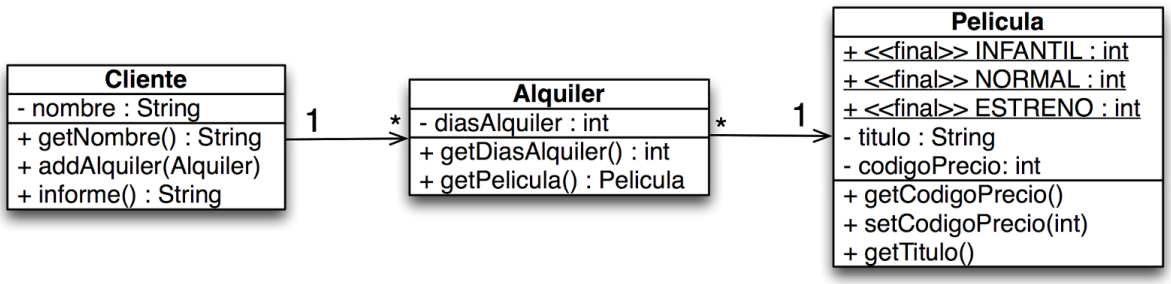
PROG3-UD10 - Mantenimiento

Hoja de reparto

Figura 1. Aplicación videoclub

Requisitos:

- Debe poder calcular e imprimir el cargo a realizar a un cliente:
    - a partir de las películas que ha alquilado y el tiempo de alquiler.
- Hay tres tipos de películas: normales, infantiles y estrenos.
- La aplicación también bonifica con puntos a los clientes que más alquilan



Programación actual para cumplir los requisitos (V1) en Figura 2

Figura 2 - Código V1

```
class Cliente {
    public String informe() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;
        Enumeration rentals = _rentals.elements();
        String result = "Rental Record for " + getName() + "\n";

        while (rentals.hasMoreElements()) {
               double thisAmount = 0;
               Alquiler each = (Alquiler) rentals.nextElement();

               //determine amounts for each line
               switch (each.getPelicula().getCodigoPrecio()) {
               case Pelicula.NORMAL:
                      thisAmount += 2;
                      if (each.getDiasAlquiler() > 2)
                             thisAmount += (each.getDiasAlquiler() - 2) * 1.5;
                      break;
               case Pelicula.ESTRENO:
                      thisAmount += each.getDiasAlquiler() * 3;
                      break;
               case Pelicula.INFANTIL:
                      thisAmount += 1.5;
                      if (each.getDiasAlquiler() > 3)
                             thisAmount += (each.getDiasAlquiler() - 3) * 1.5;
                      break;
               }
               // add frequent renter points
               frequentRenterPoints++;
               // add bonus for a two day new release rental
               if ((each.getPelicula().getCodigoPrecio() == Pelicula.ESTRENO)
                             && each.getDiasAlquiler() > 1)
                      frequentRenterPoints++;

               // show figures for this rental
               result += "\t" + each.getPelicula().getTitulo()
                             + "\t" + String.valueOf(thisAmount) + "\n";

               totalAmount += thisAmount;
        } // END WHILE
        // add footer lines
        result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
        result += "You earned " + String.valueOf(frequentRenterPoints)
        + " frequent renter points";

        return result;
    }
}
```

Figura 2

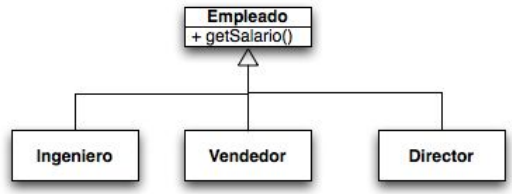
```
public String informe() {
...
    while (rentals.hasMoreElements()) {
       double thisAmount = 0;
       Alquiler each = (Alquiler) rentals.nextElement();

       thisAmount = calculaCargo(each);
. . .
}
private double calculaCargo(Alquiler alq) {
    double cargo=0;
    switch (alq.getPelicula().getCodigoPrecio()) {
    case Pelicula.NORMAL:
       cargo += 2;
       if (alq.getDiasAlquiler() > 2)
       cargo += (alq.getDiasAlquiler() - 2) * 1.5;
       break;
    case Pelicula.ESTRENO:
       cargo += alq.getDiasAlquiler() * 3;
       break;
    case Pelicula.INFANTIL:
       cargo += 1.5;
       if (alq.getDiasAlquiler() > 3)
            cargo += (alq.getDiasAlquiler() - 3) * 1.5;
       break;
    }
    return cargo;
}
```

Figura 3

Código sospechoso

```
double getSalario() {
  switch(tipoEmpleado()) {
       case INGENIERO: return salarioBase + productividad; break;
       case VENDEDOR: return salarioBase + ventas*comision; break;
       case DIRECTOR: return salarioBase + bonificacion+ dietas; break;
       default : throw new RuntimeException("Tipo de empleado incorrecto");
  }
}
```

Código refactorizado




```
class Empleado {
  abstract double getSalario(); …
}

class Ingeniero {
  @Override double getSalario() { return salarioBase + productividad; }
...}
class Vendedor {
  @Override double getSalario() { return salarioBase + ventas*comision; }
...}
class Director {
  @Override double getSalario() {return salarioBase+bonificacion+dietas;}
...}
```

(véase https://refactoring.guru/catalog)

Figura 4

En Eclipse, Botón derecho > Refactor