DOCUMENT DE PROVES PROJECTE M3 UF5

Héctor Lara, Marc Sánchez i Eric Morales
2n DAMvi A
2024/2025

Índex

Patró Factory

Per comprovar que la factory de les naus funciona correctament, hem creat una cua de naus, la nauFactory i hem creat tres naus diferents i al final les imprimim amb un iterador.

```
Queue<Nau> llistaNaus = new LinkedList<>();

NauFactory nFactory = new NauFactory();
//Creem les naus amb la factory.
llistaNaus.add(nFactory.crearNau(recolectorPunts, tipus: "Nau lleugera", punts: 10.0, nom: "Llança de la llibertat", saldo: 1000));
llistaNaus.add(nFactory.crearNau(recolectorPunts2, tipus: "Nau pesada", punts: 50.0, nom: "Escut de la democràcia", saldo: 10000));
llistaNaus.add(nFactory.crearNau(recolectorPunts3, tipus: "Nau exploracio", punts: 5.0, nom: "Mapa de les estrelles", saldo: 500));

Iterator<Nau> itNaus = llistaNaus.iterator();
while(itNaus.hasNext()) {
    Nau nau = itNaus.next();
    System.out.println(nau);
}
```

A continuació el resultat:

```
NauLleugera [nom=Llança de la llibertat, punts=10.0, saldo=1000]
NauPesada [nom=Escut de la democràcia, punts=50.0, saldo=10000]
NauExploracio [nom=Mapa de les estrelles, punts=5.0, saldo=500]
```

Podem veure que les naus s'han creat correctament i, per tant, que la facotry de les naus funciona correctament!

Patró Abstract Factory

Per comprovar que l'abstarct factory funciona bé, hem creat una jocFactory que segons el tipus d'objecte (enemic, recurs…) agafa una factory diferent i a continuació creem un objecte en específic i al final imprimim tots els objectes creats.

```
//Punt 2
JocFactory jocFactory;


//Creem un objecte de cada amb l'abstract factory.
jocFactory = FactoryProvider.getFactory( choice: "Enemic");
Enemic naucombat = (Enemic) jocFactory.create( tipus: "Enemic",  nom: "Nau de combat");

jocFactory = FactoryProvider.getFactory( choice: "Recurs");
Recurs kebab = (Recurs) jocFactory.create( tipus: "Recurs",  nom: "Kebab");

jocFactory = FactoryProvider.getFactory( choice: "Equipament");
Equipament escut = (Equipament) jocFactory.create( tipus: "Equipament",  nom: "Escut protector");

jocFactory = FactoryProvider.getFactory( choice: "Bonus");
Bonus robot = (Bonus) jocFactory.create( tipus: "Bonus",  nom: "Robot reparador");

jocFactory = FactoryProvider.getFactory( choice: "Galàxia");
Galaxia espiral = (Galaxia) jocFactory.create( tipus: "Galàxia",  nom: "Espiral");

String resultat = "Nau " + naucombat + ", Kebab: " + kebab + ", escut: " + escut + ", robot: " + robot + ", espiral: " + espiral;
System.out.println(resultat);
```

A contiunació el resultat:

Nau NauCombat [nom=Nau de combat, puntsRestar=-10, tipus=Enemic], Kebab: Kebab [nom=Kebab, puntsSumar=25.0, tipus=Recurs], escut: Escut [nom=Escut protector, factor=0.75, tipus=Equipament], robot: RobotReparador [nom=robot reparador, puntsSumar=5, tipus=Bonus], espiral: Espiral [nom=Espiral, tipus=Galàxia]

Com es pot veure, ha creat tots els objectes correctament!

No he pogut posar la captura del print perquè la línia era molt llarga i es veia molt malament.

Patró Observer

Per comprovar si el nostre patró Observer funciona correctament hem creat un atribut de la classe de l'observer (en el nostre cas anomenat RecolectorPunts) per cada nau que hem creat, per poder afegir-los en la creació de les naus.

```
RecolectorPunts recolectorPunts = new RecolectorPunts(); //Observer de les naus
RecolectorPunts recolectorPunts2 = new RecolectorPunts(); //Observer de les naus
RecolectorPunts recolectorPunts3 = new RecolectorPunts(); //Observer de les naus

Queue<Nau> llistaNaus = new LinkedList<>();

NauFactory nFactory = new NauFactory();
//Creem les naus amb la factory.
llistaNaus.add(nFactory.crearNau(recolectorPunts, tipus: "Nau lleugera", punts: 10.0, nom: "Llança de la llibertat", saldo: 1000));
llistaNaus.add(nFactory.crearNau(recolectorPunts2, tipus: "Nau pesada", punts: 50.0, nom: "Escut de la democràcia", saldo: 10000));
llistaNaus.add(nFactory.crearNau(recolectorPunts3, tipus: "Nau exploracio", punts: 5.0, nom: "Mapa de les estrelles", saldo: 500));
```

Una vegada hem afegit l'observer a les naus comprovem si la funcionalitat és correcte creant diversos ObjecteCapturat per sumar o restar la puntuació de la nau i després desubsribim les naus creades anteriorment de l'observer en un foreach de la cua creada.

```
//Punt 3
System.out.println("Punts del kebab: " + kebab.puntsASumar());
System.out.println("Punts abans: " + aux.getPunts());

//La nau es troba amb un recurs (un kebab)
ObjecteCapturat objc = new ObjecteCapturat(aux, kebab);
recolectorPunts.setPunts(objc);

System.out.println("Punts despres: " + aux.getPunts());
//La nau es troba amb un enemic (una nau de combat)
ObjecteCapturat objc0 = new ObjecteCapturat(aux, naucombat);
recolectorPunts.setPunts(objc0);
System.out.println("Punts despres: " + aux.getPunts());

//Desubscribim les naus de l'observer.
for(Nau nau : llistaNaus) {
    if(nau.getNau() instanceof NauExploracio) {
        recolectorPunts3.removePropertyChangeListener((NauExploracio)nau.getNau());
    }
    else if(nau.getNau() instanceof NauLleugera) {
        recolectorPunts.removePropertyChangeListener((NauLleugera)nau.getNau());
    }
    else if(nau.getNau() instanceof NauPesada) {
        recolectorPunts2.removePropertyChangeListener((NauPesada)nau.getNau());
    }
}
```

I el resultat que ens dóna és:

```
Punts del kebab: 25.0
Punts abans: 10.0
Objecte que arriba a l'observer: Kebab [nom=Kebab, puntsSumar=25.0, tipus=Recurs]
Punts de la nau abans de sumar: 10.0
Punts a sumar: 25.0
Sense factor a aplicar
Punts de la Nau Lleugera després del càlcul: 35.0
Punts despres: 35.0
Objecte que arriba a l'observer: NauCombat [nom=Nau de combat, puntsRestar=-10, tipus=Enemic]
Punts de la nau abans de restar: 35.0
Punts a restar: -10
Sense factor a aplicar
Punts de la Nau Lleugera després del càlcul: 25.0
Punts despres: 25.0
```

Patró Decorator

Per comprovar que el decorator funciona bé creem una nau nova i diferents equipaments nous. Seguidament, equipem els decoradors a la nau creada (color i 2 equipaments) i, finalment, imprimim el resultat.

```
RecolectorPunts recolectorPunts4 = new RecolectorPunts();
Nau aux1 = nFactory.crearNau(recolectorPunts4, tipus: "Nau exploracio", punts: 5.0, nom: "Mincron", saldo: 100);
jocFactory = FactoryProvider.getFactory( choice: "Equipament");
Equipament canon = (Equipament) jocFactory.create( tipus: "Equipament", nom: "Cano");
Equipament escut1 = (Equipament) jocFactory.create( tipus: "Equipament", nom: "Escut protector");

//Afegim decorators a una nau
Nau n2 = new EquipamentDecorator(aux1, escut1);
Nau n3 = new Color(n2, ColorEnum.AQUA);
Nau n4 = new EquipamentDecorator(n3, canon);
System.out.println(n4.getDescripcio());
System.out.println(n4.getNau());
```

A continuació el resultat:

```
Nau Exploració amb Escut [nom=Escut protector, factor=0.75, tipus=Equipament] de color AQUA amb Cano [nom=Canó, factor=0.33, tipus=Equipament]
NauExploracio [nom=Mincron, punts=5.0, saldo=100]
```

Com es pot veure, la nau ha sigut equipada amb tots els decoradors i en fer el getNau() ens retorna la nau base (nau d'exploració).

Patró Adapter

Per poder comprovar correctament el patró Adapter, hem modificat el funcionament del sumatori dels punts a la funció pròpia que està subscrita a l'observer (propertyChange). En aquesta el que hem fet és comprovar quin tipus d'ObjecteCapturat se'ns passa (si és bo o dolent) i segons el tipus s'envia a un dels dos adapters que hem creat).

```
//Punt 4 i 5
//Creem un observer nou per la nau que crearem a continuació.
//Necessitem un de nou perquè si no estariem cridant a la nau anterior
RecolectorPunts recolectorPunts4 = new RecolectorPunts();
Nau aux1 = nFactory.crearNau(recolectorPunts4, tipus: "Nau exploracio", punts: 5.0, nom: "Mincron", saldo: 100);
jocFactory = FactoryProvider.getFactory( choice: "Equipament");
Equipament canon = (Equipament) jocFactory.create( tipus: "Equipament", nom: "Cano");
Equipament escut1 = (Equipament) jocFactory.create( tipus: "Equipament", nom: "Escut protector");

//Afegim decorators a una nau
Nau n2 = new EquipamentDecorator(aux1, escut1);
Nau n3 = new Color(n2, ColorEnum.AQUA);
Nau n4 = new EquipamentDecorator(n3, canon);
System.out.println(n4.getDescripcio());
System.out.println(n4.getNau());

//La nau amb decorators es troba amb un enemic (nau de combat)
ObjecteCapturat objc1 = new ObjecteCapturat(n4, naucombat);
recolectorPunts4.setPunts(objc1);
```

El resultat de la prova és el següent:

NauExploracio [nom=Mincron, punts=5.0, saldo=100]

Objecte que arriba a l'observer: NauCombat [nom=Nau de combat, puntsRestar=-10, tipus=Enemic]

Punts de la nau abans de restar: 5.0

Punts a restar: -10

Factor a multiplicar: 0.33

Punts de la Nau d'Exploració després del càlcul: 1.6999999999999997

Com es pot veure tant el factor a multiplicar com el tipus d'objecte que arriba es calculen correctament.

Connexió amb la base de dades

Insert:

Per comprovar que l'insert funciona bé primer mirem que les naus a afegir no estiguin ja afegides a la base de dades i si no ho estan, fem l'insert del nom, els punts i el saldo de la nau i finalment imprimim el resultat de com queda la bd després de l'operació.

```
Connection conn;
Statement st = null;
ResultSet rs = null;

try {
    conn = DriverManager.getConnection( url: "jdbc:mysql://localhost/videojocjdbc?" + "user=root&password=super3");
    st = conn.createStatement();
    rs = st.executeQuery( sql: "SELECT * FROM jugador");
    boolean trobat = false;

    for (Nau nau: llistaNausDecorator) {
        while(rs.next()) {

            //Mirem que la nau no estigui ja posada a la base de dades.
            if (nau instanceof EquipamentDecorator){
                if (!rs.getString( columnLabel: "nom").equals(nau.getNau().getNom())) {
                    trobat = true;
                    break;
                }
            }
            else if (nau instanceof GalaxiaDecorator) {
                if (!rs.getString( columnLabel: "nom").equals(nau.getNau().getNom())) {
                    trobat = true;
                    break;
                }
            }
            else if (nau instanceof Color){
                if (!rs.getString( columnLabel: "nom").equals(nau.getNau().getNom())) {
                    trobat = true;
                    break;
                }
            }
        }
```

A continuació el resultat:

```
    if(!trobat) {
        String sqlInsertar="INSERT INTO jugador (nom, punts, saldo) VALUES (?, ?, ?)";
        PreparedStatement ps = conn.prepareStatement(sqlInsertar);
        ps.setString( parameterIndex: 1, nau.getNau().getNom());
        ps.setDouble( parameterIndex: 2,  x: 0);
        ps.setDouble( parameterIndex: 3, nau.getNau().getSaldoRecursos());
        ps.executeUpdate();
    }
    trobat = false;
}

st = conn.createStatement();
rs = st.executeQuery( sql: "SELECT * FROM jugador");

//Imprimim el resultat del select.
System.out.println("Naus després de l'insert:");
while(rs.next()){
    System.out.println("ID: " + rs.getInt( columnLabel: "idjugador"));
    System.out.println("Nom: " + rs.getString( columnLabel: "nom"));
    System.out.println("Saldo: " + rs.getString( columnLabel: "saldo"));
    System.out.println("Punts: " + rs.getString( columnLabel: "punts"));
}
```

```
Naus després de l'insert:
ID: 308
Nom: Mincron
Saldo: 100.00
Punts: 0.00
ID: 309
Nom: Rocinante
Saldo: 80.00
Punts: 0.00
ID: 310
Nom: Urithiru
Saldo: 10.00
Punts: 0.00
```

Com es pot veure les naus han sigut inserides correctament amb els valors corresponents. Una vegada s'han inserit les naus actualitzem la puntuació de les naus amb ObjectesCapturats per poder comprovar la actualitzacióeliminació de naus de la base de dades.

```
jocFactory = FactoryProvider.getFactory( choice: "Enemic");
Enemic meteorit = (Enemic) jocFactory.create( tipus: "Enemic",  nom: "Meteorit");

//Funció per variar el número de punts de les naus i que funcioni la funció de la base de dades següent

for(Nau nau : llistaNausDecorator) {
    ObjecteCapturat objc2 = new ObjecteCapturat(nau, kebab);
    ObjecteCapturat objc3 = new ObjecteCapturat(nau, naucombat);
    ObjecteCapturat objc4 = new ObjecteCapturat(nau, meteorit);
    ObjecteCapturat objc5 = new ObjecteCapturat(nau, robot);
    if(nau.getNau() instanceof NauExploracio) {
        recolectorPunts4.setPunts(objc2);
        recolectorPunts4.setPunts(objc3);
        recolectorPunts4.setPunts(objc4);
        recolectorPunts4.setPunts(objc5);
    }
    else if(nau.getNau() instanceof NauLleugera) {
        recolectorPunts6.setPunts(objc2);
        recolectorPunts6.setPunts(objc3);
        recolectorPunts6.setPunts(objc4);
        recolectorPunts6.setPunts(objc5);
    }
    else if(nau.getNau() instanceof NauPesada) {
        recolectorPunts5.setPunts(objc2);
        recolectorPunts5.setPunts(objc3);
        recolectorPunts5.setPunts(objc4);
        recolectorPunts5.setPunts(objc5);
    }
}
```

Si bé es veu que creem un nou enemic amb el jocFactory, la resta els utilitzem de punts anteriors el mai.

El resultat per pantalla és:

Objecte que arriba a l'observer: Kebab [nom=Kebab, puntsSumar=25.0, tipus=Recurs]
Punts de la nau abans de sumar: 1.6999999999999997
Punts a sumar: 75.75757575757575
Factor a dividir: 0.33
Punts de la Nau d'Exploració després del càlcul: 77.45757575757575
Objecte que arriba a l'observer: NauCombat [nom=Nau de combat, puntsRestar=-10, tipus=Enemic]
Punts de la nau abans de restar: 77.45757575757575
Punts a restar: -10
Factor a multiplicar: 0.33
Punts de la Nau d'Exploració després del càlcul: 74.15757575757576
Objecte que arriba a l'observer: Meteorit [nom=Meteorit, puntsRestar=-15, tipus=Enemic]
Punts de la nau abans de restar: 74.15757575757576
Punts a restar: -15
Factor a multiplicar: 0.33
Punts de la Nau d'Exploració després del càlcul: 69.20757575757575

Objecte que arriba a l'observer: RobotReparador [nom=robot reparador, puntsSumar=5, tipus=Bonus]
Punts de la nau abans de sumar: 69.20757575757575
Punts a sumar: 15.15151515151515
Factor a dividir: 0.33
Punts de la Nau d'Exploració després del càlcul: 84.35909090909091
Objecte que arriba a l'observer: Kebab [nom=Kebab, puntsSumar=25.0, tipus=Recurs]
Punts de la nau abans de sumar: 500.0
Punts a sumar: 25.0
Sense factor a aplicar
Punts de la Nau Pesada després del càlcul: 525.0
Objecte que arriba a l'observer: NauCombat [nom=Nau de combat, puntsRestar=-10, tipus=Enemic]
Punts de la nau abans de restar: 525.0
Punts a restar: -10
Sense factor a aplicar
Punts de la Nau Pesada després del càlcul: 515.0
Objecte que arriba a l'observer: Meteorit [nom=Meteorit, puntsRestar=-15, tipus=Enemic]
Punts de la nau abans de restar: 515.0
Punts a restar: -15
Sense factor a aplicar
Punts de la Nau Pesada després del càlcul: 500.0
Objecte que arriba a l'observer: RobotReparador [nom=robot reparador, puntsSumar=5, tipus=Bonus]
Punts de la nau abans de sumar: 500.0
Punts a sumar: 5
Sense factor a aplicar
Punts de la Nau Pesada després del càlcul: 505.0
Objecte que arriba a l'observer: Kebab [nom=Kebab, puntsSumar=25.0, tipus=Recurs]
Punts de la nau abans de sumar: 10.0
Punts a sumar: 25.0
Sense factor a aplicar
Punts de la Nau Lleugera després del càlcul: 35.0
Objecte que arriba a l'observer: NauCombat [nom=Nau de combat, puntsRestar=-10, tipus=Enemic]
Punts de la nau abans de restar: 35.0
Punts a restar: -10
Sense factor a aplicar
Punts de la Nau Lleugera després del càlcul: 25.0
Objecte que arriba a l'observer: Meteorit [nom=Meteorit, puntsRestar=-15, tipus=Enemic]
Punts de la nau abans de restar: 25.0

Punts a restar: -15
Sense factor a aplicar
Punts de la Nau Lleugera després del càlcul: 10.0
Objecte que arriba a l'observer: RobotReparador [nom=robot reparador, puntsSumar=5, tipus=Bonus]
Punts de la nau abans de sumar: 10.0
Punts a sumar: 5
Sense factor a aplicar
Punts de la Nau Lleugera després del càlcul: 15.0

Una vegada tenim les naus amb puntuacions diferents, realitzem l'actualització i eliminació de les naus corresponents segons la puntuació que tinguin, imprimint en tot moment les naus que queden a la cua.

```
try {
    conn = DriverManager.getConnection( url: "jdbc:mysql://localhost/videojocjdbc?" + "user=root&password=super3");
    st = conn.createStatement();
    rs = st.executeQuery( sql: "SELECT * FROM jugador");

    for (Nau nau2 : llistaNausDecorator) {
        while (rs.next()) {
            if (rs.getString( columnLabel: "nom").equals(nau2.getNau().getNom())) {
                String updatear = "UPDATE jugador SET punts = ? WHERE nom = ?";
                PreparedStatement ps = conn.prepareStatement(updatear);
                ps.setDouble( parameterIndex: 1, nau2.getNau().getPunts());
                ps.setString( parameterIndex: 2, nau2.getNau().getNom());
                ps.executeUpdate();
                break;
            }
        }
    }
}
catch (SQLException ex) {
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
}
finally {
    if (rs != null) {
        try {
            rs.close();
        } catch (SQLException sqlEx) {
        }
        rs = null;
    }
    if (st != null) {
        try {
            st.close();
        } catch (SQLException sqlEx) {
        }
        st = null;
    }
}
```

Aquí realitzem l'actualització de totes les naus a la base de dades.

```
try {
    conn = DriverManager.getConnection( url: "jdbc:mysql://localhost/videojocjdbc?" + "user=root&password=super3");
    st = conn.createStatement();
    rs = st.executeQuery( sql: "SELECT * FROM jugador");

    while (rs.next()) {
        if (rs.getDouble( columnLabel: "punts") < 400) {
            Iterator<Nau> it2 = llistaNausDecorator.iterator();
            while(it2.hasNext()) {
                Nau nau = it2.next();
                if(nau.getNau().getNom().equals(rs.getString( columnLabel: "nom"))) {
                    it2.remove();
                    break;
                }
            }
            String borrarSql ="DELETE FROM jugador WHERE nom = ?";
            PreparedStatement ps = conn.prepareStatement(borrarSql);
            ps.setString( parameterIndex: 1, rs.getString( columnLabel: "nom"));
            ps.executeUpdate();
        }
    }
}
catch(SQLException ex){
    // handle any errors
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
}
finally {
    if (rs != null) {
        try {
            rs.close();
        } catch (SQLException sqlEx) { }
        rs = null;
    }
    if (st != null) {
        try {
            st.close();
        } catch (SQLException sqlEx) { }
        st = null;
    }
}
```

Aquí realitzem l'eliminació de les naus (tant de la base de dades com de la llista que tenim al main) si no arriben a un total superior a 400 punts.

Naus després d'eliminar i actualitzar les corresponents:
Nom: Rocinante, Saldo: 80, Punts: 505.0

Una vegada hem fet totes les operacions amb la base de dades, imprimim les naus ordenades per nom i per punts. Primer ho ordenem per nom perquè es vegi la diferència. Per poder fer això hem hagut de ficar comparables a les naus i els decoradors com es pot veure a continuació:

```java
/**
 * @param o Nau a comparar amb l'actual
 * @return retorna l'ordre de les naus segons el nom de manera ascendent.
 */
@Override  Keita58
public int compareTo(Nau o) { return this.getNom().compareTo(o.getNom()); }
```

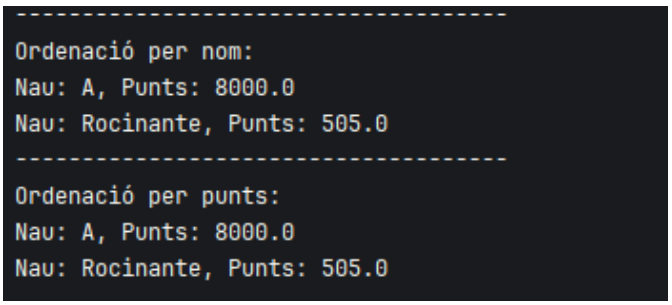
A més, hem creat una classe PuntsComparator que ordena pels punts:

```java
public class PuntsComparator implements Comparator<Nau> {  2 usages  Keita58

    /**
     * @param a Primera nau a comparar
     * @param b Segona naou a comparar
     * @return retorna l'ordenació descendent de les naus segons la quantitat de punts.
     */
    public int compare(Nau a, Nau b) { return Double.compare(b.getPunts(), a.getPunts()); }
}
```

LLavors per comprovar-ho fem un Sorted set per comprovar l'ordenació per noms i un altre sorted set amb el contingut de l'anterior, però que inclou el comparador:

```java
SortedSet<Nau> naves= new TreeSet<>();
naves.addAll(llistaNausDecorator);

System.out.println("-------------------------------------");
System.out.println("Ordenació per nom:");

//Mostrem les naus ordenades per nom.
for(Nau nau : naves) {
    System.out.println("Nau: "+nau.getNom()+", Punts: "+nau.getPunts());
}

//Ho posem tot en un SortedSet per poder utilitzar la classe PuntsComparat
//ordenat anteriorment
SortedSet<Nau> naves2 = new TreeSet<>(new PuntsComparator());
naves2.addAll(naves);

System.out.println("-------------------------------------");
System.out.println("Ordenació per punts:");

//Mostrem les naus ordenades per punts i per nom.
for(Nau nau : naves2) {
    System.out.println("Nau: "+nau.getNom()+", Punts: "+nau.getPunts());
}
```

A continuació el resultat:



Com es pot veure ha ordenat primer per nom, i després per punts i nom.