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

Guillermo Vigueras

guillermo.vigueras@imdea.org
Julio Garc´ıa

juliomanuel.garcia@upm.es
Lars-˚Ake Fredlund

lfredlund@fi.upm.es
Manuel Carro Lin˜ares

mcarro@fi.upm.es
Marina A´lvarez

marina.alvarez@upm.es
Pablo Nogueira

pnogueira@fi.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 13 de octubre 13:00 (15:00) horas 10
- Hasta el miércoles 14 de octubre, 13:00 (15:00) horas 8
- Hasta el jueves 15 de octubre, 13:00 (15:00) horas 6
- Hasta el viernes 16 de octubre, 13:00 (15:00) horas 4

Después la máxima puntuación será 0.

- Explicamos la solución tras el viernes 16 de octubre.
- Se comprobará plagio y se actuará sobre los detectados.
- Usad las tutorías.
- 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.

- Fichero(s) a subir en la sesión de hoy: TimeHMS.java, TimeSec.java y IntervalTime.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. Deber´ıa 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 IntervalTime en el proyecto aed, dentro de src.
- Aula Virtual → AED → Sesiones de laboratorio → Laboratorio3 → Laboratorio3.zip; descomprimidlo.
- Contenido de Laboratorio3.zip
  - Tester.java, Time.java, TimeHMS.java, TimeSec.java y IntervalTime.java

Configuración previa al desarrollo del ejercicio.

- Importad al paquete IntervalTime las fuentes que habéis descargado (Tester.java, Time.java, TimeHMS.java, TimeSec.java y IntervalTime).
- Ejecutad Tester. Veréis que imprime un mensaje de error.

Tarea para hoy

En la clases Time, TimeSec y TimeHMS

- En el laboratorio de hoy vamos a utilizar los elementos relacionados con el tiempo que construimos en el Laboratorio 2 (Time, TimeSec y TimeHMS) y vamos a construir intervalos de tiempo (clase IntervalTime).

- Hemos incorporado a nuestra representación del tiempo Time las relaciones de orden less y lessEq

```
public interface Time{
    public int getHour(); // Devuelve las horas (sin signo)
    public int getMins(); // Devuelve los minutos (sin signo)
    public int getSecs(); // Devuelve los segundos (sin signo)
    public int getSign(); // Devuelve el signo
    public int timeInSeconds(); // Devuelve el tiempo en // segundos (con signo)
    public Time suma(Time t); // Suma de horas
    public Time resta(Time t); // Resta de horas
    public boolean less (Time t); // NUEVA Relación <
    public boolean lessEq (Time t); // NUEVA Relación <=
}
```

- Ahora tenéis que implementar estas operaciones en las clases TimeHMS y TimeSec.

```
public boolean less(Time t) {
  // TODO Auto-generated method stub
  return false;
}

public boolean lessEq(Time t) {
  // TODO Auto-generated method stub
  return false;
}
```

Tarea para hoy

En la clase IntervalTime (archivo IntervalTime.java)

- Tenemos que implementar el método “equals” IntervalTime.

```
// ¿Está t contenido en el intervalo "this"?
  public boolean contains(Time t) {
    // COMPLETAR
  }
// Relación de igualdad entre intervalos
  public boolean equals(IntervalTime that) {
   // COMPLETAR
  }
// ¿Está el intervalo "this" incluido en intervalo "that", "this" C
"that"
  public boolean isIncluded (IntervalTime that)  {

    // COMPLETAR
  }
```

- Además, y utilizando los métodos anteriores, tenemos que resolver 3 problemas de recorrido sobre arrays de intervalos de tiempo.

```
  // Determina si todos los intervalos del array están incluidos en
  // orden creciente (cada uno en el siguiente).
  // allIncluded (<i1,i2,...,ik-1, ik>) = i1 C i2 ... C ik-1 C ik

     public static boolean allIncluded (IntervalTime[] arr) {
        // COMPLETAR
     }

  // Calcula la posición de "this" en un array "arr" que está ordenado
  // en orden creciente
  //
  // Devuelve la posición de "this" si está en "arr" y -1  en otro caso.
  // Realizar la búsqueda binaria.

     public int posInArray (IntervalTime arr[]) {
        // COMPLETAR
     }

  // Está "this" en el array "arr"
     public int posInArray (IntervalTime arr[]) {
        // COMPLETAR
     }
```

Tarea para hoy

- El proyecto debe compilar sin errores y debe cumplirse la especificación de los métodos a completar.
- Debe ejecutar Tester 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.