

PROGRAMACIÓ AVANÇADA

TRIMESTRE 1. CURS 2022/23

PRÀCTICA 1

Objectiu: Aprendre un nou llenguatge de programació: C++

- Treball amb taules: vector **dinàmic** i matriu **estàtica**
- Treballar els conceptes bàsics de la P.O.O.: **classes, atributs, mètodes i constructor.**
- **Sobrecarrega d'operadors.**

Durada: Una sessió.

Lliurament: Llistat imprès dels fonts i penjar el projecte al Moodle.

Data Lliurament: Abans de la sessió de la Pràctica 2.

Un dels objectius de la pràctica és aprendre un nou llenguatge de programació, el C++, per això és necessari:

- 1.- Instal·lar l'entorn per treballar amb C++:

Code::Blocks [escull l'opció que inclou el compilador](#)

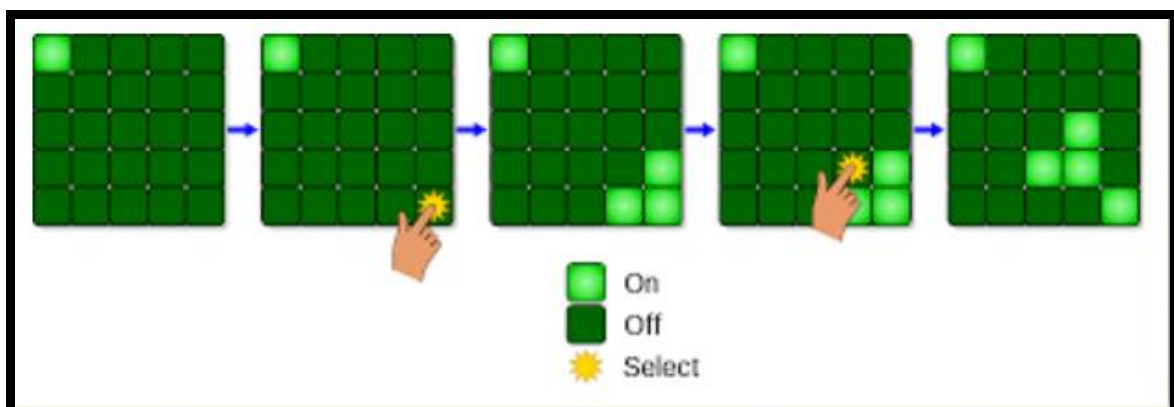
- 2.- Llegir el document: Apunts de Programació. C++. Part 1 i 2

- 3.- Programar en C++ un disseny de classes per provar els elements més bàsics d'aquest llenguatge de programació.

Enunciat


JOC. Fora llums! (Lights Out).

Aquest joc és un trencaclosques lògic que es juga amb un taulell de botons, concretament 25 elements disposats en 5 files i 5 columnes. Cada botó representa una bombeta i prement-lo aquest canvia d'estat, si està encès s'apaga i si està apagat s'encén, però a més a més les bombetes veïnes (amunt, baix, dreta i esquerra) també canvien el seu estat. L'**objectiu** del joc consisteix en prémer els botons necessaris per **apagar totes les llums** partint d'un estat inicial de bombetes enceses i d'altres apagades.



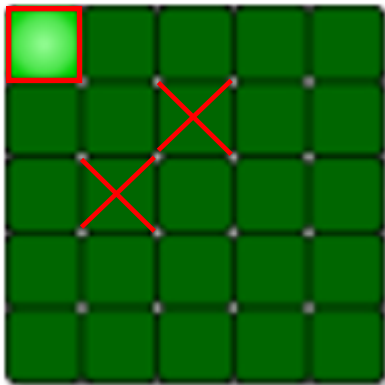
En el joc original les veïnes d'una bombeta són les que estan ubicades dalt, baix, dreta i esquerra d'aquesta. En la versió que se us demana de programar es consideraran **les 8 possibles veïnes distants** que pot tenir una bombeta, les caselles a les que es pot accedir fent **els moviments permesos per la peça del cavall en el joc dels escacs**.

Recordeu, un cavall en el joc dels escacs pot fer 8 moviments diferents, evidentment segons on estigui ubicat no sempre seran tots possibles (no es pot sortir del taulell de joc).

	6		7	
5				8
				
4				1
	3		2	

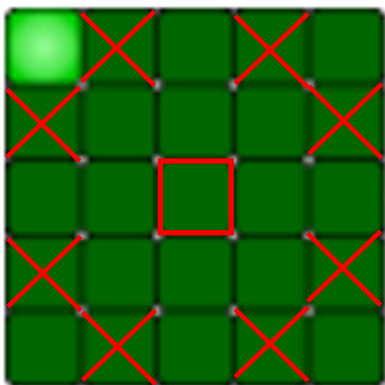
Així doncs, si la bombeta que prem l'usuari es correspon amb la posició on està ubicat el cavall les caselles que canviarien d'estat, **a més de la casella premuda** seran les enumerades de l'u al vuit.

Exemple 1

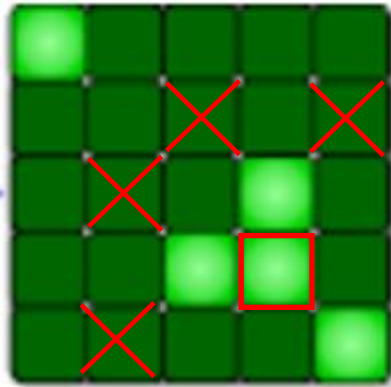


Si es prem la bombeta **emmarcada**, canvien d'estat, a més d'ella mateixa, les marcades amb una creu. Té **només té 2 veïnes distants**

Exemple 2



Si es prem la bombeta **emmarcada** canvien d'estat les marcades i ella mateixa. Té **8 veïnes distants**

Exemple 3

Si es prem la bombeta **emmarcada** canvien d'estat les marcades i ella mateixa. Té 4 veïnes distants

Especificacions del joc

- El taulell el representareu mitjançant una matriu.
- Les dimensions del taulell seran de 8x8. No treballarem amb les dimensions de taulell del joc original.
- Inicialment s'encendran un número de bombetes especificat pel jugador, i **aleatòriament** es generaran les posicions que han d'estar enceses inicialment. Controleu les repeticions de caselles. Si el jugador especifica un número inicial de bombetes enceses igual o superior al número de caselles del taulell no s'acceptarà el valor i **s'insistirà** fins que sigui menor. Evidentment també ha de ser un valor positiu major a zero.
- El jugador abans de començar el joc ha d'indicar a l'aplicació el número **màxim** de "premudes que vol fer" (ha de ser un valor major de 0) i l'aplicatiu a l'assolir-les **ha de parar** i mostrar resultats finals. A cada premuda n'hi direm **tirada**.
- Si abans d'assolir el número màxim de tirades el taulell està tot apagat l'aplicatiu també ha de finalitzar. **S'ha assolit l'objectiu del joc.**
- L'usuari per indicar la bombeta del taulell que vol prémer haurà d'especificar les seves coordenades (fila, columna). Si refereixen a una casella fora del taulell s'ha d'indicar a l'usuari i no comptabilitzar-la com a tirada.
- **Després de cada tirada a l'igual que un cop creat s'ha de mostrar el taulell de joc.** Useu una representació gràfica del taulell on es pugui veure clarament quines bombetes estan il·luminades i quines apagades.
- **Quan finalitza el joc, tant si és amb èxit com perquè s'han esgotat les tirades, mostreu un resum de l'evolució del joc:**
 - 1.- després de cada tirada cal saber quantes bombetes estan enceses i quantes apagades però la visualització d'aquestes dades s'ha de fer un cop **acabi el joc**. Com s'ha de mostrar al final, cal anar emmagatzemant les dades. **Useu un vector de tantes posicions com el número de tirades a fer**, i en cada posició un cop efectuada la tirada emmagatzemeu-hi el número de bombetes enceses ó apagades. Al finalitzar només caldrà fer un recorregut sobre aquest vector.
 - 2.- indiqueu la tirada que ha generat més fosc al taulell, és a dir, la que té menys bombetes enceses. Si dos o més tirades generen la mateixa fosc és irrellevant la tirada triada. Amb el mateix recorregut podeu resoldre ambdues parts.
- El jugador pot fer tantes partides com vulgui, un cop finalitzada se li demana si vol o no jugar altre cop. Si es torna a jugar s'ha d'apagar tot el taulell i demanar novament les dades, en cas contrari s'ha d'acomiar!

Pauta de Programació. Obligat compliment

IMPORTANT: quan tingueu que accedir a les bombetes **veïnes distants** useu una **estructura iterativa**.

Cal un recorregut pels 8 moviments:

```
for (moviment=1; moviment<=8; moviment++)
    Accés a la veïna distant si existeix aplicant el moviment
```

Per poder muntar aquesta iteració cal **emmagatzemar "els desplaçaments relatius" a la casella premuda en una matriu de 2 files i 8 columnes**. A la fila d'índex 0 hi emmagatzemeu els valors relatius a les files i a la fila d'índex 1 els de les columnes.

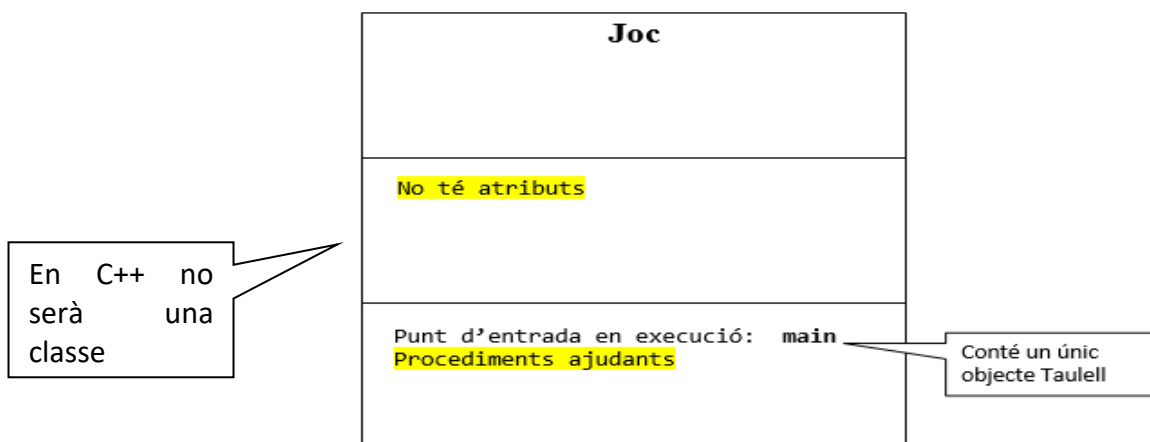
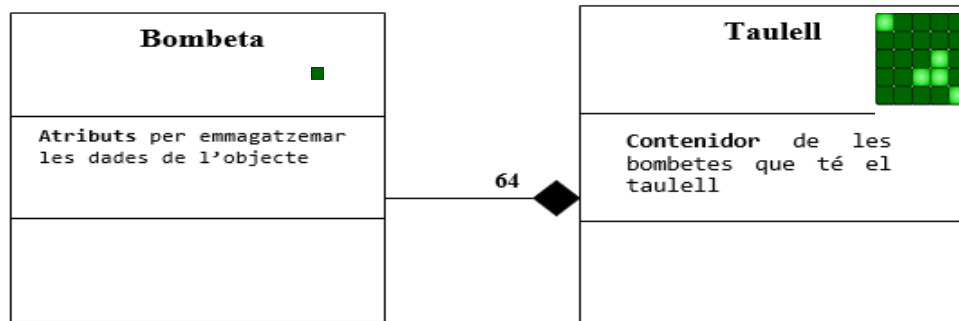
	Moviment 1	Moviment 2	Moviment 2	Moviment 3	Moviment 4	Moviment 5	Moviment 6	Moviment 7	Moviment 8
fila	1								
col	2								

Exemple:

Si es prem la bombeta (2,4) aplicar el moviment 1 vol dir: sumar 1 a la fila i 2 a la columna, generant la posició (3,6)

Estructura. Disseny de classes:

Consta de tres classes i relació de continença (agregació).



La classe Bombeta

Aquesta classe té un únic **atribut**, que és privat, per emmagatzemar si la bombeta està encesa o apagada, ha de ser de tipus booleà. Decideix el nom que li vols donar.

La classe té els següents **mètodes**:

- **Constructor** sense paràmetres. Construeix una bombeta apagada.
- Mètode **getter** per consultar l'estat de la bombeta, és a dir l'atribut de l'objecte.
- Mètode **setter** per modificar l'estat de la bombeta al valor que arriba per paràmetre.
- Mètode que visualitza a pantalla l'estat de la bombeta, useu un caràcter que permeti diferenciar clarament encesa o apagada.

void visualitza()

Per exemple



El quadrat per representar que està encesa i el guió per apagat.

- **Sobrecàrrega dels operadors: == !=** . Aquests mètodes se'ls hi dona la funcionalitat de determinar si dues bombetes estan en el mateix estat, és a dir retorna true si ambdues estan enceses o apagades, fals en cas contrari.

bool operator==(Bombeta)

//són iguals si ambdues estan enceses o apagades

bool operator!=(Bombeta)

//no ho són si una està encesa i l'altra apagada.

Adoneu-vos que el mètode només té un paràmetre tot i ser un operador binari.

En la implementació és imprescindible que un mètode invoqui a l'altre usant la simbologia de l'operador.

La classe Taulell

Aquesta classe té els següents **atributs**, tots ells de visibilitat privada. El nom, tipus i funcionalitat estan descrits a continuació:

- **contenedor**: per representar el taulell de joc, evidentment és una matriu d'objectes Bombeta.

Bombeta contenedor[8][8];

Adoneu-vos que la matriu es crea en la declaració (**no cal** usar **new** en C++ és tenen dues opcions, treballar amb taules dinàmiques o estàtiques). **En el moment d'aquesta declaració es creen els 64 objectes Bombeta invocant automàticament al constructor de la classe Bombeta que no té paràmetres).**

- **files**: enter per emmagatzemar el número de files que té el taulell de joc, serà exactament 8. Useu aquest atribut **sempre** que necessiteu treballar amb el número de files del taulell (si en qualsevol moment es modifiqués el número de files del taulell no caldria modificar la programació, només el contingut d'aquest atribut).
- **columnes**: enter per emmagatzemar-hi el número de columnes del taulell de joc. El mateix que les files però ara les columnes.
- **enceses**: enter per comptabilitzar el número de bombetes enceses que té el taulell en tot moment (**cada cop que es modifica l'estat d'una bombeta aquest atribut s'ha d'actualitzar**).

I els següents **mètodes** tots ells de visibilitat pública:

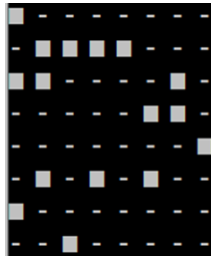
- Un mètode constructor per crear el taulell de joc. Tots els atributs de l'objecte construït han de quedar inicialitzats. Totes les bombetes estan apagades.

- Un mètode per encendre N bombetes del taulell, essent N un paràmetre del mètode. Genereu aleatòriament **quines** bombetes s'han d'encendre. S'han d'encendre exactament N bombetes i no poden estar repetides. Supposeu que el valor del paràmetre N **sempre** és correcte. Per generar un número aleatori useu la funció **rand()** de la llibreria `<stdlib.h>`

void encendre(int quantes)

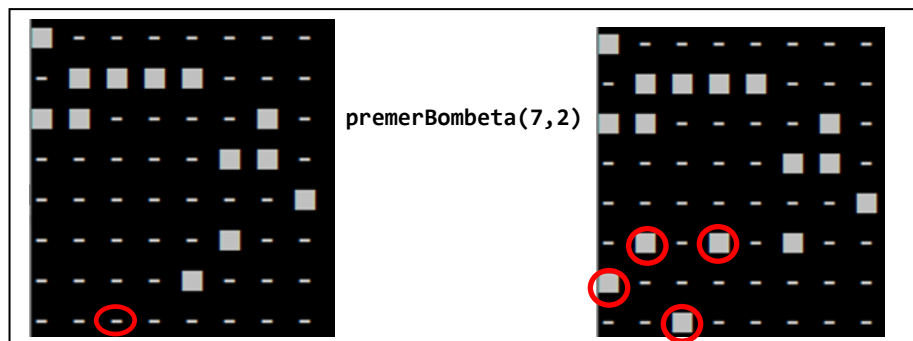
- Un mètode per mostrar a pantalla l'atribut **contenidor** amb tot el seu contingut, en format de matriu.

void visualitzar()



- Un mètode per modificar l'estat d'una bombeta del contenidor. Al mètode mitjançant paràmetres se li indica quina bombeta es vol modificar, indicant fila i columna. Si la bombeta està encesa s'ha d'apagar i en cas contrari encendre. **Aquest mètode també ha de canviar l'estat de les bombetes veïnes distants (moviments del cavall). El mètode simula una premuda d'un botó del taulell del joc.**

void premerBombeta(int f, int c);



Imprescindible que aquest mètode mantingui el quart atribut de l'objecte actualitzat. Supposeu que el valor dels paràmetres són sempre correctes.

- Un mètode que apagui totes les bombetes del contenidor. Ha d'apagar totes les bombetes i posar el darrer atribut a zero.

void apagar()

- Mètodes **getters** per consultar tots els atributs de l'objecte **a excepció del contenidor que no es pot consultar.**
- Mètode que retorni la bombeta que ocupa la posició indicada pels paràmetres. Supposeu que el valor dels paràmetres són sempre correctes.

Bombeta getBombeta(int, int)

Nota: cap d'aquests mètodes pot tenir operacions de lectura (**cin**>> equivalent a Keyboard de Java), les dades necessàries li arriben via paràmetres. **Només el mètode que ha de visualitzar el taulell pot fer ús de les operacions d'escriptura (cout<< equivalent a System.out.print de Java).**

La classe Joc

Ha d'incloure com a mínim el punt d'entrada en execució del programa, el procediment **main** i el procediment ajudant que es demana.

En aquesta classe, s'ha de:

- sol·licitar a l'usuari mitjançant operacions de lectura totes les dades que requereix el joc,
- controlar la lògica del joc invocant als mètodes de la classe Taulell,
- s'ha d'emmagatzemar el número de bombetes que estan enceses ó apagades després de cada tirada, informació que s'ha de mostrar al finalitzar el joc, a més d'indicar quina ha estat la tirada en què hi ha hagut més fosc,
- i gestionar la continuïtat en jugar o no novament amb el mateix taulell i número de bombetes enceses inicialment. Per sol·licitar aquesta dada i en cas de continuïtat preparar el taulell per la nova partida és imprescindible escriure un procediment:

bool tornarJugar(Taulell *tau)

El procediment demana a l'usuari si vol fer altra partida. En cas afirmatiu ha d'apagar tot el taulell per iniciar-la. Imprescindible que el paràmetre sigui **passat per referència**, cal treballar amb l'objecte original. El procediment retorna fals si el programa ha d'acabar.

Nota: la simbologia * fa referència a un pas per referència. En la invocació cal tenir-ho present.

El mètode **main** ha de treballar amb un únic objecte Taulell estàtic. Queda prohibit usar cap variable de tipus matriu per gestionar el joc.

Taulell tau();

La sentència **crea** una variable anomenada **tau** de tipus Taulell, invocant automàticament al constructor que no té paràmetres per la seva creació. En Java tots els objectes són dinàmics, s'han de crear amb l'operador new, en C++ tenim ambdues opcions.

El mètode **main** treballarà amb un vector, amb tantes posicions com el número de tirades que especifica l'usuari per emmagatzemar les dades que s'han de printar al final de cada partida. En aquest cas el vector **ha de ser obligatòriament dinàmic** ja que la dimensió depèn d'una entrada especificada per l'usuari. S'haurà de crear mitjançant l'operador **new** i alliberar l'espai que ocupa quan ja no cal usar-lo més o acaba el programa, mitjançant l'operador **delete**.

int *vector;

Es **valorarà positivament** l'escriptura de procediments privats per aquelles parts que per sí soles tenen una clara funcionalitat i també per aquelles que s'han d'executar varies vegades amb l'objectiu de reaprofitar codi.

Serà en aquesta classe **on es controlarà la correctesa de totes les dades entrades per l'usuari** i no pas als mètodes de la classe Taulell, que les rebran via paràmetre i les consideraran correctes.

Denoteu que tota la part que fa referència a modificacions i consultes del taulell les haureu de fer invocant als mètodes de la classe Taulell, tots ells amb visibilitat pública.

Aquesta classe no tindrà atributs, únicament el mètode main i procediments privats per ajudar aquest main. **Ni el main ni els procediments poden fer ús de variables de tipus matriu i únicament el main declararà i crearà un objecte de la classe Taulell.**

Exemple d'execució

```
Taulell de 8x8
Indica quantes bombetes vols encendre
Especifica un valor dins del interval [1, 64]
7
Encenem la posicio: 1,1
Encenem la posicio: 1,3
Encenem la posicio: 1,4
Encenem la posicio: 2,0
Encenem la posicio: 3,6
Encenem la posicio: 6,4
Encenem la posicio: 6,6
- - - - -
- ■ - ■ ■ - - -
■ - - - - -
- - - - - ■ -
- - - - - -
- - - - - -
- - - ■ - ■ -
- - - - -
Especifica numero maxim de tirades
5
Tirada : 1
*****
Indica les coordenades de la bombeta a premer
Indica la fila:
-----
Especifica un valor dins del interval [0, 7]
0
Indica la columna:
-----
Especifica un valor dins del interval [0, 7]
0
■ - - - - -
- ■ ■ ■ - -
■ ■ - - - -
- - - - - ■ -
- - - - - -
- - - ■ - ■ -
- - - - -
Tirada : 2
*****
Indica les coordenades de la bombeta a premer
Indica la fila:
-----
Especifica un valor dins del interval [0, 7]
4
Indica la columna:
-----
Especifica un valor dins del interval [0, 7]
7
```

Posicions generades
aleatòriament. No poden
estar repetides!!!

Les dades entrades per l'usuari han
d'estar **testejades**, cal insistir mentre
el valor demanat no sigui correcte

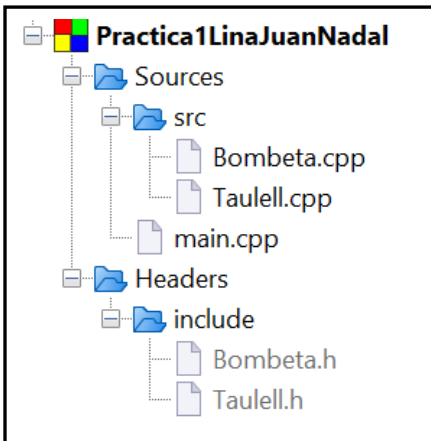
Les dades entrades per l'usuari han
d'estar testejades, cal insistir mentre
el valor demanat no sigui correcte

ORGANITZACIÓ. ESTRUCTURA DEL PROJECTE

Cadascuna de les pràctiques que desenvolupareu en aquesta assignatura ha de ser un projecte diferent. Les pràctiques **obligatòriament** s'han de fer en grups de 2 estudiants.

Els noms dels vostres projectes, en les pràctiques d'aquesta assignatura han de seguir el següent patró: PràcticaXCognom1Nom1&&Cognom2Nom2. En aquesta pràctica X és 1.

Aquesta pràctica ha de tenir la següent estructura:



Cada **classe** té el seu fitxer ***.h** amb la declaració dels seus membres (atributs i membres) i el seu ***.cpp** amb la implementació dels mètodes.

El mètode main i els seus procediments ajudants estaran ubicats dins d'un fitxer anomenat **main.cpp**

Què se us subministra?

- Fitxer amb l'enunciat de la pràctica. Document que esteu llegint.
- Aclariments Pràctica 1.
- Document: Apunts de Programació: C++. Part 1 i Part 2.
- Document: Variable dinàmica en C++.

Què s'ha de lliurar i com?

S'ha de lliurar la carpeta que conté el **projecte** Code::Blocks amb el vostre desenvolupament de la pràctica. La carpeta s'ha de lliurar amb tot el seu contingut i comprimida amb ZIP o RAR.

També s'ha de lliurar el **llistat en paper** del codi desenvolupat. El format de lliurament d'aquest codi ha de seguir el patró indicat en la presentació de l'assignatura: amb portada, índex, número de pàgina, tabulació ... En **aquest llistat** cal que indiqueu quina ha estat **la distribució de la feina entre els dos estudiants. És a dir, el grau de participació de cada membre del grup en la realització d'aquesta activitat. Explica les principals dificultats en el seu desenvolupament, valoració del codi lliurat i funcionament de la pràctica.**

On s'ha de lliurar?

El lliurament del projecte es farà a través de la plataforma Moodle i no s'acceptarà cap altra via. Feu atenció a la data i hora límit.

El lliurament en paper es farà directament a la professora a l'inici de la pràctica 2.

Quan s'ha de lliurar?

El lliurament a la plataforma es podrà fer fins el dia i hora indicats. Tingueu present que a partir d'aquesta hora el sistema bloquejarà, de manera automàtica, la possibilitat de lliurament.

Tots els grups → 19 d'octubre a les 8:00h

Rubrica de correcció

La pràctica serà puntuada sobre 10. Concretament la distribució de punts es farà de la següent forma:

Classe Bombeta → 1 punt

Classe Taulell → 4 punts

Mètode main i ajudants → 5 punts

Es valorarà l'eficiència de les implementacions. Per una banda, si amb un únic recorregut i/o cerca es pot determinar el que ha de calcular el mètode no se'n faran ni dos, ni més. **Per altra banda,** s'ha d'aplicar anàlisi descendent en la implementació d'aquells mètodes que poden reaprofitar codi o senzillament són llargs i es trosseggen per millorar la seva bona entesa. **Imprescindible seguir les pautes indicades a l'enunciat.**

Si el programa no funciona la màxima puntuació és un 4.