Laboratorio A.E.D. Ejercicio Individual 5

Guillermo Román
guillermo.roman@upm.es

Lars-Åke Fredlund
lfredlund@fi.upm.es

Manuel Carro
mcarro@fi.upm.es

Marina Álvarez
marina.alvarez@upm.es

Julio García
juliomanuel.garcia@upm.es

Tonghong Li
tonghong@fi.upm.es

Sergio Paraiso
sergio.paraiso@upm.es

Normas

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

Hasta el Jueves 7 de Noviembre, 23:59 horas	10
Hasta el Viernes 7 de Noviembre, 23:59 horas	8
Hasta el Sabado 8 de Noviembre, 23:59 horas	6

- Después la máxima puntuación será 0
- Se comprobará plagio y se actuará sobre los detectados

Sistema de Entrega

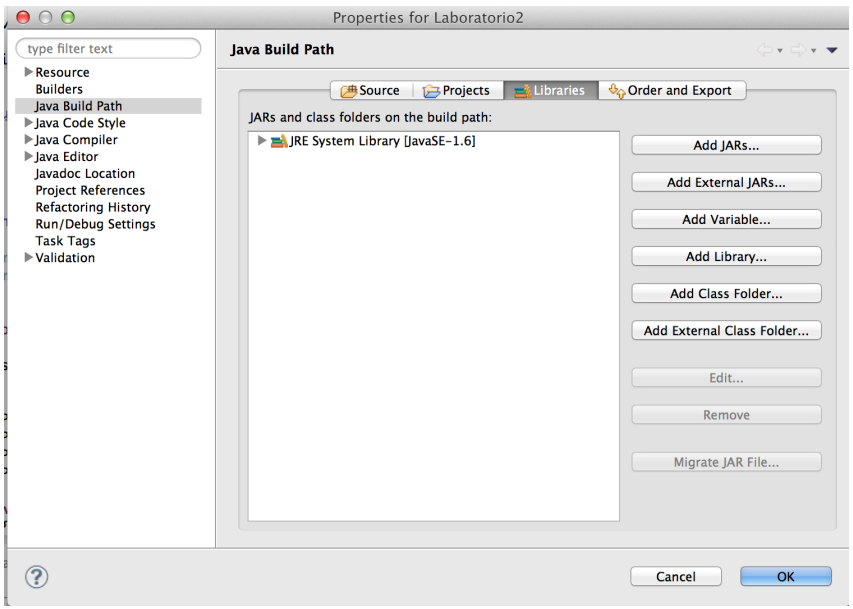
- Todas los ejercicios de laboratorio se deben entregar a través de la web `http://costa.ls.fi.upm.es/entrega`
- Los ficheros a subir son `TempUtils.java`

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.invididual5` en el proyecto `aed`, dentro de `src`
- Aula Virtual → AED → Laboratorios y Entregas Individuales → Individual 5 → `Individual5.zip`; descomprimidlo
- Contenido de Individual5.zip
  - `TesterInd5.java`, `TempData.java`, `TempUtils.java`
- Descargad también el fichero `aedlib.jar`

Configuración previa al desarrollo del ejercicio.

- Importad al paquete `aed.invididual5` los fuentes que habéis descargado (`TesterInd5.java`, `TempData.java`, `TempUtils.java`)
- Añadid al proyecto `aed` la librería `aedlib.jar` que habéis descargado. Para ello:
- Project → Properties. Se abrirá una ventana como esta:



- Java Build Path → Libraries → Add external JARs → Seleccionad el fichero `aedlib.jar` que os habéis descargado
- Ejecutad `TesterInd5`. Veréis que imprimen un mensaje de

Tarea para hoy: Entrega Individual 5

- Terminar la implementación de la clase `TempUtils`, es decir, completar la implementación de los métodos: `maxTemperatures(...)` y `maxTemperatureInRegion(...)`

Notas máximas

- Si los dos métodos se implementan correctamente, la nota máxima es 10. Si se implementan correctamente un método, la máxima nota será 6.

La clase `TempData`

La clase `TempData` contiene datos sobre la temperatura en un lugar español en un hora dada, y tiene 3 métodos "getters":

- `string getLocation()` – devuelve el lugar donde se ha medido la temperatura
- `long getTime()` – devuelve la hora a la que se ha medido la temperatura
- `int getTemperature()` – devuelve la temperatura medida

Por ejemplo, `new TempData("Madrid",300,25)` expresa que la temperatura en Madrid en la hora `300` ha sido de 25 grados.

Implementación

- El método

```
public static Map<String,Integer>
  maxTemperatures(long startTime,long endTime,
                  TempData[] tempData)
```

  tiene los parámetros `startTime` (hora inicial), `endTime` (hora final), y `tempData` (un array con las temperaturas medidas).
- El método debe devolver un map (usad la clase `HashTableMap` para crear el map) que para cada ciudad donde se ha medido la temperatura en el intervalo [startTime...endTime] contiene un `Entry<String,Integer>` con la ciudad como clave y como valor la temperatura **máxima** medida.

Ejemplo

```
maxTemperatures(310,330,
                [TempData("Madrid",300,40),
                 TempData("Madrid",310,20),
                 TempData("Madrid",320,30)])
       ==>  [("Madrid",30)]
            // Notad que el primer dato es fuera del intervalo

maxTemperatures(310,330,
                [TempData("Madrid",310,20),
                 TempData("Malaga",330,35),
                 TempData("Madrid",320,30)])
       ==>  [("Madrid",30),("Malaga",35)]
```

Implementación

- El método

```
public static Map<String,Integer>
  maxTemperaturesInRegion(long startTime,long endTime,
                          String region,
                          TempData[] tempData,
                          Map<String,String> ciudadEnRegion)
```

  recibe una hora inicial – `startTime`, una hora final – `endTime`, una region, un array con datos de temperaturas – `tempData`, y un mapa que asocia un ciudad (clave) a una region (valor).
- El método debería devolver un `Pair<String,Integer>`, con una ciudad donde se ha medido la temperatura máxima en el intervalo [startTime...endTime], **en la región especificado**.
- Si no hay ninguna medida el método debería devolver `null`.
- El parámetro `ciudadEnRegion` asocia a cada ciudad una región. Por ejemplo, "Malaga" (clave) esta asociada a "Andalucia" (valor).

Ejemplo

```
maxTemperatureInRegion(310,330,

                       "Madrid",

                       [TempData("Mostoles",300,40),
                        TempData("Malaga",330,35),
                        TempData("Leganes",310,20),
                        TempData("Getafe",320,30)],

                       [("Mostoles","Madrid"),
                        ("Getafe","Madrid"),
                        ("Leganes","Madrid"),
                        ("Malaga","Andalucia")])

    ==>  [("Getafe",30)]

    // Notad que se pide datos sobre ciudades en la comunidad
    // de Madrid, y en el intervalo [310..330]
```

Comentarios generales

- Debe ejecutar `TesterInd5` 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