FINAL GENER

Assignatura: **Disseny de Software**

Data: Dimarts, 11 de gener de 2022 Curs: **2021/22**

UNIVERSITAT DE BARCELONA

Nom:_____**DNI:**_____

Aula: _____ **Fila:**_____**Columna:**_____

Temps total ESTIMAT 2:00 h

PART I - DISSENY DE MODEL DE DOMINI: 4 punts sobre 10. Temps estimat: 45 min

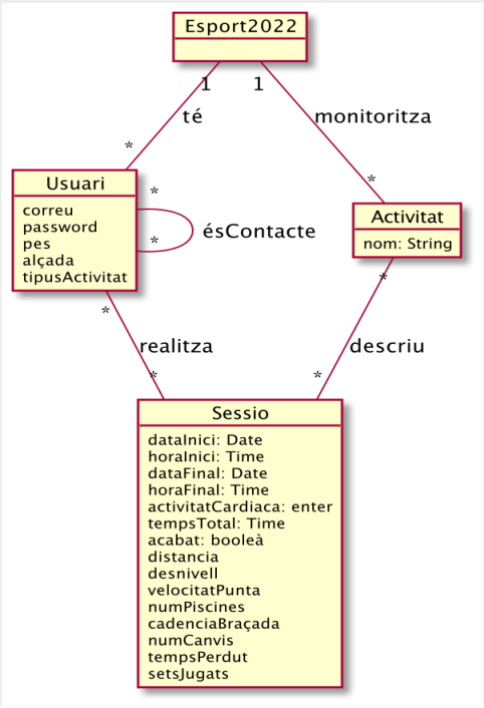
Es vol dissenyar una aplicació per a poder monitoritzar l'esport que realitza un usuari i així també poder-ho compartir entre els seus contactes.

L'usuari estarà autenticat a l'aplicació amb el seu correu i una contrasenya. En un primer moment, es demanaran les dades generals, com l'edat, el pes, l'alçada i el tipus d'activitat esportiva que acostuma a fer (baixa, moderada o alta). Cada vegada que l'usuari comenci una activitat física, podrà seleccionar el tipus d'activitat. En aquesta versió inicial, l'aplicació permet seleccionar entre les activitats de running, bicicleta, natació, tenis i esquí, tot i que en una versió posterior es pretén incloure'n més.

Quan es realitza una activitat es mesura el temps i l'activitat cardíaca. A part, en les activitats de running, bicicleta i esquí es mesura també la distància recorreguda, el desnivell acumulat i la velocitat punta. Per la bici, es vol contar també el nombre de vegades que s'ha canviat de marxa. Per a l'esquí també es vol mesurar el temps perdut en els telecadires. En cas del tenis, es mesuren els números de jocs jugats i en el cas de la natació, es conten el nombre de piscines fetes i la cadència de la braçada.

L'usuari pot aturar temporalment l'activitat i activar-la. Es vol saber també quan s'acaba del tot, per poder calcular totes les mesures i així poder-les compartir amb els seus contactes.

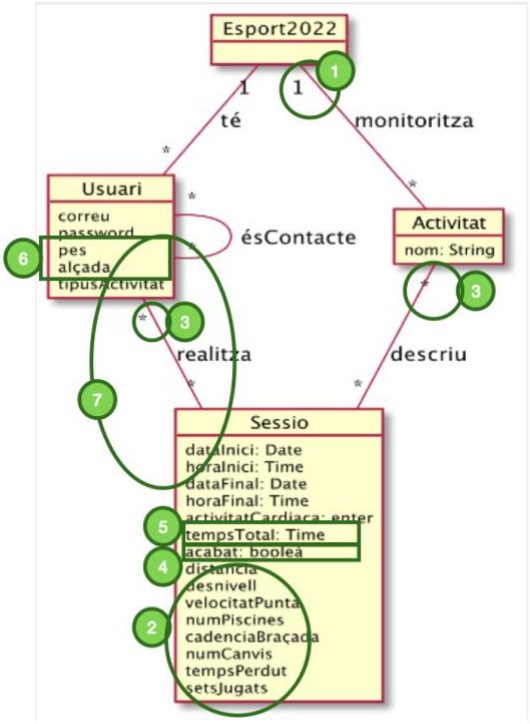
Un dissenyador de software novell ha començat a fer el següent Model de Domini, però no sap com seguir i tampoc ha inclòs la part de poder saber la llista de sessions compartides amb un contacte concret.



Analitza el model de domini que ha obtingut el dissenyador i marca en el seu model de domini **els principals errors**, si és que en detectes, explica **per què són errors** i proposa els **canvis en un nou model de domini**.

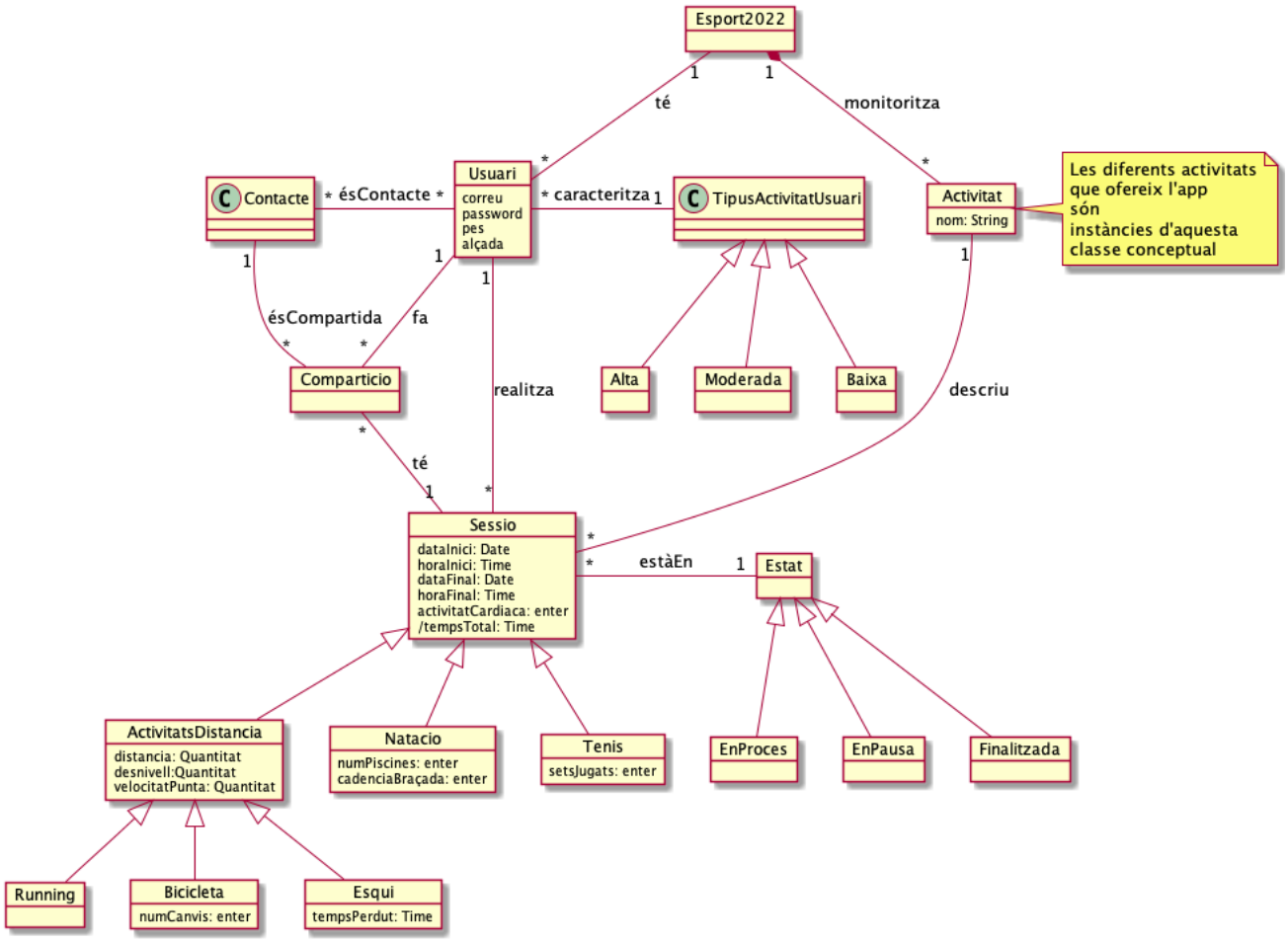
Encercla les parts del model de domini posant-hi un número al costat per poder explicar a la teva resposta per què estan malament o incompletes.

En el model final has d'indicar els tipus dels atributs de les classes conceptuals. **Indica si fas servir classes d'especificació, herències, i/o composicions/agregacions i justifica per què.**

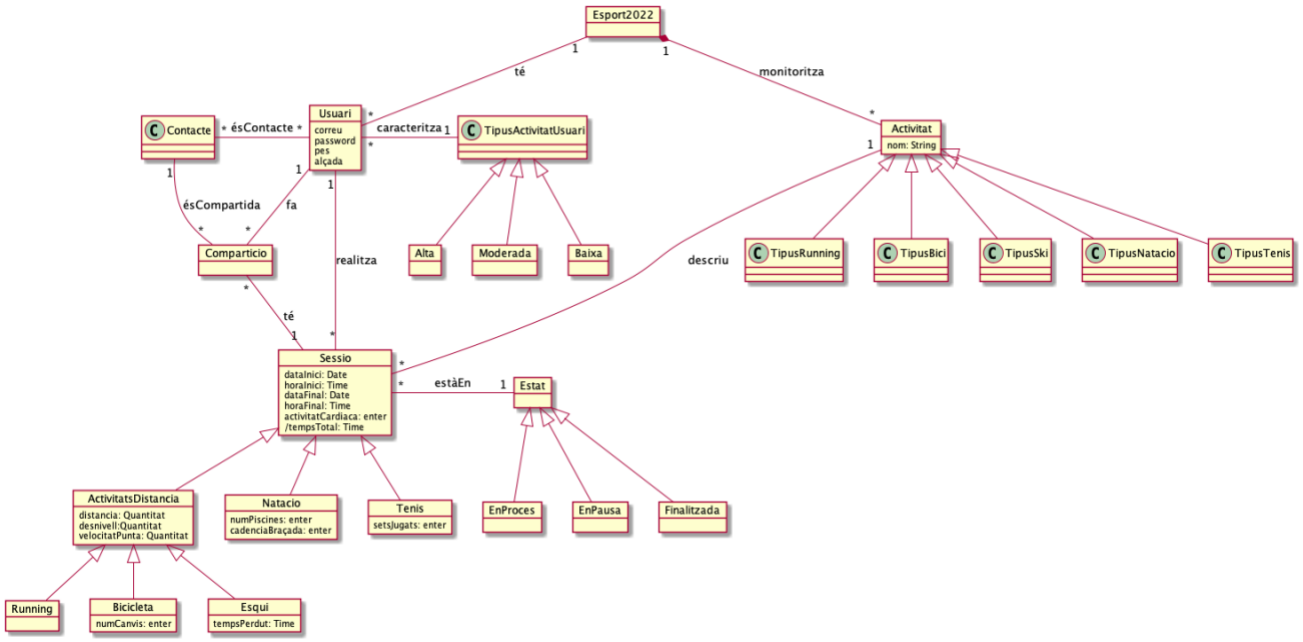


1. Es pot considerar una Composició, ja que l'aplicació Esport2022 sense activitats no té sentit
2. Aquests atributs son diferents per cada tipus d'activitat. Això no expressa prou els diferents tipus d'atributs que poden tenir sessions relatives a diferents activitats. Això porta a dissenyar una herència
3. Cardinalitat incorrecta en ambdós casos, han de ser 1 ja que una sessió només es refereix a una activitat i és només d'un usuari concret
4. L'atribut acabat no modela bé el fet d'haver començat, estar en pausa i ja haver finalitzat l'activitat. Per això es fa una classe Estat
5. L'atribut tempsTotal pot ser derivat per que es pot calcular a partir dels atribts dataInici, horaInici, dataFinal, horaFinal
6. Els atributs pes i alçada han de ser de tipus Quantitat. Falta també l'atribut edat
7. No s'ha modelat el fet de compartir la sessió amb altres usuaris, fet que no permet contestar a la consulta: *"saber la llista de sessions compartides amb un contacte concret"*

Una possible Solució:



Una altra possible solució: (se n'han admès d'altres)



FINAL GENER

Assignatura: **Disseny de Software**

Data: Dimarts, 11 de gener de 2022 Curs: **2021/22**

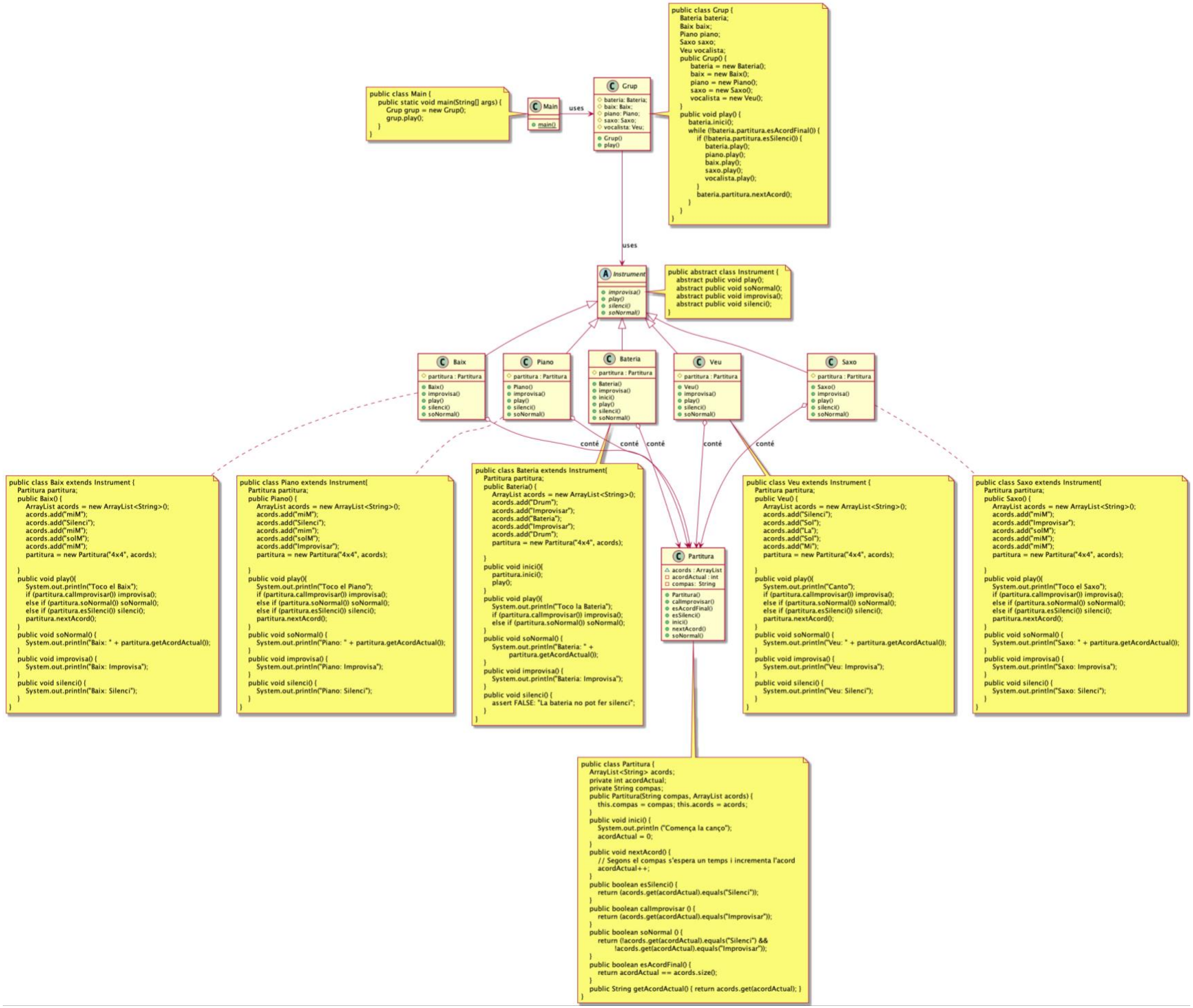
UNIVERSITAT DE BARCELONA

Nom:_____**DNI:**_____

PART II – PRINCIPIS GRASP-SOLID. APLICACIÓ DE PATRONS DE DISSENY: 6 punts sobre 10. Temps estimat 1h

Es vol simular un grup de música format per una bateria, un baix, un saxo, un piano i un/a vocalista. Per a tenir sincronitzats tots els instruments, la bateria és qui marca el ritme per a executar la partitura, acord a acord. A cada ritme de la bateria, cada instrument, seguint la seva partitura, simula el seu so corresponent a aquell acord. Per simplificar una mica el procés a cada acord, un instrument pot simular el seu so, fer silenci o bé improvisar la nota que consideri.

Davant d'aquest problema, un dissenyador de software, ha desenvolupat una aplicació per a poder executar una partitura amb aquest grup de música. Ha proposat el següent disseny de classes, on es té el programa principal a la classe Main que crea el grup i es fa el play de la cançó. No n'està gaire convençut per què, si en un futur, el grup es fa molt gran, amb d'altres instruments o amb segones veus, el codi no es gaire mantenible....


public class Main {
public static void main(String[] args) {
Grup grup = new Grup();
grup.play();
}
}
Main
main()
uses
Grup
bateria: Bateria;
baix: Baix;
piano: Piano;
saxo: Saxo;
vocalista: Veu;
Grup()
play()
public class Grup {
Bateria bateria;
Baix baix;
Piano piano;
Saxo saxo;
Veu vocalista;
public Grup() {
bateria = new Bateria();
baix = new Baix();
piano = new Piano();
saxo = new Saxo();
vocalista = new Veu();
}
public void play() {
bateria.inici();
while (!bateria.partitura.esAcordFinal()) {
if (!bateria.partitura.esSilenci()) {
bateria.play();
piano.play();
baix.play();
saxo.play();
vocalista.play();
}
bateria.partitura.nextAcord();
}
}
}
uses
Instrument
improvisa()
play()
silenci()
soNormal()
public abstract class Instrument {
abstract public void play();
abstract public void soNormal();
abstract public void improvisa();
abstract public void silenci();
}
Baix
partitura : Partitura
Baix()
improvisa()
play()
silenci()
soNormal()
Piano
partitura : Partitura
Piano()
improvisa()
play()
silenci()
soNormal()
Bateria
partitura : Partitura
Bateria()
improvisa()
inici()
play()
silenci()
soNormal()
Veu
partitura : Partitura
Veu()
improvisa()
play()
silenci()
soNormal()
Saxo
partitura : Partitura
Saxo()
improvisa()
play()
silenci()
soNormal()
conté
conté
conté
conté
conté
public class Baix extends Instrument {
Partitura partitura;
public Baix() {
ArrayList acords = new ArrayList<String>();
acords.add("miM");
acords.add("Silenci");
acords.add("miM");
acords.add("solM");
acords.add("miM");
partitura = new Partitura("4x4", acords);
}
public void play(){
System.out.println("Toco el Baix");
if (partitura.callmprovisar()) improvisa();
else if (partitura.soNormal()) soNormal();
else if (partitura.esSilenci()) silenci();
partitura.nextAcord();
}
public void soNormal() {
System.out.println("Baix: " + partitura.getAcordActual());
}
public void improvisa() {
System.out.println("Baix: Improvisa");
}
public void silenci() {
System.out.println("Baix: Silenci");
}
}
public class Piano extends Instrument{
Partitura partitura;
public Piano() {
ArrayList acords = new ArrayList<String>();
acords.add("miM");
acords.add("Silenci");
acords.add("mim");
acords.add("solM");
acords.add("Improvisar");
partitura = new Partitura("4x4", acords);
}
public void play(){
System.out.println("Toco el Piano");
if (partitura.callmprovisar()) improvisa();
else if (partitura.soNormal()) soNormal();
else if (partitura.esSilenci()) silenci();
partitura.nextAcord();
}
public void soNormal() {
System.out.println("Piano: " + partitura.getAcordActual());
}
public void improvisa() {
System.out.println("Piano: Improvisa");
}
public void silenci() {
System.out.println("Piano: Silenci");
}
}
public class Bateria extends Instrument{
Partitura partitura;
public Bateria() {
ArrayList acords = new ArrayList<String>();
acords.add("Drum");
acords.add("Improvisar");
acords.add("Bateria");
acords.add("Improvisar");
acords.add("Drum");
partitura = new Partitura("4x4", acords);
}
public void inici(){
partitura.inici();
play();
}
public void play(){
System.out.println("Toco la Bateria");
if (partitura.callmprovisar()) improvisa();
else if (partitura.soNormal()) soNormal();
}
public void soNormal() {
System.out.println("Bateria: " +
partitura.getAcordActual());
}
public void improvisa() {
System.out.println("Bateria: Improvisa");
}
public void silenci() {
assert FALSE: "La bateria no pot fer silenci";
}
}
Partitura
acords : ArrayList
acordActual : int
compas: String
Partitura()
callmprovisar()
esAcordFinal()
esSilenci()
inici()
nextAcord()
soNormal()
public class Veu extends Instrument {
Partitura partitura;
public Veu() {
ArrayList acords = new ArrayList<String>();
acords.add("Silenci");
acords.add("Sol");
acords.add("La");
acords.add("Sol");
acords.add("Mi");
partitura = new Partitura("4x4", acords);
}
public void play(){
System.out.println("Canto");
if (partitura.callmprovisar()) improvisa();
else if (partitura.soNormal()) soNormal();
else if (partitura.esSilenci()) silenci();
partitura.nextAcord();
}
public void soNormal() {
System.out.println("Veu: " + partitura.getAcordActual());
}
public void improvisa() {
System.out.println("Veu: Improvisa");
}
public void silenci() {
System.out.println("Veu: Silenci");
}
}
public class Saxo extends Instrument{
Partitura partitura;
public Saxo() {
ArrayList acords = new ArrayList<String>();
acords.add("miM");
acords.add("Improvisar");
acords.add("solM");
acords.add("miM");
acords.add("miM");
partitura = new Partitura("4x4", acords);
}
public void play(){
System.out.println("Toco el Saxo");
if (partitura.callmprovisar()) improvisa();
else if (partitura.soNormal()) soNormal();
else if (partitura.esSilenci()) silenci();
partitura.nextAcord();
}
public void soNormal() {
System.out.println("Saxo: " + partitura.getAcordActual());
}
public void improvisa() {
System.out.println("Saxo: Improvisa");
}
public void silenci() {
System.out.println("Saxo: Silenci");
}
}
public class Partitura {
ArrayList<String> acords;
private int acordActual;
private String compas;
public Partitura(String compas, ArrayList acords) {
this.compas = compas; this.acords = acords;
}
public void inici() {
System.out.println ("Comença la canço");
acordActual = 0;
}
public void nextAcord() {
// Segons el compas s'espera un temps i incrementa l'acord
acordActual++;
}
public boolean esSilenci() {
return (acords.get(acordActual).equals("Silenci"));
}
public boolean callmprovisar () {
return (acords.get(acordActual).equals("Improvisar"));
}
public boolean soNormal () {
return (!acords.get(acordActual).equals("Silenci") &&
!acords.get(acordActual).equals("Improvisar"));
}
public boolean esAcordFinal() {
return acordActual == acords.size();
}
public String getAcordActual() { return acords.get(acordActual); }
}

Davant aquest disseny, respon a les següents preguntes:

a) Què en pots dir sobre la vulneració de principis S.O.L.I.D. d'aquesta solució? Repassa cada principi SOLID un a un i digues si es vulnera o no, tot justificant-ho.

S: Single Responsability. A la classe Grup, no nomes es fa el play sinó que ella mateixa és la que s'encarrega de controlar la sincronització de tot el grup. Es pot considerar una vulneració del principi, és més no segueix el patró Expert de GRASP

O: Open-Closed Principle. Es vulnera en tots els mètodes Play de tots els instruments, ja que si en una partitura hi hagués més tipus d'acords, s'haurien de modificar tots. També es vulnera en la composició del grup, ja que si es posessin més instruments, s'hauria de modificar. Aquesta darrera vulneració és la que portarà a l'aplicació del patró Observer.

L: Liskov, en principi es vulnera ja que la Bateria no pot fer silenci, i llença una excepció. També es podia dir que no es vulnerava per que el silenci de la Bateria es la seva forma de fer silenci

I: Interface Segregation Principle. No es vulnera, ja que no hi ha interfícies

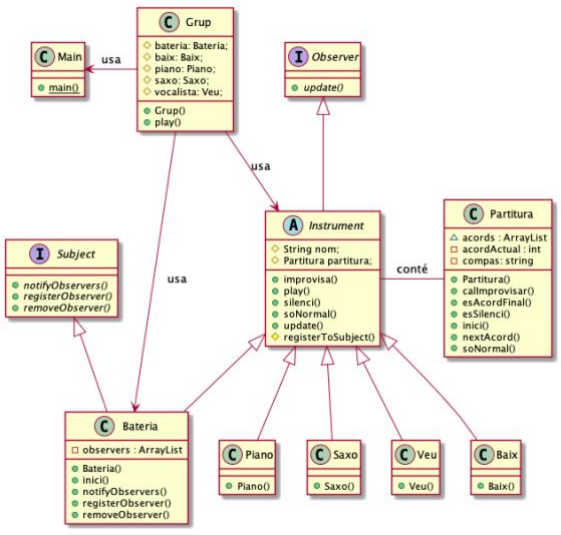
D: Dependency Inversion Principle. Es vulnera a tots els instruments per la Partitura. També es pot dir que es vulnera per que el Grup es depenent dels instruments i els crea ella, sense que ningú no els injecti.

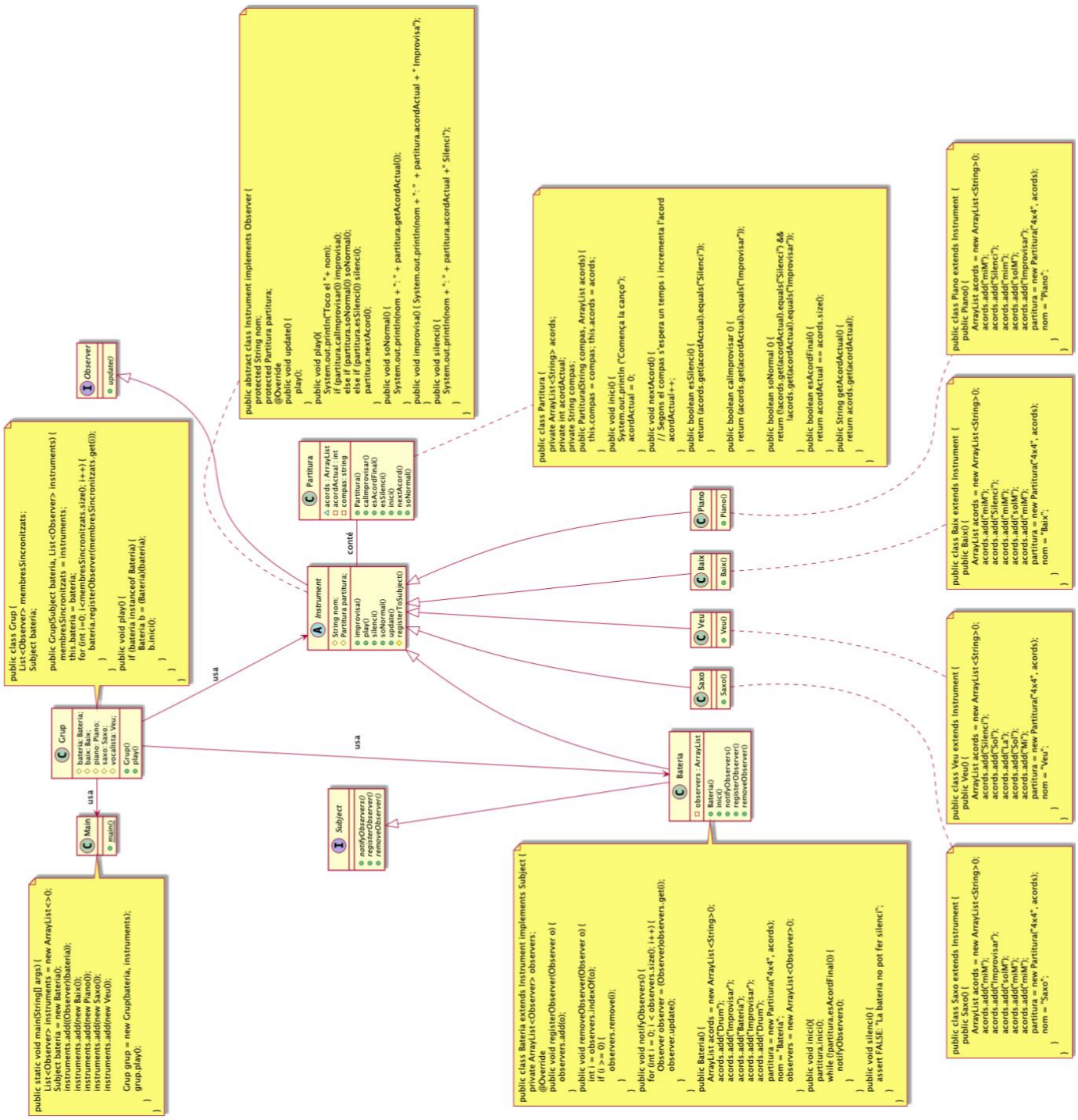
b) Com redissenyaries aquest disseny? Quin/s patró/ns de disseny faries servir? Per a contestar aquest apartat omple els apartats següents.

b.1. Digues el nom del patró principal i tipus de patró (de comportament, de creació o d'estructura). Raona breument perquè tries aquest patró.

Observer. Comportament. En ser la bateria qui mana en el play, a cada cop de bateria, es vol fer el play de tota la resta d'instruments. Això fa que el Subject (o classe observada) es la Bateria i els Observers són la resta d'instruments (i inclús es pot considerar ella mateixa un observar per a fer el seu play).

b.2. Aplicació del patró (Dibuixa el diagrama de classes obtingut després d'aplicar el patró, quines classes es corresponent en el patró original i explica els detalls més rellevants del teu disseny.





b.3. Anàlisi del patró aplicat en relació als principis S.O.L.I.D. Raona si es segueix vulnerant algun patró S.O.L.I.D.

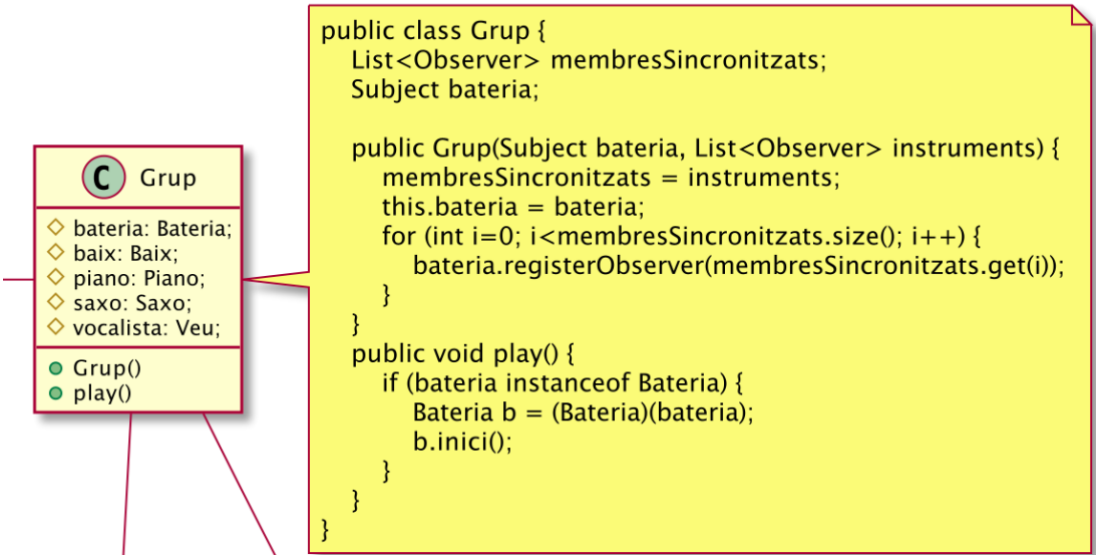
S: Ja no es vulnera el SRP ja que Grup delega a bateria l'inici i ja mitjançant el Observer es poden sincronitzar tots els elements en el play

O: Tot i que es segueix vulnerant el principi OpenClosed pel tema de la creació dels instruments del Grup des del Main, es podria solucionar utilitzant reflexivitat de Java i carregant la llista d'instruments segons les classes que se'n deriven. També es segueix vulnerant el OCP en el tema dels acords que es comentava a l'apartat a)

L: Es segueix vulnerant pel tema del silenci() a la Bateria, però es podria admetre que es el seu comportament de silenci i per tant cap partitura de bateria tindrà silenci, tal i com es comenta a l'apartat a)
D: Es segueix vulnerant a nivell de Partitura però ja no en relació als instruments dels Grup que han passat a ser Observers i Subject. Per ara la única dependència que queda és amb la classe Bateria a l'hora d'iniciar la peça. Si el mètode inici es posés a la interfície de Subject, podria solucionar-se però llavors la interfície Subject tindria dues responsabilitats, el notificar i el fer el play. Es podria pensar una interfície que fes només play – o inici()- (estil DirectorsGrup) i així podríem fer que un Grup pogués estar portat per algun instrument diferent de la Bateria.

b.4. Detalla com t'han quedat els mètodes de la classe Grup, un cop has aplicat el patró per veure com s'usaria el patró proposat. Si et cal referenciar el codi d'alguna altra classe, posa'l també per aclarir el funcionament del patró.

Grup és qui enregistra els observadors al subject (bateria)



Detallar com queda el Main en utilitzar el Grup i els instruments:

```
public static void main(String[] args) {
        List<Observer> instruments = new ArrayList<>();
        Subject bateria = new Bateria();
        instruments.add((Observer)(bateria));
        instruments.add(new Baix());
        instruments.add(new Piano());
        instruments.add(new Saxo());
        instruments.add(new Veu());

        Grup grup = new Grup(bateria, instruments);
        grup.play();
    }
}
```

Cal comentar també que a la bateria, cada cop que es salta d'acord, es quan es fa el Update de tots els observadors, ella mateixa inclosa, en ser també un Observer (via la classe Instrument).

```
public abstract class Instrument implements Observer {
   protected String nom;
   protected Partitura partitura;
   @Override
   public void update() {
      play();
   }
   public void play(){
      System.out.println("Toco el "+ nom);
      if (partitura.calImprovisar()) improvisa();
      else if (partitura.soNormal()) soNormal();
      else if (partitura.esSilenci()) silenci();
      partitura.nextAcord();
   }
   public void soNormal() {
      System.out.println(nom + ": " + partitura.getAcordActual());
   }
   public void improvisa() { System.out.println(nom + ": " + partitura.acordActual + " Improvisa");
   }
   public void silenci() {
      System.out.println(nom + ": " + partitura.acordActual +" Silenci");
   }
}
```

```
public class Bateria extends Instrument implements Subject {
   private ArrayList observers;
   public void registerObserver(Observer o) {
      observers.add(o);
   }
   public void removeObserver(Observer o) {
      int i = observers.indexOf(o);
      if (i >= 0) {
         observers.remove(i);
      }
   }
   public void notifyObservers() {
      for (int i = 0; i < observers.size(); i++) {
         Observer observer = (Observer)observers.get(i);
         observer.update();
      }
   }
   public Bateria() {
      ArrayList acords = new ArrayList<String>();
      acords.add("Drum");
      acords.add("Improvisar");
      acords.add("Bateria");
      acords.add("Improvisar");
      acords.add("Drum");
      partitura = new Partitura("4x4", acords);
      nom = "Bateria";
   }
   public void inici(){
      partitura.inici();
      while (!partitura.esAcordFinal()) {
         notifyObservers();
      }
   }
   public void silenci() {
      assert FALSE: "La bateria no pot fer silenci";
   }
}
```

b.5. Observacions addicionals (podries aplicar algun altre patró? Podries millorar el codi per evitar males olors?). Descriu en un màxim de 10 línies aquestes observacions.

Per evitar el OpenClosed dels Instruments faria servir la reflexivitat de Java. Així, des del Main podria carregar un grup d'instruments definit per classes en una carpeta del projecte.

En aquest cas, cada instrument té el tocar l'acord, el fer silenci i l'improvisar diferent i són mètodes que no es volen canviar dinàmicament en un instrument determinat. Així que no es consider l'ús del patró Strategy.

Per evitar repetició de codi, s'han pujat tots els mètodes de les classes filles d'instruments a la classe Instrument.

Per evitar la dependència de partitura es podrien pensar en partitures depenent del tipus d'instruments i aplicar un AbstractFactory quan es donés la partitura en formar de l'instrument corresponent. Aquest Abstract Factory es cridaria des de la classe Main.

c) En una segona versió, per fer més conjuntada la veu amb els instruments, es volen establir unes certes regles pel vocalista. Així, enlloc de partitura, actuarà segons les següents regles. Inicialment està en silenci en el primer acord. Només podrà cantar si en el mateix acord, el piano o el saxo toquen i cantarà la nota del piano. Si ha cantat en el darrer acord, seguirà cantant en l'acord actual, però si en l'acord anterior, el baix o el saxo improvisen, llavors el vocalista improvisarà. Seguirà improvisant fins que el saxo estigui en silenci. Quin patró usaries per al dissenyar el vocalista? Raona la teva resposta en 10 línies com a màxim.

Aquí es podria usar el patró State, on es tindria l'estat Cantant, EnSilenci i Improvisant i es podria passar d'un a l'altra per poder fer el play pertinent a cada acord i segons l'estatActual definit segons l'enunciat.