PROGRAMACIÓN II

PROYECTO

MiniMus

versión: 1.0.2

Grado en Ingeniería de Tecnologías de Telecomunicación
Curso 2019/20

HISTORIAL DE CAMBIOS

Versión	Cambios
1.0.2	Corregido el ejemplo del apartado 3.2.3. Corregido, en el apartado 3.2.1, el formato de invocación de la aplicación en modo 2. Corregido error en el ejemplo de fichero de comandos del apéndice A.4 Armonizado el término MiniMus
1.0.1	Corregido error en apartado 2.2 relativo a la descripción en el lance de juego Corregido error en ejemplo del apartado 3.1.3

1. INTRODUCCIÓN

El objetivo del proyecto de la asignatura Programación II en el presente curso es desarrollar una aplicación para jugar a un juego de cartas denominado MiniMus. Este juego, en pocas palabras, es una simplificación del juego del Mus, sin apuestas y con las puntuaciones ligeramente modificadas para incrementar la indeterminación en el resultado de las competiciones.

El Mus (y por lo tanto el MiniMus) no es un juego de ganar bazas, sino que cada jugador, con las 4 cartas que tiene en su mano, comprobará frente a los jugadores contrarios, en cada una de las jugadas, quién posee cartas superiores atendiendo a un determinado tipo de desafío.

En el presente documento se detallan, en el Apartado 2, las características y reglas del juego MiniMus y, en el Apartado 3, las particularidades de la aplicación a desarrollar. En el Apartado 4 se describen las normas de entrega del proyecto y el método de evaluación del mismo.

ATENCIÓN: *Con el fin de mantener una visión uniforme sobre la funcionalidad de la aplicación a desarrollar en el proyecto y de que todos los alumnos tengan la misma información respecto al mismo, cualquier duda que surja relativa a la interpretación de la presente especificación deberá ser planteada en el foro de consultas de FaiTIC destinado a tal efecto. Los profesores de la asignatura no atenderán a cuestiones de este tipo de manera individual.*

2. DESCRIPCIÓN DEL JUEGO

2.1 Baraja

Para jugar al MiniMus se utiliza una Baraja española de 40 Cartas o naipes, tal como se muestra en la Fig. 1 (no se utilizan las cartas con denominación 8 y 9 de la Baraja española de 48 cartas).

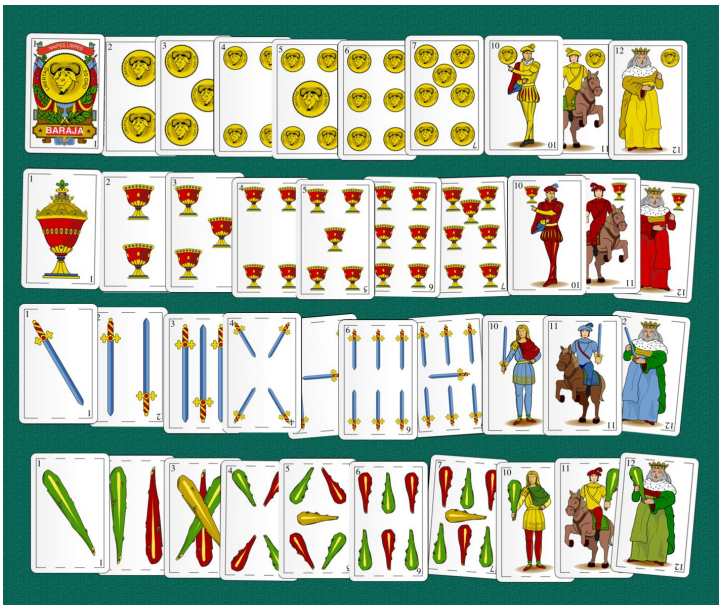

Figura 1. Baraja española de 40 cartas[1]

Las Cartas se encuentran agrupadas en 4 Palos: oros, copas, espadas y bastos. Cada Palo está conformado por 10 Cartas: las 7 primeras denominadas con los números del 1 al 7, y las tres últimas como sota (S), caballo (C) y rey (R).

En los ejemplos que siguen en este documento cada Carta se identificará con dos caracteres, donde el primero representa la denominación (1, 2, 3, 4, 5, 6, 7, S, C, R) y el segundo el palo (O, C, E, B). Por ejemplo, 4B es el cuatro de bastos y CC es el caballo de copas.

En el MiniMus, los treses se consideran reyes, y los doses se consideran ases. Es decir, a efectos prácticos se trata de una baraja con 16 figuras (cartas con las denominaciones S, C, R y 3), 8 ases (cartas con denominaciones 1 y 2) y 16 cartas normales (cartas con las denominaciones 4, 5, 6 y 7).

Las puntuaciones de las cartas en el MiniMus son las siguientes:

- Las figuras valen 10 puntos. Es decir, las cartas con las denominaciones 3, S, C y R valen 10 puntos cada una de ellas.

[1] Imagen disponible en https://es.wikipedia.org/wiki/Archivo:Baraja_de_40_cartas.png

- Los ases valen 1 punto. Es decir, las cartas con las denominaciones 1 y 2 valen 1 punto cada una de ellas.
- El resto de las cartas valen según su denominación. Por ejemplo, la carta con denominación 5 vale 5 puntos.

La puntuación de las cartas no depende del palo al que pertenecen. Por ejemplo, el rey de copas (RC) y el tres de oros (3O) tienen la misma puntuación, 10 puntos.

2.2 Desarrollo de una Partida

Para jugar una Partida de MiniMus se necesitan dos Parejas de dos Jugadores. Se disponen los Jugadores de manera que se alterna un Jugador de cada Pareja. Si llamamos a las respectivas Parejas PA y PB, y a los Jugadores de una Pareja J1 y J2, los Jugadores se ordenan en la secuencia J1PA, J1PB, J2PA y J2PB. Este orden es relevante únicamente para designar en cada Jugada qué Jugador tiene el rol de Mano (ver más abajo).

El objetivo del juego es sumar el mayor número de Piedras (o tantos) posible. Una Partida consta de una sucesión de Jugadas que se termina cuando alguna de las Parejas consigue 40 Piedras o más. La Pareja que consigue más Piedras es la Pareja ganadora. En caso de empate, gana la Pareja a la que pertenecía el Mano de la última Jugada.

Al principio de cada Jugada, se reparten aleatoriamente 4 Cartas de la Baraja completa a cada Jugador, es decir, se reparten aleatoriamente 16 Cartas del total de 40 (cf. Fig. 2).

Figura 2. Jugada en una partida de MiniMus, con las cartas dispuestas para contabilizar las Piedras que gana cada Pareja (véanse las Piedras en el centro y en la derecha)[2]

[2] Imágen disponible en: https://es.wikipedia.org/wiki/Mus#/media/Archivo:Tapete.jpg

En los ejemplos que siguen en el presente documento, las cartas que tiene un Jugador se representarán mediante cuatro Cartas entre paréntesis. Por ejemplo, para el jugador de la derecha en la Fig. 2 tendremos (RO, 3B, 3O, 3C) y para el de la izquierda (CB, 1C, 2E, 2C).

Para la primera Jugada, se elige aleatoriamente un Jugador como Mano. Para el resto de las Jugadas, el rol de Mano va rotando de acuerdo con el orden de Jugadores establecido anteriormente.

Ejemplo. Si el primer Jugador Mano es J2PA, los Jugadores Mano de las siguientes Jugadas serán, por este orden, J2PB, J1PA, J1PB, J2PA, y así sucesivamente.

En cada Jugada, una vez repartidas las Cartas, se computan las Piedras obtenidas por cada Pareja de Jugadores de acuerdo a cuatro tipos de Lances (o desafíos) según las siguientes reglas:

- **Lance de Grande:** Gana la Pareja en la que uno de sus Jugadores tiene las cartas de mayor valor. Las cartas de mayor tamaño son los reyes, seguidas de caballos, sotas, sietes, seises, cincos, cuatros y ases, por este orden. Así, para decidir qué Jugador tiene las cartas de mayor tamaño, se cuenta primero el número de reyes de cada Jugador (es decir, cartas con la denominación 3 o R). Gana el Jugador que tiene mayor número de reyes. Si hay empate en el número de reyes, se cuenta el número de caballos (cartas con la denominación C). Si sigue el empate se continúa con el resto de las cartas hasta llegar a los ases (cartas con la denominación 1 o 2). La Pareja del Jugador que tiene la mejor Grande gana tres Piedras. En caso de empate, ambas Parejas consiguen una Piedra. Sólo cuentan los Jugadores de cada Pareja con las Cartas de mayor tamaño.

  Ejemplos:
  - (3O, RO, 1C, 6B) gana a (RC, RB, 5C, 4E) a Grande
  - (RB, 3C, 2O, 2B) gana a (3O, CC, CB, SB) a Grande

- **Lance de Chica**: Gana la Pareja en la que uno de sus Jugadores tiene las cartas de menor valor. De manera análoga al lance de Grande, se cuenta el número de ases de cada Jugador (es decir, cartas con la denominación 1 o 2). Si hay empate en el número de ases, se cuenta el número de cuatros. Si sigue el empate se continúa con el resto de las cartas hasta llegar a los reyes (o treses). La Pareja del Jugador que tiene la mejor Chica gana tres Piedras. Si sigue habiendo empate, las dos Parejas ganan una Piedra cada una. Sólo cuentan los

Jugadores de cada Pareja con las Cartas de menor tamaño.

Ejemplos:
- (3O, RO, 1C, 6B) gana a (RC, RB, 5C, 4E) a Chica
- (RB, 3C, 2O, 1B) gana a (3O, CC, CB, SB) a Chica

- **Lance de Pares:** Se cuenta el número de Pares de cada jugador. Los Pares pueden ser:
  - Dúplex: dos parejas de cartas iguales, o cuatro cartas iguales.
  - Medias: tres cartas iguales.
  - Par: dos cartas iguales.

Ejemplos:
- (3O, RO, 5C, 4B) y (3O, CC, CB, SB) son Pares
- (RC, RB, 3C, 2E) son Medias
- (RB, 3C, 2O, 1B) y (RO, 3C, 3O, RB) son Dúplex

Se suma a la puntuación de cada pareja:

- 1 Piedra por cada Par que tenga cada jugador de dicha Pareja.
- 2 Piedras por cada Medias.
- 3 Piedras por cada Dúplex.

- **Lance de Juego:** Se suma la puntuación de las cartas de cada Jugador (ojo: el valor es distinto de la denominación; como se especificó en el apartado 2.1, todas las figuras, es decir, la cartas con denominación S, C, R y 3, valen 10 puntos). Si algún Jugador tiene entre 31 puntos y 40 puntos, se dice que tiene Juego. Gana la pareja del jugador que tenga mejor Juego. El valor del Juego es, de mayor a menor valor, 31, 32, 40, 37, 36, 35, 34, y 33. Si hay empate entre los jugadores que tienen (mejor) Juego de cada pareja, se comprueban las cartas de los dos Jugadores restantes. Si sigue habiendo empate, gana la Pareja del Jugador que es Mano. Si en caso de empate los dos miembros restantes de la pareja no tienen Juego, también gana la pareja cuyo jugador que tiene Juego es Mano. La Pareja que gana suma 2 Piedras. Además, cada Jugador que tenga Juego aportará a la puntuación de su Pareja 3 Piedras adicionales si tiene 31 y 2 Piedras por cualquier otro juego.

Ejemplo. Imaginemos que los jugadores de la Pareja PA tienen (3O, RO, 5C, 4B) y (RC, RB, 3C, 2E), mientras que los de PB tienen (RB, 3C, 2O, 1B) y (3O, CC, CB, SB). Ambas parejas tienen Juego. J2PA tiene 31, y J2PB tiene 40. PA gana el Juego y suma 2 Piedras. J2PA aporta 3 Piedras más a la puntuación de PA. J2PB aporta 2 Piedras a PB. En total PA gana 5 Piedras y PB 2 Piedras en el Lance de Juego.

Sólamente si ningún Jugador tiene Juego, se juega al Punto: gana la Pareja a la que pertenece el Jugador cuyas cartas suman el valor más alto, entre 30 (mejor Punto) y 4 (peor Punto posible). La Pareja que gana suma 1 Piedra. Si hay empate, se comprueban las cartas de los dos Jugadores restantes. Si sigue habiendo empate, gana la Pareja del Jugador que es Mano.

Jugada de ejemplo. Supongamos que se reparten las siguientes cartas:

J1PA: *(3O, RO, 1C, 2B)
J1PB: (4C, 4B, 4E, SB)
J2PA: (5E, 4C, 3E, 3B)
J2PB: (6B, 6E, RB, CC)

Y que es Mano J1PA. Las puntuaciones serían las siguientes:

- Lance de Grande: tanto J1PA como J2PA tienen 2 reyes. La pareja A gana 3 Piedras.
- Lance de Chica: J1PA tiene 2 ases. La pareja A gana también la chica y obtiene 3 Piedras.
- Lance de Pares: J1PA tiene Dúplex y suma 3 Piedras; J1PB tiene Medias y suma 2 Piedras; J2PA y J2PB tienen cada un Par, con lo que suma cada uno 1 Piedra.
- Lance de Juego. Las cartas de los jugadores suman respectivamente 22, 22, 29, y 32. Por lo tanto, la pareja B gana 2 Piedras por tener el mejor Juego, y luego 2 Piedras más por el juego de 32.

En total, las Piedras sumadas en esta Jugada serían:

- Pareja A: 3 + 3 + 3 + 1 = 10 Piedras.
- Pareja B: 2 + 1 + 2 + 2 = 7 Piedras.

3. APLICACIÓN MiniMus

Los alumnos deberán desarrollar una aplicación Java siguiendo el paradigma de Programación Orientada a Objetos (POO) que implemente el juego de MiniMus descrito en el apartado anterior. Esta aplicación funcionará en modo CLI (Command-Line Interface), es decir, de manera análoga a un comando en línea de Linux, con un comportamiento variable que dependerá de los parámetros utilizados en su invocación.

```
$> java MiniMus [parámetros]
```

La aplicación dispondrá de 3 modos de funcionamiento: i) juego autónomo, ii) juego con reparto de cartas preestablecido y iii) ejecución de comandos. En los siguientes sub-apartados se describen las particularidades de cada modo de funcionamiento.

3.1 Modo 1: Juego autónomo

3.1.1 Invocación

```
$> java MiniMus [-p fichero_jugadores] [-o fichero_partida]
```

3.1.2 Funcionalidad

La aplicación juega una partida completa de MiniMus de manera autónoma. En caso de que en la invocación de la aplicación se utilice el parámetro `-p fichero_jugadores`, la información de las parejas y jugadores que participarán en el juego se obtendrá del fichero especificado (en el Apéndice A.1 se detalla el formato de este fichero). En concreto, se seleccionarán las dos primeras parejas disjuntas que aparezcan en el fichero. Por ejemplo, en el caso de que el `fichero_jugadores` contuviese la siguiente información:

```
J 666 Pepa Pérez
J 777 Jon Nieve
J 4 Antonio Melero
P 666 666 777 Los Mejores
P 555 4 777 Los Segundos Mejores
J 5 Ana María Toledano
P 77 4 5 Las Notas
```

jugarían las parejas “Los Mejores” (conformada por los jugadores “Pepa Pérez” y “Jon Nieve”) y “Las Notas” (conformada por los jugadores “Antonio Melero” y “Ana María Toledano”).

En el caso de que no se utilice el parámetro `-p fichero_jugadores`, o de que la información contenida en fichero especificado no sea suficiente para configurar las dos parejas, la aplicación completará las parejas y/o jugadores con los siguientes valores por defecto:

- Primera pareja: “Pareja A”, con los jugadores “Jugador 1A” y “Jugador 2A”.
- Segunda pareja: “Pareja B” con los jugadores “Jugador 1B” y “Jugador 2B”.

Una vez establecidas las parejas, la aplicación generará y resolverá una secuencia de jugadas en las que repartirá cartas aleatoriamente hasta que alguna de las parejas consigue 40 Piedras o más.

El resultado de la partida se volcará en el fichero especificado en el parámetro `-o fichero_partida`, siempre y cuando se haya utilizado cuando se invoca la aplicación, o por salida estándar si este no hubiese sido establecido. En ambos casos, la salida tendrá el formato que se detalla en el Apéndice A.2.

3.1.3 Ejemplo

A continuación se presenta un ejemplo de invocación del juego en modo autónomo que utiliza la información de jugadores contenida en el fichero “misjugadores.txt” y que muestra la salida por pantalla.

```
$> java MiniMus -p misjugadores.txt

Los Mejores: Pepa Pérez y Jon Nieve.
Las Notas: Antonio Melero y Ana María Toledano.
Mano: Pepa Pérez.
*(3O, RO, 1C, 2B)-(4C, 4B, 4E, SB)-(5E, 4O, 3E, 3B)-(6B, 6E, RB, CC)
Grande 3 0 Chica 3 0 Pares 4 3 Juego 0 4 – 10 07
-(6B, SB, 1C, 1O)*(6E, 4O, RO, 3B)-(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C)
Grande 0 3 Chica 3 0 Pares 4 4 Juego 0 4 – 17 18
            ...
            ...
            ...
*(CO, RO, SB, 4E)-(SO, CC, 5B, 5O)-(1C, 4C, 4O, 3B)-(7E, 2B, 2O, CE)
Grande 3 0 Chica 0 3 Pares 1 2 Juego 4 0 – 42 28
Gana: Los Mejores. Número total de jugadas: 14.
```

Como se puede observar, las 2 primeras líneas incluyen información sobre las parejas de jugadores y la línea 3 informa qué jugador es mano de salida. La siguientes líneas contienen información sobre las sucesivas jugadas: primero se presenta en una línea el reparto de cartas entre los 4 jugadores, según el orden J1PA, J1PB, J2PA, J2PB (el reparto del jugador Mano aparece con un *) y en la siguiente línea se presentan las piedras obtenidas por cada pareja en cada lance, según el orden PA, PB, así como el número de piedras acumulado. La última línea indica qué pareja resulta ganadora y el número de jugadas realizadas. Nótese que esta última línea termina con un “.”, y que que la línea anterior tiene que reflejar un número de piedras acumulado mayor o igual que 40 para al menos una pareja. En el apéndice A.2 se describe formalmente el formato de salida.

3.1.4 Gestión de errores

En las pruebas que se realicen a la aplicación MiniMus, los ficheros con información de entrada siempre serán correctos, tanto sintácticamente (es decir, cumplirán estrictamente el formato definido en los apéndices) como semánticamente (es decir, la información en ellos será siempre coherente). Por tanto, la aplicación no considerará necesariamente errores de parseado de ficheros. En lo que respecta a los ficheros, la aplicación únicamente deberá considerar necesariamente errores de lectura/escritura.

De manera análoga, la invocación de la aplicación siempre se realizará de manera consistente en las pruebas: los parámetros siempre serán utilizados de manera correcta.

3.2 Modo 2: Juego con reparto de cartas preestablecido

3.2.1 Invocación

```
$> java MiniMus -j fichero_jugadas [-p fichero_jugadores] [-o
fichero_partida]
```

3.2.2 Funcionalidad

La aplicación funciona de manera análoga al modo autónomo, pero en este caso el reparto de cartas está predeterminado según el contenido del fichero `fichero_jugadas`. Este fichero contiene en cada línea el reparto de cartas de una jugada, con el mismo formato del fichero de salida antes tratado, salvo que el asterisco indicando el jugador o jugadora Mano siempre aparece únicamente en la primera jugada (y, por tanto, el fichero determina quién es Mano inicialmente). Para el resto de las jugadas, el o la jugadora mano será el que corresponda según las reglas del juego. Cada jugada está compuesta de 16 cartas diferentes distribuidas en 4 grupos de 4 cartas. Su formato está descrito en el Apéndice A.3. A continuación se muestra un ejemplo:

```
-(3O, RO, 1C, 2B)-(4C, 4B, 4E, SB)*(5E, 4O, 3E, 3B)-(6B, 6E, RB, CC)
-(6B, SB, 1C, 1O)-(6E, 4O, RO, 3B)-(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C)
       ...
       ...
       ...
-(CO, RO, SB, 4E)-(SO, CC, 5B, 5O)-(1C, 4C, 4O, 3B)-(7E, 2B, 2O, CE)
```

En el caso de el fichero de jugadas no contenga suficientes jugadas para que alguna de las parejas alcance las 40 Piedras, la última línea del `fichero_partida` (o la salida estándar si este fichero no se ha especificado) contendrá la expresión:

`Partida incompleta. Número total de jugadas: <n>.`

donde <n> es el número de jugadas incluidas en el fichero de jugadas (en lugar de especificar la pareja ganadora, como ocurre en el ejemplo mostrado en el Apartado 3.1.3).

3.2.3 Ejemplo

A continuación se muestra un ejemplo de invocación de la aplicación en modo de juego con reparto preestablecido, donde las jugadas están descritas en el fichero “misjugadas.txt” y el resultado de la partida se vuelca en el fichero “mipartida.txt”. No se especifica fichero de

jugadores, con lo que se utilizarán los jugadores por defecto.

```
$> java MiniMus -j misjugadas.txt -o miresultado.txt
```

En el caso de que el fichero “misjugadas.txt” solo contuviese dos jugadas, insuficientes para alcanzar el final de la partidas, el contenido del fichero “miresultado.txt” podría ser:

```
Pareja A: Jugador 1A y Jugador 2A.
Pareja B: Jugador 1B y Jugador 2B.
Mano: Jugador 1B.
-(3O, RO, 1C, 2B)*(4C, 4B, 4E, SB)-(5E, 4C, 3E, 3B)-(6B, 6E, RB, CC)
Grande 3 0 Chica 3 0 Pares 4 3 Juego 0 4 - 10 07
-(6B, SB, 1C, 1O)-(6E, 4O, RO, 3B)*(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C)
Grande 0 3 Chica 3 0 Pares 4 4 Juego 0 4 - 17 18
Partida incompleta. Número total de jugadas: 2.
```

3.2.4 Gestión de errores

Deben contemplarse las mismas consideraciones que se han descrito en el apartado 3.1.4.

3.3 Modo 3: Ejecución de comandos

3.3.1 Invocación

```
$> java MiniMus -c fichero_comandos [-o fichero_salida]
```

3.3.2 Funcionalidad

En el modo de ejecución de comandos la aplicación no juega una partida, sino que lleva a cabo una serie de instrucciones o comandos predefinidos. Estos comandos afectan a jugadas o lances individuales, a la configuración del juego, o a la gestión de jugadores, parejas y cartas. Los comandos a ejecutar por parte de la aplicación se almacenan en el fichero especificado en el parámetro de invocación: `-c fichero_comandos`. En este fichero, cada comando ocupa una única línea, y sólo puede haber un comando por línea. Las líneas que comiencen por el carácter “#” se consideran comentarios y se ignoran. Las líneas en blanco también se ignoran. Los resultados de la ejecución se presentan en la salida estándar, a no ser que se especifique el parámetro de invocación `-o fichero_salida`, en cuyo caso los resultados se depositan en el fichero especificado. En caso de que el `fichero_salida` especificado ya exista en el directorio desde el que se invoca la aplicación, éste se sobreescribirá.

Los comandos que puede ejecutar la aplicación son los siguientes:

Comando 1	`NewPlayer <id> <nombre>`
descripción	Se crea un nuevo jugador con identificador <id> y nombre <nombre>. El identificador será un número entre 0 y 999.
errores	• En caso de que ya exista un jugador con el <id> especificado en la sesión se considerará un error.

salida	En caso de éxito, se genera en una línea la expresión:
	`NewPlayer <id> <nombre>: OK.`
	En caso de error, se genera la expresión:
	`NewPlayer <id> <nombre>: FAIL.`

ejemplo	`NewPlayer 432 Johann Sebastian Bach`

Comando 2	`DeletePlayer <id>`
descripción	Se elimina el jugador con identificador <id>.
errores	• En caso de que no exista un jugador con el identificador <id> especificado porque no se había creado previamente en esta sesión, se genera un error.

salida	En caso de éxito, se genera la expresión:
	`DeletePlayer <id>: OK.`
	En caso de fallo, la expresión:
	`DeletePlayer <id>: FAIL.`

ejemplo	`DeletePlayer 42`

Comando 3	`NewCouple <id> <id_j1> <id_j2> <nombre>`
descripción	Se crea una pareja con identificador <id> y nombre <nombre>, compuesta por los jugadores cuyos identificadores son <id_j1> y <id_j2>.
errores	• En caso de que ya exista una pareja con identificador <id>, se genera un error. • En caso de que no exista alguno de los jugadores se genera un error.

salida	En caso de éxito se genera la expresión:
	`NewCouple <id> <id_j1> <id_j2> <nombre>: OK.`
	En caso de error, la expresión:
	`NewCouple <id> <id_j1> <id_j2> <nombre>: FAIL.`

ejemplo	`NewCouple 666 666 777 Los Mejores`

Comando 4	`DeleteCouple <id>`
descripción	Se borra la pareja cuyo identificador es <id>.
errores	• En caso de que no exista una pareja con identificador <id> se genera un error.

salida	En caso de éxito se genera la expresión:
	`DeleteCouple <id>: OK.`
	En caso de error:
	`DeleteCouple <id>: FAIL.`

ejemplo	`DeleteCouple 666`

Comando 5	`DumpPlayers <fichero_jugadores>`
descripción	Se genera un fichero <fichero_jugadores> con el formato descrito en el Apéndice A.1, donde se almacenan todos los jugadores y parejas creadas en esta sesión. Se almacenan por orden de identificador, primero todos los jugadores y a continuación todas las parejas. Si ya existía un fichero <fichero_jugadores>, se sobreescribe con la nueva información.
errores	• Si no se puede escribir el fichero especificado se produce un error.

salida	En caso de éxito se genera la expresión
	
	`DumpPlayers <fichero_jugadores>: OK.`
	
	En caso de error, la expresión:
	
	`DumpPlayers <fichero_jugadores>: FAIL.`

ejemplo	`DumpPlayers misjugadores.txt`

Comando 6	`ResetPlayers`
descripción	Borra toda la información de jugadores y parejas generada en esta sesión. No modifica ningún fichero de jugadores que pudiera existir.
errores	
salida	Se genera la expresión: `ResetPlayers: OK.`
ejemplo	`ResetPlayers`

Comando 7	`LoadPlayers <fichero_jugadores>`
descripción	Carga en memoria la información de jugadores y parejas contenida en el fichero <fichero_jugadores> y se añade a la existente en la sesión. Si en esta sesión ya existían jugadores o parejas con el mismo número de referencia, se sobreescriben los jugadores o parejas existentes.
errores	• Si no se puede leer el fichero especificado se produce un error.

salida	En caso de éxito se genera la expresión:
	`LoadPlayers <fichero_jugadores>: OK.`
	En caso de fallo, la expresión:
	`LoadPlayers <fichero_jugadores>: FAIL.`

ejemplo	`LoadPlayers players.txt`

Comando 8	`GenerateRandomDelivery <n_jug> <id_mano> <fichero_jugadas>`
descripción	Se genera un fichero de jugadas de nombre <fichero_jugadas>, con un total de <n_jug> jugadas obtenidas aleatoriamente. Se marca como jugador Mano el jugador de orden <id_mano>, un número entre 1 y 4 que se corresponden, respectivamente, a J1PA, J1PB, J2PA y J2PB .
errores	• Si no se puede escribir el fichero especificado se produce un error.

salida	En caso de éxito se muestra la expresión:
	`GenerateRandomDelivery <n_jug> <id_mano> <fichero_jugadas>: OK.`
	En caso de fallo, la expresión:
	`GenerateRandomDelivery <n_jug> <id_mano> <fichero_jugadas>: FAIL.`

ejemplo	`GenerateRandomDelivery 35 3 mipartida.txt`

Comando 9	`PlayGame <fichero_jugadas> <fichero_partida>`

descripción	Se juega una partida con <fichero_jugadas>. Se generan jugadores y parejas por defecto, a no ser que se hayan creado ya al menos 4 jugadores y 2 parejas disjuntas en la misma sesión. El resultado de la partida se almacena en el fichero <fichero_partida> (debe tenerse en cuenta que el <fichero_jugadas> puede no contener suficientes jugadas como para alcanzar un final de partida, tal como se describe en el Apartado 3.2.2). En caso de que el <fichero_partida> ya exista, este se sobreescribirá.
errores	• Si no se puede leer el <fichero_jugadas> se produce un error. • Si no se puede escribir el <fichero_partida> se produce un error.

salida	En caso de éxito se muestra la expresión:
	
	`PlayGame <fichero_jugadas> <fichero_partida>: OK.`
	
	En caso de fallo, la expresión:
	
	`PlayGame <fichero_jugadas> <fichero_partida>: FAIL.`

ejemplo	`PlayGame mydealing.txt gameoutput.txt`

Comando 10	`PlayHand <jugada>`
descripción	Se resuelven los cuatro lances para la jugada <jugada>, especificada de acuerdo a lo indicado anteriormente, incluyendo un “*” para indicar qué jugador es mano.
errores	
salida	Se genera una línea de la forma: `PlayHand <jugada>: Grande <P PA> <P PB>, Chica <P PA> <P PB>, Pares <P PA> <P PB>, Juego <P PA> <P PB>` donde <P Px> se corresponde con las piedras obtenidas por la Pareja x en un determinado lance

ejemplo	`PlayHand -(6B, SB, 1C, 1O)*(6E, 4O, RO, 3B)-(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C)`
	
	presentaría el resultado:
	`PlayHand -(6B, SB, 1C, 1O)*(6E, 4O, RO, 3B)-(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C): Grande 0 3, Chica 3 0, Pares 4 4, Juego 0 4`

Comando 11	`ResolvePares <jugada>`
descripción	Se computa un lance de Pares para la jugada `jugada`, indicando los puntos obtenidos por cada jugador y por cada pareja.
errores	
salida	Se genera una línea de la forma: `ResolvePares <jugada>: <P J1PA> <P J1PB> <P J2PA> <P J2PB> - <P PA> <P PB>` donde: • <P JxPy> se corresponde con las piedras obtenidas por el Jugador x de la Pareja y • <P Pz> se corresponde con las piedras obtenidas por la Pareja z

ejemplo	`ResolvePares -(6B, SB, 1C, 1O)*(6E, 4O, RO, 3B)-(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C)`
	
	presentaría el resultado:
	
	`ResolvePares -(6B, SB, 1C, 1O)*(6E, 4O, RO, 3B)-(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C): 1 1 3 3 - 4 4`

Comando 12	`ResolveJuego <jugada>`
descripción	Se computa un lance de Juego para la jugada <jugada>, indicando los puntos obtenidos por cada jugador y por cada pareja.
errores	
salida	Se genera una línea de la forma: `ResolveJuego <jugada>: <P J1PA> <P J1PB> <P J2PA> <P J2PB> - <P PA> <P PB>` donde: • <P JxPy> se corresponde con las piedras obtenidas por el Jugador x de la Pareja y • <P Pz> se corresponde con las piedras obtenidas por la Pareja z

ejemplo	`ResolveJuego -(6B, SB, 1C, 1O)*(6E, 4O, RO, 3B)-(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C)`
	
	Presenta como resultado:
	
	`ResolveJuego -(6B, SB, 1C, 1O)*(6E, 4O, RO, 3B)-(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C): 0 0 0 4 - 0 4`

Comando 13	`ResolveGrande <jugada>`
descripción	Se computa un lance de Grande para la jugada <jugada>, indicando los puntos obtenidos por cada jugador y por cada pareja.
errores	
salida	Se genera una línea de la forma: `ResolveGrande <jugada>: <P J1PA> <P J1PB> <P J2PA> <P J2PB> - <P PA> <P PB>` donde: • <P JxPy> se corresponde con las piedras obtenidas por el Jugador x de la Pareja y • <P Pz> se corresponde con las piedras obtenidas por la Pareja z

ejemplo	`ResolveGrande -(6B, SB, 1C, 1O)*(6E, 4O, RO, 3B)-(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C)`
	
	Presenta como resultado:
	
	`ResolveGrande -(6B, SB, 1C, 1O)*(6E, 4O, RO, 3B)-(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C): 0 3 0 0 - 0 3`

Comando 14	`ResolveChica <jugada>`
descripción	Se computa un lance de Grande para la jugada <jugada>, indicando los puntos obtenidos por cada jugador y por cada pareja.
errores	

salida	Se genera una línea de la forma: `ResolveChica <jugada>: <P J1PA> <P J1PB> <P J2PA> <P J2PB> - <P PA> <P PB>` donde: • <P JxPy> se corresponde con las piedras obtenidas por el Jugador x de la Pareja y • <P Pz> se corresponde con las piedras obtenidas por la Pareja z

ejemplo	`ResolveChica -(6B, SB, 1C, 1O)*(6E, 4O, RO, 3B)-(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C)`
	
	Presenta como resultado:
	
	`ResolveChica -(6B, SB, 1C, 1O)*(6E, 4O, RO, 3B)-(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C): 0 0 3 0 - 3 0`

3.3.3 Ejemplo

Ejemplo de invocación de la aplicación en modo ejecución de comandos donde los comandos se especifican en el fichero comandos.txt y la salida se deposita en el fichero trescom.txt:

```
$> java MiniMus -c comandos.txt -o trescom.txt
```

Si el fichero “comandos.txt” contiene la siguiente información:

```
# Generación de un nuevo fichero de jugadores
ResetPlayers
NewPlayer 10 Shakira
NewPlayer 20 Justin Timberlake
NewCouple 1 10 20 Los cantantes
NewCouple 2 3 4 Los jinetes del apocalipsis
NewPlayer 1 Uma Thurman
NewPlayer 2 Jean-Claude Van Damme
NewCouple 2 1 2 Los actores
DumpPlayers misjugadores.txt
```

Se generará el fichero “misjugadores.txt”, según el formato especificado en el Apéndice A.1.

```
J 1 Uma Thurman
J 2 Jean-Claude Van Damme
J 10 Shakira
J 20 Justin Timberlake
P 1 10 20 Los cantantes
P 2 1 2 Los actores
```

y el fichero de salida “trescom.txt” con el siguiente contenido:

```
ResetPlayers: OK.
NewPlayer 10 Shakira: OK.
NewPlayer 20 Justin Timberlake: OK.
NewCouple 1 10 20 Los cantantes: OK.
NewCouple 2 3 4 Los jinetes del apocalipsis: FAIL.
NewPlayer 1 Uma Thurman: OK.
NewPlayer 2 Jean-Claude Van Damme: OK.
NewCouple 2 1 2 Los actores: OK.
DumpPlayers misjugadores.txt: OK.
```

3.3.4 Gestión de errores

Mismas consideraciones que en el apartado 3.1.4.

4. DESARROLLO, ENTREGA Y EVALUACIÓN

4.1 Desarrollo del proyecto

El desarrollo del proyecto se llevará a cabo siguiendo el paradigma de Programación Orientada a Objetos. Este desarrollo se articula en dos fases principales:

1. La primera fase consiste en el diseño de un **diagrama UML de clases de alto nivel** adecuado para el proyecto, en el que se identificarán:
    a. Las principales clases e interfaces, así como las relaciones en el dominio del problema (relaciones de agregación, composición, herencia, realización de interfaces, etc.).
    b. Los atributos de cada una de dichas clases/interfaces, multiplicidad, restricciones de cardinalidad, roles, etc.
2. La segunda fase consiste en la implementación de la aplicación. De manera iterativa e incremental, durante esta fase:
    a. El diagrama de clases será refinado para la obtención de un diseño detallado

mediante un **diagrama UML de clases de bajo nivel** en el que se añadirán todos los métodos necesarios para poder implementar la funcionalidad exigida en el proyecto.

b. Se codificará el modelo orientado a objetos diseñado utilizando el lenguaje de programación multiplataforma Java.

Los alumnos podrán utilizar cualquier herramienta (plugin o standalone) conforme con UML para la realización del diseño de la solución. Se recomienda, no obstante, el uso de Visual Paradigm for UML Community Edition (disponible de manera gratuita para entornos educativos en https://www.visual-paradigm.com/download/community.jsp). Para la codificación se utilizará el Java SE Development Kit 8 de Oracle (disponible en https://www.oracle.com/technetwork/pt/java/javase/downloads/jdk8-downloads-2133151.html). Para la edición del código se podrá utilizar cualquier entorno o editor de programación. Se recomienda, en cualquier caso:

- El Entorno de Desarrollo Integrado (IDE) de código abierto Eclipse (disponible en https://www.eclipse.org/downloads/packages/).
- El editor de código abierto Visual Code Studio (disponible en https://code.visualstudio.com/)

4.2 Entregas

Los alumnos deberán entregar, dependiendo del modo de evaluación escogido, lo siguiente:

- **Evaluación Continua (EC)**: El desarrollo de la aplicación se realizará en grupos de dos alumnos, constituidos a principios de curso para la realización de las prácticas de iniciación en Java. En esta modalidad de evaluación se realizarán dos entregas:

    1. *Contenido*: Diseño de clases de alto nivel.
       Fecha: fecha oficial aprobada por la CAG y publicada al comienzo del cuatrimestre de impartición de la asignatura.
       Formato: PDF.
       Calificación Máxima Global: 0,5 puntos.

    2. *Contenido*: diagrama de clases de bajo nivel (en el que se hayan incorporado todas las modificaciones que el alumnado considere oportunas desde la primera entrega) junto con el código y la documentación Javadoc del proyecto.
       Fecha: A final de curso (en la fecha oficial aprobada por la CAG).
       Formato: fichero ZIP.

Calificación Máxima Global: 3 puntos.

La entrega en ambos casos se realizará a través de FaiTIC, mediante un único fichero por grupo.

- **Evaluación Única (EU)**: El desarrollo del proyecto se realizará de forma individual, efectuando una única entrega al final del cuatrimestre (en la fecha oficial aprobada por la CAG), en la que se aportará el diagrama de clases de bajo nivel, el código y su documentación Javadoc. La calificación máxima global será de 5 puntos, 1 punto para el diagrama UML y 4 puntos para el código y documentación.

4.3 Evaluación

Todos los alumnos, con independencia de la modalidad de evaluación elegida (EC o EU) deberán realizar un examen práctico individual (en la fecha oficial aprobada por la CAG) en el que se les pedirá una modificación menor de la implementación del proyecto entregada. La calificación de dicha prueba será APTO o NO APTO. Sólo los alumnos que superen el examen práctico optarán a la evaluación del proyecto; al resto se les asignará directamente una nota de 0 puntos en la implementación del proyecto. La calificación se basará en la corrección, adecuación al paradigmas POO y calidad del diseño, código y documentación entregada.

APÉNDICES. FORMATOS DE FICHEROS

A.1 Fichero de jugadores

El fichero de jugadores contiene información sobre jugadores y parejas. Se compone de un conjunto de líneas, donde en cada una de ellas se define o bien un jugador o bien una pareja.

- Una línea de jugador comienza con la letra “J” seguida de un identificador numérico del jugador (de cero a tres dígitos decimales significativos con un valor entre 0 y 999) y de una cadena de caracteres con el nombre del jugador o jugadora. Cada uno de estos elementos está separado por un espacio en blanco.
- Una línea de pareja comienza con la letra “P” seguida de un identificador numérico de la pareja (de cero a tres dígitos decimales significativos con un valor entre 0 y 999), de los identificadores numéricos de los jugadores que la conforman, y de una cadena de caracteres que indica el nombre de la pareja. Para que una definición de pareja sea válida tienen que haberse definido previamente sus dos jugadores. Cada uno de estos elementos está separado por un espacio en blanco.

Ejemplo de fichero de jugadores:

```
J 10 Pitufo Goloso
J 12 Bebé Pitufo
J 4 Pitufina
P 10 10 12 Pitufos
P 777 4 12 Más Pitufos
J 5 Sonic the Hedgehog
P 11 4 5 Personajes Azules
```

A.2 Fichero de partida

El fichero de partida describe el contexto y el transcurso de una partida de MiniMus. Contiene la siguiente información:

- Las tres primeras líneas indican las parejas que juegan y el jugador o jugadora mano inicial, según el siguiente formato:

  NombrePA: NombreJ1 y NombreJ2.

  NombrePB: NombreJ1 y NombreJ2.

  Mano: NombreJugadorMano.

  Se utiliza “: ” para separar el nombre de la pareja de sus integrantes, un “ y ” entre los nombres de los dos jugadores y un “.” para terminar la línea. En la tercera línea, aparece la expresión “Mano: ” seguida del nombre del jugador que es mano y un “.”.

- A continuación, se representan de manera consecutiva los resultados de una serie de Jugadas, donde cada jugada se compone de 2 líneas:
  - La primera línea representa el reparto de cartas mediante 4 conjuntos de 4 cartas entre paréntesis, en el orden J1PA, J1PB, J2PA, J2PB, con el mismo formato que en el fichero de Jugadas (ver Apéndice A.3) y siempre incluirá un asterisco delante de las cartas del Jugador mano para dicha Jugada.
  - La segunda línea representa el resultado en piedras de cada lance para cada pareja (Grande, Chica, Pares, Juego), en el orden PA PB, y el número de piedras acumulado hasta el momento por cada pareja tras completarse la jugada actual, también en el orden PA PB. Los resultados de los lances están separados del número total de Piedras acumulado por “ - ” (un espacio en blanco, un guión, y otro espacio en blanco).

- La última línea indica el desenlace de la partida:
    - En el caso de que ésta llegue a término, se representa qué pareja ganó la partida y el número total de jugadas completadas. La línea comienza con la expresión “Gana: ”, seguido por el nombre de la Pareja ganadora terminado por un “.” y un espacio en blanco. A continuación aparece la expresión “Número total de jugadas: ” seguido por el número total de Jugadas incluídas en el fichero, y terminado con “.”:

        Gana: <NombreParejaGanadora>. Número total de jugadas: <n>.

    - En el caso de que la partida termine sin que ninguna pareja haya alcanzado una puntuación de 40 piedras, la línea tendrá el formato:

        Partida incompleta. Número total de jugadas: <n>.

        donde <n> es el número total de jugadas realizadas hasta ese momento.

Ejemplo de fichero de partida:

```
Pareja 1: Jugador11 y Jugador21.
Pareja 2: Jugador12 y Jugador22.
Mano: Jugador11.
*(3O, RO, 1C, 2B)-(4C, 4B, 4E, SB)-(5E, 4C, 3E, 3B)-(6B, 6E, RB, CC)
Grande 3 0 Chica 3 0 Pares 4 3 Juego 0 4 – 10 07
-(6B, SB, 1C, 1O)*(6E, 4O, RO, 3B)-(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C)
Grande 0 3 Chica 3 0 Pares 4 4 Juego 0 4 – 17 18
…
…
…
*(CO, RO, SB, 4E)-(SO, CC, 5B, 5O)-(1C, 4C, 4O, 3B)-(7E, 2B, 2O, CE)
Grande 3 0 Chica 0 3 Pares 1 2 Juego 4 0 – 42 28
Gana: Pareja 1. Número total de jugadas: 14.
```

A.3. Fichero de jugadas

Se compone de un conjunto de líneas, donde en cada una de ellas se representa el reparto de 16 cartas a los jugadores de una partida, a razón de 4 cartas para cada uno de los 4 jugadores, con el siguiente formato:

```
?(c1, c2, c3, c4)?(c1, c2, c3, c4)?(c1, c2, c3, c4)?(c1, c2, c3, c4)
```

donde ? puede ser un “-” o un “*” y c1, c2, c3 y c4 se corresponde con la identificación de una

carta.

Como se puede observar, cada línea consta de 4 grupos (con el formato `?(c1, c2, c3, c4)`), que indican las 4 cartas asignadas a cada uno de los 4 jugadores de una partida según el orden J1PA, J1PB, J2PA, J2PB. Por defecto, las cartas del jugador están precedidas por un "-", pero cuando este es mano puede indicarse este hecho sustituyendo el guión por un "*".

Ejemplo de fichero de Jugadas:

```
-(3O, RO, 1C, 2B)-(4C, 4B, 4E, SB)*(5E, 4O, 3E, 3B)-(6B, 6E, RB, CC)
-(6B, SB, 1C, 1O)-(6E, 4O, RO, 3B)-(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C)
 …
 …
 …
-(CO, RO, SB, 4E)-(SO, CC, 5B, 5O)-(1C, 4C, 4O, 3B)-(7E, 2B, 2O, CE)
```

En este ejemplo, el jugador mano inicial sería el J2PA, es decir el jugador 2 de la pareja A. En las siguientes jugadas no está explicitado el jugador mano.

A.4. Fichero de comandos

Incluye la relación de comandos a ejecutar cuando se invoca el programa MiniMus en modo "ejecución de comandos" (ver Apartado 3.3). Cada comando ocupa una única línea del fichero, y sólo puede haber un comando en una línea dada. Las líneas que comienzan por el carácter "#" se consideran comentarios. Las líneas con comentarios y las líneas en blanco se ignoran.

Ejemplo de fichero de comandos:

```
#################################
# Fichero de comandos de ejemplo #
#################################

NewPlayer 666 Pepa Blanco
NewPlayer 777 Antonio Cuadrado
NewCouple 555 666 777 Pareja Happy

# Repartimos cartas y jugamos

GenerateRandomDelivery 35 3 mipartida.txt
PlayGame mipartida.txt

# Resolvemos algunas manos

PlayHand -(6B, SB, 1C, 1O)*(6E, 4O, RO, 3B)-(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C)
ResolveJuego -(6B, SB, 1C, 1O)*(6E, 4O, RO, 3B)-(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C)
ResolveChica -(6B, SB, 1C, 1O)*(6E, 4O, RO, 3B)-(1O, 2C, 1B, 2E)-(CC, CO, 7E, 7C)
```