Programación I
Grado en Informática
Curso 2018 / 2019

Práctica obligatoria – 1ª Convocatoria Ordinaria

Tabla de contenidos

1. Descripción de la aplicación

Blablakid es una aplicación que sirve para organizar el transporte de los niños de una o varias familias a las distintas actividades semanales, como ir y volver del colegio o asistir a las actividades extraescolares durante la semana laboral. El objetivo es organizar los viajes que deben hacer los padres para que los niños puedan asistir a todas sus actividades.

2. Reglas generales de funcionamiento

Utilizando una interfaz de tipo texto, se propone implementar una versión de la aplicación *Blablakid* en la que pueden participar un número máximo prefijado de `kids`, un número máximo prefijado de `parents`, que es igual al doble del número de niños, un número máximo prefijado de `activities`, que son tres por niño, y un número también acotado máximo de `rides` por día que puede ser distinto para cada padre.

La aplicación recibe como parámetro el número máximo de niños y permitirá añadir a los niños, añadir a los padres, añadir las actividades y añadir los trayectos. La aplicación deberá comprobar que todos los desplazamientos a las actividades están cubiertos por un trayecto y que todos los niños pueden asistir a todas sus actividades. La aplicación llega a su fin cuando el usuario solicita salir a través de la interfaz de texto.

3. Diseño de la aplicación

El diseño de la aplicación debe ser tal que permita generar situación de objetos en memoria que recoge la Figura 1.

La aplicación mantiene una lista de niños y una lista de padres. Cada niño puede tener actividades asignadas. Las actividades tienen un nombre, se celebran un día de la semana, tienen hora de comienzo y fin, y tienen un trayecto asignado antes y otro después. Los trayectos deberán estar cubiertos por algún padre. Cada padre mantiene la información sobre los trayectos que debe llevar a cabo cada día laborable de la semana. También mantiene referencia a cada uno de sus hijos.

En la versión inicial de la aplicación, los padres no saben a qué niño o niños llevan en cada trayecto y los niños no saben qué padre está encargado de sus trayectos. Esta funcionalidad se añadirá en la siguiente versión de la aplicación (segunda convocatoria).

4. Funcionamiento de la aplicación

4.1. Configuración inicial

La aplicación recibe como parámetro de entrada el número máximo de niños (número entero). El número máximo de padres se calculará como el doble del número máximo de niños, y el número máximo de trayectos que un padre puede llevar a cabo por día será un dato de entrada que se lea de teclado al introducir a cada uno de los padres, ya que no tiene que ser el mismo para todos los padres. En caso de que el usuario no respete la sintaxis de la aplicación, ésta deberá emitir un mensaje de error lo más detallado posible respecto al error cometido y mostrará el siguiente mensaje antes de terminar la ejecución:

```
Sintaxis del programa: Blablakid maxKids
```

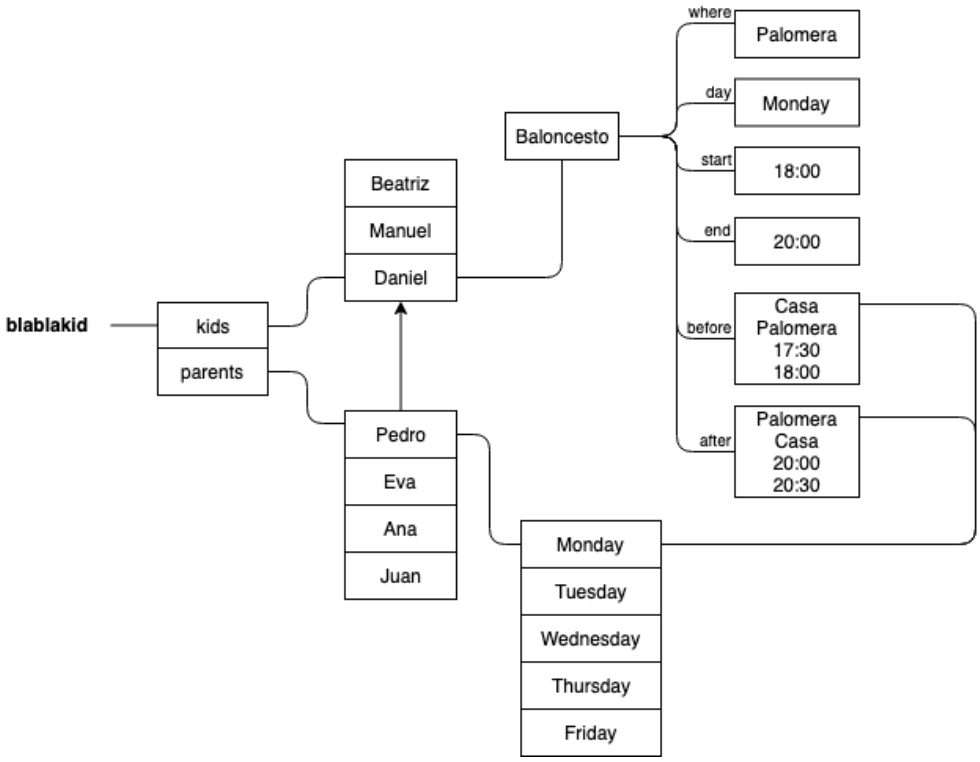

Figura 1. Ejemplo de estructura de objetos en memoria

4.2. Seguimiento del funcionamiento de la aplicación

Una vez arrancada la aplicación, se le muestra al usuario la situación inicial en la que tan solo se dispone del menú inicial. La salida por pantalla sería la siguiente:

```
---------
Blablakid
---------
Select an option:
 1 – Add kid
 2 – Remove kid
 3 – Add parent
 4 – Remove parent
 5 – Add activity
 6 – Remove activity
 7 – Add ride
 8 – Remove ride
 9 – Show summary
 10 – Check status
 0 – Exit
```

A partir de ese momento, el usuario podrá introducir una de las opciones disponibles. La aplicación deberá comprobar que el usuario introduce una opción válida, emitiendo un mensaje de error en caso contrario.

4.3. Log de la aplicación

Durante cada ejecución de la aplicación se crea un fichero de registro con la información generada. El contenido de este fichero deberá ser tal que, con la menor cantidad de información posible, nos permita reproducir fielmente el desarrollo de una sesión de ejecución de la aplicación, incluidos los posibles errores cometidos por el usuario.

Para generar los ficheros de registro se deberá utilizar obligatoriamente `log4j`. No se admitirá ninguna otra solución. El uso directo de ficheros de texto queda excluido.

4.4. Funcionamiento de la aplicación

A continuación se muestra un ejemplo de ejecución de la aplicación. La información del estado de la aplicación se muestra en todo momento solamente por motivos de claridad. En la versión final de la aplicación dicha información solamente aparecería cuando el usuario la solicitara a través de la correspondiente opción de menú. Para la entrega de la práctica se deberá mantener esa información.

Para el desarrollo del ejemplo se contemplará la siguiente situación:

- **Kids:**
  - **Daniel**
    - **Activities:**
      - **Baloncesto (Palomera)**
        - **Lunes – 18:00/20:00**
        - Miércoles – 17:00/19:00
  - **Beatriz**
    - **Activities:**
      - **Música (Conservatorio)**
        - **Martes – 17:30/19:00**
        - Jueves – 17:00/18:30
  - **Manuel**
    - **Activities**
      - **Francés (EOI)**
        - **Lunes – 17:00/19:00**
        - Miércoles – 17:00/19:00
- **Parents:**
  - **Pedro (padre de Daniel)**
    - **Lunes – Manuel – Casa – EOI – 16:30/17:00**
    - **Lunes – Daniel – Casa – Palomera – 17:30/18:00**
  - **Eva (madre de Daniel)**
    - Miércoles – Daniel – Casa – Palomera – 16:30/17:00
    - Miércoles – Daniel – Palomera – Casa – 19:00/19:30
  - **Ana (madre de Beatriz y Manuel)**
    - **Lunes – Manuel – EOI – Casa – 19:00/19:30**
    - **Lunes – Daniel – Palomera – Casa – 20:00/20:30**
    - **Martes – Beatriz – Casa – Conservatorio – 17:00/17:30**
    - Jueves – Beatriz – Conservatorio – Casa – 18:30/19:00
  - **Juan (padre de Beatriz y Manuel)**
    - **Martes – Beatriz – Conservatorio – Casa – 19:00/19:30**
    - Miércoles – Manuel – Casa – EOI – 16:30/17:00
    - Miércoles – Manuel – EOI – Casa – 19:00/19:30
    - Jueves – Beatriz – Casa – Conservatorio – 16:30/17:00

La ejecución de la aplicación ha sido la siguiente:

```
java –cp classes es.unileon.prg1.blablakid.MainBlablakid 3
```

```
//////////////////////

KIDS:

****** Daniel ******
Baloncesto (Palomera – MONDAY)18:00 > 20:00
Casa > Palomera : 17:30/18:00
Palomera > Casa : 20:00/20:30
****** Beatriz ******
Musica (Conservatorio – TUESDAY)17:30 > 19:00
Casa > Conservatorio : 17:00/17:30
Conservatorio > Casa : 19:00/19:30
****** Manuel ******
Frances (EOI – MONDAY)17:00 > 19:00
Casa > EOI : 16:30/17:00
EOI > Casa : 19:00/19:30

PARENTS:

###### Pedro ######
Kids:
Daniel
Rides:
MONDAY
Casa > Palomera : 17:30/18:00
Casa > EOI : 16:30/17:00
###### Eva ######
Kids:
Daniel
Rides:
###### Ana ######
Kids:
Beatriz Manuel
Rides:
MONDAY
Palomera > Casa : 20:00/20:30
EOI > Casa : 19:00/19:30
TUESDAY
Casa > Conservatorio : 17:00/17:30
###### Juan ######
Kids:
Beatriz Manuel
Rides:
TUESDAY
Conservatorio > Casa : 19:00/19:30
//////////////////////
---------
Blablakid
---------
Select an option:
 1 – Add kid
 2 – Remove kid
 3 – Add parent
 4 – Remove parent
 5 – Add activity
 6 – Remove activity
 7 – Add ride
 8 – Remove ride
 9 – Show summary
 10 – Check status
 0 – Exit
```

4.5. Ejemplos de ejecución

A continuación se presentan ejemplos de ejecución de cada una de las opciones del menú partiendo de la siguiente ejecución de la aplicación:

```
java –cp classes es.unileon.prg1.blablakid.MainBlablakid 3
```

Opción 1 – Añadir un niño

El ejemplo muestra el comportamiento de la aplicación al añadir a Daniel como niño. Una vez añadido, si se intenta añadir de nuevo, la aplicación deberá mostrar un mensaje de error. El nombre del niño actúa como identificador.

```
//////////////////////

KIDS:


PARENTS:

//////////////////////
_________
Blablakid
_________
Select an option:
 1 – Add kid
 2 – Remove kid
 3 – Add parent
 4 – Remove parent
 5 – Add activity
 6 – Remove activity
 7 – Add ride
 8 – Remove ride
 9 – Show summary
 10 – Check status
 0 – Exit
1
Name of the kid to add:
Daniel
//////////////////////

KIDS:

****** Daniel ******

PARENTS:

//////////////////////
_________
Blablakid
_________
Select an option:
 1 – Add kid
 2 – Remove kid
 3 – Add parent
 4 – Remove parent
 5 – Add activity
 6 – Remove activity
 7 – Add ride
```

```
 8 – Remove ride
 9 – Show summary
 10 – Check status
 0 – Exit
```

Opción 2 – Eliminar un niño

El ejemplo muestra el comportamiento de la aplicación al eliminar a Daniel como niño. Una vez eliminado, si se intenta eliminar de nuevo, la aplicación deberá mostrar un mensaje de error.

```
//////////////////////

KIDS:

****** Daniel ******
****** Manuel ******
****** Beatriz ******

PARENTS:

//////////////////////
---------
Blablakid
---------
Select an option:
 1 – Add kid
 2 – Remove kid
 3 – Add parent
 4 – Remove parent
 5 – Add activity
 6 – Remove activity
 7 – Add ride
 8 – Remove ride
 9 – Show summary
 10 – Check status
 0 – Exit
2
Name of the kid to remove:
Daniel
//////////////////////

KIDS:

****** Manuel ******
****** Beatriz ******

PARENTS:

//////////////////////
---------
Blablakid
---------
Select an option:
 1 – Add kid
 2 – Remove kid
 3 – Add parent
 4 – Remove parent
 5 – Add activity
 6 – Remove activity
 7 – Add ride
 8 – Remove ride
```

```
 9 - Show summary
 10 - Check status
 0 - Exit
2
Name of the kid to remove:
Daniel
Remove kid error: Kid Daniel not found
//////////////////////

KIDS:

****** Manuel ******
****** Beatriz ******

PARENTS:

//////////////////////
---------
Blablakid
---------
Select an option:
 1 - Add kid
 2 - Remove kid
 3 - Add parent
 4 - Remove parent
 5 - Add activity
 6 - Remove activity
 7 - Add ride
 8 - Remove ride
 9 - Show summary
 10 - Check status
 0 - Exit
```

Opción 3 – Añadir un padre

El ejemplo muestra el comportamiento de la aplicación al añadir a un padre.

```
//////////////////////

KIDS:

****** Daniel ******

PARENTS:

//////////////////////
---------
Blablakid
---------
Select an option:
 1 - Add kid
 2 - Remove kid
 3 - Add parent
 4 - Remove parent
 5 - Add activity
 6 - Remove activity
 7 - Add ride
 8 - Remove ride
 9 - Show summary
 10 - Check status
 0 - Exit
3
```

```
Name of the parent to add:
Pedro
How many kids does Pedro have?
1
How many rides can Pedro make per day?
2
Who is Pedro's kid number 01?
Daniel
//////////////////////

KIDS:

****** Daniel ******

PARENTS:

###### Pedro ######
Kids:
Daniel
Rides:
//////////////////////
---------
Blablakid
---------
Select an option:
 1 – Add kid
 2 – Remove kid
 3 – Add parent
 4 – Remove parent
 5 – Add activity
 6 – Remove activity
 7 – Add ride
 8 – Remove ride
 9 – Show summary
 10 – Check status
 0 – Exit
```

Una vez añadido un padre, si se intenta añadir otro padre con el mismo nombre, la aplicación deberá mostrar un mensaje de error. El nombre del padre actúa como identificador.

Opción 4 – Eliminar un padre

El ejemplo muestra el comportamiento de la aplicación al eliminar un padre.

```
//////////////////////

KIDS:

****** Daniel ******

PARENTS:

###### Pedro ######
Kids:
Daniel
Rides:
//////////////////////
---------
Blablakid
```

```
---------
Select an option:
 1 - Add kid
 2 - Remove kid
 3 - Add parent
 4 - Remove parent
 5 - Add activity
 6 - Remove activity
 7 - Add ride
 8 - Remove ride
 9 - Show summary
 10 - Check status
 0 - Exit
4
Name of the parent to remove:
Pedro
//////////////////////

KIDS:

****** Daniel ******

PARENTS:

//////////////////////
---------
Blablakid
---------
Select an option:
 1 - Add kid
 2 - Remove kid
 3 - Add parent
 4 - Remove parent
 5 - Add activity
 6 - Remove activity
 7 - Add ride
 8 - Remove ride
 9 - Show summary
 10 - Check status
 0 - Exit
```

Si se intenta eliminar un padre que no existe la aplicación deberá mostrar un mensaje de error.

Opción 5 – Añadir actividad

```
//////////////////////

KIDS:


PARENTS:

//////////////////////
---------
Blablakid
---------
Select an option:
 1 - Add kid
 2 - Remove kid
 3 - Add parent
 4 - Remove parent
```

```
 5 - Add activity
 6 - Remove activity
 7 - Add ride
 8 - Remove ride
 9 - Show summary
 10 - Check status
 0 - Exit
1
Name of the kid to add:
Daniel
//////////////////////

KIDS:

****** Daniel ******

PARENTS:

//////////////////////
---------
Blablakid
---------
Select an option:
 1 - Add kid
 2 - Remove kid
 3 - Add parent
 4 - Remove parent
 5 - Add activity
 6 - Remove activity
 7 - Add ride
 8 - Remove ride
 9 - Show summary
 10 - Check status
 0 - Exit
5
Name of the activity:
Baloncesto
Where does the activity Baloncesto takes place?
Palomera
Day of the week for the activity:
Insert the number of the day of the week:
0 - Monday / 1- Tuesday / 2 - Wednesday / 3 - Thursday / 4 - Friday
0
Name of the kid taking the activity:
Daniel
When does the activity start?
Insert hour:
18
Insert minute:
00
When does the activity end?
Insert hour:
20
Insert minute:
00
//////////////////////

KIDS:

****** Daniel ******
Baloncesto (Palomera - MONDAY): 18:00 > 20:00
No ride before Baloncesto assigned
```

```
No ride after Baloncesto assigned

PARENTS:

//////////////////////
---------
Blablakid
---------
Select an option:
 1 - Add kid
 2 - Remove kid
 3 - Add parent
 4 - Remove parent
 5 - Add activity
 6 - Remove activity
 7 - Add ride
 8 - Remove ride
 9 - Show summary
 10 - Check status
 0 - Exit
```

Opción 6 – Eliminar actividad

```
//////////////////////

KIDS:

****** Daniel ******
Baloncesto (Palomera - MONDAY): 18:00 > 20:00
No ride before Baloncesto assigned
No ride after Baloncesto assigned

PARENTS:

//////////////////////
---------
Blablakid
---------
Select an option:
 1 - Add kid
 2 - Remove kid
 3 - Add parent
 4 - Remove parent
 5 - Add activity
 6 - Remove activity
 7 - Add ride
 8 - Remove ride
 9 - Show summary
 10 - Check status
 0 - Exit
6
Name of the kid taking the activity to remove:
Daniel
Name of the activity to remove:
Baloncesto
Insert the number of the day of the week:
0 - Monday / 1- Tuesday / 2 - Wednesday / 3 - Thursday / 4 - Friday
0
//////////////////////

KIDS:
```

```
****** Daniel ******

PARENTS:

//////////////////////
---------
Blablakid
---------
Select an option:
 1 - Add kid
 2 - Remove kid
 3 - Add parent
 4 - Remove parent
 5 - Add activity
 6 - Remove activity
 7 - Add ride
 8 - Remove ride
 9 - Show summary
 10 - Check status
 0 - Exit
```

Opción 7 – Añadir trayecto

```
//////////////////////

KIDS:

****** Daniel ******
Baloncesto (Palomera - MONDAY): 18:00 > 20:00
No ride before Baloncesto assigned
No ride after Baloncesto assigned

PARENTS:

###### Pedro ######
Kids:
Daniel
Rides:
//////////////////////
---------
Blablakid
---------
Select an option:
 1 - Add kid
 2 - Remove kid
 3 - Add parent
 4 - Remove parent
 5 - Add activity
 6 - Remove activity
 7 - Add ride
 8 - Remove ride
 9 - Show summary
 10 - Check status
 0 - Exit
7
Name of the parent in charge of the ride:
Pedro
Name of the activity of the ride:
Baloncesto
Name of the kid taking the activity:
```

```
Daniel
Where does the ride start?
Casa
Where does the ride end?
Palomera
When does the ride start?
Insert hour:
17
Insert minute:
30
When does the ride end?
Insert hour:
18
Insert minute:
00
Day of the week for the ride:
Insert the number of the day of the week:
0 – Monday / 1– Tuesday / 2 – Wednesday / 3 – Thursday / 4 – Friday
0
//////////////////////

KIDS:

****** Daniel ******
Baloncesto (Palomera – MONDAY): 18:00 > 20:00
Casa > Palomera : 17:30/18:00
No ride after Baloncesto assigned

PARENTS:

###### Pedro ######
Kids:
Daniel
Rides:
MONDAY
Casa > Palomera : 17:30/18:00
//////////////////////
---------
Blablakid
---------
Select an option:
 1 – Add kid
 2 – Remove kid
 3 – Add parent
 4 – Remove parent
 5 – Add activity
 6 – Remove activity
 7 – Add ride
 8 – Remove ride
 9 – Show summary
 10 – Check status
 0 – Exit
```

Opción 8 – Eliminar trayecto

```
//////////////////////

KIDS:

****** Daniel ******
Baloncesto (Palomera – MONDAY): 18:00 > 20:00
```

```
Casa > Palomera : 17:30/18:00
No ride after Baloncesto assigned

PARENTS:

###### Pedro ######
Kids:
Daniel
Rides:
MONDAY
Casa > Palomera : 17:30/18:00
//////////////////////
---------
Blablakid
---------
Select an option:
 1 - Add kid
 2 - Remove kid
 3 - Add parent
 4 - Remove parent
 5 - Add activity
 6 - Remove activity
 7 - Add ride
 8 - Remove ride
 9 - Show summary
 10 - Check status
 0 - Exit
8
Name of the parent in charge of the ride:
Pedro
Day of the week for the ride:
Insert the number of the day of the week:
0 - Monday / 1- Tuesday / 2 - Wednesday / 3 - Thursday / 4 - Friday
0
Where does the ride start?
Casa
Where does the ride end?
Palomera

//////////////////////

KIDS:

****** Daniel ******
Baloncesto (Palomera - MONDAY): 18:00 > 20:00
No ride before Baloncesto assigned
No ride after Baloncesto assigned

PARENTS:

###### Pedro ######
Kids:
Daniel
Rides:
//////////////////////
---------
Blablakid
---------
Select an option:
 1 - Add kid
 2 - Remove kid
 3 - Add parent
```

```
4 – Remove parent
5 – Add activity
6 – Remove activity
7 – Add ride
8 – Remove ride
9 – Show summary
10 – Check status
0 – Exit
```

Opción 9 – Mostrar el resumen

La opción de mostrar el resumen muestra por pantalla la información del estado de la aplicación en todo momento y que coincide con lo que estamos mostrando cada vez que mostramos el menú principal.

Opción 10 – Comprobar el estado

Esta opción deberá mostrar información sobre los trayectos que faltan por cubrir.

Opción 0 – Salir y opciones erróneas de menú

Si el usuario introduce una opción de menú incorrecta, la aplicación deberá mostrar un mensaje de error y volverá a mostrar el menú.

Cuando el usuario seleccione la opción 0, finalizará la ejecución de la aplicación emitiendo un mensaje de despedida.

4.6. Casos de prueba

Se deberán llevar a cabo tests para comprobar el correcto funcionamiento de todo el código desarrollado para cada uno de los casos mostrados en la sección anterior. Además, se deberá contemplar, al menos, como situación inicial para los casos de prueba la señalada en negrita en el apartado 4.4, y se deberán comprobar las siguientes situaciones para todos los casos en los que exista una matriz de objetos:

1. Introducir un elemento en la matriz cuando está llena
2. Introducir un elemento repetido
3. Eliminar un elemento que no existe
4. Eliminar un elemento en posición intermedia eliminando el hueco que deja
5. Buscar un elemento que existe
6. Buscar un elemento que no existe
7. Generar las excepciones en todos los casos posibles

5. Ejecución de la aplicación

La ejecución del programa debe hacerse *obligatoriamente* de la siguiente forma:

```
java –cp classes es.unileon.prg1.blablakid.MainBlablakid maxKids
```

donde:

- `maxKids` es el número máximo de niños

6. Normas de entrega

- La fecha límite de entrega será el día anterior al examen de la práctica, al mediodía (12:00 horas del día 30 de mayo de 2019).
- La práctica se encontrará en el correspondiente repositorio de GitHub y contendrá la siguiente estructura de subdirectorios:
  - `src`: ficheros con el código fuente (ficheros `.java`)
  - `test`: ficheros con el código fuente de los tests de `JUnit`
  - `classes`: ficheros compilados (ficheros `.class`) – **No incluido en GitHub (generado por el `build.xml`)**
  - `lib`: librerías utilizadas
  - `doc`: documentación generada por `javadoc` (ficheros `.html`) – **No incluido en GitHub (generado por el `build.xml`)**
  - `etc`: `build.xml` y diagrama de clases UML (archivo generado con el dia http://projects.gnome.org/dia/ o con http://draw.io **en formato `.jpg`**)
  - `log`: Ficheros de registro – **No incluido en GitHub (generado por el `build.xml`)**
- El código fuente de todos los ficheros de la práctica deberá incluir los nombres de los autores, utilizando la etiqueta `@author` de `javadoc`.
- El código estará incluido en el paquete `es.unileon.prg1.blablakid`
- Es *condición imprescindible* para aprobar la práctica que los ficheros `.java` entregados compilen correctamente, es decir, sin generar errores.
- Es *condición imprescindible* para aprobar la práctica que los ficheros `.class` generados a partir de los fuentes entregados ejecuten correctamente, es decir, no den errores de ejecución y ofrezcan la funcionalidad requerida.
- Es *condición imprescindible* para aprobar la práctica la entrega de la documentación correctamente generada, es decir, habiéndola generado haciendo uso de las opciones necesarias para que se genere documentación de todas las clases, atributos y métodos.
- Es *condición imprescindible* para aprobar la práctica que se incluya el correspondiente fichero `build.xml` con los target necesarios para compilar, generar la documentación, probar (pasar los tests) y ejecutar la práctica. Cada uno de los targets deberá establecer las debidas dependencias respecto de los demás (incluido el target dedicado a limpiar).

- Es *condición imprescindible* para aprobar la práctica que genere ficheros de registro de cada ejecución que permitan reproducir cada una de las ejecuciones.
- Es *condición imprescindible* para aprobar la práctica que se incluyan tests de JUnit para cada una de las clases desarrolladas que comprueben la funcionalidad de cada clase desarrollada. **La cobertura del código debe ser del 100%. No se llevarán a cabo tests de la clase Teclado, de la clase que contiene el main, y de la clase encargada de la interfaz de usuario.**
- Es *condición imprescindible* realizar *commits* a medida que la práctica evoluciona. No se considerarán las prácticas con actividad en el repositorio solo en los últimos días previos a la entrega.