CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

ACTIVITAT AVALUABLE

Programació
CFGS DAW

Autors:
Salvador Rue Orquin – s.rueorquin@edu.gva.es
Guillermo Garrido – g.garridoportes@edu.gva.es

2023/2024

Llicència

‘EL POBLE DORMIT’

1. INTRODUCCIÓ

L'objectiu d'aquest exercici consisteix principalment en fer-vos utilitzar el conjunt de conceptes apresos, tant en la programació orientada a objectes com en la gestió d'excepcions. És per açò, que en alguns casos es pot pensar que alguns elements dels que es demanen no són estrictament necessaris o que es poden aplicar solucions més simples, però amb això no aconseguiríem l'objectiu i competències que amb aquest exercici es persegueixen.

Per tant, en aquest exercici es donen especificacions molt concretes (sense indicar la solució) per a que s'utilitzen les instruccions i estructures Java adequades. És molt important que les seguiu ja que la rubrica contemplarà si es fa ús d'aquestes o no.

Igualment, donada la complexitat de les dependències, és important que vos ajusteu a la nomenclatura específica que es proporciona al llarg de l'exercici.

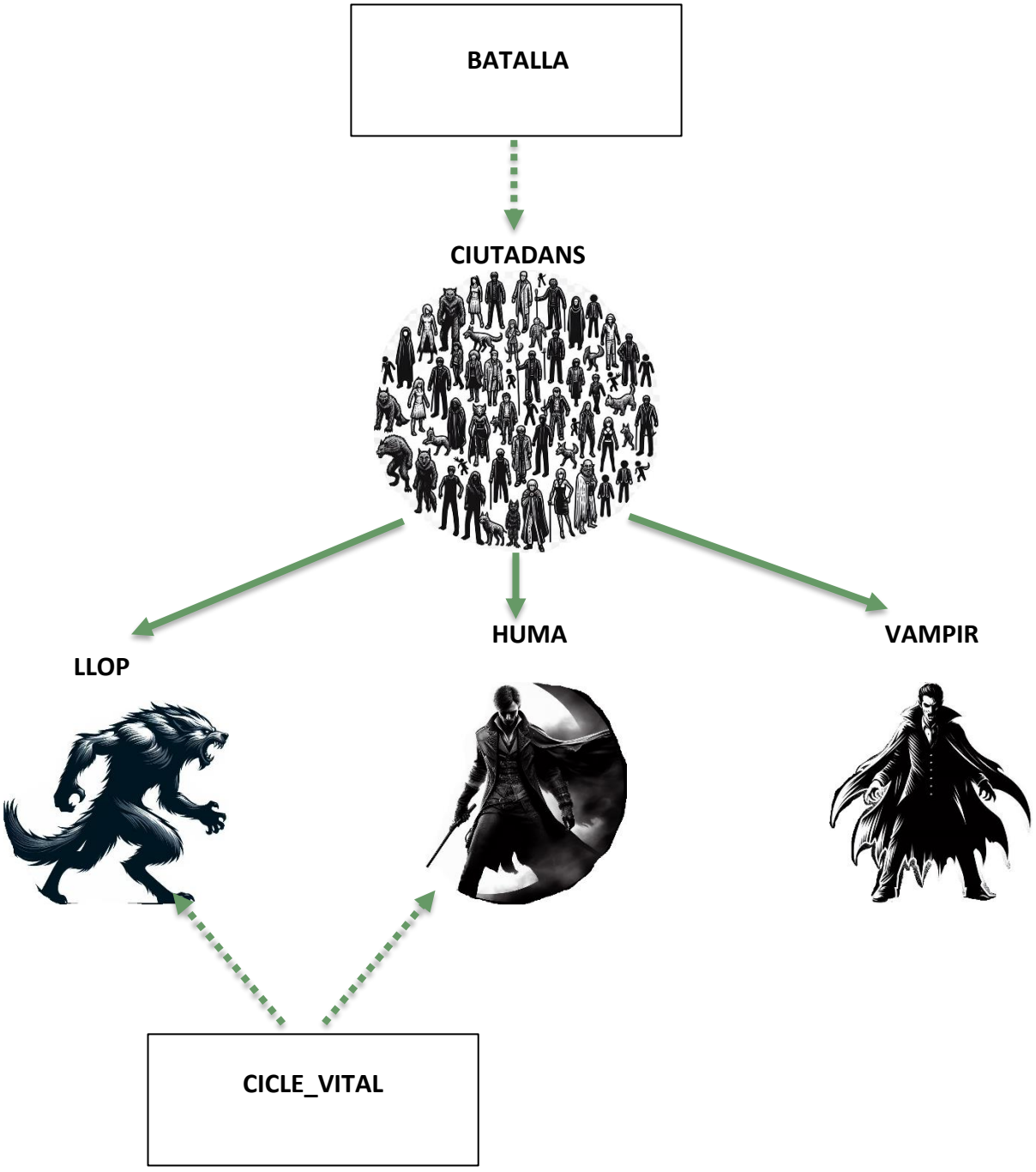
2. DESCRIPCIÓ DEL JOC: EL POBLE DORMIT

En este projecte simularem una ciutat on conviuen ciutadans de tres tipus, humans, vampirs i dones/homes llop (diguem-li llops), cadascun vulnerable al següent, d'aquesta manera els vampirs són vulnerables als llops que són vulnerables als humans, que són vulnerables als vampirs.

Són veïns conflictius, així, que són de tipus combatent, cada any s'enfronten de nit atacant-se entre ells, però compte, cadascun no sap a quin ésser ataca, es probable que muira, o qui sap, si son homònims acaben reproduint-se.

Hem de tindre en compte que tots tres són ciutadans d'igual dret i figuren al cens com a tal, però els llops i humans són éssers vius, i implementen les funcions de tot ésser viu, així com els vampirs són morts que en compte de reproduir-se, convertixen l'humà que ataquen.

La gestió d'esta població tant fascinant com tortuosa s'ha de realitzar amb un programa Java que l'implemente mitjançant la relació entre les classes/interfícies i els mètodes adequats.


BATALLA
CIUTADANS
LLOP
HUMA
VAMPIR
CICLE_VITAL

3. ESTRUCTURA BASE

Definirem dos esquemes des d'on s'establiran determinats atributs i mètodes que la resta de components hauran d'implementar. D'esta manera podrem variar el comportament de la vida i de les batalles de forma fàcil simplement canviant alguns valors.

3.1. VULNERABLE

Redueix el tipus de vulnerabilitat de cada individu a un dels tres valors, Vampir, Humà i Llop.

3.2. BATALLA

Conté el mètode que definix la batalla.

- **combat(...)**: on cada tipus d'ésser especificarà com combateix, que ocorre si troba la seua la seua presa natural o ser al qual és vulnerable. Per tant, serà necessari que entre un oponent i ens retorne el perdedor.

3.3. CICLEVITAL

Es definirà una constant màxima per a la natalitat (el nombre de fills), com una altra màxima per a la mortalitat (els anys de vida).

- NATALITAT_MAXIMA = 1
- VITALITAT_MAXIMA = 2

També, indicarà els mètodes que tots els éssers vius hauran d'implementar quan es programen:

- **reproduir(...)**, que tindrà com a finalitat que ésser dels vius en genere nous ciutadans d'este tipus que sumarà a la població.

- **envellir(...)**, on cada any que passe és un any menys de vida de l'individu i que cause la mort natural del mateix si no li quedaren anys de vida.

4. LA SUPERCLASSE: CIUTADA

La classe Ciutadà servirà com a superclasse a partir de la qual es definiran les altres classes relacionades amb els ciutadans del poble.

Esta superclasse podrà ser heretada per altres classes, però no es podrà instanciar directament. En esta classe es proporcionarà la base per a les definicions del comportament que tindran els ciutadans en general. En alguns casos, estos comportaments es definiran completament, mentre que en altres casos s'exigirà que les subclasses implementen estos comportaments.

4.1. ATRIBUTS DE CLASSE

- poblacio: aquest atribut s'incrementarà cada vegada que es creï un nou ciutadà i es reduirà quan un ciutadà mor.

4.2. ATRIBUTS D'INSTÀNCIA

- nom (String): un text descriptor.

Aquest atribut només podrà ser accedit o modificat des de la mateixa classe Ciutadà.

4.3. CONSTRUCTOR

- El nom del ciutadà es proporcionarà com a paràmetre en el constructor.

No s'ha d'oblidar d'actualitzar l'atribut de classe població en aquest punt.

4.4. MÈTODES

- int getPoblacio(): Retorna el nombre total de ciutadans (mètode de classe).
- void setPoblacio(int numero): Actualitza el nombre total de ciutadans (mètode de classe).
- String getNom(): Retorna el nom del ciutadà.
- void setNom(String nom): Actualitza el nom del ciutadà.
- String toString(): Retorna una representació en cadena del ciutadà. Exemple: “Nom del ciutada: XXXX".
- void censar(ArrayList<Ciutada> ciutadans): Imprimeix els ciutadans i imprimeix les poblacions totals (mètode de classe).
- void poblacionsTotals(): Imprimeix el cens total de ciutadans, humans, llops i vampirs (mètode de classe).
- void morir(ArrayList<Ciutada> ciutadans): Indica el comportament de quan un ciutadà mor (mètode no implementat en la classe).

5. La classe: Huma

Heretarà de la superclasse. Podrà crear instancies però no permetrà la creació de subclasses. Esta classe representa un esser viu.

5.1. ATRIBUTS DE CLASSE

- totalHumans: Enter que representa el número total de humans.
- ultimHuma: Enter que indica el número d'últim humà creat.
- ALEATORI: Objecte Random utilitzat para generar números aleatoris.

5.2. ATRIBUTS D'INSTÀNCIA

VULNERABLE: Tipo de vulnerabilitat del humà.
vida: Enter que representa la vida restant del humà. Es genera aleatòriament i serà definida per la VITALITAT_MAXIMA.

5.3. CONSTRUCTOR

Constructor que crea un nou humà amb un nom assignat basat en l'últim humà creat (exemple HUMA25), una vulnerabilitat a Vampir i una vida aleatòria.

5.4. MÈTODES

- int getPoblacio(): Retorna el nombre total de humans (mètode de classe).
- void setPoblacio(int numero): Actualitza el nombre total de humans (mètode de classe).
- Ciutada combat(Ciutada oponent): Realitza una batalla amb un oponent i retorna el perdedor. Cal afegir un text informat del que ocorre.
- void reproduir(ArrayList<Ciutada> ciutadans): Produeix una reproducció amb un nombre aleatori de fills utilitzant la NATALITAT_MAXIMA. Crea els fills de la classe.
- void morir(ArrayList<Ciutada> ciutadans): Indica el comportament de quan un humà mor. Disminueix la població i ja no forma part dels ciutadans.
- void envellir(ArrayList<Ciutada> ciutadans): Redueix la vida d'un humà d'un en un i el mata si la vida arriba a zero.
- Vulnerable getVulnerable(): Retorna el tipus de vulnerabilitat del humà.
- String toString(): Retorna una representació en cadena del humà, incloent la seva vida i la seva vulnerabilitat.

6. La classe: Llop

Heretarà de la superclasse. Podrà crear instancies però no permetrà la creació de subclasses. Esta classe representa un esser viu.

6.1. ATRIBUTS DE CLASSE

- totalLlops: Enter que representa el número total de llops.
- ultimLlop: Enter que indica el número d'últim llop creat.
- ALEATORI: Objecte Random utilitzat para generar números aleatoris.

6.2. ATRIBUTS D'INSTÀNCIA

VULNERABLE: Tipo de vulnerabilitat del llop.
vida: Enter que representa la vida restant del llop. Es genera aleatòriament i serà el doble de la VITALITAT_MAXIMA.

6.3. CONSTRUCTOR

Constructor que crea un nou llop amb un nom assignat basat en l'últim llop creat (exemple LLOP24), una vulnerabilitat a Huma i una vida aleatòria.

6.4. MÈTODES

- int getPoblacio(): Retorna el nombre total de llops (mètode de classe).
- void setPoblacio(int numero): Actualitza el nombre total de llops (mètode de classe).
- Ciutada combat(Ciutada oponent): Realitza una batalla amb un oponent i retorna el perdedor. Cal afegir un text informat del que ocorre.
- void reproduir(ArrayList<Ciutada> ciutadans): Produeix una reproducció amb un nombre aleatori de cadells utilitzant el doble de la NATALITAT_MAXIMA. Crea els cadells de la classe.
- void morir(ArrayList<Ciutada> ciutadans): Indica el comportament de quan un llop mor. Disminueix la població i ja no forma part dels ciutadans.
- void envellir(ArrayList<Ciutada> ciutadans): El temps passa més de pressa per als canins així que la vida d'un llop es redueix el doble ràpid que la d'un humà. Quan la vida arriba a zero mor.
- Vulnerable getVulnerable(): Retorna el tipus de vulnerabilitat del llop.
- String toString(): Retorna una representació en cadena del llop, incloent la seva vida i la seva vulnerabilitat.

7. La classe: Vampir

Heretarà de la superclasse. Podrà crear instancies però no permetrà la creació de subclasses.

7.1. ATRIBUTS DE CLASSE

- totalVampirs: Enter que representa el número total de vampirs.
- ultimVampir: Enter que indica el número d'últim vampir creat.

7.2. ATRIBUTS D'INSTÀNCIA

VULNERABLE: Tipo de vulnerabilitat del vampir.

7.3. CONSTRUCTOR

Constructor que crea un nou vampir amb un nom assignat basat en l'últim vampir creat (exemple VAMPIR27), i una vulnerabilitat a Llop.

7.4. MÈTODES

- int getPoblacio(): Retorna el nombre total de vampirs (mètode de classe).
- void setPoblacio(int numero): Actualitza el nombre total de vampirs (mètode de classe).
- Ciutada combat(Ciutada oponent): Realitza una batalla amb un oponent i retorna el perdedor. Cal afegir un text informat del que ocorre.
- void morir(ArrayList<Ciutada> ciutadans): Indica el comportament de quan un vampir mor. Disminueix la població i ja no forma part dels ciutadans.
- Vulnerable getVulnerable(): Retorna el tipus de vulnerabilitat del vampir.
- String toString(): Retorna una representació en cadena del vampir, incloent la seva vida i la seva vulnerabilitat.

8. Programa Principal: ElPobleDormit

8.1. Creació de l'ecosistema

Es crearà una llista ciutadans. Este inclourà tots els ciutadans del sistema.
Hi a que establir una població mínima i màxima que no podrà canviar.
També s'utilitzaran variables per a poder generar nombres aleatoris i per a poder recuperar dades de la consola.

8.2. Mètode void main(String[] args)

Generarà una població aleatòria cada vegada que executa el joc.
Per a començar ha de informar al jugador dels ciutadans que hi ha en el joc.
A continuació i mostrarà el menú, per a que el jugador puga seleccionar que fer (1-Mostrar cens, 2-Passar un any o 3-Eixir del programa)
Verificarà si la població es tota del mateix tipus.
En cas que la població siga del mateix tipus o que el usuari selecciona "Eixir ..." s'acabarà la execució.
Si no, executa les opcions seleccionades fins que l'usuari decideixi eixir.

8.3. Mètode boolean mostrarMenu(boolean continuar):

Mostra un menú amb opcions per a l'usuari i executa l'acció corresponent segons la selecció de l'usuari.
Les opcions son:

1. Cens actual.
2. Passant un any...
3. Eixint del programa...

8.4. Mètode void generarPoblacioAleatoria():

Genera una població aleatòria de ciutadans entre el mínim i màxim establert en el sistema. Hi a que tindre en compte que a de afegir al llistat de ciutadans, un ciutadà creat de manera aleatòria.

8.5. Mètode obtenirCiutadaAleatori():

Crea i retorna un ciutadà aleatori (humà, llop o vampir).

8.6. Mètode passarAny():

Simula el pas d'un any al poble.
Cada ciutadà realitzarà una acció durant l'any.
Les accions de combatre i reproduir-se requereixen d'un segon ciutadà (oponent) que es seleccionarà de manera aleatòria.
A mes a mes s'ha de actualitzar l'edat del ciutadà.

8.7. Mètode obtindreOponentAleatori(int actual):

Retorna un oponent aleatori per a un ciutadà donat. El oponent no pot ser el mateix que el ciutadà seleccionat.

8.8. Mètode realitzarAccio(Ciutada ciutada1, Ciutada ciutada2):

Realitza una acció entre dos ciutadans, si son de diferent espècie combatran. Si son de la mateixa espècie procrearan.

8.9. Mètode combatre(Ciutada oponent1, Ciutada oponent2):

Simula una batalla entre dos ciutadans, on el perdedor pot morir o convertir-se en vampir en cas de ser un humà (de ser així, imprimirà el nou nom del ciutadà).

8.10. Mètode procrear(Ciutada amant1):

Simula la reproducció d'un ciutadà (si es pot reproduir), tenint en compte el límit de població màxima. Si el nombre de ciutadans ja a arribat a la població màxima mostrarà que no s'ha pogut reproduir.

8.11. Mètode actualizarEdad(Ciutada ciutada):

Actualitza l'edat dels ciutadans (si poden envellir), provocant que envellisquen.

8.12. Mètode verificarPoblacio():

Verifica si en la només hi ha un tipus de ciutadà a la població. Si és així, mostrarà el text "Sols queda un tipus de essers en el poble" i l'aplicació acaba.

9. EXCEPCIONS

9.1. Mètode obtindreOponentAleatori

Si no existeix cap oponent (sols un ciutadà en el sistema) llançarà una excepció de tipus IllegalStateException que dirà "No hi ha cap oponent disponible.".

9.2. Mètode realitzarAccio()

Si algun dels ciutadans que arriben en els paràmetres son null, llançarà una excepció de tipus IllegalArgumentException que dirà "Els oponents no poden ser null.".

9.3. Mètode passarAny():

Si no existeix cap ciutadà llançarà una excepció de tipus IllegalStateException que dirà "No hi ha ciutadans disponibles.".
Capturarà totes les excepcions que arriben al mètode i tornarà a llançar una excepció de tipus Exception amb el text "Error al passar l'any: " al que li se concatenarà el text de la excepció capturada.

9.4. Mètode mostrarMenu():

Serà el mètode que captura la excepció final i la mostrarà per consola.

10. EXEMPLE D'EXECUCIÓ

```
run:
Població inicial:
Nom del ciutada: VAMPIR1, vulnerable: Llop
Nom del ciutada: HUMA1, vida: 6, vulnerable: Vampir
Nom del ciutada: HUMA2, vida: 2, vulnerable: Vampir
Nom del ciutada: HUMA3, vida: 5, vulnerable: Vampir
Nom del ciutada: LLOP1, vida: 5, vulnerable: Huma
Nom del ciutada: HUMA4, vida: 1, vulnerable: Vampir
Nom del ciutada: HUMA5, vida: 4, vulnerable: Vampir
Nom del ciutada: HUMA6, vida: 4, vulnerable: Vampir
Nom del ciutada: LLOP2, vida: 7, vulnerable: Huma
Nom del ciutada: LLOP3, vida: 1, vulnerable: Huma
Nom del ciutada: LLOP4, vida: 6, vulnerable: Huma
Nom del ciutada: VAMPIR2, vulnerable: Llop
Nom del ciutada: LLOP5, vida: 2, vulnerable: Huma
Nom del ciutada: VAMPIR3, vulnerable: Llop
Nom del ciutada: VAMPIR4, vulnerable: Llop
Nom del ciutada: HUMA7, vida: 5, vulnerable: Vampir
Nom del ciutada: HUMA8, vida: 6, vulnerable: Vampir
Nom del ciutada: VAMPIR5, vulnerable: Llop
Nom del ciutada: VAMPIR6, vulnerable: Llop
Actualment hi ha un cens de: 19 ciutadans.
8 humans, 5 llops i 6 vampirs

Menú:
1. Mostrar el cens actual
2. Passar un any
3. Eixir del programa
Selecciona una opció:
```

```
Selecciona una opció: 1

Cens actual:
Nom del ciutada: VAMPIR1, vulnerable: Llop
Nom del ciutada: HUMA1, vida: 6, vulnerable: Vampir
Nom del ciutada: HUMA2, vida: 2, vulnerable: Vampir
Nom del ciutada: HUMA3, vida: 5, vulnerable: Vampir
Nom del ciutada: LLOP1, vida: 5, vulnerable: Huma
Nom del ciutada: HUMA4, vida: 1, vulnerable: Vampir
Nom del ciutada: HUMA5, vida: 4, vulnerable: Vampir
Nom del ciutada: HUMA6, vida: 4, vulnerable: Vampir
Nom del ciutada: LLOP2, vida: 7, vulnerable: Huma
Nom del ciutada: LLOP3, vida: 1, vulnerable: Huma
Nom del ciutada: LLOP4, vida: 6, vulnerable: Huma
Nom del ciutada: VAMPIR2, vulnerable: Llop
Nom del ciutada: LLOP5, vida: 2, vulnerable: Huma
Nom del ciutada: VAMPIR3, vulnerable: Llop
Nom del ciutada: VAMPIR4, vulnerable: Llop
Nom del ciutada: HUMA7, vida: 5, vulnerable: Vampir
Nom del ciutada: HUMA8, vida: 6, vulnerable: Vampir
Nom del ciutada: VAMPIR5, vulnerable: Llop
Nom del ciutada: VAMPIR6, vulnerable: Llop
Actualment hi ha un cens de: 19 ciutadans.
8 humans, 5 llops i 6 vampirs

Menú:
1. Mostrar el cens actual
2. Passar un any
3. Eixir del programa
Selecciona una opció:
```

```
Selecciona una opció: 2

Passant un any...
VAMPIR1 ataca HUMA3 guanya convertint-lo en el seu vampir personal VAMPIR7.
Estem de enhorabona! HUMA1 ha tingut 1 fills: HUMA9
HUMA2 no s'ha pogut reproduïr.
LLOP1 ataca VAMPIR4 guanya i es fa un collar amb els seus ullals.
Estem de enhorabona! HUMA4 ha tingut 1 fills: HUMA10
HUMA4 ha mort de vell.
HUMA6 ataca VAMPIR3 pero mor i es converteix en VAMPIR8.
LLOP3 ataca HUMA7 pero mor amb una bala de plata.
Estem de enhorabona! LLOP5 ha tingut 2 cadells: LLOP6, LLOP7
LLOP5 ha mort de vell.
HUMA7 ataca LLOP6 guanya i ven la seua pell per a fer abrics.
HUMA8 ataca LLOP4 guanya i ven la seua pell per a fer abrics.
VAMPIR6 ataca HUMA10 guanya convertint-lo en el seu vampir personal VAMPIR9.
VAMPIR7 ataca HUMA5 guanya convertint-lo en el seu vampir personal VAMPIR10.
VAMPIR8 ataca HUMA2 guanya convertint-lo en el seu vampir personal VAMPIR11.
VAMPIR9 ataca HUMA1 guanya convertint-lo en el seu vampir personal VAMPIR12.
VAMPIR12 ataca LLOP7 pero mor a la llum del sol.
Actualment hi ha un cens de: 16 ciutadans.
3 humans, 3 llops i 10 vampirs

Menú:
1. Mostrar el cens actual
2. Passar un any
3. Eixir del programa
Selecciona una opció:
```

```
Selecciona una opció: 3

Eixint del programa...
```

11. REGLES

Recomanem iniciar l'exercici amb les alternatives més senzilles i quan es tinga el projecte ja funcionant, si es disposa de temps, es facen les modificacions necessàries per a incorporar les alternatives complexes i millorar.

S'avaluarà també la inclusió adequada de comentaris, ordenació del codi, seguiment de la nomenclatura proposada i la programació amb un codi eficient.

El nom dels mètodes, variables o classes s'ha de respectar. Es poden afegir mes variables però no mes mètodes.

12. LLIURAMENT

S'haurà d'entregar en Aules en data i forma segons l'indicat en la tasca:

- El nom de la carpeta principal seguirà la següent nomenclatura **1erCognom-2ºCognom-Nom - Avaluable_3**
- Es crearà un arxiu **.ZIP** (NO .RAR) de la carpeta amb totes les classes e interfícies.
- S'entregarà en la tasca **UF10.- Activitat Avaluable (Enunciat)** l'arxiu ZIP generat.