Entrega Individual 1 y 2

Guillermo Román
groman@fi.upm.es
Julio García
juliomanuel.garcia@upm.es
Lars-Åke Fredlund
lfredlund@fi.upm.es
Manuel Carro
mcarro@fi.upm.es
Marina Álvarez
marina.alvarez@upm.es
Raúl Correal
raul.correal@upm.es
Tonghong Li
tonghong@fi.upm.es

Normas

- **La entrega del ejercicio es individual**
- Fechas de entrega y nota máxima alcanzable:

Hasta el Martes 20 de Septienbre, 23:59 horas	10
Hasta el Miércoles 21 de Septienbre, 23:59 horas	8
Hasta el Jueves 22 de Septienbre, 23:59 horas	6
Hasta el Viernes 23 de Septienbre, 23:59 horas	4

- Después la máxima puntuación será 0.

Sistema de Entrega

- Todas los ejercicios de laboratorio se deben entregar a través de la web `http://lml.ls.fi.upm.es/~entrega`
- Los ficheros a subir hoy son `Arrays1.java`, `Arrays2.java`
- Cada uno de los ficheros corresponde con un ejercicio individual distinto
  - `Arrays1` corresponde a la "Entrega Individual 1"
  - `Arrays2` corresponde a la "Entrega Individual 2"
- Se puede entregar únicamente uno de ellos

Configuración previa al desarrollo del ejercicio.

- Arrancad Eclipse. Es suficiente con que tengáis 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* `aed.invididual1` en el proyecto `aed`, dentro de `src`
- Aula Virtual → AED → Sesiones de laboratorio → Entrega Individual 1 y 2 → `EntregaIndividual12.zip`; descomprimidlo
- Contenido de EntregaIndividual12.zip
  - `Tester1.java`, `Tester2.java`, `Arrays1.java`, `Arrays2.java`

Configuración previa al desarrollo del ejercicio.

- Importad al paquete `aed.invididual1` las fuentes que habéis descargado (`Tester1.java`, `Tester2.java`, `Arrays1.java`, `Arrays2.java`).
- Ejecutad `Tester[1-2]`. Veréis que imprimen un mensaje de error

Tarea para hoy: Entrega Individual 1

- Realizar en `Arrays1` una implementación del método:

  `Integer [] compactar (Integer [] array)`
- Recibe un vector de *Integer* y compacta los elementos consecutivos iguales a una única aparición
  - Debe devolver un *nuevo* array cuya longitud sea el número de elementos después de compactarlos (no puede ser mayor)
  - El parámetro `array` no debe ser `null` (lanzar excepción si lo es)
  - Los elementos contenidos en el array no serán `null`

`compactar({2,2,2,2})` $\rightsquigarrow$ `{2}`

`compactar({2,2,2,2,3})` $\rightsquigarrow$ `{2,3}`

`compactar({2,2,1,2})` $\rightsquigarrow$ `{2,1,2}`

`compactar({})` $\rightsquigarrow$ `{}`

`compactar({1,2,3})` $\rightsquigarrow$ `{1,2,3}`

`compactar({1})` $\rightsquigarrow$ `{1}`

`compactar(null)` $\rightsquigarrow$ `IllegalArgumentException`

Comentarios Entrega 1

- No se puede conocer *a priori* el tamaño del array resultado, éste dependerá del contenido del array
- El array DEBE ser nuevo, no se puede modificar el contenido del array de entrada
- Un sólo bucle no es suficiente para llegar a la solución
- Cuidado al comparar Integer, recordad lo que hemos comentado en clase!
- Recordad que `array` no puede ser `null`

Tarea para hoy: Entrega Individual 2

- Realizar en `Arrays2` una implementación del método:
  `boolean sonInversos (Integer [] a1, Integer [] a2)`
- Recibe dos vectores de cadenas `a1` y `a2`
- Debe devolver `true` si los arrays contienen exactamente los mismos elementos en orden inverso y `false` en caso contrario
- Los arrays `a1` y `a2` pueden ser `null`
- Los elementos del array pueden ser `null` (se considera que dos elementos `null` son iguales)

Tarea para hoy: Entrega Individual 2

- Algunos ejemplo de `sonInversos`

```
sonInversos({1,2,3},{3,2,1}) ⇝ true
sonInversos({1,2,3},{3,3,1}) ⇝ false
sonInversos(null,null) ⇝ true
sonInversos({1},null) ⇝ false
sonInversos(null,{1}) ⇝ false
sonInversos({1,2,null},{null,2,1}) ⇝ true
sonInversos({null,null},{null,null}) ⇝ true
sonInversos({1,2,3},{3,2}) ⇝ false
```

Comentarios Entrega 2

- Los arrays pueden ser `null`!
- Los arrays pueden contener elementos `null`. Cuidado con `NullPointerException`
- Cuidado con la comparación de `Integer`

Comentarios generales

- El proyecto debe compilar sin errores y debe cumplirse la especificación de los métodos a completar
- Debe ejecutar `Tester[1-2]` correctamente 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)
- Todos los ejercicios se comprueban manualmente antes de dar la nota final
- El sistema de entrega tiene dos entregas independientes para cada uno de los ejercicios