Sesión 13: *Packages*

Hoja de problemas

Programación 2

Ángel Herranz
aherranz@fi.upm.es
Universidad Politécnica de Madrid

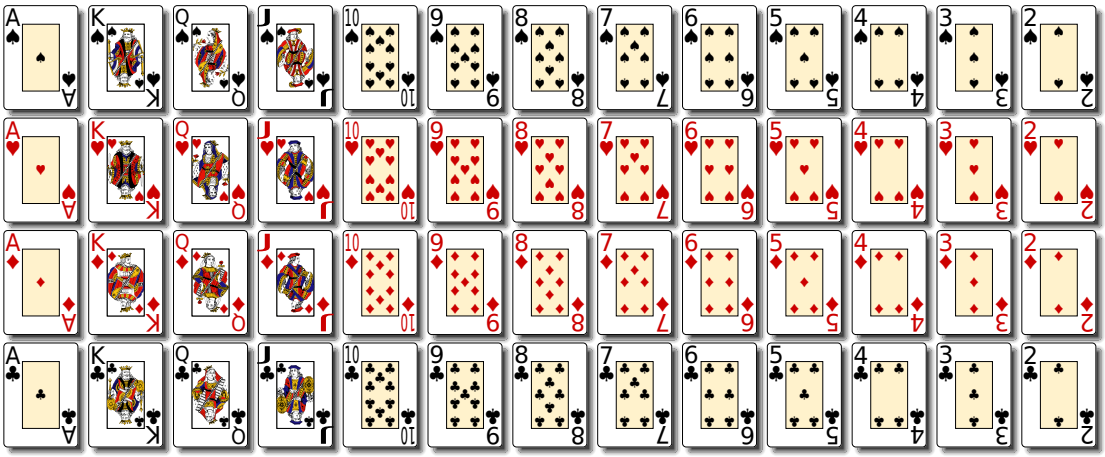
2023-2024

Ejercicio 1. Ya hemos mencionado el juego del *Poker* en las sesiones 6 y 7. En estos ejercicios vamos a profundizar en le juego a través de una variante del mismo: *Texas Hold'em*. Vamos a utilizar esta variante como ejemplo de modelización y, por primera vez, de modularización. Tu tarea empieza entendiendo en qué consiste el juego. Aunque lo voy a intentar explicar, te recomiendo una visita a la página de *Texas Hold'em* en Wikipedia

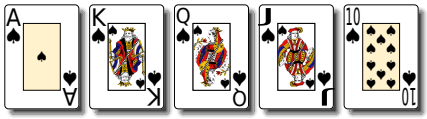
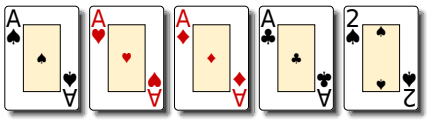
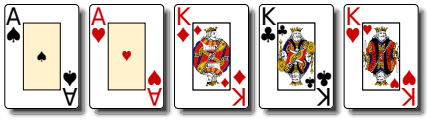
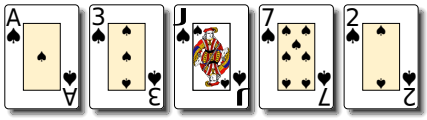
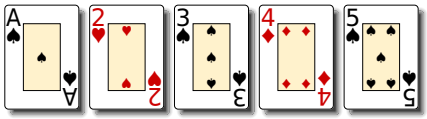
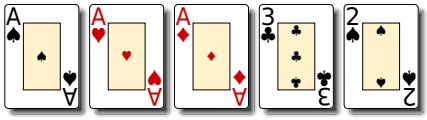
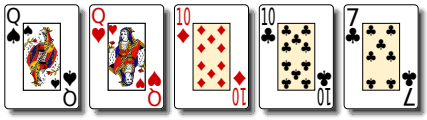
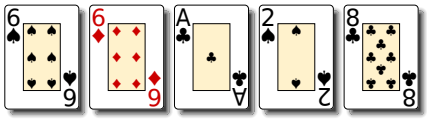
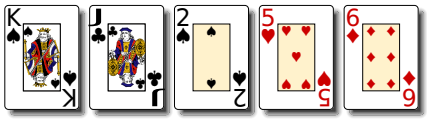
Ejercicio 2. *Texas Hold'em*, a partir de ahora Texas, es una variante del Poker y como tal se juega con la baraja francesa: 52 cartas organizadas en 4 palos y 13 valores:

Palos		**Valores**							
♠	Picas	A	As	K	Rey	Q	Dama	J	Valet
♥	Corazones	10	Diez	9	Nueve	8	Ocho	7	Siete
♦	Diamantes	6	Seis	5	Cinco	4	Cuatro	3	Tres
♣	Treboles	2	Dos						

Ejercicio 3. En este ejercicio bastará con recuperar tu modelización de las cartas: una clase `Naipe` capaz de representar las cartas de la baraja francesa:



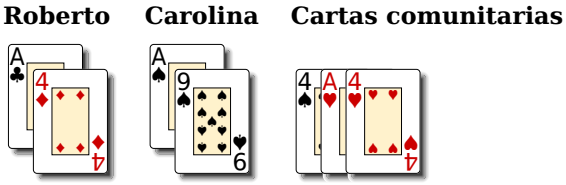
Ejercicio 4. ¿Cómo se juega a Texas? Empecemos un momento por el Poker. El Poker cada jugador tiene 5 cartas que forman una mano. La mano puede tener más o menos valor dependiendo de dichas cartas. La siguiente tabla muestra el *ranking* de manos del poker, de más a menos valor:

Mano	**Explicación y ejemplo**
Escalera de color	Cartas de valor correlativo del mismo palo 
Poker	Cuatro cartas del mismo valor 
Full	Dos cartas de un valor y tres de otro 
Color	Cinco cartas del mismo palo 
Escalera	Cartas correlativas no todas del mismo palo 
Trío	Tres cartas del mismo valor 
Dobles parejas	Dos y dos cartas del mismo valor 
Pareja	Dos cartas del mismo valor 
Carta más alta	Sin combinaciones, cuenta la carta de más valor 

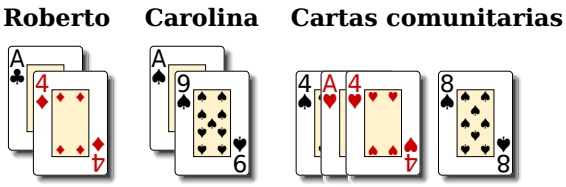
Ejercicio 5. En Texas, en vez de jugar con 5 cartas propias, cada jugador juega con 2 cartas propias y 3 cartas más a elegir entre varias cartas comunitarias. El juego se desarrolla en 4 etapas que os explico a continuación con algún ejemplo:

1. El juego comienza repartiendo 2 cartas a cada jugador, sólo el jugador puede verlas. A esto se le denomina **preflop**.

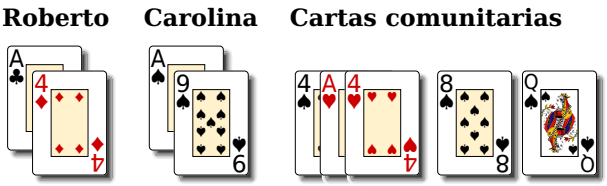
2. La siguiente etapa consiste en colocar sobre la mesa boca arriba 3 cartas comunitarias. A eso se le denomina **flop**. En ese momento, cada jugador tiene una jugada de poker combinando sus 2 cartas con las 3 comunitarias.



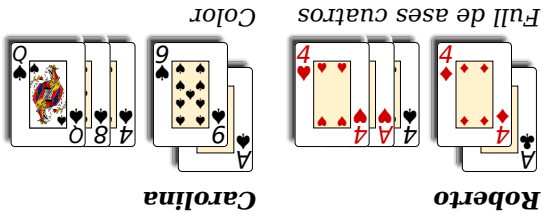
3. En la siguiente etapa se añade una carta comunitaria más. A esa etapa se le llama **turn**. Ahora el jugador lo tiene más complicado, tiene sus 2 cartas y puede elegir 3 de las 4 comunitarias.



4. La última etapa añade una nueva carta comunitaria, la quinta carta comunitaria. Esta etapa se llama **river**. Los jugadores montan definitivamente su mejor jugada con sus 2 cartas y con 3 de las 5 comunitarias.



Ejercicio 6. ¿Quién habría ganado la mano anterior? Para ello, Roberto y Carolina tienen que elegir tres cartas comunitarias, en particular las tres que maximicen su jugada.



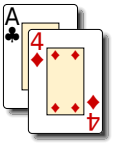
Ejercicio 7. Como puedes ver la cosa se empieza a poner complicada. Tendrás que representar los naipes, el reparto en manos de cada jugador, el reparto de las cartas comunitarias, etc.

Bueno, pues aún queda la parte más interesante del juego: las apuestas. En cada etapa los jugadores realizan sus apuestas teniendo en cuenta sus cartas (recuerda que no pueden ver las cartas de los demás) y las cartas comunitarias.

Las apuestas en Texas funcionan como en casi todos los juegos de apuestas: un parte apuesta una cantidad y las otras partes tienen la opción de **pasar** (y perder todo lo que se hubiera apostado hasta ese momento), de **igualar** la apuesta, o de **subirla**[1].

Supongamos que en el ejemplo de mano anterior entre Roberto y Carolina, ámbos disponen de 50$ para apostar.

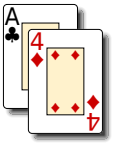
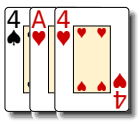
1. **Preflop:** Roberto y Carolina ven sus cartas

Roberto	**Carolina**	**Cartas comunitarias**
		
50$	50$	Bote: 0$

   Comienzan las apuestas, supongamos que empieza siempre Roberto:

   - Roberto: “subo 5$” (apostado: 5$)
   - Carolina: “igualo” (apostado: 5$)

2. **Flop:**

Roberto	**Carolina**	**Cartas comunitarias**
		
45$	45$	Bote: 10$

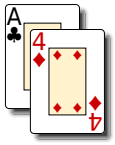
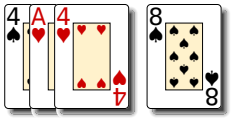
   Apuestas en el *flop*:

[1] Voy a realizar algunas simplificaciones en la estructura de apuestas pero creo que mantengo la esencia del juego

- Roberto: “paso” (apostado: 0$)
- Carolina: “subo 5$” (apostado: 5$)
- Roberto: “subo 5$” (apostado: 10$)
- Carolina: “igualo” (apostado: 10$)

Observa que Roberto ha pasado pero que como no había aún apuestas realizadas en esta etapa es como ceder el turno a Carolina.

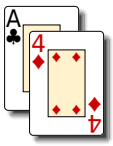
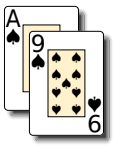
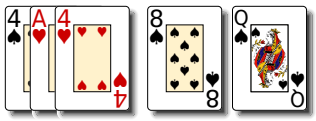
3. **Turn:**

Roberto	**Carolina**	**Cartas comunitarias**
		
35$	35$	Bote: 30$

Apuestas en el *turn*:

- Roberto: “subo 10$” (apostado 10$)
- Carolina: “igualo” (apostado 10$)

4. **River:**

Roberto	**Carolina**	**Cartas comunitarias**
		
25$	25$	Bote: 50$

Apuestas en el *river*:

- Roberto: “subo 5$” (apostado 5$)
- Carolina: “subo 3$” (apostado 8$)
- Roberto: “subo 5$” (apostado 13$)
- Carolina: “igualo” (apostado 13$)

Las apuestas quedan de la siguiente forma:

Roberto	**Carolina**	**Bote**
12$	12$	Bote: 76$

Carolina gana

En la siguiente mano, Carolina jugará con 88$ y Roberto tendrá que hacerlo con 12$.

Ejercicio 8. Tu tarea consistirá en implementar un programa *conversacional* que juegue contra un humano al Texas. Supongamos que el programa se llama `Texas` y que lo ponemos en marcha. Supongamos que el programa hace de Roberto y nosotros ante la consola hacemos de Carolina. La sesión completa con el ejemplo podría ser esta:

```
C:\ Sesion13> java -cp texas.jar Texas
Hola. Tenemos 50 para apostar cada uno.
Preflop: A♠ 9♠
Yo: 5
Tú: _
```

Tu programa ha hecho muchas cosas hasta ese punto en el que está esperando que *Carolina* diga si quiere igualar la apuesta, subirla o pasar. Hasta ese punto, tu programa:

- Mantiene el disponible de cada jugador: el suyo y el de *Carolina*
- Ha repartido dos cartas para él y dos cartas para *Carolina*, las que nos ha enseñado
- Ha apostado 5 (así que su disponible es de 45 ahora)

Carolina puede pasar, igualar o subir. El programa puede usar el siguiente criterio para comunicarse con el usuario:

- `P`: paso
- `I`: igualo
- Número entero: subo

La sesión seguiría así:

```
C:\ Sesion13> java -cp texas.java Texas
Hola. Tenemos 50 para apostar cada uno.
Preflop: A♠ 9♠
Yo: 5
Tú: I
Tengo 40
Tienes 40
Bote 10
Flop: 4♠ A♥ 4♥
Yo: P
Tú: _
```

En este punto Carolina decidía apostar 5:

```
Tú: 5
Yo: 5
Tú: _
```

Y aquí Carolina volvía a igualar:

```
Tú: I
Tengo 35
Tienes 35
Bote 30
Turn: 4♠ A♥ 4♥ 8♠
Yo: 10
Tú: _
```

Carolina vuelve a igualar:

```
Tú: I
Tengo 25
Tienes 25
Bote 50
Turn: 4♠ A♥ 4♥ 8♠ Q♠
Yo: 5
Tú: _
```

Carolina sube 3:

```
Tú: 3
Yo: 5
Tú: _
```

Carolina cierra las apuestas igualando y entonces el programa tiene que terminar diciendo:

```
Tú: I
Mis cartas: A♣ 4♦
Tus cartas: A♠ 9♠
Cartas comunitarias: 4♠ A♥ 4♥ 8♠ Q♠
Tú ganas
Tengo 12
Tienes 88
C:\ Sesion13> _
```

¿Cómo lo ves? ¿Complicadillo? Bueno, vamos a dividir un poco el problema y a organizarlo para poder atacarlo mejor.

Ejercicio 9. Comienza con un paquete, digamos `es.upm.texas.cartas` para organizar las clases que saben de cartas: `Naipe`, `Baraja`, `Par` (las dos cartas de un jugador) y `Mesa` (cartas comunitarias).

Para ello tienes que organizar el directorio y decidir la *responsabilidad* de cada clase. Para echar un mano pongo aquí el posible API de baraja:

```
package es.upm.texas.cartas;

public class Baraja {
  /**
   * Crea una baraja con todas las cartas
   */
  public Baraja() {
```

```
    ...
  }

  /**
   * Saca una carta de la baraja (carta que no volverá a salir)
   */
  public Naipe sacar() {
    ...
  }

  /**
   * Baraja las cartas que quedan en la baraja
   */
  public void barajar() {
    ...
  }
}
```

Ejercicio 10. Vas a necesitar otro paquete con la *responsabilidad* de saber del ranking del Poker. Digamos que se llama `es.upm.texas.ranking`. Quizás con una clase sea suficiente: `Mano` (mano de 5 cartas de poker). Su responsabilidad: representar una mano de 5 cartas de Poker y poder compararla con otra mano de 5 cartas para saber si es mejor peor o igual.

Ejercicio 11. Finalmente, quizás podamos colocar la clase principal (`Texas`) en el paquete `es.upm.texas`. Quizás, al ir implementando el programa surja la necesidad de crear algún paquete extra con una responsabilidad concreta. En clase hablamos de un paquete que "supiera" de las reglas del Texas, por ejemplo.

¿Crees que estás en disposición de implementar tu programa? Para que no se te haga bola te voy a dar algunas sugerencias:

- No hace falta que sea muy listo a la hora de apostar, de hecho podría apostar aleatoriamente `;)`
- Inicialmente el ranking puede ser una función que diga cualquiera dos manos de Poker son iguales, te bastará para empezar a ver cómo se comporta tu programa
- Con estas dos ideas te podrás centrar en la estructura de las apuestas y en la conversación

Ejercicio 12. La última tarea consiste en empaquetar el juego en un `.jar` para que cualquier otro usuario pueda jugar.

Para ayudaros a organizar vuesto código y contruir un fichero `.jar` que sea usable y que contenga el código fuente os sugiero lo siguiente:

- Cread un directorio `Sesion13`
- Cread un directorio `Sesion13/src` en el que poner todo el código fuente organizado por paquetes, por ejemplo, la clase `es.upm.texas.Texas` con el juego en el fichero

`Texas.java` en el directorio `Sesion13/src/es/upm/texas`, la clase `es.upm.texas.cartas.Naipe` con el juego en el fichero `Naipe.java` en el directorio `Sesion13/src/es/upm/texas/cartas`, etc.

- Desde el directorio `Sesion13` compilad el código:

```
$ javac -d lib -cp lib -sourcepath src src/es/upm/texas/Texas.java
```

- Desde el directorio `Sesion13` cread el fichero `texas.jar` con las clases y el código fuente:

```
$ jar cvf texas.jar -C lib . -C . src
added manifest
adding: es/(in = 0) (out= 0)(stored 0%)
adding: es/upm/(in = 0) (out= 0)(stored 0%)
adding: es/upm/texas/(in = 0) (out= 0)(stored 0%)
adding: es/upm/texas/Texas.class(in = 449) (out= 311)(deflated 30%)
adding: es/upm/texas/cartas/(in = 0) (out= 0)(stored 0%)
adding: es/upm/texas/cartas/Naipe.class(in = 204) (out= 170)(deflated 16%)
adding: src/(in = 0) (out= 0)(stored 0%)
adding: src/es/(in = 0) (out= 0)(stored 0%)
adding: src/es/upm/(in = 0) (out= 0)(stored 0%)
adding: src/es/upm/texas/(in = 0) (out= 0)(stored 0%)
adding: src/es/upm/texas/Texas.java(in = 189) (out= 146)(deflated 22%)
adding: src/es/upm/texas/cartas/(in = 0) (out= 0)(stored 0%)
adding: src/es/upm/texas/cartas/Naipe.java(in = 53) (out= 55)(deflated -3%)
```

- Ahora ya puedes entregar el fichero `texas.jar` y cualquier podría usar sus clases o ejecutar el programa principal:

```
$ java -cp texas.jar es.upm.texas.Texas
Hagan juego, damas y caballeros
...
```