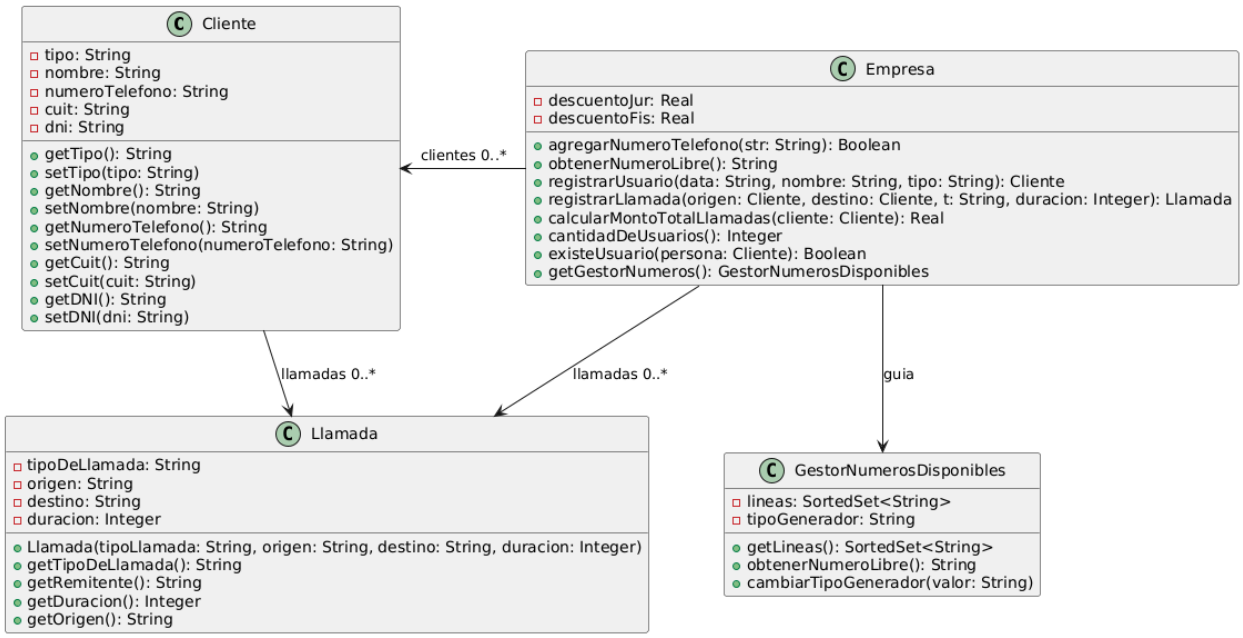
Ejercicio 5



Iteración N°1

- **Code Smell:** Switch Statement → En la clase *GestorNumerosDisponibles*, hay una variable tipo String que ayuda a decidir, junto a algunos condicionales, que debe hacer la clase en tiempo de ejecución. Estos condicionales, además, siempre preguntan por las mismas tres condiciones, por lo que podemos intuir que puede haber algún tipo de Strategy
- **Refactoring pattern a aplicar**: Replace Conditional Logic with Strategy →
  - Crear interfaz IGenerador que declare un método ObtenerNumeroLibre(*lineas*)
  - Dentro de cada subclase que implemente la interfaz, meter el fragmento de código que les corresponde
  - Finalmente, reemplazar el tipo de variable de *tipoGenerador* a **IGenerador**; cambiar la implementación de *ObtenerNumeroLibre()* de la clase *GestorNumerosDisponibles* por una simple devolución del método del

generador; cambiar el nombre del método *cambiarTipoGenerador* por cambiarGenerador, y que ahora reciba, y utilice, un objeto que implemente la interfaz IGenerador.

Iteración N°2

- **Code Smell:** Feature Envy → En la clase Empresa hay un método llamado agregarNumeroTelefono() que, dado un string, le ide a la guía que se agregue el número ella misma. El problema con esto es que está rompiendo encapsulamiento, a la vez que da olor de variable temporal y de envidia de atributos.
- **Refactoring pattern a aplicar**: Move Field, Remove Dead Code

Iteración N°3

- **Code Smell:** Switch Statement → Dentro del método calcularMontoTotalLlamadas() hay una estructura de iteración donde, en cada paso, se le pregunta a la llamada cual es su tipo, así se le cobra una u otra cosa
- **Refactoring pattern a aplicar**: Replace Conditional with Polymorphism

Iteración N°4

- **Code Smell:** Switch Statement → Los clientes tienen un tipo de descuento diferente en base a si son físicos o jurídicos. Utilizo el patrón Factory para evitar usar un if, porque la construcción de cada Cliente se repite, y de paso permito la posibilidad de introducir nuevas formas de crear clientes
- **Refactoring pattern a aplicar**: Move Creational Knowledge to Factory

Iteración N°5

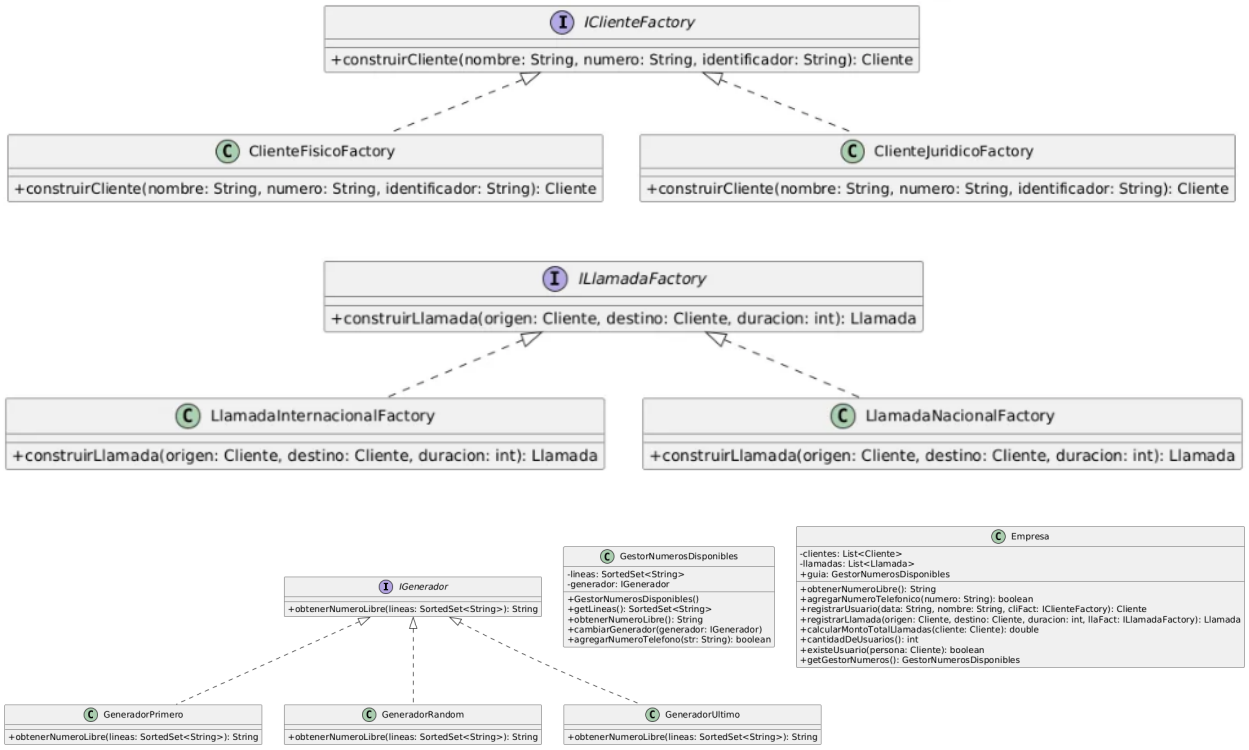
- **Code Smell:** Switch Statement → Las llamadas se construyen y se comportan de manera diferente en base a si son nacionales o internacionales. Ídem al de arriba.

- **Refactoring pattern a aplicar**: Move Creational Knowledge to Factory

Iteración N°6

- **Code Smell:** Loops → Lo que hace el método calcularMontoTotalLlamadas() es un loop grande para todas las llamadas de un cliente dado
- **Refactoring pattern a aplicar**: Replace Loop with Pipeline

Resultado final



CLIENTE.JAVA ///

```
package ar.edu.unlp.info.oo2.facturacion_llamadas;

import java.util.ArrayList;
import java.util.List;

public abstract class Cliente {
    private List<Llamada> llamadas = new ArrayList<Llamada>();
    private String nombre;
    private String numeroTelefono;

    public Cliente(String nombre, String numeroTelefono) {
        this.nombre = nombre;
        this.numeroTelefono = numeroTelefono;
    }

    public String getNombre() {
        return nombre;
```

```
    }
    public String getNumeroTelefono() {
        return numeroTelefono;
    }
    public List<Llamada> getLlamadas() {
        return this.llamadas;
    }
    public void agregarLlamada(Llamada llamada) {
        this.llamadas.add(llamada);
    }

    public abstract double getDescuento(double monto);
}

CLIENTEFISICO.JAVA ///
package ar.edu.unlp.info.oo2.facturacion_llamadas;

public class ClienteFisico extends Cliente {
    private String dni;
    //
    // INTERFAZ PÚBLICA

    public ClienteFisico(String nombre, String numeroTelefono,
    String dni) {
        super(nombre, numeroTelefono);
        this.dni = dni;
    }

    public String getDni() {
        return dni;
    }

    @Override
    public double getDescuento(double monto) {
        return monto;
    }
```

```
}

CLIENTEFISICOFACTORY.JAVA ///
package ar.edu.unlp.info.oo2.facturacion_llamadas;

public class ClienteFisicoFactory implements IClienteFactory {
    //
    // INTERFAZ PÚBLICA

    @Override
    public Cliente construirCliente(String nombre, String numero,
    String identificador) {
        ClienteFisico cli = new ClienteFisico(nombre, numero,
        identificador);
        return cli;
    }
}

CLIENTEJURIDICO.JAVA ///
package ar.edu.unlp.info.oo2.facturacion_llamadas;

public class ClienteJuridico extends Cliente {
    private String cuit;
    //
    // INTERFAZ PÚBLICA

    public ClienteJuridico(String nombre, String numeroTelefono,
    String cuit) {
        super(nombre, numeroTelefono);
        this.cuit = cuit;
    }

    public String getCuit() {
        return cuit;
    }
```

```
    @Override
    public double getDescuento(double monto) {
        return monto - (monto * 0.15);
    }
}

CLIENTEJURIDICOFACTORY.JAVA ///
package ar.edu.unlp.info.oo2.facturacion_llamadas;

public class ClienteJuridicoFactory implements IClienteFactory {
    //
    // INTERFAZ PÚBLICA

    @Override
    public Cliente construirCliente(String nombre, String numero,
    String identificador) {
        ClienteJuridico cli = new ClienteJuridico(nombre, numero,
        identificador);
        return cli;
    }
}

EMPRESA.JAVA ///
package ar.edu.unlp.info.oo2.facturacion_llamadas;

import java.util.ArrayList;
import java.util.List;

public class Empresa {
    private List<Cliente> clientes = new ArrayList<Cliente>();
    private List<Llamada> llamadas = new ArrayList<Llamada>();
    public GestorNumerosDisponibles guia =
                new GestorNumerosDisponibles();

    public String obtenerNumeroLibre() {
        return guia.obtenerNumeroLibre();
```

```
}

public boolean agregarNumeroTelefonico(String numero) {
    return this.guia.agregarNumeroTelefono(numero);
}

public Cliente registrarUsuario(String data, String nombre,
IClienteFactory cliFact) {
    Cliente cliente = cliFact.construirCliente(nombre, data,
    nombre);
    clientes.add(cliente);
    return cliente;
}

public Llamada registrarLlamada(Cliente origen, Cliente
destino, int duracion, ILlamadaFactory llaFact) {
    Llamada llamada = llaFact.construirLlamada(origen, destino,
    duracion);
    llamadas.add(llamada);
    origen.agregarLlamada(llamada);
    return llamada;
}

public double calcularMontoTotalLlamadas(Cliente cliente) {
    double c = cliente.getLlamadas().stream()
                            .mapToDouble(l ->
                            cliente.getDescuento(l.calcularMonto()))
                            .sum();

    return c;
}

public int cantidadDeUsuarios() {
    return clientes.size();
}
```

```
    public boolean existeUsuario(Cliente persona) {
        return clientes.contains(persona);
    }

    public GestorNumerosDisponibles getGestorNumeros() {
        return this.guia;
    }
}


GENERADORPRIMERO.JAVA ///
package ar.edu.unlp.info.oo2.facturacion_llamadas;

import java.util.SortedSet;

public class GeneradorPrimero implements IGenerador {
    //
    public String obtenerNumeroLibre(SortedSet<String> lineas) {
        String linea = lineas.first();
        lineas.remove(linea);
        return linea;
    }
}

GENERADORRANDOM.JAVA ///
package ar.edu.unlp.info.oo2.facturacion_llamadas;

import java.util.ArrayList;
import java.util.Random;
import java.util.SortedSet;

public class GeneradorRandom implements IGenerador {
    //
    public String obtenerNumeroLibre(SortedSet<String> lineas) {
        String linea = new ArrayList<String>(lineas)
                .get(new Random().nextInt(lineas.size()));
```

```
        lineas.remove(linea);
        return linea;
    }
}

GENERADORULTIMO.JAVA ///
package ar.edu.unlp.info.oo2.facturacion_llamadas;

import java.util.SortedSet;

public class GeneradorUltimo implements IGenerador {
    //
    public String obtenerNumeroLibre(SortedSet<String> lineas) {
        String linea = lineas.last();
        lineas.remove(lineas.last());
        return linea;
    }
}

GESTORNUMEROSDISPONIBLES.JAVA ///
package ar.edu.unlp.info.oo2.facturacion_llamadas;

import java.util.TreeSet;
import java.util.ArrayList;
import java.util.Random;
import java.util.SortedSet;

public class GestorNumerosDisponibles {
    private SortedSet<String> lineas;
    private IGenerador generador;

    public GestorNumerosDisponibles() {
        this.lineas = new TreeSet<String>();
        this.generador = new GeneradorUltimo();
    }
```

```
    public SortedSet<String> getLineas() {
        return lineas;
    }

    public String obtenerNumeroLibre() {
        return this.generador.obtenerNumeroLibre(this.lineas);
    }

    public void cambiarGenerador(IGenerador generador) {
        this.generador = generador;
    }

    public boolean agregarNumeroTelefono(String str) {
        if (!this.getLineas().contains(str)) {
            this.getLineas().add(str);
            return true;
        }
        return false;
    }
}


ICLIENTEFACTORY.JAVA ///
package ar.edu.unlp.info.oo2.facturacion_llamadas;

// Mezcla de Template Method y Factory Method

public interface IClienteFactory {
    //
    // INTERFAZ PÚBLICA

    public Cliente construirCliente(String nombre, String numero,
    String identificador);
}

IGENERADOR.JAVA ///
```

```
package ar.edu.unlp.info.oo2.facturacion_llamadas;

import java.util.SortedSet;

public interface IGenerador {
    public String obtenerNumeroLibre(SortedSet<String> lineas);
}

ILLAMADAFACTORY.JAVA ///
package ar.edu.unlp.info.oo2.facturacion_llamadas;

public interface ILlamadaFactory {
    //
    // INTERFAZ PÚBLICA

    public Llamada construirLlamada(Cliente origen, Cliente destino,
    int duracion);
}

LLAMADA.JAVA ///
package ar.edu.unlp.info.oo2.facturacion_llamadas;

public abstract class Llamada {
    private Cliente origen;
    private Cliente destino;
    private int duracion;

    public Llamada(Cliente origen, Cliente destino, int duracion) {
        this.origen= origen;
        this.destino= destino;
        this.duracion = duracion;
    }

    public Cliente getRemitente() {
        return destino;
    }
```

```
    public int getDuracion() {
        return this.duracion;
    }

    public Cliente getOrigen() {
        return origen;
    }

    public double calcularMonto() {
        double monto = this.getDuracion() * this.getPrecio() +
        (this.getDuracion() * this.getPrecio() * 0.21);
        return monto;
    }

    public abstract double getPrecio();
}

LLAMADAINTERNACIONAL.JAVA ///
package ar.edu.unlp.info.oo2.facturacion_llamadas;

public class LlamadaInternacional extends Llamada {
    //
    public LlamadaInternacional(Cliente origen, Cliente destino,
    int duracion) {
        super(origen, destino, duracion);
    }

    @Override
    public double getPrecio() { return 150; }

    @Override
    public double calcularMonto() {
        return super.calcularMonto() + 50;
    }
}
```

```
LLAMADAINTERNACIONALFACTORY.JAVA ///
package ar.edu.unlp.info.oo2.facturacion_llamadas;

public class LlamadaInternacionalFactory implements
ILlamadaFactory {
    //
    // INTERFAZ PÚBLICA

    @Override
    public Llamada construirLlamada(Cliente origen, Cliente destino,
    int duracion) {
        return new LlamadaInternacional(origen, destino, duracion);
    }
}

LLAMADANACIONAL.JAVA ///
package ar.edu.unlp.info.oo2.facturacion_llamadas;

public class LlamadaNacional extends Llamada {
    //
    public LlamadaNacional(Cliente origen, Cliente destino,
    int duracion) {
        super(origen, destino, duracion);
    }

    @Override
    public double getPrecio() { return 3; }
}

LLAMADANACIONALFACTORY.JAVA ///
package ar.edu.unlp.info.oo2.facturacion_llamadas;

public class LlamadaNacionalFactory implements ILlamadaFactory {
    //
    // INTERFAZ PÚBLICA
```

```
    @Override
    public Llamada construirLlamada(Cliente origen, Cliente destino,
    int duracion) {
        return new LlamadaNacional(origen, destino, duracion);
    }
}
```