PRÀCTICA Arquitectura del Software

4rt Lliurament

Grup 12

QT 15-16

Shows.com

Pere Berge Sànchez
Daniel Gil Sancho
Albert Suàrez Molgó
Víctor Pérez Martos

ÍNDEX

1. Introducció
2. Classes
    - 2.1. Entrada
    - 2.2. Espectacle
    - 2.3. Estrena
    - 2.4. EstrenaPK
    - 2.5. Local
    - 2.6. Representació
    - 2.7. RepresentacióPK
    - 2.8. Seient
    - 2.9. SeientEnRepresentació
    - 2.10. SeientEnRepresentacióPK
    - 2.11. SeientPK
    - 2.12. Sessió
    - 2.13. Showscom
3. Adapters
    - 3.1. BankServiceAdapter
    - 3.2. CurrencyConvertorAdapter
    - 3.3. IBankServiceAdapter
    - 3.4. ICurrencyConvertorAdapter
4. Factories
    - 4.1. FactoriaCtrl
    - 4.2. FactoriaServeis
    - 4.3. FactoriaUseCase
    - 4.4. ServiceLocator
5. TupleTypes
    - 5.1. TupleTypeFila
    - 5.2. TupleTypeRepresentació
    - 5.3. TupleTypeSeleccioSeients
6. UseCaseControllers
    - 6.1. CompraEntradesUseCaseController
7. UseCases
    - 7.1. ComprarEntrada

7.2. ConsultarOcupacio
7.3. ConsultarRepresentacions
8. Utils
8.1. ComboItem
8.2. Utils
9. Services
9.1. BankService
9.2. CurrencyService
10. Enums
10.1. Estat
10.2. Moneda
10.3. TipusSessió
11. PresentationControllers
11.1. ComprarEntradesController
12. Views
12.1. ComprarEntradaView
12.2. ErrorView
12.3. IniView
12.4. PagamentView
12.5. SeientsView
13. Controllers
13.1. CtrlEspectacle
13.2. CtrlRepresentació
13.3. CtrlSientEnRepresentació
14. Persistence
14.1. CtrlEspectacleDB
14.2. CtrlRepresentacioDB
14.3. CtrlSeientEnRepresentacioDB
14.4. HibernateUtils
15. Joc de proves

1.- Introducció

El document que es presenta a continuació, mostra les diferents classes que conformen un sistema que simula el sistema de compra d'entrades d'espectacles.

Les classes mencionades prèviament segueixen una estructura basada en el seguiment de diferents patrons de disseny que conformen un conjunt de bones pràctiques, permetent així obtenir qualitats com mantenibilitat o escalabilitat.

Així doncs, les classes estan organitzades de la següent manera:

- **Persistència**: Incorpora totes les classes relatives a la base de dades. Consta de diferents controladors i una classe amb funcions auxiliars d'ajuda en l'ús del hibernate.

- **Domini**: Incorpora totes les classes relatives a la lògica del projecte, està subdividit en:

- **Utils**: incorpora funcions d'utilitat per al funcionament del projecte.

- **UseCases**: incorpora els diferents casos d'ús que s'emprenen en el projecte.

- **UseCaseControllers**: incorpora els controladors dels casos d'ús, necessaris per obtenir un aïllament respecte a la implementació dels mètodes.

- **TupleTyple**: incorpora els diferents tipus de tuples emprats en el projecte.

- **Factories**: incorpora classes que creen els controladors, adaptadors, o cerquen els serveis.

- **Enums**: incorpora classes que representen restriccions en el nomenament de diferents tipus de valors.

- **Controllers**: incorpora els controladors de les diferents entitats del projecte. Ens permeten obtenir una millor abstracció del projecte.

- **Classes**: conté les diferents entitats que conformen el projecte, com espectacles, locals, seients…

- **Adapters**: incorpora les classes que permeten connectar amb serveis web externs al nostre sistema. Permeten obtenir un millor aïllament.

- **Presentació**: Incorpora totes les classes referents a la percepció visual de l'usuari sobre el projecte. Està subdividida en controladors que permeten connectar amb la capa de domini i vistes, que contenen la informació relatives a la representació gràfica que veu l'usuari.

- **Services**: Aquest paquet incorpora diferents classes que simulen el comportament de serveis web.

2.- Classes

2.1.- Entrada

```
package com.shows.as.domain.classes;

import javax.persistence.*;
import java.sql.Date;
import java.util.Collection;
import java.util.Set;
/*
Implementació de la classe Entrada del paquet domain.classes
 */
//entitat que representa una entrada a una representació
@Entity
@Table(name = "entrada", schema = "public", catalog = "postgres")
public class Entrada {

    private String identificador;
    private String dniclient;
    private Integer nombreespectadors;
    private Date data;
    private Double preu;
    private Representació representació;
    private Collection<Seientenrepresentació> seients;



    public Entrada() {

    }

    public Entrada(String identificador, String dniClient, Integer nombreEspectadors, Date
data, Double preu) {
        this.identificador = identificador;
        this.dniclient = dniClient;
        this.nombreespectadors = nombreEspectadors;
        this.data = data;
        this.preu = preu;
    }

    public void associa(Representació r, Set<Seientenrepresentació> seientRep) {
        this.representació = r;
        r.associaEntrada(this);
        this.seients = seientRep;
    }

    @Id
    @Column(name = "identificador", nullable = false, insertable = true, updatable = true,
length = 255)
    public String getIdentificador() {
        return identificador;
    }

    public void setIdentificador(String identificador) {
        this.identificador = identificador;
    }

    @Basic
    @Column(name = "dniclient", nullable = true, insertable = true, updatable = true,
length = 255)
```

```
    public String getDniclient() {
        return dniclient;
    }

    public void setDniclient(String dniclient) {
        this.dniclient = dniclient;
    }

    @Basic
    @Column(name = "nombreespectadors", nullable = true, insertable = true, updatable =
true)
    public Integer getNombreespectadors() {
        return nombreespectadors;
    }

    public void setNombreespectadors(Integer nombreespectadors) {
        this.nombreespectadors = nombreespectadors;
    }

    @Basic
    @Column(name = "data", nullable = true, insertable = true, updatable = true)
    public Date getData() {
        return data;
    }

    public void setData(Date data) {
        this.data = data;
    }

    @Basic
    @Column(name = "preu", nullable = true, insertable = true, updatable = true, precision
= 17)
    public Double getPreu() {
        return preu;
    }

    public void setPreu(Double preu) {
        this.preu = preu;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Entrada that = (Entrada) o;

        if (identificador != null ? !identificador.equals(that.identificador) :
that.identificador != null)
            return false;
        if (dniclient != null ? !dniclient.equals(that.dniclient) : that.dniclient != null)
return false;
        if (nombreespectadors != null ? !nombreespectadors.equals(that.nombreespectadors) :
that.nombreespectadors != null)
            return false;
        if (data != null ? !data.equals(that.data) : that.data != null) return false;
        if (preu != null ? !preu.equals(that.preu) : that.preu != null) return false;
        return true;
    }

    @Override
    public int hashCode() {
        int result = identificador != null ? identificador.hashCode() : 0;
        result = 31 * result + (dniclient != null ? dniclient.hashCode() : 0);
        result = 31 * result + (nombreespectadors != null ? nombreespectadors.hashCode() :
0);
```

```
        result = 31 * result + (data != null ? data.hashCode() : 0);
        result = 31 * result + (preu != null ? preu.hashCode() : 0);
        return result;
    }

    @ManyToOne
    @JoinColumns({@JoinColumn(name = "sessió", referencedColumnName = "sessió"),
@JoinColumn(name = "nomlocal", referencedColumnName ="nomlocal")})
    public Representació getRepresentació() {
        return representació;
    }

    public void setRepresentació(Representació representació) {
        this.representació = representació;
    }

    @OneToMany(mappedBy = "entradaByIdentrada")
    public Collection<Seientenrepresentació> getSeients() {
        return seients;
    }

    public void setSeients(Collection<Seientenrepresentació>
seientenrepresentaciósByIdentificador) {
        this.seients = seientenrepresentaciósByIdentificador;
    }

}
```

2.2.- Espectacle

```
package com.shows.as.domain.classes;

import com.shows.as.domain.tupleTypes.TupleTypeRepresentacio;
import com.shows.as.domain.utils.Utils;

import javax.persistence.*;
import java.util.Collection;
import java.sql.Date;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;
/*
Implementació de la classe Espectacle del paquet domain.classes
 */
//entitat que representa un espectacle
@Entity
@Table(name = "espectacle", schema = "public", catalog = "postgres")
public class Espectacle {

    public static final String TAULA = "Espectacle";

    private String títol;
    private Integer participants;
    private Set<Representació> representacions = new LinkedHashSet<Representació>();

    public Espectacle(){

    }

    public Espectacle(String a, int i) {
        this.títol = a;
        this.participants = i;

    }

    public Set<TupleTypeRepresentacio> obteRepresentacions(Date data){
        Set<TupleTypeRepresentacio> result = new LinkedHashSet<TupleTypeRepresentacio>();
        for (Representació r : getRepresentacions()){
            if (Utils.sameDay(data, r.getData())) result.add(r.obteInfo());
        }
        return result;
    }

    @Id
    @Column(name = "títol", nullable = false, insertable = true, updatable = true, length =
255)
    public String getTítol() {
        return títol;
    }

    public void setTítol(String títol) {
        this.títol = títol;
    }

    @Basic
    @Column(name = "participants", nullable = true, insertable = true, updatable = true)
    public Integer getParticipants() {
        return participants;
    }

    public void setParticipants(Integer participants) {
        this.participants = participants;
```

```
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Espectacle that = (Espectacle) o;

        if (títol != null ? !títol.equals(that.títol) : that.títol != null) return false;
        if (participants != null ? !participants.equals(that.participants) :
that.participants != null) return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = títol != null ? títol.hashCode() : 0;
        result = 31 * result + (participants != null ? participants.hashCode() : 0);
        return result;
    }

    @OneToMany(cascade = CascadeType.ALL,fetch = FetchType.EAGER)
    @JoinColumn(name = "títolesp",referencedColumnName = "títol")
    public Set<Representació> getRepresentacions() {
        return representacions;
    }

    public void setRepresentacions(Set<Representació> representaciósByTítol) {
        this.representacions = representaciósByTítol;
    }
}
```

2.3.- Estrena

```
package com.shows.as.domain.classes;

import javax.persistence.*;
import java.sql.Date;
/*
Implementació de la classe Estrena del paquet domain.classes
 */
//Entitat que representa l'estrena d'un espectacle
@Entity
@Table(name = "estrena", schema = "public", catalog = "postgres")
@PrimaryKeyJoinColumns({@PrimaryKeyJoinColumn(name = "nomlocal", referencedColumnName =
"sessió"), @PrimaryKeyJoinColumn(name = "sessió", referencedColumnName = "nomlocal")})
public class Estrena extends Representació {

    private Integer recàrrec;
    private Representació representació;

    public Estrena() {

    }

    public Estrena(Double preu, Date data, Integer nombreseientslliures, String sessió,
String nomlocal, String títolesp, Integer recàrrec) {
        super(preu, data, nombreseientslliures, sessió, nomlocal, títolesp);
        this.recàrrec = recàrrec;
    }

    @Basic
    @Column(name = "recàrrec", nullable = true, insertable = true, updatable = true)
    public Integer getRecàrrec() {
        return recàrrec;
    }

    public void setRecàrrec(Integer recàrrec) {
        this.recàrrec = recàrrec;
    }

    @OneToOne
    @JoinColumns({@JoinColumn(name = "sessió", referencedColumnName = "sessió", nullable =
false), @JoinColumn(name = "nomlocal", referencedColumnName = "nomlocal", nullable =
false)})
    public Representació getRepresentació() {
        return representació;
    }

    public void setRepresentació(Representació representació) {
        this.representació = representació;
    }

    @Override
    public boolean esEstrena() {
        return true;
    }

    @Override
    public Integer obteRecarrec() {
        return recàrrec;
    }

}
```

2.4.- Local

```
package com.shows.as.domain.classes;

import javax.persistence.*;
import java.util.Collection;
/*
Implementació de la classe Local del paquet domain.classes
 */
//entitat que representa un local on s'efectuen representacions
@Entity
@Table(name = "local", schema = "public", catalog = "postgres")
public class Local {
    private String nom;
    private String adreça;
    private Collection<Representació> representacions;
    private Collection<Seient> seients;



    public Local() {

    }

    public Local(String nom, String adreça) {
        this.nom = nom;
        this.adreça = adreça;
    }

    @Id
    @Column(name = "nom", nullable = false, insertable = true, updatable = true, length =
255)
    public String getNom() {
        return nom;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    @Basic
    @Column(name = "adreça", nullable = true, insertable = true, updatable = true, length =
255)
    public String getAdreça() {
        return adreça;
    }

    public void setAdreça(String adreça) {
        this.adreça = adreça;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Local that = (Local) o;

        if (nom != null ? !nom.equals(that.nom) : that.nom != null) return false;
        if (adreça != null ? !adreça.equals(that.adreça) : that.adreça != null) return
false;

        return true;
```

```
    }

    @Override
    public int hashCode() {
        int result = nom != null ? nom.hashCode() : 0;
        result = 31 * result + (adreça != null ? adreça.hashCode() : 0);
        return result;
    }

    @OneToMany(mappedBy = "localByNomlocal")
    public Collection<Representació> getRepresentacions() {
        return representacions;
    }

    public void setRepresentacions(Collection<Representació> representaciósByNom) {
        this.representacions = representaciósByNom;
    }

    @OneToMany(mappedBy = "localByNomlocal")
    public Collection<Seient> getSeients() {
        return seients;
    }

    public void setSeients(Collection<Seient> seientsByNom) {
        this.seients = seientsByNom;
    }
}
```

2.6.- Representació

```
package com.shows.as.domain.classes;

import com.shows.as.domain.tupleTypes.TupleTypeFilaColumna;
import com.shows.as.domain.tupleTypes.TupleTypeRepresentacio;

import javax.persistence.*;
import java.sql.Date;
import java.util.Collection;
import java.util.LinkedHashSet;
import java.util.Set;
/*
Implementació de la classe Representació del paquet domain.classes
 */
//entitat que representa  una representació d'un espectacle
@Entity
@Table(name = "representació", schema = "public", catalog = "postgres")
@IdClass(RepresentacióPK.class)
@Inheritance(strategy=InheritanceType.JOINED)
public class Representació {

    public static final String TAULA = "Representació";

    private Double preu;
    private Date data;
    private Integer nombreseientslliures;
    private String sessió;
    private String nomlocal;
    private String títolesp;
    private Set<Entrada> entradas;
    private Estrena estrena;
    private Espectacle espectacleByTítolesp;
    private Local localByNomlocal;
    private Sessió sessióBySessió;
    private Collection<Seientenrepresentació> seients;



    public Representació() {

    }

    public Representació(Double preu, Date data, Integer nombreseientslliures, String
sessió, String nomlocal, String títolesp) {
        this.preu = preu;
        this.data = data;
        this.nombreseientslliures = nombreseientslliures;
        this.sessió = sessió;
        this.nomlocal = nomlocal;
        this.títolesp = títolesp;
    }

    @Basic
    @Column(name = "preu", nullable = true, insertable = true, updatable = true, precision
= 17)
    public Double getPreu() {
        return preu;
    }

    public void setPreu(Double preu) {
        this.preu = preu;
    }
```

```
    @Basic
    @Column(name = "data", nullable = true, insertable = true, updatable = true)
    public Date getData() {
        return data;
    }

    public void setData(Date data) {
        this.data = data;
    }

    @Basic
    @Column(name = "nombreseientslliures", nullable = true, insertable = true, updatable =
true)
    public Integer getNombreseientslliures() {
        return nombreseientslliures;
    }

    public void setNombreseientslliures(Integer nombreseientslliures) {
        this.nombreseientslliures = nombreseientslliures;
    }

    @Id
    //@Column(name = "sessió", nullable = false, insertable = true, updatable = true,
length = 255)
    public String getSessió() {
        return sessió;
    }

    public void setSessió(String sessió) {
        this.sessió = sessió;
    }

    @Id
    //@Column(name = "nomlocal", nullable = false, insertable = true, updatable = true,
length = 255)
    public String getNomlocal() {
        return nomlocal;
    }

    public void setNomlocal(String nomlocal) {
        this.nomlocal = nomlocal;
    }

    @Basic
    //@Column(name = "títolesp", nullable = true, insertable = true, updatable = true,
length = 255)
    public String getTítolesp() {
        return títolesp;
    }

    public void setTítolesp(String títolesp) {
        this.títolesp = títolesp;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Representació that = (Representació) o;

        if (preu != null ? !preu.equals(that.preu) : that.preu != null) return false;
        if (data != null ? !data.equals(that.data) : that.data != null) return false;
```

```
        if (nombreseientslliures != null ?
!nombreseientslliures.equals(that.nombreseientslliures) : that.nombreseientslliures !=
null)
            return false;
        return true;
    }

    @Override
    public int hashCode() {
        return hashCode(this.nomlocal, this.sessió);
    }

    public static int hashCode(String nomlocal, String sessió){
        String sSurrogate =   String.format("%60s", nomlocal)           // 60 chars
                            + String.format("%60s", sessió);           // 60 chars

        return sSurrogate.hashCode();
    }

    @OneToMany(cascade=CascadeType.ALL, fetch=FetchType.EAGER)
    @JoinColumns({@JoinColumn(name = "sessió", referencedColumnName = "sessió"),
@JoinColumn(name = "nomlocal", referencedColumnName ="nomlocal")})
    public Set<Entrada> getEntradas() {
        return entradas;
    }

    public void setEntradas(Set<Entrada> entradas) {
        this.entradas = entradas;
    }

    @OneToOne(mappedBy = "representació")
    public Estrena getEstrena() {
        return estrena;
    }

    public void setEstrena(Estrena estrena) {
        this.estrena = estrena;
    }

    @ManyToOne
    @JoinColumn(name = "títolesp", referencedColumnName = "títol",insertable = false,
updatable = false)
    public Espectacle getEspectacleByTítolesp() {
        return espectacleByTítolesp;
    }

    public void setEspectacleByTítolesp(Espectacle espectacleByTítolesp) {
        this.espectacleByTítolesp = espectacleByTítolesp;
    }

    @ManyToOne
    @JoinColumn(name = "nomlocal", referencedColumnName = "nom", nullable =
false,insertable = false, updatable = false)
    public Local getLocalByNomlocal() {
        return localByNomlocal;
    }

    public void setLocalByNomlocal(Local localByNomlocal) {
        this.localByNomlocal = localByNomlocal;
    }

    @ManyToOne
    @JoinColumn(name = "sessió", referencedColumnName = "sessió", nullable =
false,insertable = false, updatable = false)
    public Sessió getSessióBySessió() {
        return sessióBySessió;
```

```
    }

    public void setSessióBySessió(Sessió sessióBySessió) {
        this.sessióBySessió = sessióBySessió;
    }

    @OneToMany(cascade=CascadeType.ALL, fetch=FetchType.EAGER)
    @JoinColumns({@JoinColumn(name = "sessió", referencedColumnName = "sessió"),
@JoinColumn(name = "nomlocal", referencedColumnName ="nomlocal")})
    public Collection<Seientenrepresentació> getSeients() {
        return seients;
    }

    public void setSeients(Collection<Seientenrepresentació> seientenrepresentaciós) {
        this.seients = seientenrepresentaciós;
    }

    public boolean esEstrena() {
        return false;
    }

    public void associaEntrada(Entrada entrada) {
        this.entradas.add(entrada);
    }

    public Set<TupleTypeFilaColumna> getSeients(Integer nombEspectadors) {
        if (this.nombreseientslliures < nombEspectadors) throw new
IllegalStateException("seientsNoDisp");
        Set<TupleTypeFilaColumna> seients = new LinkedHashSet<TupleTypeFilaColumna>();
        for (Seientenrepresentació sr : this.seients) {
            seients.add(sr.obteSeientFilaColumna());
        }
        return seients;
    }

    public TupleTypeRepresentacio obteInfo() {
        TupleTypeRepresentacio info = new TupleTypeRepresentacio();
        info.nomLocal = this.nomlocal;
        info.nomSessio = this.sessió;
        info.estrena = esEstrena();
        info.nombreSeientsLliures = this.nombreseientslliures;
        info.preu = this.preu;
        return info;
    }

    public Integer obteRecarrec() {
        return 0;
    }

}
```

2.7.- RepresentacióPK

```
package com.shows.as.domain.classes;

import javax.persistence.Column;
import javax.persistence.Id;
import java.io.Serializable;

/*
Implementació de la classe RepresentacioPK, primary key de la classe Representacio.
 */
//Entitat de Hibernate que representa la primary key de la classe Representació
public class RepresentacióPK implements Serializable {
    private String sessió;
    private String nomlocal;

    public RepresentacióPK() {

    }

    public RepresentacióPK(String nomlocal, String sessió) {
        this.sessió = sessió;
        this.nomlocal = nomlocal;
    }

    @Column(name = "sessió", nullable = false, insertable = true, updatable = true, length
= 255)
    @Id
    public String getSessió() {
        return sessió;
    }

    public void setSessió(String sessió) {
        this.sessió = sessió;
    }

    @Column(name = "nomlocal", nullable = false, insertable = true, updatable = true,
length = 255)
    @Id
    public String getNomlocal() {
        return nomlocal;
    }

    public void setNomlocal(String nomlocal) {
        this.nomlocal = nomlocal;
    }

    @Override
   public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        RepresentacióPK that = (RepresentacióPK) o;

        if (sessió != null ? !sessió.equals(that.sessió) : that.sessió != null) return
false;
        if (nomlocal != null ? !nomlocal.equals(that.nomlocal) : that.nomlocal != null)
return false;

        return true;
    }

    @Override
    public int hashCode() {
```

```
        int result = sessió != null ? sessió.hashCode() : 0;
        result = 31 * result + (nomlocal != null ? nomlocal.hashCode() : 0);
        return result;
    }
}
```

2.8.- Seient

```
package com.shows.as.domain.classes;

import javax.persistence.*;
import java.util.Collection;
import java.util.Set;

/*
Implementació de la classe Seient del paquet domain.classes
 */
//Entitat que representa un seient d'un local
@Entity
@Table(name = "seient", schema = "public", catalog = "postgres")
@IdClass(SeientPK.class)
public class Seient {

    private int fila;
    private int columna;
    private String nomlocal;
    private Local localByNomlocal;
    private Set<Seientenrepresentació> seientenrepresentaciós;


    public Seient() {

    }

    public Seient(int fila, int columna, String nomlocal) {
        this.fila = fila;
        this.columna = columna;
        this.nomlocal = nomlocal;
    }

    @Id
    @Column(name = "fila", nullable = false, insertable = true, updatable = true)
    public int getFila() {
        return fila;
    }

    public void setFila(int fila) {
        this.fila = fila;
    }

    @Id
    @Column(name = "columna", nullable = false, insertable = true, updatable = true)
    public int getColumna() {
        return columna;
    }

    public void setColumna(int columna) {
        this.columna = columna;
    }

    @Id
    @Column(name = "nomlocal", nullable = false, insertable = true, updatable = true,
length = 255)
    public String getNomlocal() {
        return nomlocal;
    }

    public void setNomlocal(String nomlocal) {
        this.nomlocal = nomlocal;
```

```
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Seient that = (Seient) o;

        if (fila != that.fila) return false;
        if (columna != that.columna) return false;
        if (nomlocal != null ? !nomlocal.equals(that.nomlocal) : that.nomlocal != null)
return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = fila;
        result = 31 * result + columna;
        result = 31 * result + (nomlocal != null ? nomlocal.hashCode() : 0);
        return result;
    }

    @ManyToOne
    @JoinColumn(name = "nomlocal", referencedColumnName = "nom", nullable =
false,insertable = false, updatable = false)
    public Local getLocalByNomlocal() {
        return localByNomlocal;
    }

    public void setLocalByNomlocal(Local localByNomlocal) {
        this.localByNomlocal = localByNomlocal;
    }

    @OneToMany(cascade=CascadeType.ALL, fetch=FetchType.EAGER)
    @JoinColumns({@JoinColumn(name = "nomlocal", referencedColumnName = "nomlocal"),
@JoinColumn(name = "fila", referencedColumnName ="fila"), @JoinColumn(name = "columna",
referencedColumnName = "columna")})
    public Set<Seientenrepresentació> getSeientenrepresentaciós() {
        return seientenrepresentaciós;
    }

    public void setSeientenrepresentaciós(Set<Seientenrepresentació>
seientenrepresentaciós) {
        this.seientenrepresentaciós = seientenrepresentaciós;
    }

    public void canviarEstat(Representació r) {
        for (Seientenrepresentació sr : getSeientenrepresentaciós()) {
            sr.canviarOcupat(r);
        }
    }

}
```

2.9.- SeientEnRepresentació

```
package com.shows.as.domain.classes;

import com.shows.as.domain.enums.Estat;
import com.shows.as.domain.tupleTypes.TupleTypeFilaColumna;

import javax.persistence.*;
/*
Implementació de la classe SeientEnRepresentacio del paquet domain.classes
 */
//Entitat que representa un seient durant una representació
@Entity
@Table(name = "seientenrepresentació", schema = "public", catalog = "postgres")
@IdClass(SeientenrepresentacióPK.class)
public class Seientenrepresentació {

    public static final String TAULA = "Seientenrepresentació";

    private String estat;
    private int fila;
    private int columna;
    private String nomlocal;
    private String sessió;
    private String identrada;
    private Entrada entradaByIdentrada;
    private Representació representació;
    private Seient seient;

    public Seientenrepresentació() {

    }

    public Seientenrepresentació(String estat, int fila, int columna, String nomlocal,
String sessió, String identrada) {
        this.estat = estat;
        this.fila = fila;
        this.columna = columna;
        this.nomlocal = nomlocal;
        this.sessió = sessió;
        this.identrada = identrada;
    }

    @Basic
    @Column(name = "estat", nullable = true, insertable = true, updatable = true, length =
255)
    public String getEstat() {
        return estat;
    }

    public void setEstat(String estat) {
        this.estat = estat;
    }

    @Id
    @Column(name = "fila", nullable = false, insertable = true, updatable = true)
    public int getFila() {
        return fila;
    }

    public void setFila(int fila) {
        this.fila = fila;
    }
```

```
    @Id
    @Column(name = "columna", nullable = false, insertable = true, updatable = true)
    public int getColumna() {
        return columna;
    }

    public void setColumna(int columna) {
        this.columna = columna;
    }

    @Id
    @Column(name = "nomlocal", nullable = false, insertable = true, updatable = true,
length = 255)
    public String getNomlocal() {
        return nomlocal;
    }

    public void setNomlocal(String nomlocal) {
        this.nomlocal = nomlocal;
    }

    @Id
    @Column(name = "sessió", nullable = false, insertable = true, updatable = true, length
= 255)
    public String getSessió() {
        return sessió;
    }

    public void setSessió(String sessió) {
        this.sessió = sessió;
    }

    @Basic
    @Column(name = "identrada", nullable = true, insertable = true, updatable = true,
length = 255)
    public String getIdentrada() {
        return identrada;
    }

    public void setIdentrada(String identrada) {
        this.identrada = identrada;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Seientenrepresentació that = (Seientenrepresentació) o;

        if (fila != that.fila) return false;
        if (columna != that.columna) return false;
        if (estat != null ? !estat.equals(that.estat) : that.estat != null) return false;
        if (nomlocal != null ? !nomlocal.equals(that.nomlocal) : that.nomlocal != null)
return false;
        if (sessió != null ? !sessió.equals(that.sessió) : that.sessió != null) return
false;
        if (identrada != null ? !identrada.equals(that.identrada) : that.identrada != null)
return false;

        return true;
    }

    @Override
    public int hashCode() {
        return hashCode(this.nomlocal, this.sessió, this.fila, this.columna);
```

```
    }

    public static int hashCode(String nomlocal, String sessió, Integer fila, Integer
columna){
        String sSurrogate =   String.format("%60s", nomlocal)          // 60 chars
                            + String.format("%60s", sessió)            // 60 chars
                            + String.format("%60s", fila)              // 60 chars
                            + String.format("%60s", columna);          // 60 chars

        return sSurrogate.hashCode();
    }

    @ManyToOne
    @JoinColumn(name = "identrada", referencedColumnName = "identificador",insertable =
false, updatable = false)
    public Entrada getEntradaByIdentrada() {
        return entradaByIdentrada;
    }

    public void setEntradaByIdentrada(Entrada entradaByIdentrada) {
        this.entradaByIdentrada = entradaByIdentrada;
    }

    @ManyToOne
    @JoinColumns({@JoinColumn(name = "sessió", referencedColumnName = "sessió", nullable =
false,insertable = false, updatable = false), @JoinColumn(name = "nomlocal",
referencedColumnName = "nomlocal", nullable = false,insertable = false, updatable =
false)})
    public Representació getRepresentació() {
        return representació;
    }

    public void setRepresentació(Representació representació) {
        this.representació = representació;
    }

    @ManyToOne
    @JoinColumns({@JoinColumn(name = "nomlocal", referencedColumnName = "nomlocal",
nullable = false,insertable = false, updatable = false), @JoinColumn(name = "fila",
referencedColumnName = "fila", nullable = false,insertable = false, updatable = false),
@JoinColumn(name ="columna", referencedColumnName = "columna", nullable = false,insertable
= false, updatable = false)})
    public Seient getSeient() {
        return seient;
    }

    public void setSeient(Seient seient) {
        this.seient = seient;
    }

    public TupleTypeFilaColumna obteSeientFilaColumna() {
        TupleTypeFilaColumna s = new TupleTypeFilaColumna();
        if (estat.equals(Estat.lliure.toString())) {
            s.fila = this.fila;
            s.columna = this.columna;
        }
        return s;
    }

    public void canviarOcupat(Representació r) {
        if (r.getNomlocal().equals(this.nomlocal) && r.getSessió().equals(this.sessió)) {
            this.estat = Estat.ocupat.toString();
        }
    }

}
```

2.10.- SeientEnRepresentacióPK

```
package com.shows.as.domain.classes;

import javax.persistence.Column;
import javax.persistence.Id;
import java.io.Serializable;

/*
Implementació de la classe SeientenrepresentacioPK, primary key de la classe
SeientEnRepresentacio.
 */
//Entitat propia de Hibernate que representa la primary key de la classe
SeienEnRepresentació
public class SeientenrepresentacióPK implements Serializable {

    private int fila;

    @Column(name = "fila", nullable = false, insertable = true, updatable = true)
    @Id
    public int getFila() {
        return fila;
    }

    public void setFila(int fila) {
        this.fila = fila;
    }

    private int columna;

    @Column(name = "columna", nullable = false, insertable = true, updatable = true)
    @Id
    public int getColumna() {
    1.          return columna;
    }

    public void setColumna(int columna) {
        this.columna = columna;
    }

    private String nomlocal;

    @Column(name = "nomlocal", nullable = false, insertable = true, updatable = true,
length = 255)
    @Id
    public String getNomlocal() {
        return nomlocal;
    }

    public void setNomlocal(String nomlocal) {
        this.nomlocal = nomlocal;
    }

    private String sessió;

    @Column(name = "sessió", nullable = false, insertable = true, updatable = true, length
= 255)
    @Id
    public String getSessió() {
        return sessió;
    }

    public void setSessió(String sessió) {
        this.sessió = sessió;
```

```
    }


    public SeientenrepresentacióPK() {

    }

    public SeientenrepresentacióPK(int fila, int columna, String nomlocal, String sessió) {
        this.fila = fila;
        this.columna = columna;
        this.nomlocal = nomlocal;
        this.sessió = sessió;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        SeientenrepresentacióPK that = (SeientenrepresentacióPK) o;

        if (fila != that.fila) return false;
        if (columna != that.columna) return false;
        if (nomlocal != null ? !nomlocal.equals(that.nomlocal) : that.nomlocal != null)
return false;
        if (sessió != null ? !sessió.equals(that.sessió) : that.sessió != null) return
false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = fila;
        result = 31 * result + columna;
        result = 31 * result + (nomlocal != null ? nomlocal.hashCode() : 0);
        result = 31 * result + (sessió != null ? sessió.hashCode() : 0);
        return result;
    }
}
```

2.11.- SeientPK

```
package com.shows.as.domain.classes;

import javax.persistence.Column;
import javax.persistence.Id;
import java.io.Serializable;
/*
Implementació de la classe SeientPK, primary key de la classe Seient.
 */
//Entitat de Hibernate que representa la primary key de la classe seient
public class SeientPK implements Serializable {

    private int fila;
    private int columna;
    private String nomlocal;

    @Column(name = "fila", nullable = false, insertable = true, updatable = true)
    @Id
    public int getFila() {
        return fila;
    }

    public void setFila(int fila) {
        this.fila = fila;
    }

    @Column(name = "columna", nullable = false, insertable = true, updatable = true)
    @Id
    public int getColumna() {
        return columna;
    }

    public void setColumna(int columna) {
        this.columna = columna;
    }

    @Column(name = "nomlocal", nullable = false, insertable = true, updatable = true,
length = 255)
    @Id
    public String getNomlocal() {
        return nomlocal;
    }

    public void setNomlocal(String nomlocal) {
        this.nomlocal = nomlocal;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        SeientPK that = (SeientPK) o;

        if (fila != that.fila) return false;
        if (columna != that.columna) return false;
        if (nomlocal != null ? !nomlocal.equals(that.nomlocal) : that.nomlocal != null)
return false;

        return true;
    }

    @Override
```

```
    public int hashCode() {
        int result = fila;
        result = 31 * result + columna;
        result = 31 * result + (nomlocal != null ? nomlocal.hashCode() : 0);
        return result;
    }
}
```

2.12.- Sessió

```
package com.shows.as.domain.classes;

import javax.persistence.*;
import java.util.Collection;
/*
Implementació de la classe Sessio del paquet domain.classes
 */
//Entitat que representa una sessió
@Entity
@Table(name = "sessió", schema = "public", catalog = "postgres")
public class Sessió {

    private String sessió;
    private Collection<Representació> representaciósBySessió;

    public Sessió() {
    }

    public Sessió(String sessió) {
        this.sessió = sessió;
    }

    @Id
    @Column(name = "sessió", nullable = false, insertable = true, updatable = true, length
= 255)
    public String getSessió() {
        return sessió;
    }

    public void setSessió(String sessió) {
        this.sessió = sessió;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Sessió that = (Sessió) o;

        if (sessió != null ? !sessió.equals(that.sessió) : that.sessió != null) return
false;

        return true;
    }

    @Override
    public int hashCode() {
        return sessió != null ? sessió.hashCode() : 0;
    }

    @OneToMany(mappedBy = "sessióBySessió")
    public Collection<Representació> getRepresentaciósBySessió() {
        return representaciósBySessió;
    }

    public void setRepresentaciósBySessió(Collection<Representació> representaciósBySessió)
{
        this.representaciósBySessió = representaciósBySessió;
    }
}
```

2.13.- Showscom

```java
package com.shows.as.domain.classes;

import com.shows.as.domain.enums.Moneda;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Transient;
/*
Implementació de la classe Showscom del paquet domain.classes. Aquesta classe és singleton.
 */
//Entitat que representa l'usuari Showscom
public class Showscom {

    private static Showscom instance;
    private Integer codiBanc;
    private String numeroCompte;
    private Double comissio;
    private Moneda divisa;
    private Moneda[] canvis;

    public static Showscom getInstance() {
        if (instance == null) {
            instance = new Showscom();
        }
        return instance;
    }

    private Showscom() {

    }

    public Double getComissio() {
        return comissio;
    }

    public Moneda[] getCanvis() {
        return canvis;
    }

    public Integer getCodiBanc() {
        return codiBanc;
    }

    public Moneda getDivisa() {
        return divisa;
    }

    public String getNumeroCompte() {
        return numeroCompte;
    }

    public void setCodiBanc(Integer codiBanc) {
        this.codiBanc = codiBanc;
    }

    public void setNumeroCompte(String numeroCompte) {
        this.numeroCompte = numeroCompte;
    }

    public void setComissio(Double comissio) {
        this.comissio = comissio;
```

```
    }

    public void setDivisa(Moneda divisa) {
        this.divisa = divisa;
    }

    public void setCanvis(Moneda[] canvis) {
        this.canvis = canvis;
    }
}
```

3.- Adaptadors

3.1.- BankServiceAdapter

```
package com.shows.as.domain.adapters;

import com.shows.as.domain.factories.ServiceLocator;
import com.shows.as.service.BankService;

import java.util.Date;

/*
Classe adaptador del servei de pagament
 */
public class BankServiceAdapter implements IBankServiceAdapter {
    public Boolean autoritza(String dni, Integer codiB, String numCompte, Double preu,
Integer codiBancShows, String numCompteShows, DatedAvui){
       ServiceLocator  s = ServiceLocator.getInstance();
        BankService b = s.getBankService();
        return b.autoritza(dni,codiB,numCompte,preu,codiBancShows,numCompteShows,dAvui);
    }

}
```

3.2.- CurrencyConvertorAdapter

```java
package com.shows.as.domain.adapters;

import com.shows.as.domain.enums.Moneda;
import com.shows.as.domain.factories.ServiceLocator;
import com.shows.as.service.CurrencyService;
/*
Classe adaptador del servei de conversió de moneda
 */
public class CurrencyConvertorAdapter implements ICurrencyConvertorAdapter {

    public Double conversionRate(Moneda divisa, Moneda moneda) {
        ServiceLocator s = ServiceLocator.getInstance();
        CurrencyService currencyService = s.getCurrencyService();
        return currencyService.getConversionRate(divisa, moneda);
    }

}
```

3.3.- IBankServiceAdapter

```java
package com.shows.as.domain.adapters;

import java.util.Date;

//interficie de la classe adaptadora del servei de Banc
public interface IBankServiceAdapter {

    Boolean autoritza(String dni, Integer codiB, String numCompte, Double preu, Integer
codiBancShows, String numCompteShows, Date dAvui);

}
```

3.2.- ICurrencyConvertorAdapter

```java
package com.shows.as.domain.adapters;

import com.shows.as.domain.enums.Moneda;

//interficie de classe adaptadora del servei de conversió de moneda
public interface ICurrencyConvertorAdapter {

    Double conversionRate(Moneda divisa, Moneda moneda);

}
```

4.- Factories

4.1.- FactoriaCtrl

```java
package com.shows.as.domain.factories;

import com.shows.as.domain.controllers.CtrlEspectacle;
import com.shows.as.domain.controllers.CtrlRepresentacio;
import com.shows.as.domain.controllers.CtrlSeientEnRepresentacio;
import com.shows.as.persistence.CtrlEspectacleDB;
import com.shows.as.persistence.CtrlRepresentacioDB;
import com.shows.as.persistence.CtrlSeientEnRepresentacioDB;

/*
Implementació de la classe FactoriaCtrl.
 */
//Entitat que representa una factoria de classes controlador
public class FactoriaCtrl {

    private static FactoriaCtrl instance;
    private CtrlEspectacle ctrlEspectacle;
    private CtrlRepresentacio ctrlRepresentacio;
    private CtrlSeientEnRepresentacio ctrlSeientEnRepresentacio;

    public static FactoriaCtrl getInstance() {
        if (instance == null) {
            instance = new FactoriaCtrl();
        }
        return instance;
    }

    private FactoriaCtrl() {

    }

    public CtrlEspectacle getCtrlEspectacle() {
        if (ctrlEspectacle == null) {
            ctrlEspectacle = new CtrlEspectacleDB();
        }
        return ctrlEspectacle;
    }

    public CtrlRepresentacio getCtrlRepresentacio() {
        if (ctrlRepresentacio == null) {
            ctrlRepresentacio = new CtrlRepresentacioDB();
        }
        return ctrlRepresentacio;
    }

    public CtrlSeientEnRepresentacio getCtrlSeientEnRepresentacio() {
        if (ctrlSeientEnRepresentacio == null) {
            ctrlSeientEnRepresentacio = new CtrlSeientEnRepresentacioDB();
        }
        return ctrlSeientEnRepresentacio;
    }

}
```

4.2.- FactoriaServeis

```java
package com.shows.as.domain.factories;

import com.shows.as.domain.adapters.BankServiceAdapter;
import com.shows.as.domain.adapters.CurrencyConvertorAdapter;
import com.shows.as.domain.adapters.IBankServiceAdapter;
import com.shows.as.domain.adapters.ICurrencyConvertorAdapter;
/*
Implementació de la classe FactoriaServeis
 */
//Entitat que representa una factoria de adaptadors de serveis
public class FactoriaServeis {

    private static FactoriaServeis instance;
    private ICurrencyConvertorAdapter iCurrencyConvertorAdapter;
    private IBankServiceAdapter iBankServiceAdapter;

    public static FactoriaServeis getInstance() {
        if (instance == null) {
            instance = new FactoriaServeis();
        }
        return instance;
    }

    private FactoriaServeis() {

    }

    public ICurrencyConvertorAdapter getiCurrencyConvertorAdapter() {
        if (iCurrencyConvertorAdapter == null) {
            iCurrencyConvertorAdapter = new CurrencyConvertorAdapter();
        }
        return iCurrencyConvertorAdapter;
    }

    public IBankServiceAdapter getiBankServiceAdapter() {
        if (iBankServiceAdapter == null) {
            iBankServiceAdapter = new BankServiceAdapter();
        }
        return iBankServiceAdapter;
    }

}
```

4.3.- FactoriaUseCase

```java
package com.shows.as.domain.factories;

import com.shows.as.domain.useCases.ComprarEntrada;
import com.shows.as.domain.useCases.ConsultarOcupacio;
import com.shows.as.domain.useCases.ConsultarRepresentacions;

/*
Implementació de la classe FactoriaUseCase
 */
//Entitat que representa una factoria de casos d'ús
public class FactoriaUseCase {

    private static FactoriaUseCase instance;
    private ConsultarRepresentacions consultarRepresentacions;
    private ConsultarOcupacio consultarOcupacio;
    private ComprarEntrada comprarEntrada;

    public static FactoriaUseCase getInstance() {
        if (instance == null) {
            instance = new FactoriaUseCase();
        }
        return instance;
    }

    private FactoriaUseCase() {

    }

    public ConsultarRepresentacions getConsultarRepresentacions() {
        if (consultarRepresentacions == null) {
            consultarRepresentacions = new ConsultarRepresentacions();
        }
        return consultarRepresentacions;
    }

    public ConsultarOcupacio getConsultarOcupacio() {
        if (consultarOcupacio == null) {
            consultarOcupacio = new ConsultarOcupacio();
        }
        return consultarOcupacio;
    }

    public ComprarEntrada getComprarEntrada() {
        if (comprarEntrada == null) {
            comprarEntrada = new ComprarEntrada();
        }
        return comprarEntrada;
    }

}
```

4.4.- ServiceLocator

```java
package com.shows.as.domain.factories;

import com.shows.as.service.BankService;
import com.shows.as.service.CurrencyService;

//Entitat que representa un localitzador de serveis web que utilitza el sistema
public class ServiceLocator {

    private static ServiceLocator instance;
    private CurrencyService currencyService;
    private BankService bankService;

    /** Implementació del patró Singleton. **/
    public static ServiceLocator getInstance() {
        if (instance == null) {
            instance = new ServiceLocator();
        }
        return instance;
    }

    private ServiceLocator() {

    }


    public CurrencyService getCurrencyService() {
        if (currencyService == null) {
            currencyService = new CurrencyService();
        }
        return currencyService;
    }

    public BankService getBankService() {
        if (bankService == null) {
            bankService = new BankService();
        }
        return bankService;
    }
}
```

5.- TupleTypes

5.1.- TupleTypeFilaColumna

```java
package com.shows.as.domain.tupleTypes;

//Entitat que representa una tupla
public class TupleTypeFilaColumna {

    public Integer fila;
    public Integer columna;

}
```

5.2.- TupleTypeRepresentacio

```java
package com.shows.as.domain.tupleTypes;

import com.shows.as.domain.enums.TipusSessio;

import java.util.Date;

//Entitat que representa una tupla
public class TupleTypeRepresentacio {

    public String nomLocal;
    public String nomSessio;
    public Integer nombreSeientsLliures;
    public Boolean estrena;
    public Double preu;

}
```

5.3.- TupleTypeSeleccioSeients

```java
package com.shows.as.domain.tupleTypes;

import com.shows.as.domain.enums.Moneda;

import java.util.LinkedHashSet;
import java.util.Set;

//Entitat que representa una tupla
public class TupleTypeSeleccioSeients {

    public Double preu;
    public Set<Moneda> canvis;

    public TupleTypeSeleccioSeients() {
        canvis = new LinkedHashSet<Moneda>();
    }

}
```

6.- UseCaseControllers

6.1.- CompraEntradesUseCaseControllers

```java
package com.shows.as.domain.useCaseControllers;

import com.shows.as.domain.classes.Seient;
import com.shows.as.domain.enums.Moneda;
import com.shows.as.domain.enums.TipusSessio;
import com.shows.as.domain.factories.FactoriaUseCase;
import com.shows.as.domain.tupleTypes.TupleTypeFilaColumna;
import com.shows.as.domain.tupleTypes.TupleTypeRepresentacio;
import com.shows.as.domain.tupleTypes.TupleTypeSeleccioSeients;
import com.shows.as.domain.useCases.ComprarEntrada;
import java.sql.Date;
import java.util.Set;

/*
Implementació de la classe ComprarEntradesUseCaseController. Controlador del cas d'ús
comprarEntrades
 */
//Entitat que representa un controlador del cas d'ús comprar entrades
public class ComprarEntradesUseCaseController {

    private ComprarEntrada comprarEntrada;

    public ComprarEntradesUseCaseController() {
        FactoriaUseCase factoriaUseCase = factoriaUseCase = FactoriaUseCase.getInstance();
        comprarEntrada = factoriaUseCase.getComprarEntrada();
    }

    public Set<String> obteEspectacles() {
        return comprarEntrada.obteEspectacles();
    }

    public Set<TupleTypeRepresentacio> obteRepresentacions(String titol, Date data) throws
Exception{
        return comprarEntrada.obteRepresentacions(titol, data);
    }

    public Set<TupleTypeFilaColumna> obteOcupacio(String nomLocal, String sessio, Integer
nombEspectadors) {
        return comprarEntrada.obteOcupacio(nomLocal, sessio, nombEspectadors);
    }

    public TupleTypeSeleccioSeients seleccionarSeients(Set<Seient> seients) {
        return comprarEntrada.seleccionarSeients(seients);
    }

    public Double obtePreuMoneda(Moneda moneda) {
        return comprarEntrada.obtePreuMoneda(moneda);
    }

    public void pagament(String dni, Integer codiB, String numCompte) throws Exception {
        comprarEntrada.pagament(dni, codiB, numCompte);
    }

}
```

7.- UseCases

7.1.- ComprarEntrada

```
package com.shows.as.domain.useCases;

import com.shows.as.domain.adapters.IBankServiceAdapter;
import com.shows.as.domain.adapters.ICurrencyConvertorAdapter;
import com.shows.as.domain.classes.*;
import com.shows.as.domain.controllers.CtrlRepresentacio;
import com.shows.as.domain.controllers.CtrlSeientEnRepresentacio;
import com.shows.as.domain.enums.Moneda;
import com.shows.as.domain.enums.TipusSessio;
import com.shows.as.domain.factories.FactoriaCtrl;
import com.shows.as.domain.factories.FactoriaServeis;
import com.shows.as.domain.factories.FactoriaUseCase;
import com.shows.as.domain.tupleTypes.TupleTypeFilaColumna;
import com.shows.as.domain.tupleTypes.TupleTypeRepresentacio;
import com.shows.as.domain.tupleTypes.TupleTypeSeleccioSeients;
import com.shows.as.domain.utils.Utils;
import com.shows.as.persistence.HibernateUtils;

import java.sql.Date;
import java.util.*;
/*
Implementació del cas d'ús comprarEntrada
 */
//Entitat que representa un cas d'ús de comprar entrada
public class ComprarEntrada {

    public static final String noHiHaRepresentacions = "No Hi Ha Representacions per
l'espectacle i data";
    public static final String pagamentNoAutoritzat = "El pagament no s'autoritza";

    private String titolEsp;
    private Date dataRep;
    private String nLocal;
    private String tSessio;
    private Integer nEspectadors;
    private Set<Seient> seients;
    private Double preuTotal;

    /**
     * @exception IllegalStateException noHiHaEspectacles: No hi ha espectacles
enregistrats al sistema.
     * @return result = obte el titol de tots els espectacles existents al sistema.
     */
    public Set<String> obteEspectacles() {
        FactoriaUseCase factoriaUseCase = FactoriaUseCase.getInstance();
        ConsultarRepresentacions consultarRepresentacions =
factoriaUseCase.getConsultarRepresentacions();

        return consultarRepresentacions.consultaEspectacles();
    }

    /**
     * @pre exiseixEspecacle: L'espectacle amb titol existeix.
     * @pre dataCorrecta: La data es correcta.
     * @exception IllegalStateException noHiHaRepresentacions: No Hi Ha Representacions per
l'espectacle i data.
     * @param titol Titol de l'espectacle.
     * @param data Data de l'espectacle.
```

```
     * @return result = obte els locals, sessions, el nombre de seients lliures, la
indicacio de si es estrena i el preu de totes les representacions.
     * @post emmagatzemaDades: s'emmagatzema a la capa de domini el titol i la data.
     */
    public Set<TupleTypeRepresentacio> obteRepresentacions(String titol, Date data) throws
Exception {
        FactoriaUseCase factoriaUseCase = FactoriaUseCase.getInstance();
        ConsultarRepresentacions consultarRepresentacions =
factoriaUseCase.getConsultarRepresentacions();

        Set<TupleTypeRepresentacio> result =
consultarRepresentacions.consultaRepresentacions(titol, data);

        if (result.isEmpty()) throw new Exception(noHiHaRepresentacions);

        this.titolEsp = titol;
        this.dataRep = data;

        return result;
    }

    /**
     * @pre representacioExisteix: la representacio existeix.
     * @pre nombEspectadorsOK: el nombEspectadors es mes gran que 0.
     * @exception IllegalStateException seientsNoDisp: El nombre d'espectadors es mes gran
que el nombre de seients lliures.
     * @param nomLocal El nom del local.
     * @param sessio El tipus de la sessio.
     * @param nombEspectadors El nombre d'espectadors.
     * @return result = fila i columna de tots els seients disponibles per a aquella
representacio.
     * @post emmagatzemaDades: s'emmagatzema a la capa de domini el nomLocal, sessio i
nombEspectadors.
     */
    public Set<TupleTypeFilaColumna> obteOcupacio(String nomLocal, String sessio, Integer
nombEspectadors) {
        FactoriaUseCase factoriaUseCase = FactoriaUseCase.getInstance();
        ConsultarOcupacio consultarOcupacio = factoriaUseCase.getConsultarOcupacio();

        Set<TupleTypeFilaColumna> result = consultarOcupacio.consultaOcupacio(nomLocal,
sessio, nombEspectadors);

        this.nLocal = nomLocal;
        this.tSessio = sessio;
        this.nEspectadors = nombEspectadors;

        return result;
    }

    /**
     * @pre seientsLliures: els seients estan lliures per la representacio.
     * @param seients Els seients de la representacio.
     * @return result = preu total de l'entrada i les monedes en les que pot demanar la
conversio.
     * @post emmagatzemaDades: s'emmagatzema a la capa de domini els seients i el preu
total de l'entrada.
     */
    public TupleTypeSeleccioSeients seleccionarSeients(Set<Seient> seients) {
        TupleTypeSeleccioSeients result = new TupleTypeSeleccioSeients();
        FactoriaCtrl factoriaCtrl = FactoriaCtrl.getInstance();
        CtrlRepresentacio ctrlRepresentacio = factoriaCtrl.getCtrlRepresentacio();

        Representació r = ctrlRepresentacio.getRepresentacio(this.nLocal, this.tSessio);

        Double p = r.getPreu();
        p += r.obteRecarrec();
```

```
        for (Seient s : seients) {
            s.canviarEstat(r);
        }

        Showscom showscom = Showscom.getInstance();
        result.preu = (this.nEspectadors * p) + showscom.getComissio();
        result.canvis = new LinkedHashSet<Moneda>(Arrays.asList(showscom.getCanvis()));

        this.seients = seients;
        this.preuTotal = result.preu;

        return result;
    }

    /**
     * @pre monedaExisteix: la moneda es un dels canvis permesos.
     * @exception IllegalStateException serveiNoDisponible: el servei no esta disponible.
     * @param moneda La moneda a realitzar la conversio.
     * @post el sistema crida a l'operacio <code>conversionRate(divisa,
moneda):Float</code> del servei currency convertor per obtenir el canvi entre l'euro i la
moneda.
     * @return result = preu (emmagatzemat a la capa de domini) * canvi obtingut al servei
anterior.
     */
    public Double obtePreuMoneda(Moneda moneda) {
        FactoriaServeis factoriaServeis = FactoriaServeis.getInstance();
        ICurrencyConvertorAdapter iCurrencyConvertorAdapter =
factoriaServeis.getiCurrencyConvertorAdapter();

        Showscom showscom = Showscom.getInstance();
        Moneda divisa = showscom.getDivisa();

        Double conversio = iCurrencyConvertorAdapter.conversionRate(divisa, moneda);

        return this.preuTotal * conversio;
    }

    /**
     * @exception IllegalStateException serveiNoDisponible: el servei no esta disponible o
no autoritza el pagament.
     * @exception IllegalStateException pagamentNoAutoritzat: El pagament no s'autoritza.
     * @param dni El dni de la persona que paga.
     * @param codiB El codi bancari de la persona que paga.
     * @param numCompte El numero de compte de la persona que paga.
     * @post pagament: el sistema crida a l'operacio <code>autoritza(dni, codiB, numCompte,
import, codiBancShows, numCompteShows, dAvui):Boolean</code>
     *                  del servei Bank Service amb les dades dels comptes per fer la
transferència i el preu total de l'entrada.
     * @post creacioEntrada: si el pagament s'ha produït amb èxit, es crea una instància de
entrada i de les seves associacions amb la representació i els seients de la representació.
     */
    public void pagament(String dni, Integer codiB, String numCompte) throws Exception {
        Showscom showscom = Showscom.getInstance();
        Integer cb = showscom.getCodiBanc();
        String nc = showscom.getNumeroCompte();

        FactoriaServeis factoriaServeis = FactoriaServeis.getInstance();
        IBankServiceAdapter iBankServiceAdapter = factoriaServeis.getiBankServiceAdapter();

        Date dt = new Date(Calendar.getInstance().getTimeInMillis());

        Boolean b = iBankServiceAdapter.autoritza(dni, codiB, numCompte, this.preuTotal,
cb, nc, dt);

        if (!b) throw new Exception(pagamentNoAutoritzat);
```

```
        String id = Utils.random();

        FactoriaCtrl factoriaCtrl = FactoriaCtrl.getInstance();
        CtrlRepresentacio ctrlRepresentacio = factoriaCtrl.getCtrlRepresentacio();
        CtrlSeientEnRepresentacio ctrlSeientEnRepresentacio =
factoriaCtrl.getCtrlSeientEnRepresentacio();

        Representació r = ctrlRepresentacio.getRepresentacio(this.nLocal, this.tSessio);

        Set<Seientenrepresentació> seientRep = new LinkedHashSet<Seientenrepresentació>();
        for (Seient seient : this.seients) {
            seientRep.add(ctrlSeientEnRepresentacio.getSeientEnRepresentacio(this.nLocal,
this.tSessio, seient.getFila(), seient.getColumna()));
        }

        Entrada e = new Entrada(id, dni, this.nEspectadors, dt, this.preuTotal);
        e.associa(r, seientRep);
        HibernateUtils.save(e);
    }

}
```

7.2.- ConsultarOcupacio

```java
package com.shows.as.domain.useCases;

import com.shows.as.domain.classes.Representació;
import com.shows.as.domain.controllers.CtrlRepresentacio;
import com.shows.as.domain.enums.TipusSessio;
import com.shows.as.domain.factories.FactoriaCtrl;
import com.shows.as.domain.tupleTypes.TupleTypeFilaColumna;

import java.util.Set;
/*
Implementació del cas d'ús consultarOcupació
 */
//Entitat que representa el cas d'ús consultar Ocupació
public class ConsultarOcupacio {

    /**
     * @pre representacioExisteix: la representacio existeix.
     * @pre nombEspectadorsOK: el nombEspectadors es mes gran que 0.
     * @exception IllegalStateException seientsNoDisp: El nombre d'espectadors es mes gran
que el nombre de seients lliures.
     * @param nomLocal El nom del local.
     * @param sessio El tipus de la sessio.
     * @param nombEspectadors El nombre d'espectadors.
     * @return result = fila i columna de tots els seients disponibles per a aquella
representacio.
     */
    public Set<TupleTypeFilaColumna> consultaOcupacio(String nomLocal, String sessio,
Integer nombEspectadors) {
        FactoriaCtrl factoriaCtrl = FactoriaCtrl.getInstance();
        CtrlRepresentacio ctrlRepresentacio = factoriaCtrl.getCtrlRepresentacio();

        Representació r = ctrlRepresentacio.getRepresentacio(nomLocal, sessio);
        return r.getSeients(nombEspectadors);
    }

}
```

7.3.- ConsultarRepresentacions

```java
package com.shows.as.domain.useCases;

import com.shows.as.domain.classes.Espectacle;
import com.shows.as.domain.controllers.CtrlEspectacle;
import com.shows.as.domain.factories.FactoriaCtrl;
import com.shows.as.domain.tupleTypes.TupleTypeRepresentacio;
import java.sql.Date;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;

/*
Implementació del cas d'ús consultarRepresentacions
 */
//Entitat que representa el cas dús consultar Representacions
public class ConsultarRepresentacions {

    public static final String noHiHaEspectacles = "No hi ha espectacles enregistrats al
sistema";

    /**
     * @exception IllegalStateException noHiHaEspectacles: No hi ha espectacles
enregistrats al sistema.
     * @return result = obte el titol de tots els espectacles existents al sistema.
     */
    public Set<String> consultaEspectacles() {
        FactoriaCtrl factoriaCtrl = FactoriaCtrl.getInstance();
        CtrlEspectacle ctrlEspectacle = factoriaCtrl.getCtrlEspectacle();
        List<Espectacle> espectacles = ctrlEspectacle.getAll();

        if (espectacles.size() == 0) throw new IllegalStateException(noHiHaEspectacles);

        Set<String> result = new LinkedHashSet<String>();
        for (Espectacle e : espectacles){
            result.add(e.getTítol());
        }

        return result;
    }

    /**
     * @pre exiseixEspecacle: L'espectacle amb titol existeix.
     * @pre dataCorrecta: La data es correcta.
     * @param titol Titol de l'espectacle.
     * @param data Data de l'espectacle.
     * @return result = obte els locals, sessions, el nombre de seients lliures, la
indicacio de si es estrena i el preu de totes les representacions.
     */
    public Set<TupleTypeRepresentacio> consultaRepresentacions(String titol, Date data) {
        FactoriaCtrl factoriaCtrl = FactoriaCtrl.getInstance();
        CtrlEspectacle ctrlEspectacle = factoriaCtrl.getCtrlEspectacle();

        Espectacle espectacle = ctrlEspectacle.getEspectacle(titol);

        return espectacle.obteRepresentacions(data);
    }
}
```

8.- Utils

8.1.- ComboItem

```java
package com.shows.as.domain.utils;

//Entitat que representa un ComboItem
public class ComboItem
{
    private String key;
    private String value;

    public ComboItem(String key, String value)
    {
        this.key = key;
        this.value = value;
    }

    @Override
    public String toString()
    {
        return key;
    }

    public String getKey()
    {
        return key;
    }

    public String getValue()
    {
        return value;
    }
}
```

8.2.- Utils

```java
package com.shows.as.domain.utils;

import java.util.Calendar;
import java.sql.Date;
import java.util.Random;

/*
Implementacio de la classe Utils. Aquesta classe conté diferents mètodes necessàris per
l'execució.
 */
//Classe que incorpora funcions utils per al funcionament del sistema
public class Utils {

    private final static Random generator = new Random();

    public static String random() {
        return Long.toString(generator.nextLong());
    }

    public static int randomInt(){return generator.nextInt();}

    public static Boolean sameDay(Date d1, Date d2) {
        int day, month, year;
        Calendar calendar = Calendar.getInstance();
        calendar.setTime(d1);
        day = calendar.get(Calendar.DAY_OF_MONTH);
        month = calendar.get(Calendar.MONTH);
        year = calendar.get(Calendar.YEAR);
        calendar.setTime(d2);
        return (day == calendar.get(Calendar.DAY_OF_MONTH) &&
                month == calendar.get(Calendar.MONTH) &&
                year == calendar.get(Calendar.YEAR));
    }

    public static Date createDate(int day, int month, int year) {
        Calendar cDate = Calendar.getInstance();
        cDate.set(Calendar.DATE, day);
        cDate.set(Calendar.MONTH, month);
        cDate.set(Calendar.YEAR, year);
        return new Date(cDate.getTimeInMillis());
    }

}
```

9.- Services

9.1.- BankService

```java
package com.shows.as.service;

import com.shows.as.domain.utils.Utils;

import java.util.Date;


//Entitat que simula el comportament del servei web de pagament
public class BankService {

    public Boolean autoritza(String dni, Integer codiB, String numCompte, Double preu,
Integer codiBancShows, String numCompteShows, Date dAvui){
        return (Utils.randomInt() % 4 != 0);
    }
}
```

9.2.- CurrencyService

```java
package com.shows.as.service;

import com.shows.as.domain.enums.Moneda;

//Entitat que simula el comportament d'un servei web de canvi de divisa
public class CurrencyService {

    private static final Double eurTOusd = 1.08;
    private static final Double usdTOeur = 0.92;
    private static final Double eurTOgbp = 0.75;
    private static final Double gbpTOeur = 1.33;
    private static final Double usdTOgbp = 0.69;
    private static final Double gbpTOusd = 1.44;

    public Double getConversionRate(Moneda fromCurrency, Moneda toCurrency) {
        if (fromCurrency == Moneda.EUR && toCurrency == Moneda.USD) return eurTOusd;
        if (toCurrency == Moneda.EUR && fromCurrency == Moneda.USD) return usdTOeur;
        if (fromCurrency == Moneda.EUR && toCurrency == Moneda.GBP) return eurTOgbp;
        if (toCurrency == Moneda.EUR && fromCurrency == Moneda.GBP) return gbpTOeur;
        if (fromCurrency == Moneda.USD && toCurrency == Moneda.GBP) return usdTOgbp;
        if (toCurrency == Moneda.USD && fromCurrency == Moneda.GBP) return gbpTOusd;
        return 1.0;
    }
}
```

10.- Enums

10.1.- Estat

```java
package com.shows.as.domain.enums;

//Entitat que representa una enumeració de tipus de estat de seient
public enum Estat {
    ocupat,
    lliure;
}
```

10.2.- Moneda

```java
package com.shows.as.domain.enums;

//Entitat que representa les diferents divises disponibles al sistema
public enum Moneda {
    EUR,
    USD,
    GBP
}
```

10.3.- TipusSessio

```java
package com.shows.as.domain.enums;

//Entitat que representa els diferents tipus de sessió disponibles
public enum TipusSessio {
    mati,
    tarda,
    nit
}
```

11.- PresentationControllers

11.1.- ComprarEntradesController

```java
package com.shows.as.domain.controllers;

import com.shows.as.domain.classes.Espectacle;

import java.util.List;
import java.util.Set;

/*
Implementació de la classe CtrlEspectacle del paquet domain.controllers
 */
//Entitat que representa un controlador de la classe Espectacle
public interface CtrlEspectacle {

    List<Espectacle> getAll();

    Espectacle getEspectacle(String titol);

}
```

12.- Views

12.1.- ComprarEntradesView

```
package com.shows.as.presentation.views;

import com.shows.as.domain.tupleTypes.TupleTypeRepresentacio;
import com.shows.as.domain.utils.Utils;
import com.shows.as.presentation.controllers.ComprarEntradesController;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.Set;

public class ComprarEntradesView extends JDialog {
    private JPanel contentPane;
    private JButton buttonOK;
    private JButton buttonCancel;
    private JComboBox espectaclesComboBox;
    private JComboBox comboBox2;
    private JComboBox comboBox3;
    private JComboBox comboBox4;
    private JComboBox comboBox5;
    private JComboBox comboBox6;
    private JLabel local;
    private JLabel entrades;
    private ComprarEntradesController controller;

    public ComprarEntradesView(ComprarEntradesController controller) {
        this.controller = controller;
        buttonOK.setEnabled(false);
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        int width = (int)screenSize.getWidth();
        int height = (int)screenSize.getHeight();
        setBounds((width / 2) - 225, (height / 2) - 150, 450, 300);
        setResizable(false);
        setContentPane(contentPane);
        setModal(true);
        getRootPane().setDefaultButton(buttonOK);

        //posem com a invisible tot lo de representació
        comboBox5.setVisible(false);
        comboBox6.setVisible(false);
        local.setVisible(false);
        entrades.setVisible(false);

        //inicialitzem dates i entrades
        espectaclesComboBox.addItem("-");
        comboBox2.addItem("-");
        comboBox3.addItem("-");
        comboBox4.addItem("-");
        comboBox5.addItem("-");
        for(int i = 1; i <= 31;++i){
            comboBox2.addItem(i);
            if(i < 13) comboBox3.addItem(i);
            comboBox6.addItem(i);
        }
        comboBox4.addItem(2016);

        buttonOK.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
```

```
                onOK();
            }
        });

        buttonCancel.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                onCancel();
            }
        });

// call onCancel() when cross is clicked
        setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                onCancel();
            }
        });

// call onCancel() on ESCAPE
        contentPane.registerKeyboardAction(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                onCancel();
            }
        }, KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0),
JComponent.WHEN_ANCESTOR_OF_FOCUSED_COMPONENT);
        comboBox2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                comprobaQualitatDades();
            }
        });
        comboBox3.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                comprobaQualitatDades();
            }
        });
        comboBox4.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                comprobaQualitatDades();
            }
        });
        espectaclesComboBox.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                comprobaQualitatDades();
            }
        });
        comboBox5.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                comprovaRepresentacio();
            }
        });

    }

    private void comprobaQualitatDades(){
        if(comboBox2.getSelectedIndex() != 0 && comboBox3.getSelectedIndex()!= 0
                && comboBox4.getSelectedIndex() != 0 &&
espectaclesComboBox.getSelectedIndex()!=0){
            Integer day = (Integer) comboBox2.getSelectedItem();
            Integer month = (Integer) comboBox3.getSelectedItem();
            Integer year = (Integer) comboBox4.getSelectedItem();

controller.prOkConsultaRepresentacions((String)espectaclesComboBox.getSelectedItem(),
Utils.createDate(day, month, year));
        }
    }
```

```
    private void onOK() {
        String word = (String)comboBox5.getSelectedItem();
        String[] s = word.split(" : ");
        controller.prOkConsultaOcupacio(s[0], s[1], (Integer)comboBox6.getSelectedItem());
        dispose();
    }

    private void onCancel() {
// add your code here if necessary
        controller.prCancel();
        dispose();
    }

    public static void main(String[] args) {
        //ComprarEntradesView dialog = new ComprarEntradesView();
        //dialog.pack();
        //dialog.setVisible(true);
        //System.exit(0);
    }

    public void mostraEspectacles(Set<String> espectacles) {
        if (espectacles.isEmpty()) {
            // TODO Mostrar Error View
            return;
        }
        for (String s : espectacles) {
            espectaclesComboBox.addItem(s);
        }

    }

    public void amaga() {
        comboBox5.setVisible(false);
        comboBox6.setVisible(false);
        local.setVisible(false);
        entrades.setVisible(false);
    }


    public void comprovaRepresentacio() {
        if (comboBox5.getSelectedIndex() != 0) buttonOK.setEnabled(true);
        else buttonOK.setEnabled(false);
    }

    public void mostraRepresentacions(Set<TupleTypeRepresentacio> tupleTypeRepresentacios)
{
        comboBox5.removeAllItems();
        comboBox5.addItem("-");
        for (TupleTypeRepresentacio t : tupleTypeRepresentacios) {
            comboBox5.addItem(t.nomLocal + " : " + t.nomSessio);
        }
        comboBox5.setVisible(true);
        comboBox6.setVisible(true);
        local.setVisible(true);
        entrades.setVisible(true);
    }
}
```

12.2.- ErrorView

```
package com.shows.as.presentation.views;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class ErrorView extends JDialog {
    private JPanel contentPane;
    private JButton buttonOK;
    private JLabel errorText;

    public ErrorView() {
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        int width = (int)screenSize.getWidth();
        int height = (int)screenSize.getHeight();
        setBounds((width/2)-225,(height/2)-150,450,300);
        setResizable(false);
        setContentPane(contentPane);
        setModal(true);
        getRootPane().setDefaultButton(buttonOK);
        buttonOK.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                onOK();
            }
        });

        setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                onCancel();
            }
        });

        contentPane.registerKeyboardAction(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                onCancel();
            }
        }, KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0),
JComponent.WHEN_ANCESTOR_OF_FOCUSED_COMPONENT);
    }

    private void onOK() {
        dispose();
    }

    private void onCancel() {
        dispose();
    }

    public static void main(String[] args) {
        ErrorView dialog = new ErrorView();
        dialog.pack();
        dialog.setVisible(true);
        System.exit(0);
    }

    public void mostraMissatgeError(String text) {
        errorText.setText(text);
    }
}
```

12.3.- IniView

```
package com.shows.as.presentation.views;

import com.shows.as.presentation.controllers.ComprarEntradesController;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class IniView extends JDialog {
    private JPanel contentPane;
    private JButton buttonOK;
    private ComprarEntradesController controller;

    public IniView(ComprarEntradesController controller) {
        this.controller = controller;
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        int width = (int)screenSize.getWidth();
        int height = (int)screenSize.getHeight();
        setBounds((width / 2) - 225, (height / 2) - 150, 450, 300);
        setResizable(false);
        setContentPane(contentPane);
        setModal(true);
        getRootPane().setDefaultButton(buttonOK);

        buttonOK.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                onOK();
            }
        });

        // call onCancel() when cross is clicked
        setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                onCancel();
            }
        });

        // call onCancel() on ESCAPE
        contentPane.registerKeyboardAction(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                onCancel();
            }
        }, KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0),
JComponent.WHEN_ANCESTOR_OF_FOCUSED_COMPONENT);
    }

    private void onOK() {
        controller.prConsultaEspectacles();
    }

    private void onCancel() {
        // add your code here if necessary
        dispose();
    }

    public static void main(String[] args) {

    }
}
```

12.4.- PagamentView

```
package com.shows.as.presentation.views;

import com.shows.as.domain.enums.Moneda;
import com.shows.as.domain.tupleTypes.TupleTypeSeleccioSeients;
import com.shows.as.domain.utils.ComboItem;
import com.shows.as.presentation.controllers.ComprarEntradesController;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class PagamentView extends JDialog {
    private JPanel contentPane;
    private JButton buttonOK;
    private JButton buttonCancel;
    private JTextField textField1;
    private JTextField textField2;
    private JTextField textField3;
    private JComboBox comboBox1;
    private JLabel preu;
    private ComprarEntradesController controller;

    public PagamentView(final ComprarEntradesController controller) {
        this.controller = controller;
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        int width = (int)screenSize.getWidth();
        int height = (int)screenSize.getHeight();
        setBounds((width/2)-225,(height/2)-150,450,300);
        setResizable(false);
        setContentPane(contentPane);
        setModal(true);
        getRootPane().setDefaultButton(buttonOK);

        buttonOK.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                onOK();
            }
        });

        buttonCancel.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                onCancel();
            }
        });

        comboBox1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                controller.prCanviMoneda((Moneda) comboBox1.getSelectedItem());
            }
        });

        setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                onCancel();
            }
        });

        contentPane.registerKeyboardAction(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                onCancel();
```

```
            }
        }, KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0),
JComponent.WHEN_ANCESTOR_OF_FOCUSED_COMPONENT);

    }

    private void onOK() {
        if (textField1.getText().isEmpty() || textField2.getText().isEmpty() ||
textField3.getText().isEmpty()) {
            controller.prMostraMissatgeError("Omple tots els camps");
            return;
        }
        controller.prOkPagament(textField1.getText(),
Integer.parseInt(textField2.getText()), textField3.getText());
    }

    private void onCancel() {
        dispose();
    }

    public static void main(String[] args) {

    }

    public void seientsSeleccionats(TupleTypeSeleccioSeients tupleTypeSeleccioSeients) {
        preu.setText(tupleTypeSeleccioSeients.preu.toString());
        comboBox1.addItem(Moneda.EUR);
        for (Moneda m : tupleTypeSeleccioSeients.canvis) {
            comboBox1.addItem(m);
        }
    }

    public void mostraPreu(Double preuMoneda) {
        preu.setText(preuMoneda.toString());
    }
}
```

12.5.- SeientsView

```
package com.shows.as.presentation.views;

import com.shows.as.domain.classes.Seient;
import com.shows.as.domain.tupleTypes.TupleTypeFilaColumna;
import com.shows.as.presentation.controllers.ComprarEntradesController;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.LinkedHashSet;
import java.util.Set;

public class SeientsView extends JDialog {
    private JPanel contentPane;
    private JButton buttonOK;
    private JButton buttonCancel;
    private JCheckBox c00;
    private JCheckBox c01;
    private JCheckBox c02;
    private JCheckBox c03;
    private JCheckBox c04;
    private JCheckBox c05;
    private JCheckBox c06;
    private JCheckBox c07;
    private JCheckBox c10;
    private JCheckBox c11;
    private JCheckBox c12;
    private JCheckBox c13;
    private JCheckBox c14;
    private JCheckBox c15;
    private JCheckBox c16;
    private JCheckBox c17;
    private JCheckBox c20;
    private JCheckBox c21;
    private JCheckBox c22;
    private JCheckBox c23;
    private JCheckBox c24;
    private JCheckBox c25;
    private JCheckBox c26;
    private JCheckBox c27;
    private JCheckBox c30;
    private JCheckBox c31;
    private JCheckBox c32;
    private JCheckBox c33;
    private JCheckBox c34;
    private JCheckBox c35;
    private JCheckBox c36;
    private JCheckBox c37;

    private JCheckBox[][] buttons = {{c00, c01, c02, c03, c04, c05, c06, c07}, {c10, c11,
c12, c13, c14, c15, c16, c17}, {c20, c21, c22, c23, c24, c25, c26, c27}, {c30, c31, c32,
c33, c34, c35, c36, c37}};
    private int nEspectadors;
    private String nomlocal;
    private ComprarEntradesController controller;

    public SeientsView(ComprarEntradesController controller) {
        this.controller = controller;
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        int width = (int)screenSize.getWidth();
        int height = (int)screenSize.getHeight();
```

```
        setBounds((width/2)-225,(height/2)-150,450,300);
        setResizable(false);
        setContentPane(contentPane);
        setModal(true);
        getRootPane().setDefaultButton(buttonOK);

        buttonOK.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                onOK();
            }
        });

        buttonCancel.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                onCancel();
            }
        });

        setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                onCancel();
            }
        });

        contentPane.registerKeyboardAction(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                onCancel();
            }
        }, KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0),
JComponent.WHEN_ANCESTOR_OF_FOCUSED_COMPONENT);
    }

    public void repInfo(int nEspectadors, String nomlocal) {
        this.nEspectadors = nEspectadors;
        this.nomlocal = nomlocal;
    }

    private void onOK() {
        Set<Seient> seients = new LinkedHashSet<Seient>();
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 8; j++) {
                if (buttons[i][j].isSelected()) seients.add(new Seient(i, j,
this.nomlocal));
            }
        }

        if (seients.size() != this.nEspectadors) {
            controller.prMostraMissatgeError("Has de seleccionar " + this.nEspectadors + "
seients");
            return;
        }
        controller.prOkSeleccionarSeients(seients);
        dispose();
    }

    private void onCancel() {
        dispose();
    }

    public static void main(String[] args) {

    }

    public void mostraSeients(Set<TupleTypeFilaColumna> tupleTypeFilaColumnas) {
        for (int i = 0; i < 4; i++) {
```

```
                for (int j = 0; j < 8; j++) {
                    buttons[i][j].setEnabled(false);
                }
            }
            for (TupleTypeFilaColumna t : tupleTypeFilaColumnas) {
                buttons[t.fila][t.columna].setEnabled(true);
            }
        }
    }
```

13.- Controllers

13.1.- CtrlEspectacle

```java
package com.shows.as.domain.controllers;

import com.shows.as.domain.classes.Espectacle;

import java.util.List;
import java.util.Set;

/*
Implementació de la classe CtrlEspectacle del paquet domain.controllers
 */
//Entitat que representa un controlador de la classe Espectacle
public interface CtrlEspectacle {

    List<Espectacle> getAll();

    Espectacle getEspectacle(String titol);

}
```

13.2.- CtrlRepresentacio

```java
package com.shows.as.domain.controllers;

import com.shows.as.domain.classes.Representació;
import com.shows.as.domain.enums.TipusSessio;

import java.util.List;
import java.util.Set;
/*
Implementacio de la classe CtrlSeientEnRepresentacio del paquet domain.controllers
 */
//Entitat que representa un controlador de la classe representació
public interface CtrlRepresentacio {

    Set<Representació> getAll();

    Representació getRepresentacio(String nomLocal, String sessio);

}
```

13.3.- CtrlSeientEnRepresentacio

```java
package com.shows.as.domain.controllers;

import com.shows.as.domain.classes.Seientenrepresentació;
import com.shows.as.domain.enums.TipusSessio;

import java.util.Set;
/*
Implementació de la classe CtrlSeientEnRepresentacio del paquet domain.controllers
 */
//Entitat que reprenta un controlador de la classe SeientEnRepresentació
public interface CtrlSeientEnRepresentacio {

    Set<Seientenrepresentació> getAll();

    Seientenrepresentació getSeientEnRepresentacio(String nomLocal, String sessio, Integer
fila, Integer columna);

}
```

14.- Persistence

14.1.- CtrlEspectacleDB

```
package com.shows.as.persistence;

import com.shows.as.domain.classes.Espectacle;
import com.shows.as.domain.controllers.CtrlEspectacle;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

import java.util.List;
import java.util.Set;

//Entitat que representa un controlador de la classe espectacle a la base de dades
public class CtrlEspectacleDB implements CtrlEspectacle {

    public List<Espectacle> getAll() {
        SessionFactory sf = HibernateUtils.getSessionFactory();
        Session session = sf.openSession();

        List<Espectacle> res = (List<Espectacle>) session.createQuery("from "+
Espectacle.TAULA).list();
        session.close();
        return res;
    }

    public Espectacle getEspectacle(String titol) {
        SessionFactory sf = HibernateUtils.getSessionFactory();
        Session session = sf.openSession();

        Espectacle res = (Espectacle) session.get(Espectacle.class, titol);
        session.close();
        if (res == null)
            throw new IllegalStateException("espectacleNoExisteix");
        return res;
    }
}
```

14.2.- CtrlRepresentacioDB

```
package com.shows.as.persistence;

import com.shows.as.domain.classes.Espectacle;
import com.shows.as.domain.classes.Representació;
import com.shows.as.domain.classes.RepresentacióPK;
import com.shows.as.domain.controllers.CtrlRepresentacio;
import com.shows.as.domain.enums.TipusSessio;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

import java.util.List;
import java.util.Set;
//Entitat que representa un controlador de la classe Representació a la base de dades
public class CtrlRepresentacioDB implements CtrlRepresentacio {
    public Set<Representació> getAll() {
        SessionFactory sf = HibernateUtils.getSessionFactory();
        Session session = sf.openSession();

        Set<Representació> res = (Set<Representació>) session.createQuery("from "+
Representació.TAULA).list();
        session.close();
        return res;
    }

    public Representació getRepresentacio(String nomLocal, String sessio) {
        SessionFactory sf = HibernateUtils.getSessionFactory();
        Session session = sf.openSession();

        Representació res = (Representació) session.get(Representació.class, new
RepresentacióPK(nomLocal, sessio));
        session.close();
        if (res == null)
            throw new IllegalStateException("representacioNoExisteix");
        return res;
    }

}
```

14.3.- CtrlSeientEnRepresentacioDB

```
package com.shows.as.persistence;

import com.shows.as.domain.classes.Representació;
import com.shows.as.domain.classes.Seientenrepresentació;
import com.shows.as.domain.classes.SeientenrepresentacióPK;
import com.shows.as.domain.controllers.CtrlSeientEnRepresentacio;
import com.shows.as.domain.enums.TipusSessio;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

import java.util.Set;

//Entitat que representa un controlador de la classe SeientenRepresentació de la base de
dades
public class CtrlSeientEnRepresentacioDB implements CtrlSeientEnRepresentacio {
    public Set<Seientenrepresentació> getAll() {
        SessionFactory sf = HibernateUtils.getSessionFactory();
        Session session = sf.openSession();

        Set<Seientenrepresentació> res = (Set<Seientenrepresentació>)
session.createQuery("from "+ Seientenrepresentació.TAULA).list();
        session.close();
        return res;
    }

    public Seientenrepresentació getSeientEnRepresentacio(String nomLocal, String sessio,
Integer fila, Integer columna) {
        SessionFactory sf = HibernateUtils.getSessionFactory();
        Session session = sf.openSession();

        Seientenrepresentació res = (Seientenrepresentació)
session.get(Seientenrepresentació.class, new SeientenrepresentacióPK(fila, columna,
nomLocal, sessio));
        session.close();
        if (res == null)
            throw new IllegalStateException("seientenrepresentacioNoExisteix");
        return res;
    }
}
```

14.4.- HibernateUtils

```
package com.shows.as.persistence;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.AnnotationConfiguration;

//Classe amb funcions utils per a l'ús de l'eina Hibernate
public class HibernateUtils {

    private static final SessionFactory sessionFactory = buildSessionFactory();

    private static SessionFactory buildSessionFactory() {
        try {
            return new AnnotationConfiguration()
                    .configure()
                    .buildSessionFactory();
        } catch (Throwable ex) {
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }

    public static Object save(Object o) {
        SessionFactory sf = HibernateUtils.getSessionFactory();
        Session session = sf.openSession();

        session.beginTransaction();

        session.save(o);

        session.getTransaction().commit();

        session.close();

        return o;
    }

    public static Object update(Object o) {
        SessionFactory sf = HibernateUtils.getSessionFactory();
        Session session = sf.openSession();

        session.beginTransaction();

        session.merge(o);
```

```
        session.getTransaction().commit();

        session.close();
        return o;

    }

    public static void delete(Object o) {
        SessionFactory sf = HibernateUtils.getSessionFactory();
        Session session = sf.openSession();

        session.beginTransaction();

        session.delete(o);

        session.getTransaction().commit();

        session.close();

    }

    public static int emptyTable(String myTable){
        SessionFactory sf = HibernateUtils.getSessionFactory();
        Session session = sf.openSession();

        session.beginTransaction();

        String hql = String.format("delete from %s",myTable);
        Query query = session.createQuery(hql);
        int res =  query.executeUpdate();
        session.getTransaction().commit();

        session.close();

        return res;

    }
}
```

15.- Joc de Proves

15.1.- Joc de Proves 1

```
        Showscom showscom = Showscom.getInstance();
         showscom.setComissio(1.02);
         showscom.setCodiBanc(23);
         showscom.setNumeroCompte("123456789");
         showscom.setDivisa(Moneda.EUR);
         showscom.setCanvis(new Moneda[]{Moneda.GBP, Moneda.USD});

         HibernateUtils.save(new Espectacle("Cats", 10));
         HibernateUtils.save(new Espectacle("El Rey Leon", 20));
         HibernateUtils.save(new Espectacle("La Sirenita", 20));
         HibernateUtils.save(new Espectacle("Frozen", 20));
         HibernateUtils.save(new Espectacle("La Cantina", 20));
         HibernateUtils.save(new Sessió(TipusSessio.mati.toString()));
         HibernateUtils.save(new Sessió(TipusSessio.tarda.toString()));
         HibernateUtils.save(new Sessió(TipusSessio.nit.toString()));
         HibernateUtils.save(new Local("Cyrano", "Aribau 21"));
         HibernateUtils.save(new Local("Bling Bling", "Tusset 49"));
         HibernateUtils.save(new Local("Apolo", "Paralel 845"));
         for (int i = 0; i < 4; i++) {
             for (int j = 0; j < 8; j++) {
                 HibernateUtils.save(new Seient(i, j, "Cyrano"));
             }
         }
         for (int i = 0; i < 4; i++) {
             for (int j = 0; j < 8; j++) {
                 HibernateUtils.save(new Seient(i, j, "Bling Bling"));
             }
         }
         for (int i = 0; i < 4; i++) {
             for (int j = 0; j < 8; j++) {
                 HibernateUtils.save(new Seient(i, j, "Apolo"));
             }
         }
         HibernateUtils.save(new Representació(50.0, Utils.createDate(14, 1, 2016), 32,
"mati", "Cyrano", "Cats"));
         HibernateUtils.save(new Representació(60.0, Utils.createDate(14, 1, 2016), 32,
"nit", "Cyrano", "El Rey Leon"));
         HibernateUtils.save(new Representació(70.0, Utils.createDate(15, 1, 2016), 32,
"tarda", "Cyrano", "Frozen"));
         HibernateUtils.save(new Representació(40.0, Utils.createDate(14, 1, 2016), 32,
"mati", "Bling Bling", "Cats"));
         HibernateUtils.save(new Representació(30.0, Utils.createDate(14, 1, 2016), 32,
"nit", "Bling Bling", "La Cantina"));
         HibernateUtils.save(new Representació(20.0, Utils.createDate(15, 1, 2016), 32,
"tarda", "Bling Bling", "Frozen"));
         HibernateUtils.save(new Representació(10.0, Utils.createDate(14, 1, 2016), 32,
"mati", "Apolo", "Cats"));
         HibernateUtils.save(new Representació(70.0, Utils.createDate(14, 1, 2016), 32,
"nit", "Apolo", "El Rey Leon"));
         HibernateUtils.save(new Representació(50.0, Utils.createDate(15, 1, 2016), 32,
"tarda", "Apolo", "La Sirenita"));

         HibernateUtils.save(new Estrena(60.0, Utils.createDate(14, 1, 2016), 32, "nit",
"Cyrano", "El Rey Leon",1));
         HibernateUtils.save(new Estrena(70.0, Utils.createDate(14, 1, 2016), 32, "nit",
"Apolo", "El Rey Leon",1));
         HibernateUtils.save(new Estrena(30.0, Utils.createDate(14, 1, 2016), 32, "nit",
"Bling Bling", "La Cantina",1));
```

```
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 8; j++) {
                HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Cyrano","mati",null));
            }
        }
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 8; j++) {
                HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Cyrano","tarda",null));
            }
        }
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 8; j++) {
                HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Cyrano","nit",null));
            }
        }
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 8; j++) {
                HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Apolo","nit",null));
            }
        }
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 8; j++) {
                HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Apolo","tarda",null));
            }
        }
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 8; j++) {
                HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Apolo","mati",null));
            }
        }
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 8; j++) {
                HibernateUtils.save(new Seientenrepresentació("lliure",i, j, "Bling
Bling","nit",null));
            }
        }
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 8; j++) {
                HibernateUtils.save(new Seientenrepresentació("lliure",i, j, "Bling
Bling","mati",null));
            }
        }
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 8; j++) {
                HibernateUtils.save(new Seientenrepresentació("lliure",i, j, "Bling
Bling","tarda",null));
            }
        }
```

15.2.- Joc De Proves 2

```
//Obres de Teatre (shows)
    HibernateUtils.save(new Espectacle("La Gran Ilusión", 7));
    HibernateUtils.save(new Espectacle("El Rey León", 25));
    HibernateUtils.save(new Espectacle("BITS - Tricicle", 3));
    HibernateUtils.save(new Espectacle("Berto Romero", 1));
    HibernateUtils.save(new Espectacle("Mr Snow", 1));
    HibernateUtils.save(new Espectacle("Patufet - El Musical", 1));
    HibernateUtils.save(new Espectacle("The Hole", 3));

//Sessions
    HibernateUtils.save(new Sessió(TipusSessio.mati.toString()));
    HibernateUtils.save(new Sessió(TipusSessio.tarda.toString()));
    HibernateUtils.save(new Sessió(TipusSessio.nit.toString()));

//Teatres
    HibernateUtils.save(new Local("Alexandre", "Rambla Catalunya 90"));
    HibernateUtils.save(new Local("Capitol", "Les Rambles 138"));
    HibernateUtils.save(new Local("Victòria", "Av. Paral·lel 67"));
    HibernateUtils.save(new Local("Coliseum", "Gran Via de les Corts Catalanes 595"));
    HibernateUtils.save(new Local("Borràs", "Pl. Urquinaona 9"));
    HibernateUtils.save(new Local("TNC", "Pl. de les Arts 1"));

//Crear Seients Teatres
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 8; j++) {
            HibernateUtils.save(new Seient(i, j, "Alexandre"));
        }
    }
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 8; j++) {
            HibernateUtils.save(new Seient(i, j, "Capitol"));
        }
    }
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 8; j++) {
            HibernateUtils.save(new Seient(i, j, "Victòria"));
        }
    }
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 8; j++) {
            HibernateUtils.save(new Seient(i, j, "Coliseum"));
        }
    }
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 8; j++) {
            HibernateUtils.save(new Seient(i, j, "Borràs"));
        }
    }
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 8; j++) {
            HibernateUtils.save(new Seient(i, j, "TNC"));
        }
    }

//La Gran Ilusión - Borràs
    HibernateUtils.save(new Representació(30.0, Utils.createDate(15, 1, 2016), 32, "mati",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(30.0, Utils.createDate(15, 1, 2016), 32, "tarda",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(30.0, Utils.createDate(15, 1, 2016), 32, "nit",
"Borràs", "La Gran Ilusión"));
```

```
    HibernateUtils.save(new Representació(30.0, Utils.createDate(16, 1, 2016), 32, "mati",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(30.0, Utils.createDate(16, 1, 2016), 32, "tarda",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(30.0, Utils.createDate(16, 1, 2016), 32, "nit",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(30.0, Utils.createDate(17, 1, 2016), 32, "mati",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(30.0, Utils.createDate(17, 1, 2016), 32, "tarda",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(30.0, Utils.createDate(17, 1, 2016), 32, "nit",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(30.0, Utils.createDate(18, 1, 2016), 32, "mati",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(30.0, Utils.createDate(18, 1, 2016), 32, "tarda",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(30.0, Utils.createDate(18, 1, 2016), 32, "nit",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(30.0, Utils.createDate(19, 1, 2016), 32, "mati",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(30.0, Utils.createDate(19, 1, 2016), 32, "tarda",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(30.0, Utils.createDate(19, 1, 2016), 32, "nit",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(15.0, Utils.createDate(20, 1, 2016), 32, "mati",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(15.0, Utils.createDate(20, 1, 2016), 32, "tarda",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(15.0, Utils.createDate(20, 1, 2016), 32, "nit",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(30.0, Utils.createDate(21, 1, 2016), 32, "mati",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(30.0, Utils.createDate(21, 1, 2016), 32, "tarda",
"Borràs", "La Gran Ilusión"));
    HibernateUtils.save(new Representació(30.0, Utils.createDate(21, 1, 2016), 32, "nit",
"Borràs", "La Gran Ilusión"));

//Mr Snow - Alexandre
    HibernateUtils.save(new Representació(8.0, Utils.createDate(15, 1, 2016), 32, "mati",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(15, 1, 2016), 32, "tarda",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(15, 1, 2016), 32, "nit",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(8.0, Utils.createDate(16, 1, 2016), 32, "mati",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(16, 1, 2016), 32, "tarda",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(16, 1, 2016), 32, "nit",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(8.0, Utils.createDate(17, 1, 2016), 32, "mati",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(17, 1, 2016), 32, "tarda",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(17, 1, 2016), 32, "nit",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(8.0, Utils.createDate(18, 1, 2016), 32, "mati",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(18, 1, 2016), 32, "tarda",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(18, 1, 2016), 32, "nit",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(8.0, Utils.createDate(19, 1, 2016), 32, "mati",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(19, 1, 2016), 32, "tarda",
"Alexandre", "Mr Snow"));
```

```
    HibernateUtils.save(new Representació(16.0, Utils.createDate(19, 1, 2016), 32, "nit",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(8.0, Utils.createDate(20, 1, 2016), 32, "mati",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(8.0, Utils.createDate(20, 1, 2016), 32, "tarda",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(8.0, Utils.createDate(20, 1, 2016), 32, "nit",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(8.0, Utils.createDate(21, 1, 2016), 32, "mati",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(21, 1, 2016), 32, "tarda",
"Alexandre", "Mr Snow"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(21, 1, 2016), 32, "nit",
"Alexandre", "Mr Snow"));

// Teatre Coliseum - Variat
    HibernateUtils.save(new Representació(12.0, Utils.createDate(15, 1, 2016), 32, "mati",
"Coliseum", "Patufet - El Musical"));
    HibernateUtils.save(new Representació(12.0, Utils.createDate(18, 1, 2016), 32, "mati",
"Coliseum", "Patufet - El Musical"));
    HibernateUtils.save(new Representació(12.0, Utils.createDate(19, 1, 2016), 32, "mati",
"Coliseum", "Patufet - El Musical"));
    HibernateUtils.save(new Representació(12.0, Utils.createDate(20, 1, 2016), 32, "mati",
"Coliseum", "Patufet - El Musical"));
    HibernateUtils.save(new Representació(12.0, Utils.createDate(21, 1, 2016), 32, "mati",
"Coliseum", "Patufet - El Musical"));
    HibernateUtils.save(new Representació(12.0, Utils.createDate(22, 1, 2016), 32, "mati",
"Coliseum", "Patufet - El Musical"));
    HibernateUtils.save(new Representació(12.0, Utils.createDate(25, 1, 2016), 32, "mati",
"Coliseum", "Patufet - El Musical"));
    HibernateUtils.save(new Representació(12.0, Utils.createDate(26, 1, 2016), 32, "mati",
"Coliseum", "Patufet - El Musical"));

    HibernateUtils.save(new Representació(22.0, Utils.createDate(15, 1, 2016), 32, "nit",
"Coliseum", "The Hole"));
    HibernateUtils.save(new Representació(22.0, Utils.createDate(16, 1, 2016), 32, "nit",
"Coliseum", "The Hole"));
    HibernateUtils.save(new Representació(22.0, Utils.createDate(17, 1, 2016), 32, "nit",
"Coliseum", "The Hole"));
    HibernateUtils.save(new Representació(22.0, Utils.createDate(18, 1, 2016), 32, "nit",
"Coliseum", "The Hole"));
    HibernateUtils.save(new Representació(22.0, Utils.createDate(19, 1, 2016), 32, "nit",
"Coliseum", "The Hole"));
    HibernateUtils.save(new Representació(22.0, Utils.createDate(20, 1, 2016), 32, "nit",
"Coliseum", "The Hole"));
    HibernateUtils.save(new Representació(22.0, Utils.createDate(21, 1, 2016), 32, "nit",
"Coliseum", "The Hole"));
    HibernateUtils.save(new Representació(22.0, Utils.createDate(22, 1, 2016), 32, "nit",
"Coliseum", "The Hole"));
    HibernateUtils.save(new Representació(22.0, Utils.createDate(23, 1, 2016), 32, "nit",
"Coliseum", "The Hole"));
    HibernateUtils.save(new Representació(22.0, Utils.createDate(24, 1, 2016), 32, "nit",
"Coliseum", "The Hole"));
    HibernateUtils.save(new Representació(22.0, Utils.createDate(25, 1, 2016), 32, "nit",
"Coliseum", "The Hole"));
    HibernateUtils.save(new Representació(22.0, Utils.createDate(26, 1, 2016), 32, "nit",
"Coliseum", "The Hole"));

// El Rey Leon - Capitol
    HibernateUtils.save(new Representació(58.0, Utils.createDate(15, 1, 2016), 32, "mati",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(46.0, Utils.createDate(15, 1, 2016), 32, "tarda",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(46.0, Utils.createDate(15, 1, 2016), 32, "nit",
"Capitol", "El Rey León"));
```

```
    HibernateUtils.save(new Representació(58.0, Utils.createDate(16, 1, 2016), 32, "mati",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(46.0, Utils.createDate(16, 1, 2016), 32, "tarda",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(46.0, Utils.createDate(16, 1, 2016), 32, "nit",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(58.0, Utils.createDate(17, 1, 2016), 32, "mati",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(46.0, Utils.createDate(17, 1, 2016), 32, "tarda",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(46.0, Utils.createDate(17, 1, 2016), 32, "nit",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(58.0, Utils.createDate(18, 1, 2016), 32, "mati",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(46.0, Utils.createDate(18, 1, 2016), 32, "tarda",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(46.0, Utils.createDate(18, 1, 2016), 32, "nit",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(58.0, Utils.createDate(19, 1, 2016), 32, "mati",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(46.0, Utils.createDate(19, 1, 2016), 32, "tarda",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(46.0, Utils.createDate(19, 1, 2016), 32, "nit",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(58.0, Utils.createDate(20, 1, 2016), 32, "mati",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(58.0, Utils.createDate(20, 1, 2016), 32, "tarda",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(58.0, Utils.createDate(20, 1, 2016), 32, "nit",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(58.0, Utils.createDate(21, 1, 2016), 32, "mati",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(46.0, Utils.createDate(21, 1, 2016), 32, "tarda",
"Capitol", "El Rey León"));
    HibernateUtils.save(new Representació(46.0, Utils.createDate(21, 1, 2016), 32, "nit",
"Capitol", "El Rey León"));

// Berto Romero - Victòria
    HibernateUtils.save(new Representació(16.0, Utils.createDate(15, 1, 2016), 32, "tarda",
"Victòria", "Berto Romero"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(15, 1, 2016), 32, "nit",
"Victòria", "Berto Romero"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(16, 1, 2016), 32, "tarda",
"Victòria", "Berto Romero"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(16, 1, 2016), 32, "nit",
"Victòria", "Berto Romero"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(17, 1, 2016), 32, "tarda",
"Victòria", "Berto Romero"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(17, 1, 2016), 32, "nit",
"Victòria", "Berto Romero"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(18, 1, 2016), 32, "tarda",
"Victòria", "Berto Romero"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(18, 1, 2016), 32, "nit",
"Victòria", "Berto Romero"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(19, 1, 2016), 32, "tarda",
"Victòria", "Berto Romero"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(19, 1, 2016), 32, "nit",
"Victòria", "Berto Romero"));
    HibernateUtils.save(new Representació(13.0, Utils.createDate(20, 1, 2016), 32, "tarda",
"Victòria", "Berto Romero"));
    HibernateUtils.save(new Representació(13.0, Utils.createDate(20, 1, 2016), 32, "nit",
"Victòria", "Berto Romero"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(21, 1, 2016), 32, "tarda",
"Victòria", "Berto Romero"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(21, 1, 2016), 32, "nit",
"Victòria", "Berto Romero"));
```

```
// BITS - Tricicle - TNC
    HibernateUtils.save(new Representació(26.0, Utils.createDate(15, 1, 2016), 32, "tarda",
"TNC", "BITS - Tricicle"));
    HibernateUtils.save(new Representació(26.0, Utils.createDate(15, 1, 2016), 32, "nit",
"TNC", "BITS - Tricicle"));
    HibernateUtils.save(new Representació(26.0, Utils.createDate(16, 1, 2016), 32, "tarda",
"TNC", "BITS - Tricicle"));
    HibernateUtils.save(new Representació(26.0, Utils.createDate(16, 1, 2016), 32, "nit",
"TNC", "BITS - Tricicle"));
    HibernateUtils.save(new Representació(26.0, Utils.createDate(17, 1, 2016), 32, "tarda",
"TNC", "BITS - Tricicle"));
    HibernateUtils.save(new Representació(26.0, Utils.createDate(17, 1, 2016), 32, "nit",
"TNC", "BITS - Tricicle"));
    HibernateUtils.save(new Representació(26.0, Utils.createDate(18, 1, 2016), 32, "tarda",
"TNC", "BITS - Tricicle"));
    HibernateUtils.save(new Representació(26.0, Utils.createDate(18, 1, 2016), 32, "nit",
"TNC", "BITS - Tricicle"));
    HibernateUtils.save(new Representació(26.0, Utils.createDate(19, 1, 2016), 32, "tarda",
"TNC", "BITS - Tricicle"));
    HibernateUtils.save(new Representació(26.0, Utils.createDate(19, 1, 2016), 32, "nit",
"TNC", "BITS - Tricicle"));
    HibernateUtils.save(new Representació(18.0, Utils.createDate(20, 1, 2016), 32, "tarda",
"TNC", "BITS - Tricicle"));
    HibernateUtils.save(new Representació(18.0, Utils.createDate(20, 1, 2016), 32, "nit",
"TNC", "BITS - Tricicle"));
    HibernateUtils.save(new Representació(16.0, Utils.createDate(21, 1, 2016), 32, "tarda",
"TNC", "BITS - Tricicle"));
    HibernateUtils.save(new Representació(26.0, Utils.createDate(21, 1, 2016), 32, "nit",
"TNC", "BITS - Tricicle"));

//Ocupació Seients

    //La Gran Ilusión - Borràs
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 8; j++) {
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Borràs","mati",null));
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Borràs","tarda",null));
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Borràs","nit",null));
        }
    }
    //Mr Snow - Alexandre
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 8; j++) {
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Alexandre","mati",null));
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Alexandre","tarda",null));
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Alexandre","nit",null));
        }
    }

    // Teatre Coliseum - Variat
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 8; j++) {
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Coliseum","mati",null));
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Coliseum","tarda",null));
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Coliseum","nit",null));
        }
```

```
    }

    // El Rey Leon - Capitol
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 8; j++) {
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Capitol","mati",null));
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Capitol","tarda",null));
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Capitol","nit",null));
        }
    }

    // Berto Romero - Victòria
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 8; j++) {
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Victòria","mati",null));
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Victòria","tarda",null));
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"Victòria","nit",null));
        }
    }

    // BITS - Tricicle - TNC
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 8; j++) {
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"TNC","mati",null));
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"TNC","tarda",null));
            HibernateUtils.save(new Seientenrepresentació("lliure",i, j,
"TNC","nit",null));
        }
    }

//Afegir seients ocupats

    //Alexandre
    HibernateUtils.save(new Seientenrepresentació("ocupat",1, 3, "Alexandre","mati",null));
    HibernateUtils.save(new Seientenrepresentació("ocupat",1, 4, "Alexandre","mati",null));
    HibernateUtils.save(new Seientenrepresentació("ocupat",1, 5, "Alexandre","mati",null));

    HibernateUtils.save(new Seientenrepresentació("ocupat",2, 3, "Alexandre","nit",null));
    HibernateUtils.save(new Seientenrepresentació("ocupat",2, 4, "Alexandre","nit",null));
    HibernateUtils.save(new Seientenrepresentació("ocupat",2, 5, "Alexandre","nit",null));
    HibernateUtils.save(new Seientenrepresentació("ocupat",2, 6, "Alexandre","nit",null));

    //TNC
    HibernateUtils.save(new Seientenrepresentació("ocupat",1, 1, "TNC","tarda",null));
    HibernateUtils.save(new Seientenrepresentació("ocupat",1, 2, "TNC","tarda",null));

//Afegir Estrenes
    HibernateUtils.save(new Estrena(3, mati, "TNC"));
    HibernateUtils.save(new Estrena(3, tarda, "TNC"));

    HibernateUtils.save(new Estrena(2, mati, "Capitol"));

    HibernateUtils.save(new Estrena(1, nit, "Alexandre"));
    HibernateUtils.save(new Estrena(1, tarda, "Alexandre"));

//Afegir Entrades
    HibernateUtils.save(new Estrena("IDAL3150116", "41234567T", 3, Utils.createDate(15, 1,
2016), 24.0, mati, "Alexandre"));
```

```
    HibernateUtils.save(new Estrena("IDTNC2170116", "41234568B", 2, Utils.createDate(17, 1,
2016), 52.0, tarda, "TNC"));
    HibernateUtils.save(new Estrena("IDAL4190116", "41234569N", 4, Utils.createDate(19, 1,
2016), 68.0, nit, "Alexandre"));
```