Ejercicio 5

▼ Malos olores en agregarNumeroDeTelefono(String str) de la clase Empresa

Primer mal olor

i_ Envidia de atributos

ii_ Aplicamos move method

iii_

```
public class Empresa {
   public boolean agregarNumeroTelefono(String str) {
         boolean encontre = guia.buscar(str);
         if (!encontre) {
            guia.agregarNumero(str);
            encontre= true;
            return encontre;
         }
         else {
            encontre= false;
            return encontre;
         }
      }
   }
```

```
public class Empresa {
   public boolean agregarNumeroTelefono(String str) {
         return this.guia.agregarNumeroTelefono(str);
      }
}

public class GestorNumerosDisponibles {
   public boolean agregarNumeroTelefono(String str) {
         boolean encontre = guia.getLineas().contains(str);
         if (!encontre) {
            guia.getLineas().add(str);
```

```
            encontre= true;
            return encontre;
        }
        else {
            encontre= false;
            return encontre;
    }
    }
}
```

Segundo mal olor
i_ Envidia de atributos

ii_ Aplicamos extract method

iii_

```
public class GestorNumerosDisponibles {
    public boolean buscar(String string) {
        return this.lineas.contains(string);
    }

    public boolean agregarNumeroTelefono(String str) {
        // boolean encontre = this.buscar(str);
        boolean encontre = guia.buscar(str);
        if (!encontre) {
            this.getLineas().add(str);
            encontre= true;
            return encontre;
        }
        else {
            encontre= false;
            return encontre;
        }
    }
}
```

iv_ Pasa el test.

Tercer mal olor

i_ Envidia de atributos

ii_ Aplicamos extract method

iii_

```
public class GestorNumerosDisponibles {
   public boolean agregarNumeroTelefono(String str) {
         boolean encontre = this.buscar(str);
         if (!encontre) {
            this.agregarNumero(str);
            encontre= true;
         }
         else {
            encontre= false;
         }
         return encontre;
      }
   }
      public void agregarNumero(String string) {
         return this.lineas.add(string);
      }
      public boolean buscar(String string) {
         return this.lineas.contains(string);
      }
   }
```

iv_ Pasa el test.

i_ Envidia de atributos

ii_ Aplicamos move method

iii_

```
public class Empresa {
   public boolean agregarNumeroTelefono(String str) {
         // boolean encontre = guia.getLineas().contains(str);
         boolean encontre = guia.buscar(str);
         if (!encontre) {
```

```
            guia.getLineas().add(str);
            encontre= true;
            return encontre;
        }
        else {
            encontre= false;
            return encontre;
        }
    }
}

public class GestorNumerosDisponibles {
    public boolean buscar(String string) {
        return this.lineas.contains(string);
    }
}
```

iv_ Pasa el test.

▼ Malos olores en registrar Usuario(String data, String nombre, String tipo) de la clase empresa.

Primer mal olor:

i_ Mal olor: switch statement

ii_ Aplicamos replace conditional with polymorphism.

iii_

```
public class Empresa {
    public Cliente registrarUsuario(String data, String nombre, String tipo) {
        Cliente var = new Cliente();
        if (tipo.equals("fisica")) {
            var.setNombre(nombre);
            String tel = this.obtenerNumeroLibre();
            var.setTipo(tipo);
            var.setNumeroTelefono(tel);
            var.setDNI(data);
        }
        else if (tipo.equals("juridica")) {
            String tel = this.obtenerNumeroLibre();
```

```
            var.setNombre(nombre);
            var.setTipo(tipo);
            var.setNumeroTelefono(tel);
            var.setCuit(data);
        }
        clientes.add(var);
        return var;
    }
}
```

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

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

    public void asignarNumero(String tel){
        this.numeroTelefono = tel;
    }
    // metodos
}
public class ClienteJuridica extends Cliente{
    private String cuit;

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

}
public class ClienteFisica extends Cliente{
    private String dni;

    public ClienteFisica(String nombre, String dni){
        super(nombre);
```

```
        this.dni = dni;
    }
}

public class Empresa {
    public Cliente registrarUsuario(Cliente cliente) {
            cliente.asignarNumero(this.obtenerNumeroLibre());
            clientes.add(cliente);
            return cliente;
        }
    }
```

▼ Malos olores por variables que pertenecen al tipo de cliente en clase empresa.

i_ Mal olor: envidia de atributos.

ii_ Aplicamos move field
iii_

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

    static double descuentoJur = 0.15;
    static double descuentoFis = 0;
    ....
}
```

```
public class ClienteFisica extends Cliente{
    private String dni;
    private static double descuentoFis = 0;

    public ClienteFisica(String nombre, String dni){
        super(nombre);
        this.dni = dni;
    }
```

```
}

public class ClienteJuridica extends Cliente{
   private String cuit;
   private static double descuentoJur = 0.15;

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

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

   ....
}
```

▼ Mal olor en metodo registrarLlamada(...) de la clase Empresa

i_ Mal olor: feature envy

ii_ Aplicamos move method

iii_

```
public class Empresa {
   public Llamada registrarLlamada(Cliente origen, Cliente destino, String t
         Llamada llamada = new Llamada(t, origen.getNumeroTelefono(), de
         llamadas.add(llamada);
         origen.llamadas.add(llamada);
         return llamada;
   }
}
```

```
public class Empresa {
   public Llamada registrarLlamada(Cliente origen, Cliente destino, String t
         Llamada llamada = new Llamada(t, origen.getNumeroTelefono(), de
```

```
            llamadas.add(llamada);
            origen.agregarLlamada(llamada);
            return llamada;
        }
    }
    public class Cliente {
        public void agregarLlamada(Llamada llamada) {
            this.llamadas.add(llamada);
        }
    }
```

▼ Malos olores en calcularMontoTotalLlamadas(Cliente cliente) de la clase empresa

Primer mal olor

i_ Mal olor: switch statement

ii_ Aplicamos replace conditional with polymorphysm

iii_

```
public class Empresa {
    public double calcularMontoTotalLlamadas(Cliente cliente) {
            double c = 0;
            for (Llamada l : cliente.llamadas) {
                double auxc = 0;
                if (l.getTipoDeLlamada() == "nacional") {
                    // el precio es de 3 pesos por segundo más IVA sin adicional p
                    auxc += l.getDuracion() * 3 + (l.getDuracion() * 3 * 0.21);
                } else if (l.getTipoDeLlamada() == "internacional") {
                    // el precio es de 150 pesos por segundo más IVA más 50 pes
                    auxc += l.getDuracion() * 150 + (l.getDuracion() * 150 * 0.21) +
                }

                if (cliente.getTipo() == "fisica") {
                    auxc -= auxc*descuentoFis;
                } else if(cliente.getTipo() == "juridica") {
                    auxc -= auxc*descuentoJur;
                }
                c += auxc;
```

```
        }
        return c;
    }
}
```

```
public class Llamada {
    private String origen;
    private String destino;
    private int duracion;

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

public class LlamadaInternacional() extends Llamada{

    public LlamadaInternacional(String origen, String destino, int duracion){
        super(origen, destino, duracion);
    }

    public double calcularPrecioLlamada(){
        return (this.duracion * 3) + (this.duracion * 3 * 0.21);
    }

}

public class LlamadaNacional() extends Llamada{

    public LlamadaNacional(String origen, String destino, int duracion){
        super(origen, destino, duracion);
    }

    public double calcularPrecioLlamada(){
        return (this.duracion * 150) + (this.duracion * 150 * 0.21) + 50;
    }
```

```
}

public class Empresa {
   public double calcularMontoTotalLlamadas(Cliente cliente) {
         double c = 0;
         for (Llamada l : cliente.llamadas) {
            double auxc = 0;
            auxc += l.calcularPrecioLlamada();

            if (cliente.getTipo() == "fisica") {
               auxc -= auxc*descuentoFis;
            } else if(cliente.getTipo() == "juridica") {
               auxc -= auxc*descuentoJur;
            }
            c += auxc;
         }
         return c;
      }
   }
```

Segundo mal olor

i_ Mal olor: codigo duplicado para la forma de calcular el precio de la llamada

ii_ Apicamos form templateMethod

iii_

```
public class LlamadaInternacional() extends Llamada{
   public double calcularPrecioLlamada(){
      return (this.duracion * 3) + (this.duracion * 3 * 0.21);
   }
}

public class LlamadaNacional() extends Llamada{
   public double calcularPrecioLlamada(){
      return (this.duracion * 150) + (this.duracion * 150 * 0.21) + 50;
   }
}
```

```
public class Llamada {
    ....

    protected double multiplicar() {
        return this.duracion * this.valor();
    }

    protected double porcentaje() {
        return this.multiplicar() *  0.21;
    }

    protected abstract double valor();

    protected abstract double adicional();

    public double calcularPrecioLlamada(){
        return this.multiplicar() + this.porcentaje() + this.adicional();
    }
}

public class LlamadaInternacional() extends Llamada{
    ...
    protected double valor() {
        return 3;
    }

    protected abstract double adicional() {
        return 0;
    }
}

public class LlamadaNacional() extends Llamada{
    protected double valor() {
        return 150;
    }

    protected abstract double adicional() {
        return 50;
```

```
    }
}
```

Tercer mal olor

i_ Switch statements

ii_ Move method

iii_

```
public double calcularMontoTotalLlamadas(Cliente cliente) {
        double c = 0;
        for (Llamada l : cliente.llamadas) {
            double auxc = 0;
            auxc += l.calcularPrecioLlamada();

            if (cliente.getTipo() == "fisica") {
                auxc -= auxc*descuentoFis;
            } else if(cliente.getTipo() == "juridica") {
                auxc -= auxc*descuentoJur;
            }
            c += auxc;
        }
        return c;
    }
```

```
public abstract class Cliente {
    public abstract double calcularDescuento(double auxc);
}

public class ClienteJuridica extends Cliente{
    public double calcularDescuento(double auxc) {
        return auxc*descuentoJur;
    }
}

public class ClienteFisica extends Cliente{
    public double calcularDescuento(double auxc) {
        return auxc*descuentoFis;
```

```
    }
}

public class Empresa {
    public double calcularMontoTotalLlamadas(Cliente cliente) {
            double c = 0;
            for (Llamada l : cliente.llamadas) {
                double auxc = 0;
                auxc += l.calcularPrecioLlamada();

                auxc -= cliente.calcularDescuento(auxc);

                c += auxc;
            }
            return c;
        }
    }
```

```
public class Empresa {
    public Llamada registrarLlamada(Llamada llamada, Cliente origen) {
        llamadas.add(llamada);
        origen.agregarLlamada(llamada);
        return llamada;
    }
}
```