**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

1. **Data clumps (Grupo de datos)**

Se da cuando hay grupos de datos idénticos, una forma de identificar este tipo de error es aislar un dato y ver si este tiene sentido por si solo o encuentra sentido cuando esta con un grupo de datos.

* **Tecinas de Refactoring**
  + 1. **Extract class:** moverá los atributos que sean parte del grupo de datos a una nueva clase, cuando los datos de los atributos se repitan
    2. **Introduce Parameter Object:** Si el grupo de datos es pasado como parámetro a una función, es mejor remplazarlos por un objeto.
    3. **Preserve Whole Object:** Si extraigo varios valores de un objeto para luego pasarlos como parámetros. Es mejor pasar todo el objeto

1. **Primitive Obsession (Obsesión Primitiva)**

Este tipo de error se presenta cuando en vez de representar atributos mediante una clase correcta, se la representa mediante datos primitivos como int, double etc.

* **Tecinas de Refactoring**
  + 1. **Replace Data Value With Object:** Si tenemos un grupo grande de datos, es muy posible que su lógica pertenezca a otra clase, por lo que seria mejor mover toda esa lógica a su propia clase
    2. **Introduce Parameter Object**
    3. **Replace type code with class** Si se tiene una lista de datos admisibles, el valor de estas variables no son utilizados en condicionales y no afectan el comportamiento del programa, Entonces creamos una nueva clase y utilizamos el objeto en lugar de las variables separadas.

1. **Refused Bequest (Legado Rechazado)**

Cuando el programa tiene aplicaciones incompletas de Poo se da este tipo de error. Si una subclase solo usa algunos métodos y propiedades heredadas de su padre, Entonces la jerarquía esta fuera de lugar.

* **Tecinas de Refactoring**
  + 1. **Replace inheritance with delegation:** Si tenemos una subclase que solo utiliza una porción de los métodos de su clase padre, crear un atributo del tipo de la clase padre y delegar los métodos al objeto de la clase padre y eliminar la herencia.
    2. **Extract Super Class:** Si tenemos 2 clases que comparten varios atributos y métodos iguales. Creamos una super clase en la cual estén todos los atributos y métodos iguales en ambas clases y luego las 2 clases serán hijas de la super clase que hemos creado

1. **Featured Envy (Característica Envidiada )**

Cuando una clase accede mas a la data de otra clase en vez de en la suya.

* **Tecinas de Refactoring**
  + 1. **Move method:** Mover los métodos que utilizan la data del otro objeto a ese objeto
    2. **Extract method:** Si solamente una parte del objeto accede a la data del otro. Se puede separar el método creando uno nuevo.

1. **Large class (clase larga)**

Cuando una clase tiene demasiados atributos, métodos o líneas de codigo

* **Tecinas de Refactoring**
  + 1. **Extract class**
    2. **Extract subclass:** Ayudara si parte del comportamiento de la clase puede ser implementado de otra manera creando una nueva clase que se encargue de esa seccion