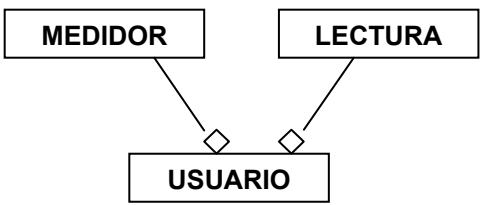
Enunciado

Los usuarios (clientes) de SPSE se identifican por:

- Numero de usuario
- Nombre y apellido
- Domicilio
- Medidor (localidad-ruta-folio)
- Lecturas (mes, estado medidor)



```
class Medidor {
 int localidad, ruta, folio;
 Medidor (int localidad, int ruta, int folio)
 {
  this.localidad = localidad;
  this.ruta = ruta;
  this.folio = folio;
 }
// métodos set y get
}

class Lectura {
 int periodo, año;
 long consumo;
 Consumo (int periodo, int año, int consumo)
 {
  this.periodo = periodo;
  this.año = año;
  this.consumo = consumo;
 }
 // métodos set y get
}
```

```
class Usuario {
  long idUsuario;
  String nombreYapellido, domicilio;
  Medidor medidor;
  Lectura [] lecturas;
  Usuario (…)
  { // inicializa atributos - crea vector}

  // metodos set/get

  void agregaLectura (int periodo, int año, long consumo)
  { lecturas[i] = new Lectura (periodo, año, consume); }

  long getConsumo (mes, año)
  { long lectura1 = lectura del medidor del periodo buscada
    long lectura2 = lectura del medidor del periodo anterior
    return lectura1-lectura2;
  }
  float calculoCostoConsumo (int periodo, int año)
  { }
}
```

Existen distintos tipos (categorías) de usuarios:

- Residenciales (familias)
- Comerciales (empresas)
- Jubilados
- Carenciados (familias de bajos recursos)

La principal diferencia reside en el cálculo del costo de consumo:

Residenciales: se emplea la siguiente tabla:

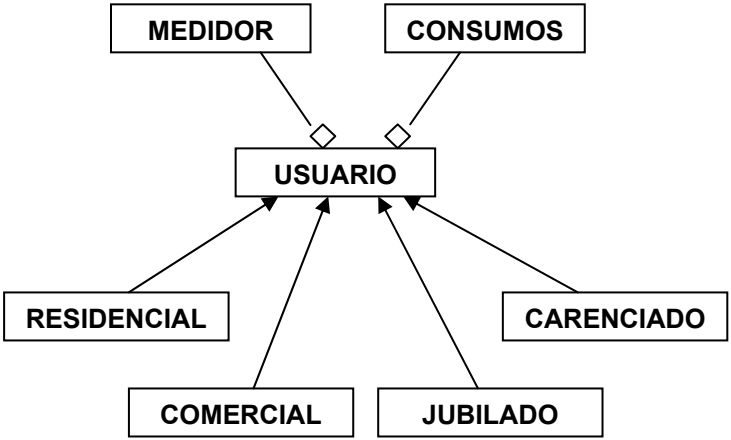
rango	Valor
0 a 250	0,05
251 a 500	0,10
Mas de 501	0,15

Además se aplica una multa de $ 6,5 si ha superado el consumo del mismo periodo del año anterior.

Comerciales: se aplica consumo * 0,25

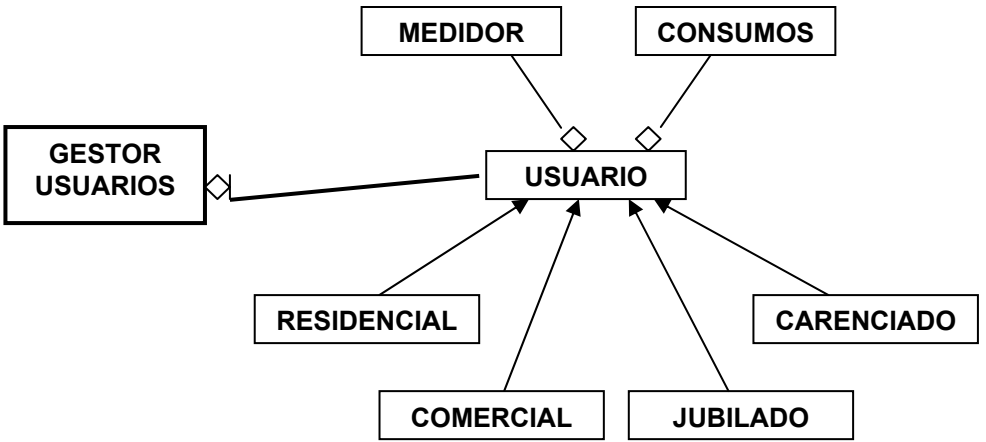
Jubilados: se aplica consumo * 0,05

Carenciados: se aplica consumo * 0,05 - 5


MEDIDOR
CONSUMOS
USUARIO
RESIDENCIAL
CARENCIADO
COMERCIAL
JUBILADO

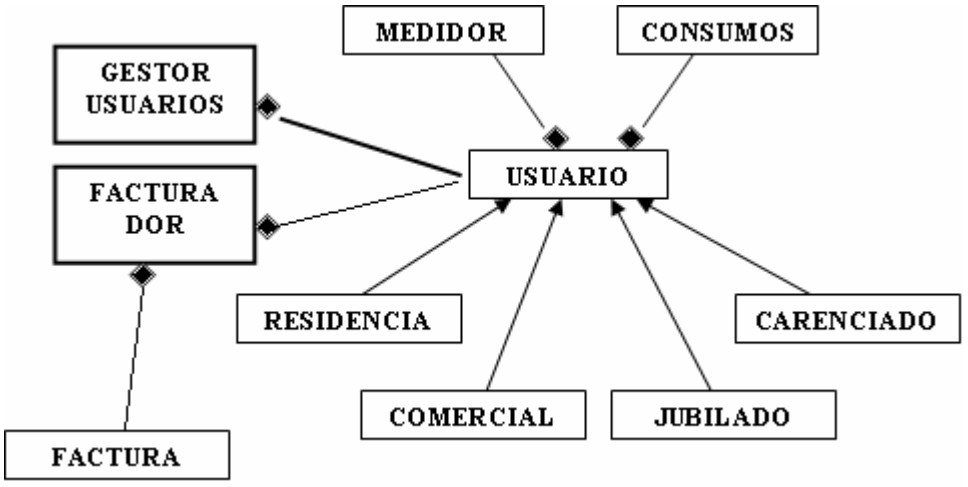
```
class Residencial extends Usuario {
 Residencial (……)
  {
    super (…..)
  }

  float calculoCostoConsumo (int mes, int año)
  { long consumo = this.getConsumo(mes,año);
    // aplica algoritmo sobre consumo
  }
}
```

Debe existir un gestor de usuarios, se trata de miles de usuarios.



```
class GestorUsuarios {
  Usuario [] usuarios;
  GestorUsuarios ()   {
   // crea vector
  }
  void agregaUsuario(Usuario u) // este método debe
                                    //sobrecargarse
  {
  }
  Usuario buscarUsuario(int usuario) {
  }
  void cargaLectura(int usuario, // datos lectura)
  { Usuario u = buscaUsuario(usuario);
    u.agregaLectura(…); }
 }
 Usuario[] getUsuarios() { return usuarios;}
 // hay métodos para listar usuarios por loc, ruta,
```

Todos los meses se realiza la facturación que implica sobre cada usuario calcular el costo del consumo e imprimir la factura.



```
class Factura {
   int numero;
   int mes, año;
   Usuario usuario;
   float costo;
Factura ()
{}
// metodos set y get
}
```

```
class Facturador {
  Factura [] facturas;
  Usuario [] usuarios;
  int ultimoNumero;
  Facturador (Usuario [] u)   {
   usuarios = u;
   // crea facturas;
  }
  void generaFacturas(int mes, int año)
  {
   for(int i=0; i<=usuarios.length; i++)
    facturas[i] = new Factura(ultimoNumero++, mes, año,
                              usuarios[i],
                    usuarios[i].calculoCostoConsumo(mes,año);
  }
  void imprimeFacturas(){
   for(int i=0; i<=facturas.length; i++)
     { System.out.println (facturas[i].get…());
      }
  }
```

```
main () // hacer menues
{
  GestorUsuarios gu = new GestorUsuarios();
  Facturador fc = new Facturador (gu.getUsuarios());
  // creo usurIos y los agrego al gu
  fc.generaFactura(---);
  fc.imprimeFacturas(--);
}
```

Qué clases se han beneficiado del polimorfismo???