

# 21779. Algorísmia i Estructures de Dades II

## Curs 2023-24

### ZenWord: Les funcions auxiliars

25 d'abril de 2024

Durant aquesta sessió de pràctiques ens encarregarem d'implementar les funcions auxiliars necessàries per a la gestió de la nostra aplicació.

## 1 Mostrar una paraula amagada

En aquest apartat s'ha d'implementar una funció, definida com:

```
1 private void mostraParaula(String s, int posicio)
```

On *s* és un *String* que conté la paraula que es vol mostrar i *posicio* indica a quina de les línies corresponents a les paraules amagades es vol mostrar aquest *String*.

A la Figura 1 es mostren els resultats d'executar les crides:

```
1 mostraParaula("neró", 4);
```

i

```
1 mostraParaula("noi", 2);
```

## 2 Mostrar la primera lletra

En aquest apartat s'ha d'implementar una funció, definida com:

```
1 private void mostraPrimeraLletra(String s, int posicio)
```

On *s* és un *String* que conté la paraula de la què es vol mostrar la primera lletra (en minúscula) i *posicio* indica a quina de les línies corresponents a les paraules amagades es vol mostrar aquesta lletra. Aquesta funció es cridarà a



Figura 1: Mostrar paraules amagades.

partir del botó d'ajuda (si es compleixen les restriccions per a poder demanar una ajuda).

A la Figura 2 es mostren els resultats d'executar les crides:

```
1  mostraPrimeraLletra("alè", 2);
i
1  mostraPrimeraLletra("foc", 3);
```

### 3 Mostrar un missatge

En aquest apartat s'ha d'implementar una funció, definida com:

```
1  private void mostraMissatge(String s, boolean llarg)
```

On *s* és un *String* que conté el missatge que es vol mostrar i *llarg* és una variable booleana que indica si el temps per mostrar el missatge és llarg o curt. Per mostrar un missatge, podem utilitzar un **Toast**.

Un toast (o avís) proporciona informació simple sobre una acció en una petita finestra emergent. Només ocupa la quantitat d'espai necessari per al



Figura 2: Mostrar la primera lletra de les paraules amagades *alè* i *foc*.

missatge i l'activitat en curs roman visible i continua admetent la interacció. Els avisos desapareixen automàticament després d'un temps d'espera.

Per a crear una instància d'un objecte *Toast*, hem d'utilitzar el mètode *makeText()*, el qual admet els següents paràmetres:

- El context de l'aplicació.
- El text que s'ha de mostrar a l'usuari.
- La durada que es mostra l'avís a la pantalla (*Toast.LENGTH\_SHORT*, *Toast.LENGTH\_LONG*).

Per a mostrar l'avís, hem de cridar al mètode *show()*:

```

1 Context context = getApplicationContext();
2 CharSequence text = "Hello toast!";
3 int duration = Toast.LENGTH_LONG;
4
5 Toast toast = Toast.makeText(context, text, duration);
6 toast.show();
```

## 4 Funcionalitat del botó *Reiniciar*

En aquest apartat s'ha d'implementar la funcionalitat del botó reiniciar, que ha de:

- Esborrar de la pantalla les caselles de les lletres amagades actuals.
- Canviar el color dels elements de la pantalla (cercle i lletres amagades).
- Mostrar les noves paraules amagades.
- Reiniciar totes les variables necessàries per començar una nova partida.

Pensau que, de moment, no podem reiniciar totes les coses (encara no tenim les paraules i no es pot determinar quantes ni quines solucions tenim, però ho podem simular amb valors constants).

## 5 Habilitar i deshabilitar *Views* de la pantalla

En aquest apartat s'han d'implementar dues funcions, definides com:

```
1  private void enableViews(int parent)
2  private void disableViews(int parent)
```

Que, a partir de l'identificador d'un element de la pantalla (el *parent*), ha d'habilitar totes les seves components (*enableViews*) o bé deshabilitar totes les seves components, excepte els botons de *Bonus* i de *Reiniciar* (*disableViews*).

Aquestes funcions s'utilitzaran quan s'acabi el joc (es desabilita tot, excepte els botons de *Bonus* i de *Reiniciar*) i quan es torni a començar una partida (s'habilita tot).

El paràmetre *parent* corresindrà a l'identificador del vostre *ConstraintLayout*.

Podreu aconseguir aquestes dues funcionalitats amb els mètodes:

- *setEnabled(boolean t)*: habilita o deshabilita un *View* (*true* o *false* al paràmetre).
- *getChildCount()*: retorna el número de fills que conté un *ViewGroup* (podeu fer un *cast* del vostre *ConstraintLayout* a un *ViewGroup*).
- *getChildAt(int i)*: agafa el *View* fill de la posició *i* d'un *ViewGroup*.

Un *ViewGroup* és un *View* que conté altres *Views* (agrupació de *Views*), anomenats fills. Així, a partir del *Layout* principal, es pot accedir a totes les seves components (és a dir, a partir del *ConstraintLayout* podeu accedir a tots els *Views* de la pantalla).

Si voleu fer un tractament diferent a alguns dels elements (botons de *Bonus* i de *Reiniciar*), ho podeu fer comparant el seu *id* amb l'*id* dels fills que heu recuperat del *ViewGroup* (*getId()*).