

Guillermo Vigueras

guillermo.vigueras@imdea.org

Julio García

juliomanuel.garcia@upm.es

Lars-Åke Fredlund

lfredlund@fi.upm.es

Manuel Carro Liñares

mcarro@fi.upm.es

Marina Álvarez

marina.alvarez@upm.es

Tonghong Li

tonghong@fi.upm.es

Normas.

- ▶ **¡Solo debe entregar una persona por grupo!**
- ▶ Fechas de entrega y nota máxima alcanzable:

Hasta el martes 10 de noviembre, 13:00 (15:00) horas	10
Hasta el miércoles 11 de noviembre, 13:00 (15:00) horas	8
Hasta el jueves 12 de noviembre, 13:00 (15:00) horas	6
Hasta el viernes 13 de noviembre, 13:00 (15:00) horas	4

Después la máxima puntuación será 0.
- ▶ Explicamos la solución después del jueves 22 de octubre.
- ▶ Se comprobará plagio y se actuará sobre los detectados.
- ▶ Usad las horas de tutoría para preguntar sobre programación – son oportunidades excelentes para aprender.

Sistema de Entrega

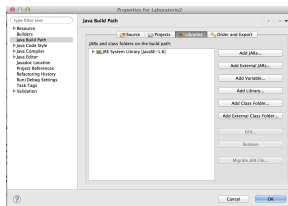
- ▶ Todos los ejercicios de laboratorio se deben entregar a través de la web `http://lml.ls.fi.upm.es/~entrega`.
- ▶ Hoy, el fichero que hay que subir es `Solitario.java`.

Configuración previa al desarrollo del ejercicio.

- ▶ Arrancad Eclipse. Debéis tener un acceso directo.
- ▶ Si trabajáis en portátil, podéis utilizar cualquier versión relativamente reciente de Eclipse. Debería valer cualquier versión entre la versión 3.7 (Indigo) o 4.3 (Kepler). Es suficiente con que instaléis la *Eclipse IDE for Java Developers*.
- ▶ Cambiad a “Java Perspective”.
- ▶ Cread un proyecto Java llamado aed:
 - ▶ Seleccionad separación de directorios de fuentes y binarios.
- ▶ Cread un *package* cartas en el proyecto aed, dentro de src.
- ▶ Aula Virtual → AED → Sesiones de laboratorio → Laboratorio 5 → Laboratorio5.zip; descomprimidlo.
- ▶ Contenido de Laboratorio5.zip:
 - ▶ `OurFIFOImplementation.java`, `Carta.java`, `Tester.java` y `Solitario.java`
 - ▶ `aedlibraries.jar`

Configuración previa al desarrollo del ejercicio.

- ▶ Importad al paquete cartas las fuentes que habéis descargado (OurFIFOImplementation.java, Carta.java, Tester.java y Solitario.java).
- ▶ Añadid al proyecto aed las librerías jar que habéis descargado. Para ello:
- ▶ Project → Properties. Se abrirá una ventana como esta:



- ▶ Java Build Path → Libraries → Add external JARs → Seleccionad el fichero aedlibraries.jar, ... que os habéis descargado
- ▶ Ejecutad Tester. Veréis que imprime un mensaje de error.

Tarea para hoy

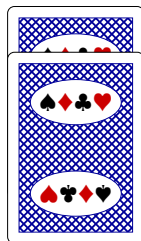
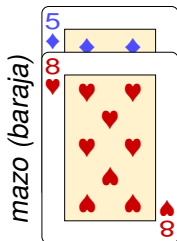
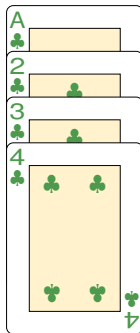
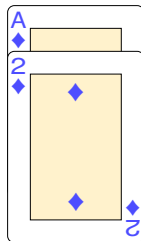
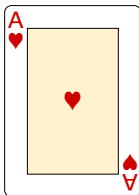
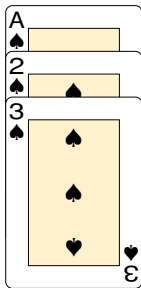
- ▶ Realizar una implementación del método:

```
void play(FIFO<Carta> mazo,  
          LIFO<Carta> [] montones)
```

dentro la clase Solitario.

- ▶ El método recibe una cola (mazo) y un array de pilas (montones). El método debe jugar una variante del juego de cartas “solitario” descrito en detalle en las siguientes páginas.
- ▶ Se usan las pilas (la interfaz LIFO<E>) y las colas (la interfaz FIFO<E>) extensivamente durante el laboratorio. Deberíais repasar ambos “APIs”.

montones de cartas



mazo (descarte)

Las Reglas

1. Saca la primera carta de la baraja (una cola), si la baraja no está vacía. Intenta ponerla en el montón correspondiente, respetando el palo, y en orden *consecutivo* 1 (As), 2, 3, ..., 10.
2. Si no se puede colocar la carta en un montón, muévela al descarte (una cola).
3. Saca la siguiente carta de la baraja (si existe). Muévela al descarte.
4. Repite los pasos (1)-(3) hasta que la baraja este vacía.
5. Intercambia la baraja (ahora vacía) y el descarte.
6. Repite los pasos (1)-(6), o termina.

El juego continua hasta que todas las cartas han sido colocadas en un montón, o, cuando no ha sido posible colocar ninguna carta de la baraja en un montón.

La Implementación

- ▶ Una carta está representado como un objeto de la clase Carta, y tiene un palo (un int entre 0 y 3), y un valor (un int entre 1 (As) y 10). Se considera que 0 representa *picas*, 1 representa *corazones*, 2 representa *diamantes* y 3 representa *tréboles*.
- ▶ Los métodos getPalo() y getValor() de una Carta devuelven el palo y el valor de la carta.
- ▶ El array montones, un parámetro al método play, esta indexado con los palos (0-3). Para tener acceso al montón que corresponde a una carta se escribe: `montones[carta.getPalo()]`.
- ▶ La baraja inicial esta disponible en el parámetro mazo del método play. *Se recomienda crear otro mazo para representar el descarte*. Créalo con: `FIFO<Carta> descarte = new OurFIFOImplementation<Carta>();`

Tareas para hoy (3)

- ▶ El proyecto debe compilar sin errores y debe cumplirse la especificación de los métodos a completar.
- ▶ Debe ejecutar el Tester sin mensajes de error.
- ▶ Nota: una ejecución sin mensajes de error no significa que el método sea correcto (es decir, que funcione bien para cada posible entrada).
- ▶ El Tester de hoy comprueba el estado de los montones (*picas*, *corazones*, *diamantes* y *tréboles*) cuando una llamada al método `play` ha terminado. Si una implementación es demasiado ineficaz (calculado como el número de llamadas a los métodos `enqueue` o `dequeue` de los mazos de entrada y descarte), o sospechosamente eficaz, el Tester imprime un mensaje de advertencia, pero admite la entrega.
- ▶ Todos los ejercicios se comprueban manualmente antes de dar la nota final.