siguiente paso será modelar el concepto de una baraja. En este caso una baraja estará formada por una array unidimensional de Cartas. En este caso modelaremos una clase base abstracta denominada **Baraja** 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 Baraja. Clase abstracta formada por:
  - numCartas: Número de cartas totales
  - numCP: Número de cartas por cada tipo
  - numTipo: Número de palos/tipos de cartas
  - Carta[] b; Vector de cartas
  - Tiene los siguientes métodos:
    - void setEstado(int pos, int e): Se encarga de cambiar el estado (e) de la Carta indicada por pos (oculta o levantada)
    - int getEstado(int pos): Devuelve el estado de la carta indicada en pos (oculta o levantada)
    - Carta getCarta(int pos): Devuelve la carta indicada en pos
  - Además está formado por tres métodos abstractos que deben implementarse en la clase derivada:
    - abstract inicializar(): Se encarga de inicializar la baraja(será llamado desde el constructor de la clase derivada)
    - abstract void repartir(): Se encarga de repartir (en muchos de los casos será aleatorio)
    - abstract void mostrar(): Se encarga de mostrar la baraja en un formato gráfico (puede valer con toString)
  - Nótese que pudieran existir otros métodos.

- Clase derivada BarajaEspanola:
  - No tiene ningún atributo adicional
  - Debe implementar los métodos abstractos
- Clase derivada BarajaFrancesa:
  - No tiene ningún atributo adicional
  - Debe implementar los métodos abstractos
- Clase derivada BarajaAlemana:
  - No tiene ningún atributo adicional
  - Debe implementar los métodos abstractos

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 JuegoCartaMásAlta
  - Un objeto de la clase BarajaFrancesa b;
  - El objetivo será instanciar dos objetos de la baraja francesa y repartir cartas. A continuación comenzará el juego realizando la comparación de ambos mazo. Como fin del programa deberá establecerse que baraja ha ganado más veces.

- Clase JuegoCartaSieteMedia:
  - Un objeto de la clase BarajaEspanola b;
  - El objetivo será instanciar dos objetos de la baraja española y repartir cartas. A continuación comenzará el juego sacando una carta. El sistema preguntará al usuario si quiere sacar otra carta, sabiendo que no puede pasarse de 7 y media. Las cartas del 1 al 7 valen su valor mientras que las figuras valen medio punto. La otra baraja será la del ordenador y su lógica será la siguiente: si lleva acumulado menos de 5 sacará otra carta automáticamente y sino se plantará. Ganará aquel que este más cerca de 7 y ½ y no se haya pasado. Hacer un bucle para jugar varias veces y determinar quién ha ganado más veces.
- Clase JuegoParesSolitario:
  - Un objeto de la clase BarajaAlemana b y un array de 6 cartas.
  - Sacando 6 cartas en la mesa es objetivo es hacer parejas. Las parejas sumando 20, pudiendo ser: 6-As, 7-K, 8-D, y 9-U, 10-10.
  - El objetivo será instanciar un objeto de la baraja Alemana y repartir 6 cartas. A continuación comenzará el juego. El sistema preguntará al usuario que seleccione dos cartas de las 6 y el sistema comprobará si realmente suman 20. Si es así elimina esas cartas y echas otras 2. El juego termina si no hay parejas disponibles o se acaba el mazo.

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)