Laboratorio A.E.D. Viernes 13:00 - 15:00 y 15:00 - 17:00

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.

- **La realización del laboratorio es opcional, por lo tanto no hay fecha de entrega.**
- Explicamos la solución en clase al finalizar los plazos.
- Tras la sesión de laboratorio tenéis disponibles las tutorías.

Configuración previa al desarrollo del ejercicio.

- Arrancad Eclipse. Debéis tener un acceso directo.
- Si trabajáis en portátil, podeis utilizar cualquier version relativamente reciente de Eclipse. Debería valer cualquier versión entre la versión 3.7 (Indigo) o 4.3 (Kepler). Es suficiente con que instaleis 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* `ArrayExamples` en el proyecto `aed`, dentro de `src`.
- Aula Virtual → AED → Sesiones de laboratorio → `Laboratorio1` → `Laboratorio1.zip`; descomprimidlo.
- Contenido de `Laboratorio1.zip`
  - `Tester1.java`, `Tester2.java`, `Tester3.java`, `Example1.java`, `Example2.java`, `Example3.java`

Configuración previa al desarrollo del ejercicio.

- Importad al paquete `ArrayExamples` las fuentes que habéis descargado (`Tester1.java`, `Tester2.java`, `Tester3.java`, `Example1.java`, `Example2.java`, `Example3.java`).
- Ejecutad `Tester[1-3]`. Veréis que imprime un mensaje de error.

Tarea para hoy

- Realizar una implementación del método:
  `int numOcString(String s, String [] v)` dentro la clase `Example1`
- Recibe un vector de cadenas (`v`), y una cadena a buscar en el vector (`s`).
- Debe devolver el número de ocurrencias de la cadena `s` en el vector `v`. Las cadenas pueden ser `null`

```
numOcString("akp",{"akp","akp","mr","akp"}) ⇝ 3
numOcString("kpa",{"akp","akp","mr","akp"}) ⇝ 0
numOcString(null,{"akp",null,"mr",null,null}) ⇝ 3
numOcString("",{"",null,"",null,null}) ⇝ 2
```

Tarea para hoy

- Realizar una implementación del método:
  `boolean compareArrays(String [] v1, String [] v2)`
  dentro la clase `Example2`
- Recibe dos vectores de cadenas `v1` y `v2`.
- Debe devolver `true` si los vectores contienen las mismas cadenas y en el mismo orden. Devolverá `false` en caso contrario. Las cadenas pueden ser `null`

```
compareArrays({"hgb","hgb","ae","ae"},{"hgb","hgb","ae","ae"})
⇝ true
compareArrays({"hgb","hgb","ae","ae"},{"ae","hgb","hgb","ae"})
⇝ false
compareArrays({"hgb",null,"ae","ae"},{"hgb",null,"ae","ae"})
⇝ true
```

Tarea para hoy

- Realizar una implementación del método:
  `String [] commons(String [] v1, String [] v2)` dentro la clase `Example3`
- Recibe dos vectores de cadenas `v1` y `v2`.
- Los vectores de entrada y las cadenas dentro de los vectores pueden ser `null`
- Debe devolver un nuevo vector de cadenas (no modificar argumentos) que contenga los elementos comunes en `v1` y `v2`. El orden de los elementos en el vector devuelto es indiferente
- La cardinalidad de un elemento en el vector devuelto será el mínimo entre el número de veces que ese elemento aparece en `v1` y el número de veces que aparece en `v2`
- Si al menos un argumento es `null` la salida será `null`

Tarea para hoy

- Ejemplos de llamadas al método commons

```
commons({"hgb","hgb","ae"},{"hgb","ae"}) ⇝ {"hgb","ae"}
commons({"hgb","hgb","hgb"},{"hgb","hgb"}) ⇝ {"hgb","hgb"}
commons({"hgb","ae","kpa"},{"kpa","hgb","ae"}) ⇝
{"hgb","ae","kpa"}
commons({null,"ae",null},{"hgb",null}) ⇝ {null}
commons({},{"hgb",null}) ⇝ {}
commons({},{}) ⇝ {}
commons(null,null) ⇝ null
commons({},null) ⇝ null
```