

Laboratorio A.E.D. Ejercicio Laboratorio 3

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

Normas.

- ▶ Fechas de entrega y la penalización aplicada a la puntuación obtenida sobre 10:

Hasta el Lunes 16 de octubre, 23:59 horas	0 %
Hasta el Martes 17 de octubre, 23:59 horas	20 %
Hasta el Miércoles 18 de octubre, 23:59 horas	40 %
Hasta el Jueves 19 de octubre, 23:59 horas	60 %
Después la puntuación máxima será 0	
- ▶ Se comprobará plagio y se actuará sobre los detectados
- ▶ Usad las horas de tutoría para preguntar sobre programación – son oportunidades excelentes para aprender

Entrega

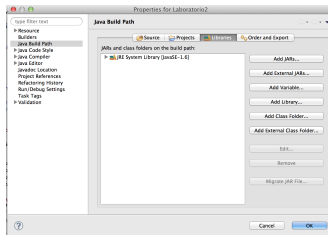
- ▶ Todos los ejercicios de laboratorio se deben entregar a través de la web `http://lm1.ls.fi.upm.es/~entrega`.
- ▶ El fichero que hay que subir es `Secretaria.java`.

Configuración previa

- ▶ Arrancad Eclipse
- ▶ Si trabajáis en portátil, podéis utilizar cualquier versión relativamente reciente de Eclipse. Debería valer cualquier versión a partir de la versión 3.7. 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* `aed.iteradores` en el proyecto `aed`, dentro de `src`
- ▶ Aula Virtual → AED → Laboratorios y Entregas Individuales → Laboratorio 3 → Laboratorio3.zip; descomprimidlo
- ▶ Contenido de Laboratorio3.zip:
 - ▶ `AsignaturaAdmin.java`, `Secretaria.java`,
`TesterLab3.java`, `Pair.java`,
`InvalidMatriculaException.java`,
`InvalidAsignaturaException.java`

Configuración previa al desarrollo del ejercicio.

- ▶ Importad al paquete `aed.iteradores` los fuentes que habéis descargado (`AsignaturaAdmin.java`, `Secretaria.java`, `TesterLab3.java`, `Pair.java`, `InvalidMatriculaException.java`, `InvalidAsignaturaException.java`)
- ▶ Añadid al proyecto `aed` la librería `aedlib.jar` que tenéis en Moodle (en Laboratorios y Entregas Individuales). Para ello:
- ▶ Project → Properties → Java Build Path. Se abrirá una ventana como esta:



- ▶ Usad la opción “Add External JARs...”.
- ▶ Intentad ejecutar `TesterLab3.java`

Documentación de la librería aedlib.jar

- ▶ La documentación de la API de la librería aedlib.jar esta disponible en
<http://lml.ls.fi.upm.es/~entrega/aed/docs/aedlib/>
- ▶ También se puede añadir la documentación de la librería a Eclipse (*no es obligatorio*): en el “Package Explorer”:
“Referenced Libraries” → aedlib.jar y elige la opción
“Properties”. Se abre una ventana donde se puede elegir
“Javadoc Location” y ahí se pone como “javadoc location
path:”
<http://lml.ls.fi.upm.es/~entrega/aed/docs/aedlib/>
y presionar el botón “Apply and Close”

Tarea: Implementar la clase Secretaria usando iteradores

- ▶ El objetivo de este laboratorio es desarrollar un sistema para administrar la matriculación de alumnos en **múltiples asignaturas**, así como consultar informes de notas
- ▶ La implementación de este sistema para múltiples asignaturas se realiza dentro la clase Secretaria
- ▶ Para gestionar los alumnos de **una asignatura**, en la implementación de Secretaria se utilizará la clase AsignaturaAdmin, que fue desarrollada en el Laboratorio 2
- ▶ El comportamiento de los métodos de la clase Secretaria esta documentado en el fichero Secretaria.java

Utilización de la clase AsignaturaAdmin

- ▶ **IMPORTANTE:** Es **obligatorio** usar la implementación de AsignaturaAdmin que esta disponible en el fichero .zip, y no la que hayáis desarrollado en el Laboratorio 2
- ▶ **Hay modificaciones** en AsignaturaAdmin.java con respecto a la del Laboratorio 2:
 - ▶ Los métodos ya no tienen parámetros de tipo PositionList<...>, y no devuelven PositionList<...>
 - ▶ Sólo utilizan objetos de tipo Iterable<...>
 - ▶ P.e., el método matricular ahora es:

```
public Iterable<String> matricular (Iterable<String> matriculas)
```

- ▶ Tiene un método nuevo

```
public Iterable<String> matriculados()
```

que devuelve un **objeto iterable** con los identificadors de los matriculados en la asignatura

La clase Secretaria

```
public class Secretaria {  
    private Iterable<AsignaturaAdmin> asignaturas;  
  
    public Secretaria(Iterable<AsignaturaAdmin> asignaturas) // Constructor  
  
    // Metodo de ejemplo  
    public Iterable<String> matricular(String asignatura,  
                                       Iterable<String> matriculas)  
        throws InvalidAsignaturaException  
  
    // Metodo de ejemplo  
    public Iterable<String> desMatricular(String asignatura,  
                                           Iterable<String> matriculas)  
        throws InvalidAsignaturaException  
  
    public double notaMediaExpediente (String matricula)  
  
    public String mejorNotaMedia()  
  
    public Iterable<Pair<String,Integer>> expediente(String matricula)  
  
    public Iterable<Pair<String,String>> asignaturasNoConflictivas ()  
  
    private boolean compartenAlumnos (AsignaturaAdmin a1, AsignaturaAdmin a2)  
}
```

Reglas de la Implementación

- ▶ Modificad solo `Secretaria.java`. El Tester hace las pruebas con los ficheros originales y si los modificáis vuestras pruebas serán diferentes de las que se realizan al entregar.
- ▶ NO esta permitido usar “for-each”. Esquemas como
`for (String matricula : asignaturaAdmin.matriculados()) {.`
están PROHIBIDOS
- ▶ NO esta permitido hacer *casting* ni usar `instanceof`
- ▶ Está permitido (y recomendado) añadir métodos auxiliares
- ▶ NO está permitido añadir nuevos atributos
- ▶ NO se debe modificar el contenido de las estructuras de datos recibidas como parámetros en los métodos
- ▶ Los métodos que devuelven un objeto `Iterable` (p.e., `matricular`) deben devolver un objeto **nuevo**
 - ▶ Podéis crear el objeto usando cualquier clase que conozcáis que implemente el interfaz `Iterable`

Consejos:

- ▶ Antes de empezar, revisad:
 - ▶ La diferencia entre un Iterable y un Iterator. ¿Cómo se obtiene un Iterator de un Iterable?
 - ▶ El comportamiento de los métodos `hasNext()` y `next()` de iteradores.
- ▶ Tenéis como ejemplo de uso de iteradores los métodos `matricular` y `desmatricular`
- ▶ Implementad y usad en la implementación de `asignaturasNoConflictivas` el método privado

```
private boolean compartenAlumnos(AsignaturaAdmin a1,  
                                   AsignaturaAdmin a2)
```

que debe devolver `true` si las asignaturas `a1` y `a2` tiene alumnos en común

Ejemplos

Asumimos que se ha construido un objeto ‘‘secretaria’’ que contiene los siguientes datos:

Asignatura PPS:

matricula: 11 22 33

nota: 2 4 7

Asignatura AED:

matricula: 22

nota: 8

Programacion I:

matricula: 55

nota: 5

```
mejorNotaMedia()          --> "AED" // la nota media de AED >
                             // la nota media de PPS y
                             // la nota media de Prog. I
```

```
notaMediaExpediente("22")  --> (4+8)/2 = 6.0
                             // La nota media de "22" en las
                             // asignaturas en las que aparece
```

```
secretaria.matricular("AED",["222"]) --> devuelve ["222"] y anade "222" a "AED"
```

```
secretaria.desmatricular("AED",["222"]) --> devuelve ["222"] y
                                             elimina "222" de "AED"
```

Ejemplos

Asumimos que se ha construido un objeto ‘‘secretaria’’ que contiene los siguientes datos:

Asignatura PPS:

matricula: 11 22 33

nota: 2 4 7

Asignatura AED:

matricula: 22

nota: 8

Programacion I:

matricula: 55

nota: 5

```
expediente("22")          --> [Pair("AED",8),Pair("PPS",4)]  
                           // Las notas de "22"  
                           // en las asignaturas
```

```
asignaturasNoConflictivas() --> [Pair("AED","Programacion_I"),  
                                 Pair("PPS","Programacion_I")]  
                           // Devuelve una lista de pares de asignaturas  
                           // que no tienen ningun alumno en comun
```

Notas

- ▶ El proyecto debe compilar sin errores y debe cumplirse la especificación de los métodos a completar
- ▶ Debe ejecutar `TesterLab3.java` correctamente y 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