- **The bloaters:**
  - *Long method:* Order.Total
    - *Continuar desde Switch Statements*
    - El shipping 15 de la última línea en una nueva variable
    - Mover la variable encima del "if USA".
    - Cambiar el contenido del if por shipping=0;
    - Realizar inlines y extract methods
  - *(3)Primitive obsession:* Customer.FormatPhoneNumber
    - *Crear clase PhoneNumber y declararla dentro de Customer.*
    - Nos dirigimos al test
    - Change signature constructor(poner en el default el valor del teléfono new PhoneNumber("") y eliminar el tipo primitivo)
    - Crear el constructor de PhoneNumber (crear el field number y ponerlo como privado)
    - Arreglar el constructor dentro de orden.
    - Cambiar el método FormatPhoneNumber para que utilize la nueva propiedad.
    - RUN TEST
    - Mover el método a la clase PhoneNumber
      - **NET**
      - Extraer método como delegado
      - Convertir a estático (borrar los this si lostiene) e insertar PhoneNumber como parámetro.
      - Quitar el método de estático.
    - Extraer métodos de country, city y localnumber.
    - RUN TEST
  - *(4)Long parameter list*: Order Constructor
    - ***NET***
    - *Extraer Parámetro de clase.*
    - *Mover clase a otro archivo. (opcionalmente convertir los a propiedades autoimplementadas)*
    - *RUN TESTS*
    - *Crear una propiedad DeliveryAddress y cambiar el nombre de las propiedades en el constructor.*
    - *Probar el uso de las propiedades, veremos q solo el contry se utiliza.*
    - *Borrar el city y el Street.*
    - *RUN TESTS*
    - *Cambiar a backing field y actualizar el contenido por DeliveryAddress.Country y luego un inline.*
    - *Borrar los que no se utilizen.*
    - *RUN TESTS*
  - *(5)Dataclumps:* street, city, contry en Order y Employee.
    - ***NET***
    - Verificar si son utilizados.
    - Reemplazarlos a mano.
- **The Object-Orientation Abusers:**
  - *Switch statements (conditional complexity):* Order.Total
    - Renombrar los discounts q tienen nombres diferentes
    - Extraer variables discount en donde no las haya y Split de la declaración
    - Mover la primera variable discount afuera de los ifs.
    - Eliminar los discounts restantes
    - Enviar fuera de los ifs al final, el itemamount=itemamount-discount.
    - Borrar todos los ítems amounts al final.
    - RUN TESTS
    - Extraer en un solo método "CalculateITotalItem", todo el foreach menos la última línea totalamoaunt +=itemamount;
    - Realizar un inline de la variable itemamount;
    - RUN TESTS
    - Mover el método extraido CalculateItemDiscount a OrderItem;
    - Renombrar CalculateItemDiscount a CalculateItemDiscount
    - Extraer método con el contenido de cada discount (ejm CalculateBikesDiscount)
    - Por cada método extraido:

            - Change Signature para q todos los métodos tengan una misma firma con parámetro OrderItem.
            - RUN TEST
        - Para cada método extraido
            - Crear una nueva clase por cada product category discount.
            - Mover los métodos creados a las nuevas clases, y cambiar el nombre del método a CalculateDiscount();
                - **NET**
                - Make-Static
                - Creando una variable dentro del método
                - Introduciendo parámetro con esa variabel
                - Make Non-Static
        - Extraer una interfaz o de cada clase productcategorydiscount y pushup del método.
        - Que cada nueva CategoryDiscount implemente la interfaz.
        - Extraer una variable de cada "new XXXDiscount()", llamarla categoryDiscount
        - Subir la variable al inicio de los ifs y borrar las variables duplicadas.
        - Bajar la sentencia categorydiscount.calculateDiscount al final de las ifs y eliminar las variables duplicadas.
        - Extraer todos los ifs en un método CreateCategoryDiscount()
        - Reordenar el método con inlines.
        - Cambiar el nombre a CalculateItemAmount a CalculatePartialTotal
    - *Temporary field*
    - *Refused bequest:* Salesman inherits from Manager.
        - Push Down subordinates, Subordinates, AddSubordinate and RemoveSubornidate
    - *Alternative Classes with Different Interfaces:* Manager.NetSalary, Salesman.NetSalary
        - Renombrar Manager a NetSalary.
        - Extraer en Salesman CalculateAdditionalBenefits de NetSalary.
        - PushDown NetSalary
            - **NET**
            - Eliminar el Net Salary en la otra clase y cambiar su calculateadditionalbenefits a protected override
- **The change preventers:**
    - *(2)Divergent Change:* Producto.ToXml
        - *Crear la clase XmlProductSerializer*
        - Extraer método delegado ToXmlDelegate
        - Convertir a estático
        - Change Signature (agregar nuevo parámetro xmlProductSerializer)
        - Convertir a no estático
        - Renombrar el método
    - *Shotgun surgery.*
    - *Parallel Inheritance Hierarchies.*
- **The Dispensables:**
    - *(5)Lazy class*: ImagenInfo
        - *Ir a los test y eliminar la instancia de ImagenInfo y reemplaazarla por una instancia de Product*
        - *Modificar el ImagenInfo.ImagenType por Product.ImageType*
        - *Crear el método dentro de Address y delegar el contenido a ImagenInfo.ImageType*
        - *RUN TESTS*
        - *Change signature al constructor de Product para quitar el ImageInfo y agregar un string.*
        - *Reutilizar la clase ImageInfo en el constructor y no borrarla, realizar un: ImagenInfo = new ImagenInfo(imagen).*
        - *Crear la propiedad string Path en Product y reemplazarla encima de ImageType para ver los errores*
        - *Copiar el contenido de ImagenInfo.ImagenType a Product.ImageType;*
        - *Corregir el constructor*
        - *RUN TESTS*
    - *Data class:*
    - *Dead code:*
    - *Duplicate code:* Manager.NetSalary, Salesman.NetSalary
        - Mirar "*Alternative Classes with Different Interfaces*"
    - *(7)Speculative generality:*

        - ThirdParty (PushDown members y arreglar el constructor)
- **The couplers:**
    - *(1)Features envy:* Order.CalculateItemAmount
        - ***NET***
        - *Make Static* Order.CalculateItemAmount
        - *Borrar Parámetro Order que no se usa*
        - *Make Non Static*
    - *(Ultimo) Inappropriate intimacy (*Indecent Exposure*)*: Order.OrderItems
        - ***NET***
        - *Ir al test y modificar el AddOrderItem la llamada Order.Items.Add para q sea Order.Add*
        - ** Convertimos la propiedad a backingfield* para que utilice la referencia ítems y no Items.
        - *Modificar la clase Order para crear el método anterior.*
        - RUN TEST
        - *Ir al test y modificar el AddOrderItem la llamada Order.Items.Count para q sea Order.Count*
        - *Modificar la clase Order para crear el método anterior.*
            - ***NET***
            - *Vemos q la clase solo tiene referencias internas*
        - *Movemos la llamada del constructor new List() al mismo field*
            - ***NET***
            - *Eliminamos la referencia al set de la propiedad.*
        - *RUN TEST*
        - Exponemos la colección como no modificable.
    - *Message Chains:*
    - *Middle man*
- **Comments:**
    - Fields: Salesman
    - Variables: Order
    - Inside Methods Order
    - Obvios: Comentarios que no dan ninguna información adicional más que ruido: Constructor OrderItem, Order.Total
    - Mandatorios: Es simplemente absurdo tener una regla que dice que cada función debe tener un javadoc, o cada variable debe tener un comentario. Comentarios como este sólo el desorden el código, se encuentra la puerta propaganda, y se prestan a la confusión general y la desorganización. Este desorden no aporta nada y sólo sirve para ocultar el código y crear la potencial de la mentira y la mala dirección. Salesman.UpdateQuota
    - Scary Noise: Product fields.