

APAC1. Dibuixant Figures

Joan Gerard Camarena Estruch

The image features a purple diagonal banner with white text. The text reads "Accès à Dades" on the top line and "Pràctiques" on the bottom line. The banner is positioned over a yellow triangle. Inside the yellow triangle is a blue digital interface with various icons and data points, suggesting a technical or data-related theme.



Continguts

1 Descripció de l'activitat. Que es demana	3
1.1 Punt de partida	3
1.2 Opcions implementades	4
1.3 Noves funcionalitats a implementar	5
1.4 Estructura del projecte	9
2 Elements del projecte. Tasques a realitzar	10
2.1 La classe FileManager	11
3 Lliurament	13
3.1 Fluxe de treball	14

1. Descripció de l'activitat. Que es demana

En aquesta primera APAC d'Accés a dades, anem a implementar la part de persistència amb fitxers d'un projecte de figures geomètriques semblant al realitzat a la unitat introductòria. Aquest programa s'encarrega de crear una sèrie de figures, de distint tipus i ens ofereix la possibilitat de representar-les gràficament. A l'igual que tots els programes, oferirem la possibilitat de exportar i importar figures ja creades. Els formats de guardar/carregar imatges seran 4, com es veurà a continuació: text, binari, json i xml. Ademès, farem compatible el format xml en un famós programa de disseny vectorial, [Inkscape](#).

1.1. Punt de partida

Se us proporciona un fitxer comprimit amb el projecte en Gradle [APAC1_AD](#), amb la implementació de la jerarquia de figures i diverses utilitats.

El projecte s'ha creat amb `gradle init`, que ens crea una menuda estructura de directoris i fitxers, com els llançadors gradlew i gradlew.bat (que no utilitzarem).

Per tal de construir el projecte farem:

```
$ gradle build  
BUILD SUCCESSFUL in 1s  
5 actionable tasks: 5 up-to-date
```

I per llançar-la, podem fer-ho amb:

```
$ gradle run --console plain
```

L'opció `--console plain`, fa que no mostre contínuament un molest missatge indicant que s'està executant.

Si desitgem passar-li arguments a l'aplicació (admet dos valors que són les dimensions del llençol de dibuix que podem generar al final de la mateixa), podeu fer-ho amb:

```
gradle run --console plain --args="500 500"
```

Altra alternativa és crear a l'arrel del projecte el següent fitxer `gradle.properties`:

```
org.gradle.console=plain
```

Una vegada en execució, el nostre programa mostrarà el següent prompt, esperant que li introduïm dades:

```
# Figura:
```

Anem a veure que ens ofereix

1.2. Opcions implementades

Que admet les següents opcions:

- `dimensions ample alt`: Que admint ample i alt com a valors numèrics, i que estableix les dimensions del llençol de treball.
- `cercle x y radi #color`: Que afig un cercle a l'escena, i admet tres paràmetres enters, amb la posició i el radi, i un valor de color, en format `#RRGGBB`. (Sent RR un valor hexadecimal amb la quantitat de roig, GG un hexadecimal amb la quantitat de verd, i BB hexadecimal amb la quantitat de blau).
- `rectangle x1 y1 alt ample #color`: Que afig una figura rectangle, des de les posicions (x_1, y_1) fins a (x_2, y_2) , en el color indicat.
- `linia x1 y1 x2 y2 #color`: Que dibuixa una línia, de grossor preestablert 3 pixels, entre el punt (x_1, y_1) i el (x_2, y_2) , del color indicat.
- `draw`: Que obre una finestra amb `JavaFX` i dibuixa l'escena que hem creat. Quan tanquem aquesta finestra, l'aplicació finalitzarà.

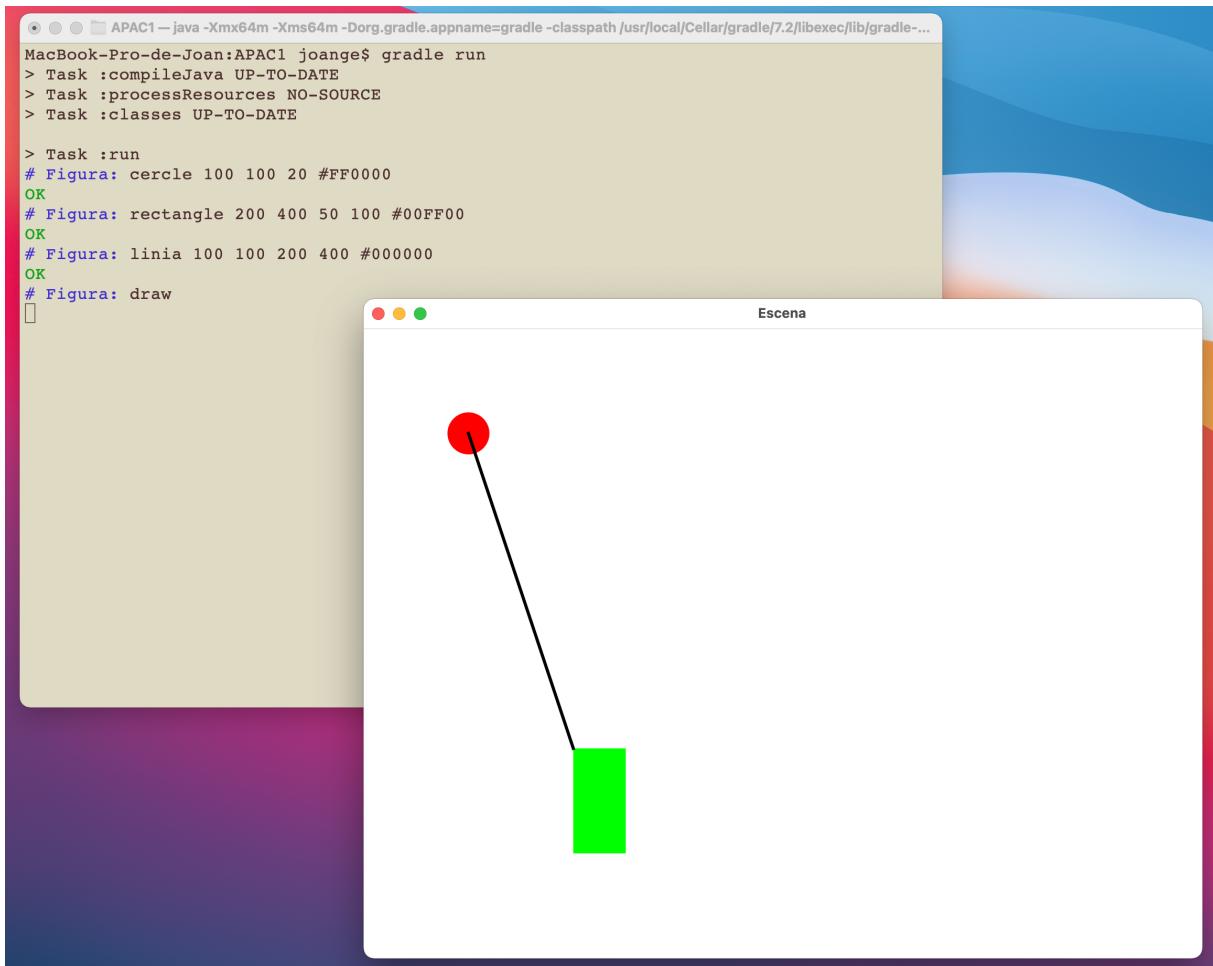
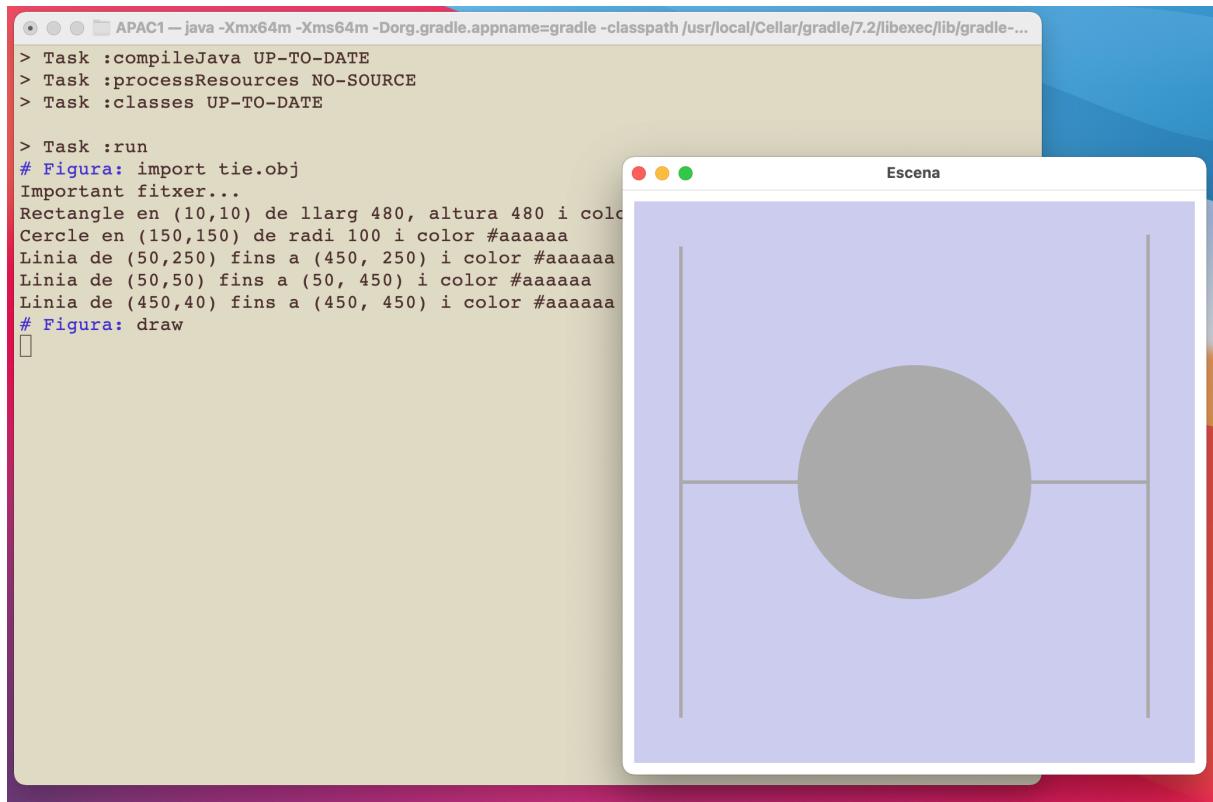


Figura 1: FEnt dibuixets

1.3. Noves funcionalitats a implementar

La nostra tasca consistirà en realitzar la implementació de dos ordres més:

- **import** [fitxer.txt | fitxer.obj]: Que permetrà importar fitxers d'instruccions en format text o en format objecte (seriats).
- **export** [fitxer.txt | fitxer.obj | fitxer.svg | fitxer.json]: Que permetrà exportar el fitxer a format de text d'instruccions, seriar els objectes, obtindre un **svg**-tipus especial d'XML-, i en format JSON.

**Figura 2:** Un Tie-Fighter

Veiem un exemple dels diferents fitxers (el format obj no el mostrem, per raons òbvies):

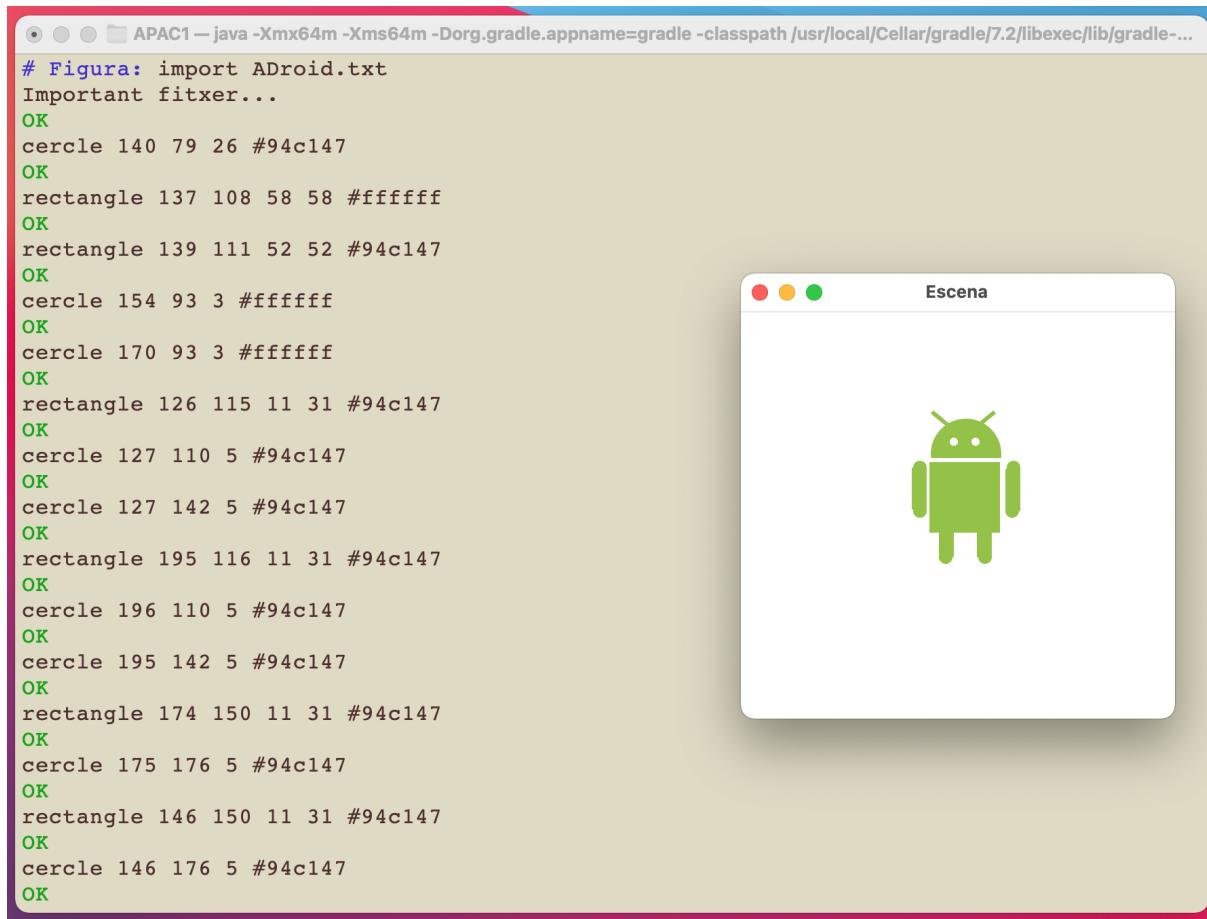
■ **Fitxer en format txt:**

```
dimensions 500 500
rectangle 10 10 480 480 #ccccce
cercle 250 250 100 #aaaaaa
linia 50 250 450 250 #aaaaaa
linia 50 50 50 450 #aaaaaa
linia 450 40 450 450 #aaaaaa
```

■ **Fitxer en format SVG (XML):**

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg height="500" width="500">
  <rect fill="#ccccce" height="480" width="480" x="10" y="10"/>
  <circle cx="250" cy="250" fill="#aaaaaa" r="100"/>
  <line stroke="#aaaaaa" stroke-width="3" x1="50" x2="450" y1="250" y2="250"/>
  <line stroke="#aaaaaa" stroke-width="3" x1="50" x2="50" y1="50" y2="450"/>
```

```
<line stroke="#aaaaaa" stroke-width="3" x1="450" x2="450" y1="40" y2="450"/>
</svg>
```



```
# Figura: import ADroid.txt
Important fitxer...
OK
cercle 140 79 26 #94c147
OK
rectangle 137 108 58 58 #ffffff
OK
rectangle 139 111 52 52 #94c147
OK
cercle 154 93 3 #ffffff
OK
cercle 170 93 3 #ffffff
OK
rectangle 126 115 11 31 #94c147
OK
cercle 127 110 5 #94c147
OK
cercle 127 142 5 #94c147
OK
rectangle 195 116 11 31 #94c147
OK
cercle 196 110 5 #94c147
OK
cercle 195 142 5 #94c147
OK
rectangle 174 150 11 31 #94c147
OK
cercle 175 176 5 #94c147
OK
rectangle 146 150 11 31 #94c147
OK
cercle 146 176 5 #94c147
OK
```

Figura 3: Dibuixant android

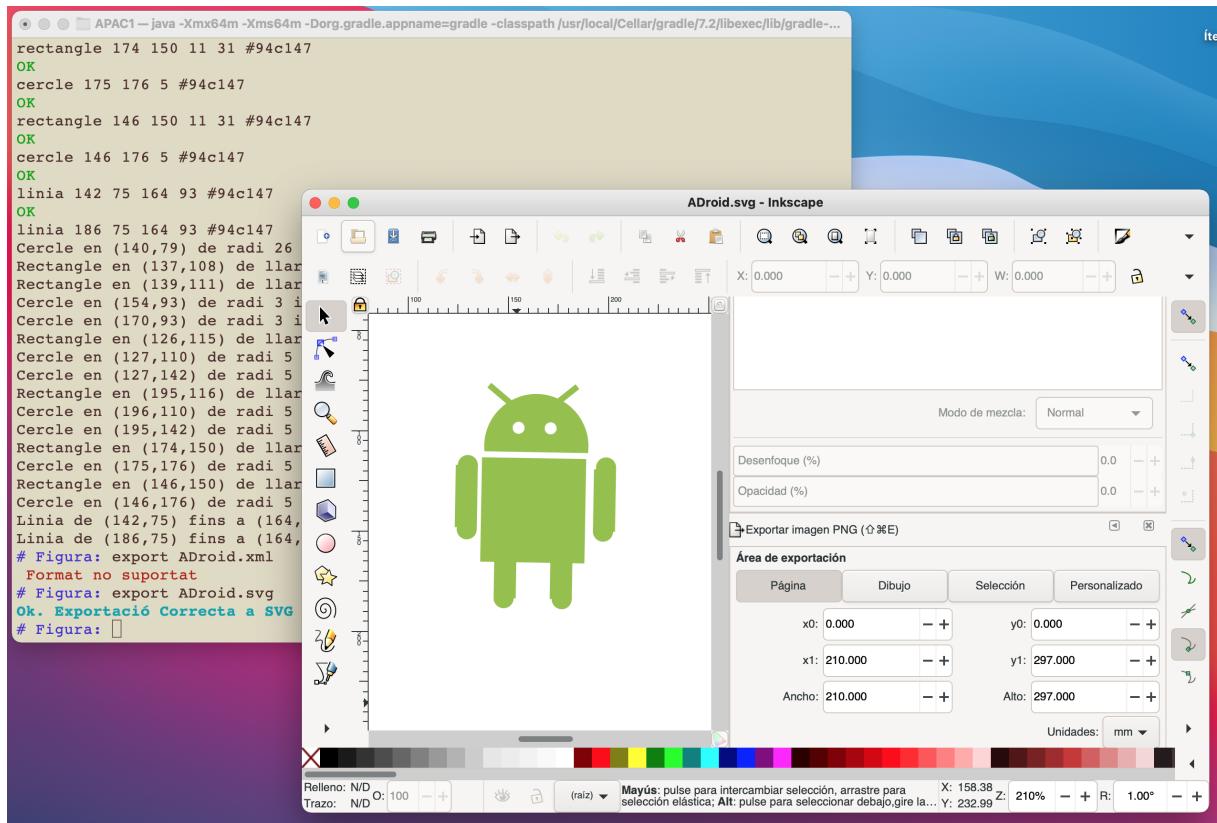


Figura 4: Obrint amb Inkscape

Aquests fitxers podreu obrir-los en aplicacions de disseny vectorial, com [Inkscape](#). Feu la prova per comprovar que heu realitzat bé la implementació!

- **Fitxer en format JSON:**

```
{"escena": {
    "figures": [
        {"rectangle": {
            "x": "10",
            "width": "480",
            "y": "10",
            "fill": "#cccccc",
            "height": "480"
        }},
        {"cerkle": {
            "r": "100",
            "cx": "250",
            "cy": "250",
            "fill": "#aaaaaa"
        }},
        {"linia": {
            "x": "10",
            "y": "10",
            "x2": "250",
            "y2": "250"
        }}
    ]
}}
```

```

        "y1": "250",
        "x1": "50",
        "y2": "250",
        "x2": "450",
        "stroke-width": "3",
        "stroke": "#aaaaaa"
    },
    {"linia": {
        "y1": "50",
        "x1": "50",
        "y2": "450",
        "x2": "50",
        "stroke-width": "3",
        "stroke": "#aaaaaa"
    }},
    {"linia": {
        "y1": "40",
        "x1": "450",
        "y2": "450",
        "x2": "450",
        "stroke-width": "3",
        "stroke": "#aaaaaa"
    }}
],
"width": 500,
"height": 500
}
}

```

1.4. Estructura del projecte

L'estructura del projecte és:

```

.
|-- build.gradle
|-- gradle
    |-- wrapper
        |-- gradle-wrapper.jar
        |-- gradle-wrapper.properties
|-- gradlew
|-- gradlew.bat
|-- settings.gradle
|-- src
    |-- main
        |-- java
            |-- com
                |-- ieseljust
                    |-- ad
                        |-- figures
                            |-- App.java

```

```
-- cercle.java  
-- escena.java  
-- figura.java  
-- FileManager.java  
-- HexColorValidator.java  
-- Linia.java  
-- punt.java  
-- Rectangle.java  
-- resources
```

Dels fitxers de la carpeta arrel, ens interessa el fitxer `build.gradle`, que conté la informació sobre la construcció del projecte. Aquest ja inclou les llibreries per a la gestió dels gràfics, i la llibreria `org.JSON`, **pel que no farà falta modificar-lo**.

2. Elements del projecte. Tasques a realitzar

El que ens interessa, és tot allò ubicat dins de `src/main/java/com/ieseljust/ad/figures`. Aci tenim:

- **App.java:** Es tracta del punt d'entrada a l'aplicació, qui s'encarrega de mostrar el prompt i crear l'Escena, formada per la llista de figures. **En principi, en aquesta classe no caldrà modificar res.**
- **Figura.java:** Es tracta de la superclasse a partir de la qual es deriven les classes cercle, rectangle i línia. Emmagatzema la posició (amb un objecte de tipus punt) i el color de la figura. En aquesta classe no caldrà fer grans canvis, però si que haurà de definir com a mètodes abstractes, els que anem afegint a la resta de figures comuns a la mateixa classe Figura. A més, caldrà importar algunes llibreries, per a la seriació i la manipulació de fitxers XML i JSON.
- **Cercle.java:** Especialització de figura que dibuixa un cercle. A més dels atributs heretats de figura, incorpora un nou atribut que representa el radi. Haurà d'implementar alguns mètodes addicionals per obtenir una representació de la figura en diferents formats (text, XML, JSON), i incorporar les llibreries pertinents.
- **Linia.java:** Especialització de figura que dibuixa una línia de grossor predefinit 3 píxels. A més dels atributs heretats de figura, incorpora un nou atribut de la classe punt (anomenat `vector`), que diu el punt final de la línia. Haurà d'implementar alguns mètodes addicionals per obtenir una representació de la figura en diferents formats (text, XML, JSON), i incorporar les llibreries pertinents.
- **Rectangle.java:** Especialització de figura que dibuixa un rectangle. A més dels atributs heretats de figura, incorpora dos nous atributs, que són l'alt i l'ample. Haurà d'implementar alguns mètodes addicionals per obtenir una representació de la figura en diferents formats (text, XML, JSON), i incorporar les llibreries pertinents.

- **Punt.java:** Classe auxiliar per als punts de les figures, i no herata de figura. En principi, **no haurà d'afegir cap mètode nou**, la única cosa que caldrà modificar és (per a quan emmagatzem els objectes en format binari) que li haurem d'indicar que la classe es pot seriar (ja que en seriar un objecte, comença a seriar objectes recursivament fins arribar als tipus bàsics, i ha de saber, per tant, que punt també és serialable.)
- **HexColorValidator.java:** Classe auxiliar que verifica que els colors introduits tinguin un format correcte. **No s'ha de modificar.**
- **Escena.java:** Representa l'escena en sí. Té un parell d'atributs, que podem modificar i consultar (ens farà falta en algun moment), i una llista de figures, que haurem de recórrer per tal de guardar-la en disc.
- **FileManager.java:** Classe que s'encarregarà de la gestió de l'emmagatzemament. Serà el que haurem d'implementar, i per això li dedicarem un apartat.

2.1. La classe FileManager

La classe `FileManager` gestionarà l'emmagatzemament i recuperació de fitxers en diferents formats. Haurà d'implementar els següents mètodes:

Mètode `Exists`

```
public Boolean Exists(String file)
```

Retornarà un valor lògic indicant si el fitxer donat existeix o no.

Mètode `importFromText`

```
public Escena importFromText(String file)
```

Que importa una escena en format text, en la formatació que s'indica al principi d'aquest document. A la carpeta imgs, disposeu d'un parell d'imatges per provar. Com veieu, rep el camí al fitxer en format `String`, i retornarà un objecte de tipus `Escena`.

Tingueu en compte que, a més de les figures, pot aparéixer l'ordre `dimensions`, que ens indicarà que haurem de modificar les dimensions de l'Escena.

Mètode `importFromObj`

```
public Escena importFromObj(String file)
```

Que importa una escena seriada en format d'objectes, en la formatació que s'indica al principi d'aquest document. A la carpeta imgs, disposeu d'un parell d'imatges per provar. Com veieu, rep el camí al fitxer en format `String`, i retornarà un objecte de tipus `Escena`.

Mètode `exportText`

```
public Boolean exportText(Escena escena, String file)
```

Aquest mètode exportarà una escena donada a un fitxer de text, en el format especificat més amunt, per poder llegir-les amb `importFromText`.

El mètode rebrà l'objecte de tipus `Escena`, i un `String` amb el nom del fitxer a guardar, amb extensió `.txt`.

Per a la implementació d'aquest mètode, us resultarà útil implementar un mètode anomenat `getAsString` en cada tipus de figura, i que retorne la pròpia figura en el format que interessa. D'aquesta manera, per exportar l'escena, recorrerem les diferents figures i anirem obtenint la representació de cadascuna.

Mètode `exportObj`

```
public Boolean exportObj(Escena escena, String file)
```

Aquest mètode exportarà una escena donada a un fitxer seriat d'objectes, per poder llegir-les amb `importFromObj`.

El mètode rebrà l'objecte de tipus `Escena`, i un `String` amb el nom del fitxer a guardar, amb extensió `.obj`.

Tingueu en compte, que caldrà també indicar que la classe `Figura` i la classe `Punt` són seriabls.

Mètode `exportSVG`

```
public Boolean exportSVG(Escena escena, String file)
```

Aquest mètode exportarà una escena donada a un fitxer SVG que és un format basat en XML. Al principi del document teniu un exemple d'aquest tipus de fitxer. Una vegada exportat, podreu obrir-lo amb alguna aplicació de disseny vectorial, com l'`Inkscape`.

Veiem alguns detall del format:

- L'etiqueta `<?xml version="1.0" encoding="UTF-8" standalone="no"?>` del principi s'inclou automàticament quan creem el document, pel que no cal fer res perquè aparega.
- L'element arrel és del tipus `<svg>`, i té els atributs `height` i `width`, que caldrà incorporar emb el mètode `setAttribute()`.
- Aquest arrel contindrà una llista d'elements de tipus geomètric: `rect`, `circle` o `line`, amb diferents atributs. Mireu l'exemple del principi del document per indicar el nom dels atributs correctament.
- Per a la conversió de la classe `Cercle`, cal tindre en compte que, tal i com representem a l'aplicació la informació, per al bon posicionament dels cercles en el format SVG, les posicions

`cx i cy` (atributs de `<circle>`) s'obtindran amb `this.posicio.getX() +this.radi i this.posicio.getY() +this.radi`

El mètode rebrà l'objecte de tipus `Escena`, i un `String` amb el nom del fitxer a guardar, amb extensió `.svg`.

Per a la implementació d'aquest mètode, us resultarà útil implementar un mètode anomenat `getAsXML` en cada tipus de figura, i que retorne la pròpia figura en el format que interessa. Per a la implementació d'aquest mètode a les classes de figures, caldrà que li passem una referència a l'objecte `Document` amb el que estem creant el document a la classe `FileManager`, ja que tots els elements del mateix document s'han de crear a partir del mateix Factory. En cas contrari, obtindríem l'error:

```
org.w3c.dom.DOMException: WRONG_DOCUMENT_ERR: A node is used in a different document than the one that created it.
```

Recordeu també que per a aquest mètode caldrà fer alguns imports en les diferents classes, tant per a la manipulació de fitxers XML com per a la transformació a text.

Mètode `exportJSON`

```
public Boolean exportJSON(Escena escena, String filename)
```

Aquest mètode exportarà una escena donada a un fitxer JSON. Al principi del document teniu un exemple d'aquest tipus de fitxer.

El mètode rebrà l'objecte de tipus `Escena`, i un `String` amb el nom del fitxer a guardar, amb extensió `.json`.

Per a la implementació d'aquest mètode, us resultarà útil implementar un mètode anomenat `getAsJSON` en cada tipus de figura, i que retorne la pròpia figura en el format que interessa. Recordeu que anem a utilitzar una llibreria externa, que ja va referenciada com a dependència del projecte Gradle, només haurem de tindre en compte de fer els `imports` corresponents.

3. Lliurament

Per al lliurament, calrà entregar un fitxer comprimit amb el projecte preparat per construir. Es recomana que feu un `gradle clean` abans de fer l'entrega, per netejar el projecte i deixar-lo preparat per a poder-lo construir de nou.

Haureu d'entregar la ruta a un repositori de github on teniu la tasca

3.1. Fluxe de treball

Es recomana que l'ordre d'implementació dels mètodes de la classe `FileManager` siga:

- `importFromText`, `exportText` (per tal de poder importar i exportar imatges i no haver de dibuixar l'escena cada vegada que volgams fer una prova);
- `exportObj`, `importObj` (per treballar la importació i exportació d'objectes), i finalment
- `exportSVG` i `exportJSON`;
 - Anar implementant els diferents mètodes a les figures segons anem necessitant-los (`getAsText`, `getAsXML`, `getAsJSON`)

Llicència



Joan Gerard Camarena Estruch & Jose Alfredo Murcia

Reconeixement - NoComercial - CompartirIgual (by-nc-sa):

No es permet un ús comercial de l'obra original ni de les possibles obres derivades, la distribució de les quals s'ha de fer amb una llicència igual a què regula l'obra original