

Practico 3

Diagramas de clases

Ejercicios oblicagorios: 3, 5, 7

Entregar ejercicio 8

1) Dibuje un posible diagrama de clases para el siguiente fragmento de código:

```
Compas unCompas = new Compas();
Circulo unCirculo = unCompas.dibujarCirculo(5);
Pincel unPincel = Cartuchera.getPinceles().get(2);
unPincel.setColor(new Color("#336677"));
unPincel.pintar(unCirculo);
int superficie = unCirculo.getSuperficie();
```

2) Construya a partir del código Java dado a continuación un **modelo de diseño** usando diagramas de clases en UML.

```
package ar.edu.unsl.tuw.ingsw;
public interface Roedor {
    public void comer();
    public void dormir();
    public void saltar();
}
package ar.edu.unsl.tuw.ingsw;
public class Gerbil implements Roedor {
    public void saltar() {System.out.println("Saltar de Gerbo");}
    public void comer() {System.out.println("Comer de Gerbo");}
    public void dormir() {System.out.println("Dormir de Gerbo");}
}
package ar.edu.unsl.tuw.ingsw;
public class Hamster implements Roedor {
    public void comer() {System.out.println("Comer de Hamster");}
    public void dormir() {System.out.println("Dormir de Hamster");}
    public void saltar() {System.out.println("");}
}
package ar.edu.unsl.tuw.ingsw;
public class Tester {
    public static void main(String[] args) {
        Roedor arrayOfRoedores[] = new Roedor[2];
        arrayOfRoedores[0] = new Gerbil();
        arrayOfRoedores[1] = new Hamster();
        for (int i = 0; i < arrayOfRoedores.length; i++) {
            arrayOfRoedores[i].saltar();
            arrayOfRoedores[i].comer();
            arrayOfRoedores[i].dormir();
        }
    }
}
```

3) Dado el siguiente código Java, construya un diagrama de clase que incluya clases, asociaciones, atributos, métodos, visibilidad de métodos, atributos y asociaciones.

```
package ar.edu.unsl.tuw.ingsw;
import java.util.Enumeration;
import java.util.Vector;

public abstract class Price {
    abstract int getPriceCode();
    abstract double getCharge(int daysRented);
    int getFrequentRenterPoints(int daysRented) {return 1;}
}

class Movie {
    private String _title;
    private Price _price;
    public Movie(String name, Price price) {
        _title = name;
        _price = price;
    }
    int getFrequentRenterPoints(int daysRented) {
        return _price.getFrequentRenterPoints(daysRented);
    }
    double getCharge(int daysRented) {return _price.getCharge(daysRented);}
    String getTitle() {return _title;}
}

class Rental {
    private Movie _movie;
    private int _daysRented;
    public Rental(Movie movie, int daysRented) {
        _movie = movie;
        _daysRented = daysRented;
    }
    public int getDaysRented() {return _daysRented;}
    public Movie getMovie() {return _movie;}
    public double getCharge() {return _movie.getCharge(_daysRented);}
    public int getFrequentRenterPoints() {
        return _movie.getFrequentRenterPoints(_daysRented);}
}

class Customer {
    private String _name;
    private Vector _rentals = new Vector();
    public Customer(String name) {
        _name = name;
    }
    public void addRental(Rental arg) {_rentals.addElement(arg);}
    public String getName() {return _name;}
}
```

```

    public String htmlStatement() {
        Enumeration rentals = _rentals.elements();
        String result =
            "<H1> Rentals for >EM>" + getName() + "</EM></H1><P>\n";
        while (rentals.hasMoreElements()) {
            Rental each = (Rental) rentals.nextElement();
            result += each.getMovie().getTitle() + ": "
                + String.valueOf(each.getCharge()) + "<BR>\n";
        }
        return result;
    }

    public double getTotalCharge() { ... }
    public int getTotalFrequentRenterPoints() { . . . . . }
}

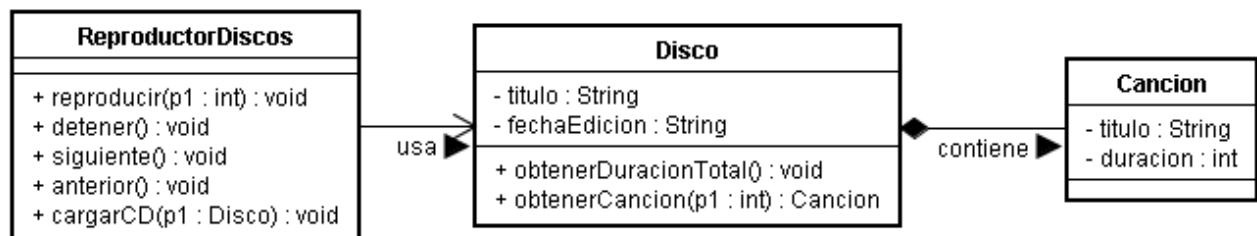
class ChildrensPrice extends Price {
    @Override
    int getPriceCode() {...}
    @Override
    double getCharge(int daysRented) {...}
}

class NewReleasePrice extends Price {
    double getCharge(int daysRented) {return daysRented * 3;}
    int getFrequentRenterPoints(int daysRented) {return daysRented;}
    @Override
    int getPriceCode() {return 0;}
}

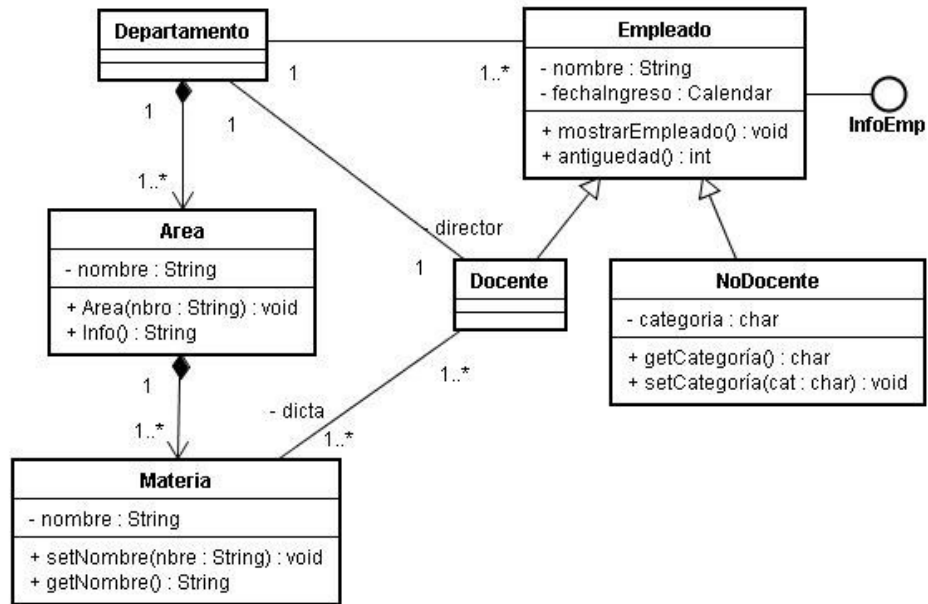
class RegularPrice extends Price {
    double getCharge(int daysRented) {return daysRented;}
    @Override
    int getPriceCode() {return 0;}
}

```

4) Escriba el código correspondiente al siguiente diagrama de clases (no escriba el cuerpo de los métodos, sino solo su declaración)



5) Dado el siguiente diagrama de clases, genere el código Java correspondiente.



6) Construya el **Modelo del Dominio** para el caso de estudio “Casa de Seguros”.

7) Construya el **Modelo del Dominio** para un caso de estudio a su elección.

8) Construya el **Modelo del Dominio** para el **Trabajo Especial**.