Technical Debt + Code Smells

Conexión:

- Duración: 5 minutos

A. En parejas discutir una situación cuando llegaron a un proyecto o les pasaron un proyecto en el cual el código al que se enfrentaron era malísimo, les daba dolores de cabeza y su mantenibilidad era una tarea heróica.

B. Preguntas:
    - Del 1 al 5, ¿Cómo nos sentimos acerca de la calidad de nuestro código ahora?
    - Del 1 al 5, ¿Cómo creemos que será la calidad de nuestro código dentro de 6 meses?

Conceptos:

- Technical Debt
    - Repartición de hojas *"Technical Debt".*
    - En grupos responder las preguntas. (15 minutos)
    - Construir colaborativamente la definición. (15 minutos)
- Code Smells
- Repartición de hojas: *"CodeSmells".*
- Repartición de las balotas (hay que preparlas y priorizarlas).
- Mostrar los atributos de calidad en el proyector.
- Iteración 1 (individual)
    - Duración: 15
    - Lectura del Code Smell que le tocó
    - Entendimiento
    - Búsqueda de ejemplo simple
    - Relación con atributos de calidad
- Iteración 2 (parejas)
    - Duración: 10
    - Compartir el Code Smell de cada uno
    - Aportar al conocimiento/entendimiento del Code Smell del otro
- Iteración 3 (grupos 4)
    - Duración: 10
    - Compartir el Code Smell de cada uno
    - Aportar al conocimiento/entendimiento del Code Smell del otro
    - Usar pizarra o papelografo
- Iteración 4 (todos)
    - Duración: 20
    - Se muestra uno a uno los Code Smell (dibujos con analogías graciosas)
    - A quien le tocó ese Code Smell se levanta y comparte lo entendido del mismo
    - Se pregunta al resto si hay dudas

- Se resuelve las dudas

Concreción:

- Entregar a cada persona código real.
- Iteración 1:
  - Cada pareja deberá identificar dentro del código todos los smells que le han sido asignados al grupo.
  - Duración: 20 minutos
- Iteración 2:
  - En grupos discutir y contrastar resultados con las otras parejas del grupo.
  - Duración: 10 minutos
- Iteración 3:
  - Se pregunta uno a uno por cada Code Smell.
  - Cualquier miembro del grupo se levanta y comparte en donde identificó el smell.
  - Se pregunta si hay alguna duda.
  - Se resuelven las dudas.
  - Duración: 10 minutos

Conclusión:

- Creación de Mapa Mental Colaborativo (grupos 4)
  - Duración: 15 minutos
- Explicación del mapa mental
  - Duración: 20 minutos

Refactoring

Conexión

- Duración: 5 minutos

A. Discutir en parejas sobre lo más relevante del día anterior

B.

Concepto

- Intro a Refactoring. (10 min)
- Iteración 1:
    - Repartir las hojas de Refactoring: *"Refactoring"*.
    - En parejas identificar que refáctorings del catálogo aplicarían para solucionar cada uno de todos los Code Smells.
    - Duración 20 minutos.
- Iteración 2:
    - En grupo discutir y contrastar los resultados con las otras parejas.
    - Duración 10 minutos.
- Iteración 3:
    - Uno a uno se nombra cada Code Smell, cualquier persona de la clase se levanta y explica cuál es el refactoring que utilizaría.
    - Duración 10 minutos.
- Intro a ¿Cómo hacer Refactoring?

Concreción

- Refactoring by the book
    - Baby Steps
    - Ejecutar Tests Continuamente
    - No Refactoring Tools
- Inverse Refactoring
    - Aprender Refactorings Adicionales
    - Duración: 10 minutos
- Refuctoring / Movie Rental
    - Refactoring guiado con herramientas
    - Duración: 30 minutos
- Refactoring Golf
    - Ejecutar los refáctorings identificados anteriormente (Concepto) en iteraciones.
    - Duración: 45 minutos

Conclusión

- Creación de Mapa Mental Colaborativo (grupos 4)
    - Duración: 15 minutos
- Explicación del mapa mental
    - Duración: 20 minutos

Refactoring Patterns

Conexión

¿?

Concepto y Concreción

- Unified Methods
    - Conceptos: 2 Min
    - Concreción: 10 Min
- Narrowed Change y Parallel Change
    - Conceptos: 5 Min
    - Concreción: 30 Min

Conclusión

¿?

Cierre

- 1 minuto de silencio sobre el curso
- Escribir en post-its que he aprendido del curso
- Pasar al frente y ponerlos en la pared
- Escribir que acción concreta (una) voy a aplicar esta semana con relación a lo aprendido
- Compartir en parejas las acción y crear compromiso compartido

Code Smells para Sortear

1	Long Method
2	Dead Code
3	Long Parameter List
4	Duplicated code
5	Data Clumps
6	Primitive Obsession
7	Refused Bequest
8	Switch Statements
9	Shotgun Surgery
10	Lazy Class
11	Indecent Exposure
12	Feature Envy
13	Speculative Generality
14	Divergent Change
15	Comments

16	Long Method
17	Dead Code
18	Long Parameter List
19	Duplicated code
20	Data Clumps
21	Primitive Obsession
22	Refused Bequest
23	Switch Statements
24	Shotgun Surgery
25	Lazy Class
26	Indecent Exposure
26	Feature Envy
27	Speculative Generality
28	Divergent Change
30	Comments

Code Smell	Donde se Encuentra (class.method)	Refactoring
Long Method	Order.Total	Extract Method
Primitive Obsession	ThirdParty.FormatPhoneNumber	Extract Class
Long Parameter List	Order._constructor	Introduce Parameter Object
Dataclums	Variables street, city, country en Order y Employee	Extract Class
Switch Statements	Order.Total	Replace Conditional with Polymorphism
Refused Bequest	Herencia Employee -> Salesman (.Subordinates, .AddSubordinate, .RemoveSubordinate)	Push Down Field/Method
Divergent Change	Product.ToXml	Extract Class
Lazy Class	ImageInfo	Inline Class
Duplicate Code	Manager.NetSalary, Salesman.NetSalary	Pull Up Field/Method
Speculative Generality	ThirdPary	Collapse Hierarchy
Feature Envy	Order.CalculateItemAmount	Move Method
Indecent Exposure (Inappropriate Intimacy)	Order.OrderItems	Encapsulate Collection
Comments	- Fields, Inside/Outside Methods (Debido a la falta de claridad o nombres que no revelan el propósito real): Salesman, Order.Total - Obvios (Comentarios que no dan ninguna información adicional): OrderItem._create, Order.Total. - Scary Noise(Ruidosos): Product fields. - Mandatorios (No existe una regla universal que diga que todos los métodos deben tener un javadoc, o cada parámetro debe tener un comentario. El mejor contexto para utilizar estos javadoc es en frameworks o libreríos, no tanto en aplicaciones). Este tipo de comentarios se prestan a la confusión general, desorganización, mentira y ocultan el código real.) Salesman.UpdateQuota	Extract Method Rename

Inverse Refactorings

A continuación se muestra una lista de refáctorings. Al lado de cada refactoring, escribe el nombre del refactoring que revierte su cambio.

Refactoring	Inverse
Collapse Hierarchy	Extract SubClass
Extract Method	Inline Method
Inline Class	Extract Class
Rename Method	Rename Method
Inline Temp	Introduce Explaining Variable
Pull Up Method	Push Down Method

- **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.