



A la recerca de l'Arca de l'Aliança



Any 1936, Cindy Jones és una professora universitària d'arqueologia, però alhora una aventurera a la recerca de valuoses relíquies històriques. Després de tornar d'una infructuosa missió a Amèrica del Sud, el govern nord-americà li encarrega una nova comesa: trobar l'Arca de l'Aliança, on es creu que els hebreus van dipositar els manaments que Déu havia atorgat a Moisès, abans que els nazis la trobin i en facin un mal ús. Segons la llegenda, l'objecte atribueix un poder invencible a qui la posseeixi, permetent destruir muntanyes i regions senceres.



Cindy inicia la seva missió als estats units però la seva aventura la porta a recórrer diferents països (vegeu la línia vermella). En el seu periple, Cindy s'ha anat trobat a diferents persones com Marius (el fill d'un arqueòleg amic de Cindy) i el Coronel Diettrich (l'enemic nazi que també està a la recerca de l'Arca). El seu amic Marius l'ha acompanyada en alguns moments però finalment va ser capturat pels nazis que el tenen retingut. Ara, Cindy està a punt de finalitzar la seva missió, ha arribat al pou de les ànimes, que conté amagat sota terra un temple on suposadament es troba l'Arca de l'Aliança. El temple està format per diferents nivells i a cada nivell estan situades diferents cambres.

Ajuda a Cindy en la seva etapa final de la recerca de l'Arca de l'Aliança per evitar que els nazis la trobin. Cindy ha entrat al pou de les ànimes i està just entrant al primer nivell i la primera cambra del temple soterrat. A les diferents cambres Cindy s'anirà trobant diferents objectes: una clau, un diamant, una arca, un amulet, etc. que li serviran per completar la seva missió. El que no sap Cindy és que el coronel Diettrich la persegueix acompanyada del seu amic Marius i d'un munt de soldats nazis que s'estan dispersant pel temple soterrat. Cindy haurà d'agafar diamants màgics que li aporten energia per mantenir la seva vida i no morir-se de cansament.

A les teves mans et deixem l'inici del joc que et permetrà controlar els moviments de l'arqueòloga Cindy Jones i guiar-la per les perilloses cambres del temple soterrat mitjançant les fletxes d'adreça del cursor, 'm' per PAUSAR el joc i així pensar per on seguir, i 'ESC' per SORTIR.

Després de provar el joc, l'hauràs de finalitzar. Aquí podràs aplicar el que has après sobre lògica de programació durant aquests mesos, modificant el codi font del joc que et proporcionem.

El codi base

El codi base (`RecercaArcaAlianca.zip` en el campus virtual) és el conjunt de classes que et proporcionem per fer funcionar el joc amb un mínim d'utilitats, inclosa la que conté el `main`. Fixa't en l'estil i la documentació del codi que et proporcionem i documenta apropiadament el teu codi.

Estructures de dades

El **temple** soterrat està format per varis nivells, cadascun d'ells té nombre qualsevol de cambres.

Cadascuna de les cambres en les que es desenvolupa el joc es divideix en cel·les de mida fixa, formant una quadricula de 17 x 25 cel·les, sobre una cel·la es pot situar un mur, una porta o res (espai lliure d'obstacles).

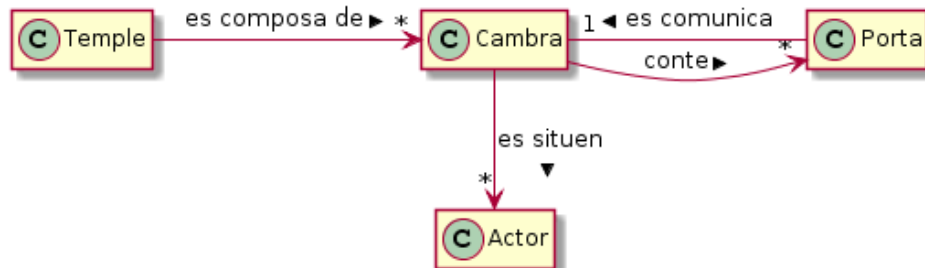
La definició de cada **cambra** es realitza mitjançant un fitxer de text pla que conté determinats caràcters: 'P' porta, 'M' mur, '_' terra. Cada caràcter correspon a una de les cel·les de la quadricula. Els fitxers amb les definicions de les cambres han de situar-se en el directori de "`src/recursos`".

A continuació podem veure el contingut del fitxer "`c0_0.txt`" que correspon a una de les cambres del programa base (17 files i 25 columnes):

```
M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M
M,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,M
M,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,M
P,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,M
M,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,M
M,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,M
M,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,M
M,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,P
M,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,M
M,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,M
M,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,M
M,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,M
M,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,M
M,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,M
M,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,M
M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M,M
```

Es pot veure que la cambra conté un mur perifèric i dues **portes**, una situada a l'esquerra i una altra situada a la dreta. Així com un passadís en forma de L invertida.

A cada cambra es situaran diversos objectes com diamants, un arca, soldats, etcètera, als que anomenarem **Actors**. El següent diagrama mostra com es relacionen tots els conceptes dels que hem tractat fins aquest moment.



A continuació teniu la pantalla de benvinguda del joc:




A sota tenim una captura en un moment del joc, corresponent al mapa que hem dibuixat anteriorment (c0_0.txt) :




Personatges




 **Cindy:** Aquest personatge és la intrèpida i aventurera arqueòloga de la nostra història. Esta disposada a trobar l'Arca de l'Aliança i evitar que els nazis s'apoderin d'ella.



 **Coronel Diettrich:** És el cap dels soldats nazis que es volen apoderar de l'Arca de l'Aliança i que persegueix a Cindy per trobar-la abans que ella. Lluitarà amb ella per tal d'apoderar-se de l'Arca.



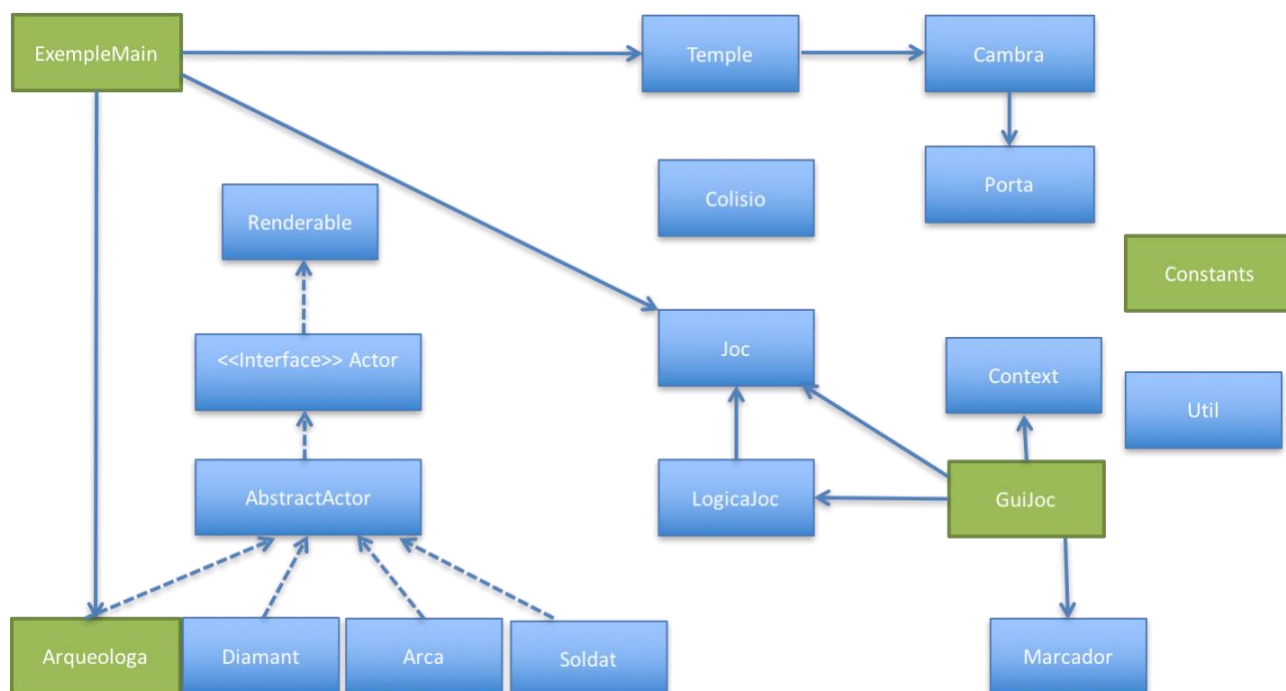
 **Soldats Nazis:** Són soldats altament entrenats per lluitar i sense escrúpols que obeeixen cegament al seu cap el Coronel Diettrich. Es troben situats a qualsevol part del temple i, no dubtaran en atacar-te per eliminar-te si et descobreixen!

Estructura del codi:

El joc consta de diferents parts funcionals:

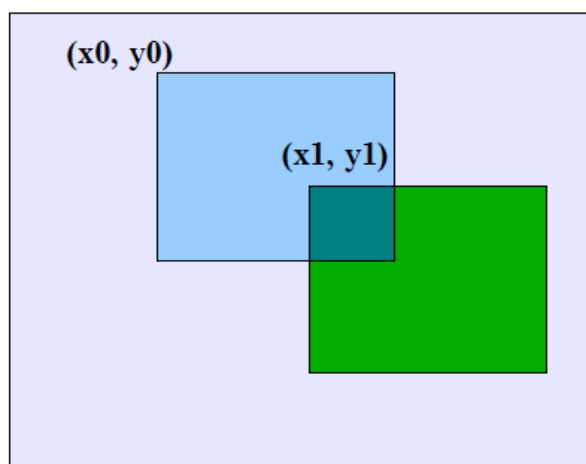
- **El motor del joc:** controla l'animació i coordina les accions que s'han de realitzar a cada moment del joc i es troba a la classe `Joc.java`.
- **La lògica del joc:** que estableix uns estats: Menú, Jugar, Fi del Joc (*GameOver*) i Sortida (*Exit*) i es troba a la classe `LogicaJoc.java`.
- **Els tipus de dades:** són les classes que representen les dades sobre les que es desenvolupa el joc: *Temple*, *Cambra*, *Joc*, *Marcador*, etc., i les classes que modelen cadascun dels objectes (*Actors*) que apareixen al joc: *Arqueologa*, *Diamant*, *Arca*, *Soldat*, etc.
- **Els recursos:** les imatges que s'usen per crear els escenaris. Podeu trobar-ne més a: <http://www.iconarchive.com>, <http://www.veryicon.com>, <http://www.iconseeker.com>

`http://images.google.com/images?q=sprite+sheet,...`). Les imatges es guardaran a la carpeta `src/recursos`



Descripció del funcionament intern del joc:

L'animació s'executa varies vegades per segon (la freqüència és d'uns 60 frames per segon). Per cada frame, s'executa l'animació que implica que els actors actualitzen la seva posició i estat, i poden interactuar amb altres actors, modificant algun dels seus atributs. Per exemple, l'objecte Diamant interacciona amb l'arqueòloga, augmentant la seva vida cada cop que "col·lisionen". Per detectar col·lisions, cada actor definirà la seva dimensió (ample i alt en píxels) com un rectangle situat en una posició X, Y. Dos actors en les posicions (x0, y0) i (x1, y1) col·lisionaran quan els seus rectangles delimitadors interseccionen en algun punt:



Quan es detecta alguna col·lisió d'aquest tipus, el mètode `tractarColisio(Colisio colisio)` de cada actor implicat és executat, de forma que el actor podrà modificar el seu estat, si és necessari. La definició completa de la interfície `Actor` és la següent:

```
public void inicializar();
public void setPosicioInicial(int x, int y);
public int[] getPosicioInicial();
public void setPosicio(int x, int y);
```

```

public int[] getPosicio();
public void actualitzar(Context context);
public Rectangle getLimits();
public void tractarColisio(Colisio colisio);
public float getForca();
public void setForca(float forca);
public int getEstat();
public void setEstat(int estat);

```

Consulta la documentació per conèixer més sobre cada mètode.

La protagonista, en un instant determinat del joc, està en un únic nivell i cambra. La classe `Temple` proporciona tots els mètodes necessaris per modificar la cambra i el nivell actual, així com per afegir i accedir a les diferents cambres.

```

public void addCambra(int nivell, int numCambra, Cambra cambra)
public void setNivell(int nivell)
public int getNivell()
public void setNumCambra(int cambra)
public int getNumCambra()
public Cambra getCambra()
public Cambra getCambra(int numNivell, int numCambra)

public Cambra[] getCambres(int numNivell)
public int getNumNivells()
public int getNumCambres(int nivell)

```

Com us podeu imaginar la classe `Temple` conté un atribut de tipus taula que guarda l'estructura de les cambres que hi ha a cada nivell:

```

// cambres es una matriu de nivells i les cambres que hi ha a cada nivell
// Aclariment: el nombre de cambres és el mateix en tots els nivells
private Cambra[][] cambres;
...
// en el constructor de Temple es realitza la següent instrucció:
cambres = new Cambra[numNivells][numCambresPerNivell];

```

Per comunicar les cambres entre sí utilitzarem la classe `Porta`, que conté la cambra i la porta amb la que es comunica. Les portes serveixen per a comunicar cambres del mateix nivell i també per comunicar cambres de diferents nivells. Podeu veure al mètode `actualitzar()` de la classe **Arqueologa** com la nostra protagonista es canvia de cambra fent servir les portes.

Exemple:

A mode d'exemple, es proporciona una classe principal `ExempleMain.java` que crea un joc amb 3 cambres i alguns objectes (Soldats) distribuïts per elles. Ni han tres tipus de soldats (veure `Soldat.java`): comandant, capità i soldat ras, cadascun amb una força de 20, 12 i 8 respectivament. Com es pot veure al mètode `distribuirSoldats`, inicialment es situen un nombre aleatori de soldats dels diferents tipus en cada cambra. Aquí es mostra cadascun dels mètodes que construeix una cambra inicial:

```

private Main() {

    Temple temple = new Temple(1, 3); //1 nivell, 3 cambres per nivell
    temple.addCambra(0, 0, crearCambra0Nivell0()); //crea cambra 0 del nivell 0
    ...

    Arqueologa cindy_jones = new Arqueologa();
    Cambra cam = temple.getCambra(0, 0);
    int[] p = cam.getPosicioCela(10, 10);
    cindy_jones.setPosicioInicial(p[0], p[1]);

    distribuirSoldats();
    ...
    //inicialització del joc
    Joc joc = new Joc(temple, cindy_jones);
    GuiJoc gui = new GuiJoc(joc);
}

```

```

}

private Cambra crearCambraONivell0() {
    Cambra cam = Util.carregarCambra("c0_0.txt");

    Porta porta = cam.getPorta(14, 24); //fila, columna
    porta.setNumNivellDesti(0);
    porta.setNumCambraDesti(1);
    porta.setPosicioCambraDesti(cam.getPosicioCela(1, 1));

    porta = cam.getPorta(0, 10);
    porta.setNumNivellDesti(0);
    porta.setNumCambraDesti(2);
    porta.setPosicioCambraDesti(cam.getPosicioCela(14, 10));

    return cam;
}

private void distribuirSoldats() {
    int [] num = {1,3,6}; //nombre de soldats a distribuir de cada tipus

    for (int i = 0; i < Soldat.TipusSoldat.values().length ; i++) {
        for(int k = 0; k < num[i] ; k++){
            int nivell = (int) (Math.random() * temple.getNumNivells());
            int numCambra = (int) (Math.random() * temple.getNumCambres(nivell));
            Cambra cam = temple.getCambra(nivell, numCambra);
            int[] cela = obtenirCelaLliure(cam);
            Soldat s = new Soldat(Soldat.TipusSoldat.values()[i]);

            int[] posicio = cam.getPosicioCela(cela[0], cela[1]);
            s.setPosicioInicial(posicio[0], posicio[1]);
            cam.addActor(s);
        }
    }
}

```

Compilació i Execució des de Netbeans

Descomprimeix el fitxer `RecercaArcaAlianca.zip` que pots trobar al CV, obre el projecte Netbeans i executa'l.

Implementació

Seguiu els 5 passos descrits a continuació per a implementar el joc.

PAS 1 – CLASSE INICIAL: Crea un arxiu amb el nom **Practica.java** que contingui un mètode **main**. Aquesta classe serà de l'estil de la classe de l'exemple *ExempleMain.java* que es proporciona. Fixa't que hauràs de compilar l'arxiu `Practica.java` i no `ExempleMain.java`, como fins a aquest moment. Elimina l'arxiu `ExempleMain.java` del projecte de Netbeans. Elimineu només aquest arxiu, la resta del fitxers `.java` es queden com estan.

PAS 2 – CONFIGURACIÓ DEL TERRITORI: Crea un temple amb com a mínim dos nivells i tres cambres per nivell. Per a cada cambra hauràs de crear un fitxer `.txt` amb la definició de la cambra. Si ho desitges, pots copiar l'arxiu "c0_2.txt" que defineix una cambra buida i el modifiques segons desitgis. Recorda que les cambres es situen en el directori `src/recursos`.

PAS 3 – CONFIGURACIÓ DE LES CAMBRES: A la classe `Practica.java`, crea les cambres i les portes que comuniquen les cambres entre sí segons desitgis. Tingues en compte que la informació de l'arxiu `.txt` que defineix la cambra ha de ser coherent amb les instruccions que realitzes a la classe `Practica.java`

PAS 4 – DIAMANTS I ARCA: Crea un nou actor de tipus **Diamant**. Cada diamant té un color i un valor d'energia associat, el color pot ser *vermell*, *blau*, *verd*, *groc* i *taronja*. L'energia inicial de cada

diamant està en funció del color. Un diamant vermell inicialment té una energia de 15 unitats, un diamant blau té una energia de 10 unitats, els diamants de color verd, groc i taronja tenen una energia de 5 unitats. Si la nostra arqueòloga col·lisió, recull el diamant i augmenta la seva vida en funció del color del diamant. Per a facilitar la depuració del codi, imprimeix a consola la vida total que té la nostra arqueòloga.

Afegeix a `Constants.java` el següent `enum` anomenat **ColorDiamant**. Aquest `enum` permet manipular els diamants de cada color. Vegeu l'atribut `color` de la classe **Diamant**, és de tipus `ColorDiamant`:

```
enum ColorDiamant {
    VERMELL(0,15),
    BLAU(1,10),
    VERD(2,5),
    GROC(3,5),
    TARONJA(4,5);
    private final int energia;
    private final int colorNum;

    ColorDiamants(int color, int energia){
        this.colorNum = color;
        this.energia = energia;
    }
    public int getEnergia() {
        return this.energia;
    }
    public int getColorNum(){
        return this.colorNum;
    }
}
```

A continuació teniu la definició de la classe **Diamant**. Heu d'implementar el constructor i el mètode `tractarCol·lisió`. En aquest PAS 4 només heu d'implementar el constructor. Podeu veure que la classe **Diamant** és semblant a la classe **Soldat**, la principal diferència són els atributs i el mètode `actualitzar`. Aquest mètode està buit a la classe **Diamant** perquè un diamant no es mou.

Al constructor, per a carregar l'imatge d'un diamant heu d'utilitzar la crida a `Color`. Per exemple:

`image = Util.carregarImatge("Diamond.png", new Color(255, 0, 0, 0))`, carrega un diamant de color vermell. Els tres primers paràmetres de `Color` indiquen els valors R (red), G (green), B (blue). El vermell s'indica amb (255, 0, 0, 0), el verd amb (0, 255, 0, 0), el blau amb (0, 0, 255, 0), el groc amb (255, 255, 0, 0) i el taronja amb (255, 128, 64, 0).

```
public class Diamant extends AbstractActor {
    //atributs de la classe
    private Image image;
    private final int amplada = 20;
    private final int alcada = 37;
    private Constants.ColorDiamant color;

    public Diamant(Constants.ColorDiamant colorDiamant){
        // AQUI EL TEU CODI
    }
    public Constants.ColorDiamant getColor() {
        return color;
    }
    public void setColor(Constants.ColorDiamant color){
        this.color = color;
    }

    public void actualitzar(Context context) {
        // no fem res, es estatic (no se mou)
    }
    public Rectangle getLimits() {
        // es important per tractar les colisions des de la classe GuiJoc al metode actualitzarJoc
        return new Rectangle(getX(), getY(), amplada, alcada);
    }
    public void tractarColisio(Colisio colisio) {
        Actor actor = colisio.getActor();
    }
}
```



```

if (actor instanceof Arqueologa) {
    Arqueologa arqueologa = (Arqueologa) actor;
    // AQUI EL TEU CODI
}
}

public void render(Graphics2D g) {
    //Com dibuixar a la pantalla principal
    g.drawImage(image, getX(), getY(), observer);
}
}

```

Distribueix 10 diamants aleatòriament pel temple, 2 vermells, 3 blaus, 3 verds, 3 grocs i 2 de color taronja.

Crea un nou actor de tipus **Arca** (nou fitxer **Arca.java**) i situa l'Arca de l'Aliança a l'última cambra de l'últim nivell. Per a crear l'arca pots inspirar-te en la classe **Diamant.java**. Com a imatge de l'arca utilitza el recurs *arca32.png*.

La distribució dels diamants i la colocació de l'arca has de fer-lo a *Practica.java*. Per allò, crea els mètodes *distribuirDiamants()* i *posicionarArca()*. Pots inspirar-te en el mètode **DistribuirSoldats.java**.

PAS 5 – FINAL PER DIAMANTS: modifica la col·lisió amb els diamants (mètode *tractarColisio()* de la classe **Diamant**) i guarda un diamant de cada tipus (hi ha un màxim de 5 tipus) que l'arqueòloga es va trobant. Per fer-ho:

- Afegeix a la classe **Arqueologa** els atributs *diamantsTrobats*, *arcaTrobada* i *vida*. Obligatoriament *diamantsTrobats* ha de ser un array que permetrà saber si l'arqueologa ha trobat un diamant de cada color (penseu el tipus dels elements de l'array). *arcaTrobada* és un boolean que permet saber si l'arqueologa ha trobat l'arca. *vida* es un valor decimal que conté la vida de l'arqueologa, inicialment a 100.0. El jugador perd el joc quan la vida de Cindy és inferior a 0. Una vegada heu afegit l'atribut *vida*, heu de descomentar el codi del mètode *calcularNivellVida()*, que decrementa la vida de l'arqueologa amb el pas del temps. Recordeu inicialitzar aquests atributs de l'arqueologa.
- Implementa el mètode *addDiamant(Diamant diamant)* de la classe **Arqueologa**, que rep el diamant i, si no existeix un diamant d'aquest color anota que ja en té un i augmenta la vida de Cindy. Si ja existeix un diamant d'aquest color, només s'augmenta la vida de Cindy segons el tipus de diamant.
- Implementa el mètode *tractarColisio()* de la classe **Diamant**. Aquest mètode s'executarà quan l'arqueologa col·lisió amb un diamant, serà dins de *tractarColisio()* d'on hem de cridar *addDiamant()*. Recordeu que en col·lisionar l'arqueologa amb el diamant, aquest ha de desaparèixer. Revisa abans el mètode *tractarColisio()* de la classe **Soldat**.

Fes que el **joc finalitzi** quan la protagonista trobi l'Arca de l'aliança i pugui afagar-la. Per agafar l'arca, l'arqueologa ha d'haver **trobat almenys un diamant de cada color** i la seva vida ha de ser superior a un valor predeterminat (per exemple 40). S'ha d'implementar el mètode *boolean haTrobatElsDiamants()* de la classe **Arqueologa** per comprovar si ha trobat (o no) un diamant de cada color. Altres mètodes a implementar són: *haMort()* de la classe **Arqueologa** per a saber si la protagonista ha mort, i el mètode *haTrobatArcaPerduda()* per a saber si la protagonista ha trobat (ha col·lisionat) amb l'arca.

Fixeu-vos que el final del joc es controla des del mètode `actualitzarJoc()` de la classe **GuiJoc**. Mireu també com la classe **LogicaJoc** controla l'estat del joc.

En el cas que l'arqueòloga **col·lisió amb un soldat de qualsevol tipus, és possible que perdi un diamant**. Quan hi ha col·lisió amb el soldat, aleatòriament es decideix si perd o no un diamant i, en el cas de perdre'l, es decideix també de forma aleatòria quin diamant es perd de tots els que ella té (en cas de no tenir cap, es quedarà igual). En totes les col·lisions de l'arqueòloga amb els soldats, la vida de l'arqueòloga es redueix tantes unitats com força tingui el soldat. Un comandant té una força de 20, un capità té una força de 12 i un soldat ras té una força de 8. Per a implementar aquesta funcionalitat heu d'implementar el mètode `tractarColisio()` de la classe **Soldat**, i el mètode `perdreDiamant()` de la classe **Arqueologa**.