

Refactoring to patterns

Ejercicio 5

Clase Cliente

Bad smells:

- Rompe el encapsulamiento: (variable llamadas es pública)

Refactoring: Encapsulate Field (se cambia la visibilidad de la variable llamadas a privada)

Clase GestorNumerosDisponibles

Bad Smell:

- Switch Statements - Obsesión por los primitivos: el método obtenerNumeroLibre está lleno de if-else (switch).

Refactoring: Replace Conditionals with Strategy (se crea strategy abstracta con subclasses (concrete strategies) que la extienden.

Clase Empresa

Bad Smell:

- Envidia de atributos (Feature Envy - mala asignación responsabilidad): **boolean encuentre = guia.getLineas().contains(str);** en el método agregarNumeroTelefono hay envidia de atributos. La clase Empresa realiza lógica que debería resolver la clase GestorNumerosDisponibles
- **guia.getLineas().add(str);** idem inciso anterior
- Temporary Field: la variable temporal encuentre se puede obviar.

Refactoring: Move Method (se mueve el método a GestorNumerosDisponibles).

Se ve nuevo bad smell: Long Method (método agregarNumeroTelefono en Gestor es muy largo -> muchas responsabilidades)

Refactoring: extract method. se crean nuevos métodos para subdividir tareas

- Switch Statements - Obsesión por los primitivos: en el método registrarUsuario se hacen uso de sentencias if-else(switch).

Refactoring: Replace Conditionals with Polymorphism (Cliente pasa a ser clase abstracta, se crean clases concretas ClienteFisico y ClienteJuridico)

Había long method: agregarUsuario.

Además, se hizo extract method. se dividió el método agregarUsuario en 2: agregarClienteFisico y agregarClienteJuridico.

Código repetido: **this.clientes.add(cliente);**

Se hace extract method y se crea método agregarCliente

- Envidia de atributos (feature envy - mala asignación responsabilidad): **origen.llamadas.add(llamada);** (debe hacerlo la clase Cliente)

Refactoring: extract method. se crea método agregarLlamada en la clase Cliente.

- método calcularMontoTotalLlamadas: Switch statements - obsesión por los primitivos. Además, es una lógica que debería estar en la clase Cliente.

Refactoring: Replace Conditionals with Polymorphism (Llamada pasa a ser clase abstracta, se crean clases concretas LlamadaInternacional y LlamadaNacional)

Refactoring: extract method, registrarLlamada se divide en 2: registrarLlamadaInternacional y registrarLlamadaNacional

- Long method -> calcularMontoTotalLlamadas es método largo

Refactoring: extract method y Move method -> se crea calcularMontoTotal en la clase Cliente

- Reinventa La Rueda: hace uso de un for

Refactoring: replace loop with pipeline (se usan streams en vez de un for)

- Long method -> calcularMontoTotal es método largo

Refactoring: extract method y Move method -> se crea calcularMontoLlamada en la clase Llamada

- Long method -> calcularMontoLlamada es método largo

Refactoring: extract method y Move method -> se crean métodos obtenerPrecioLlamada, obtenerIVA, aplicarAdicional