Supongamos que tenemos un videoclub con una serie de películas. Cada película tiene una serie de estados que son diferentes. Cuando una película se estrena es novedad pero pasado un tiempo deja de serlo. El cliente quiere poder introducir nuevos tipos de película. Tenemos que huir de la lógica condicional hacia el polimorfismo.

Veamos el código:

Clase película que encapsula el comportamiento típico de una película:

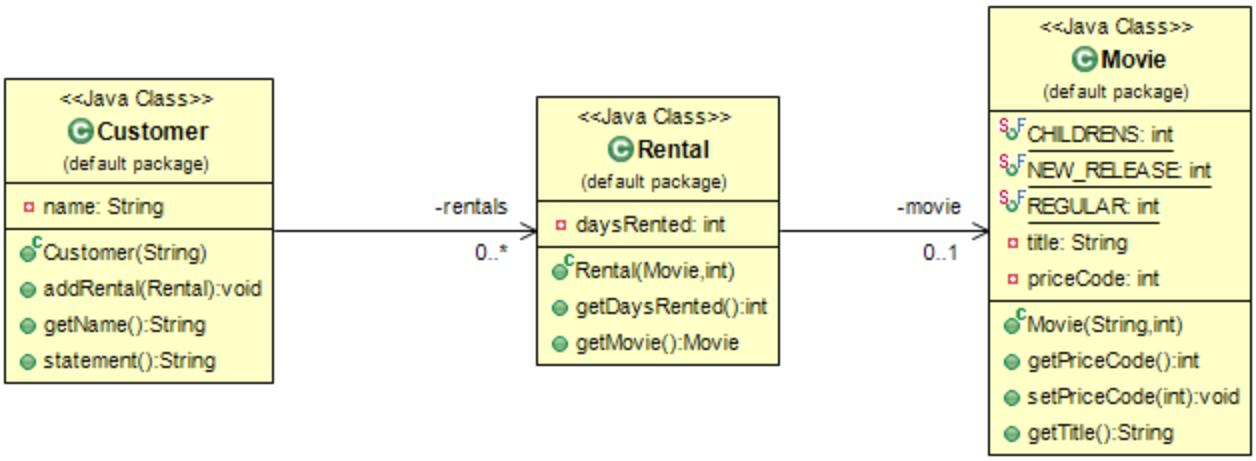
```
public class Movie
{
	public static final int CHILDRENS = 2;
	public static final int NEW_RELEASE = 1;
	public static final int REGULAR = 0;
	private String title;
	private int priceCode;

	public Movie(String title, int priceCode) {
		this.title = title;
		this.priceCode = priceCode;
	}

	public int getPriceCode() {
		return priceCode;
	}

	public void setPriceCode(int priceCode){
		this.priceCode = priceCode;
	}

	public String getTitle()
	{
		return title;
	}
}
```

Clase Rental o Alquiler que representa los días que una película ha estado alquilada.

```
public class Rental
{
	private Movie movie;
	private int daysRented;

	public Rental(Movie movie, int daysRented)
	{
		this.movie = movie;
		this.daysRented = daysRented;
	}

	public int getDaysRented()
	{
		return daysRented;
	}

	public Movie getMovie()
	{
		return movie;
	}
}
```

La clase Customer o Cliente que tendrá una lista de los alquileres que tiene.

```
public class Customer
{
	private String name;
	private List<Rental> rentals = new ArrayList<Rental>();

	public Customer(String name)
	{
		this.name = name;
	}

	public void addRental(Rental rental)
	{
		rentals.add(rental);
	}

	public String getName()
	{
		return name;
	}

	public String statement()
	{
		double totalAmount = 0;
		int frequentRenterPoints = 0;
		String result = "Rental Record for " + getName() + "\n";

		for (Rental each : rentals) {
			double thisAmount = 0;
			// Calcula el importe de cada alquiler
			switch (each.getMovie().getPriceCode()) {
			case Movie.REGULAR:
				thisAmount += 2;
				if (each.getDaysRented() > 2)
					thisAmount += (each.getDaysRented() - 2) * 1.5;
				break;
			case Movie.NEW_RELEASE:
				thisAmount += each.getDaysRented() * 3;
				break;
			case Movie.CHILDRENS:
				thisAmount += 1.5;
				if (each.getDaysRented() > 3)
					thisAmount += (each.getDaysRented() - 3) * 1.5;
				break;
			}
			// Añade los puntos de alquiler frecuente
			frequentRenterPoints++;
			// Un punto extra en el caso de las novedades alquiladas
			// por un  período de dos o más días
			if ((each.getMovie().getPriceCode() == Movie.NEW_RELEASE) &&
			     each.getDaysRented() > 1)
			       frequentRenterPoints++;
			// Muestra el importe de esta película alquilada
			result += "\t" + each.getMovie().getTitle() + "\t" +
			          String.valueOf(thisAmount) + "\n";
			totalAmount += thisAmount;
			// Añade las líneas de total
			result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
			result += "You earned " + String.valueOf(frequentRenterPoints) +
			        "   frequent renter points";
		}
		return result;
	}
}
```


<<Java Class>>
Customer
(default package)
name: String
Customer(String)
addRental(Rental):void
getName():String
statement():String
-rentals
0..*
<<Java Class>>
Rental
(default package)
daysRented: int
Rental(Movie,int)
getDaysRented():int
getMovie():Movie
-movie
0..1
<<Java Class>>
Movie
(default package)
CHILDRENS: int
NEW_RELEASE: int
REGULAR: int
title: String
priceCode: int
Movie(String,int)
getPriceCode():int
setPriceCode(int):void
getTitle():String