# Refactoring

* Primitive Obsession – Obsesión Primitiva

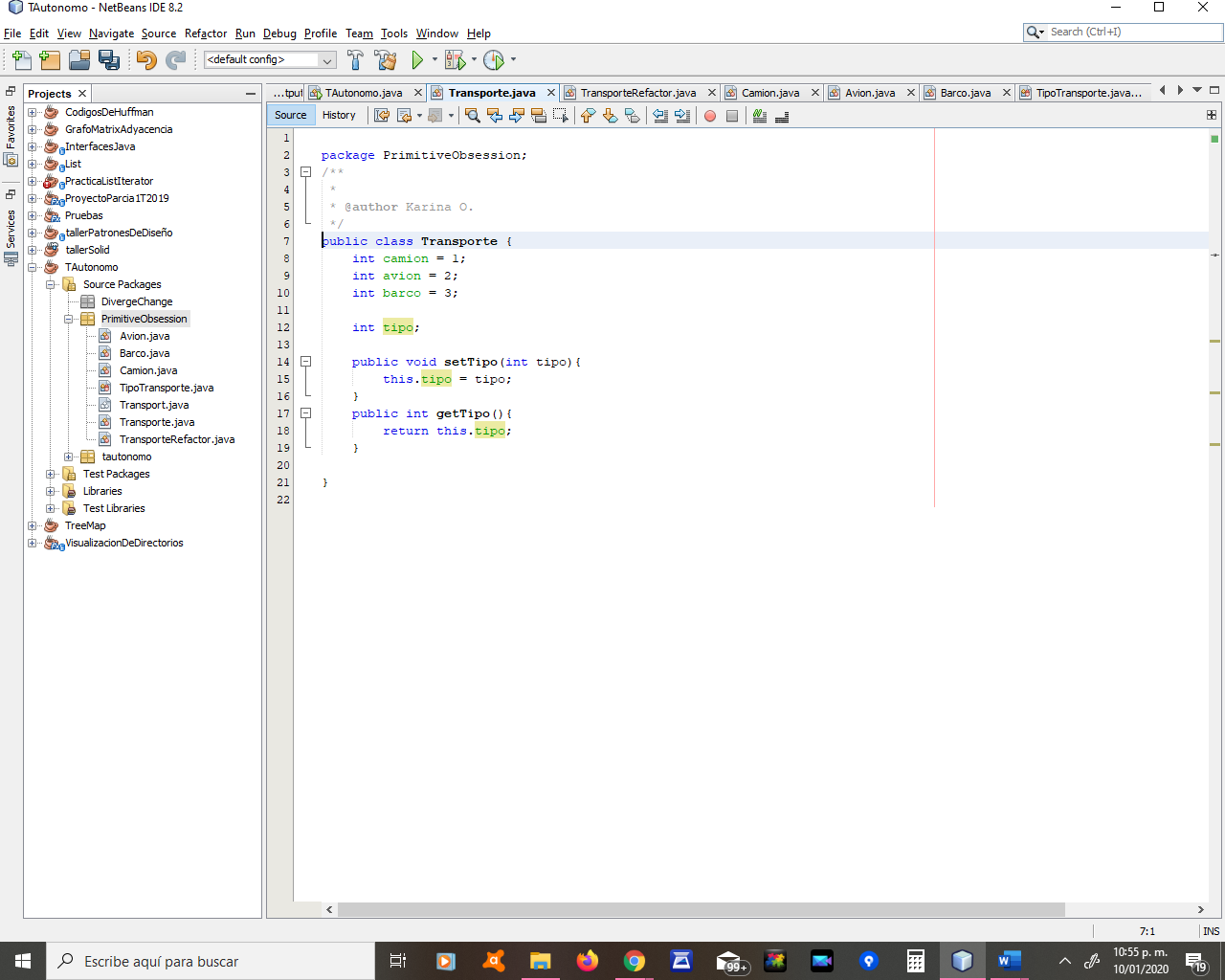
Uso de tipo de datos primitivos en lugar de usar objetos. El code smell de este tipo se puede identificar con el uso de constantes para codificar información.

Las técnicas de refactorización posibles para este mal olor son las siguientes:

* Replace Data Value with object: Se debe considerar que los datos primitivos pueden formar parte de una nueva clase cuyos atributos tienen su propio comportamiento.
* Extract class: Cuando una clase realiza más de una responsabilidad o el trabajo de otra clase es necesario considerar extraer este código para formar una clase nueva.
* Replacetypecodewithclass: El código de tipo lo podemos notar cuando, en lugar de un tipo de datos separado, se muestra un conjunto de números o cadenas formando una lista de valores para alguna entidad.

Ejemplo:

La clase Transporte tiene constantes con tipos de datos primitivos.



Al aplicar la técnica de refactorización Replace data value with object estos datos primitivos pasan a formar parte de una clase nueva. (package primitiveObsession)

* Divergent Change – Cambio divergente

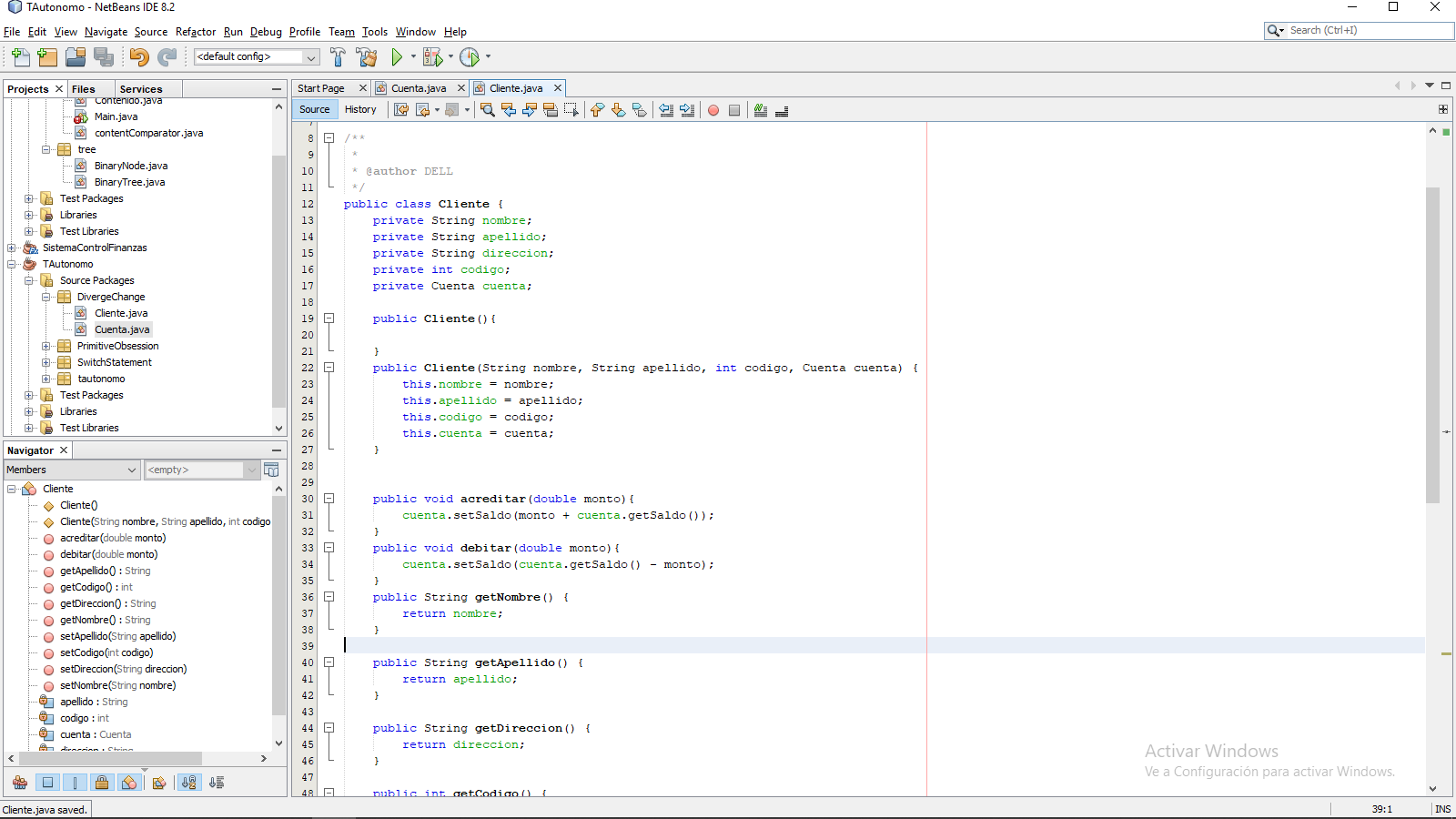
Este mal olor en el código ocurre cuando se necesita modificar una clase porque se hicieron muchos cambios sobre sus métodos.

Las técnicas de refactorización posibles a aplicar son las siguientes:

* Extract Class: Cuando una clase realiza más de una responsabilidad o el trabajo de otra clase es necesario considerar extraer este código para formar una clase nueva.
* Extract SuperClass: Cuando dos clases con campos y métodos comunes aparecen en nuestro código. Se debe considerar extraer esta porción de código y crear una nueva superclase compartida entre las clases que comparten los campos y métodos.
* Extract SubClass: Cuando una clase tiene características que se usan solo en ciertos casos. Para estos casos se debe crear una subclase y usarla en estos casos.

Ejemplo:

La clase Cliente tiene un atributo cuenta de tipo Cuenta, a la cual se accede para modificar el saldo de la cuenta, utilizando los métodos de otra clase. Este cambio en el valor del saldo es responsabilidad de la propia clase Cuenta. Para lo cual se extraen estos métodos de la clase Cliente.



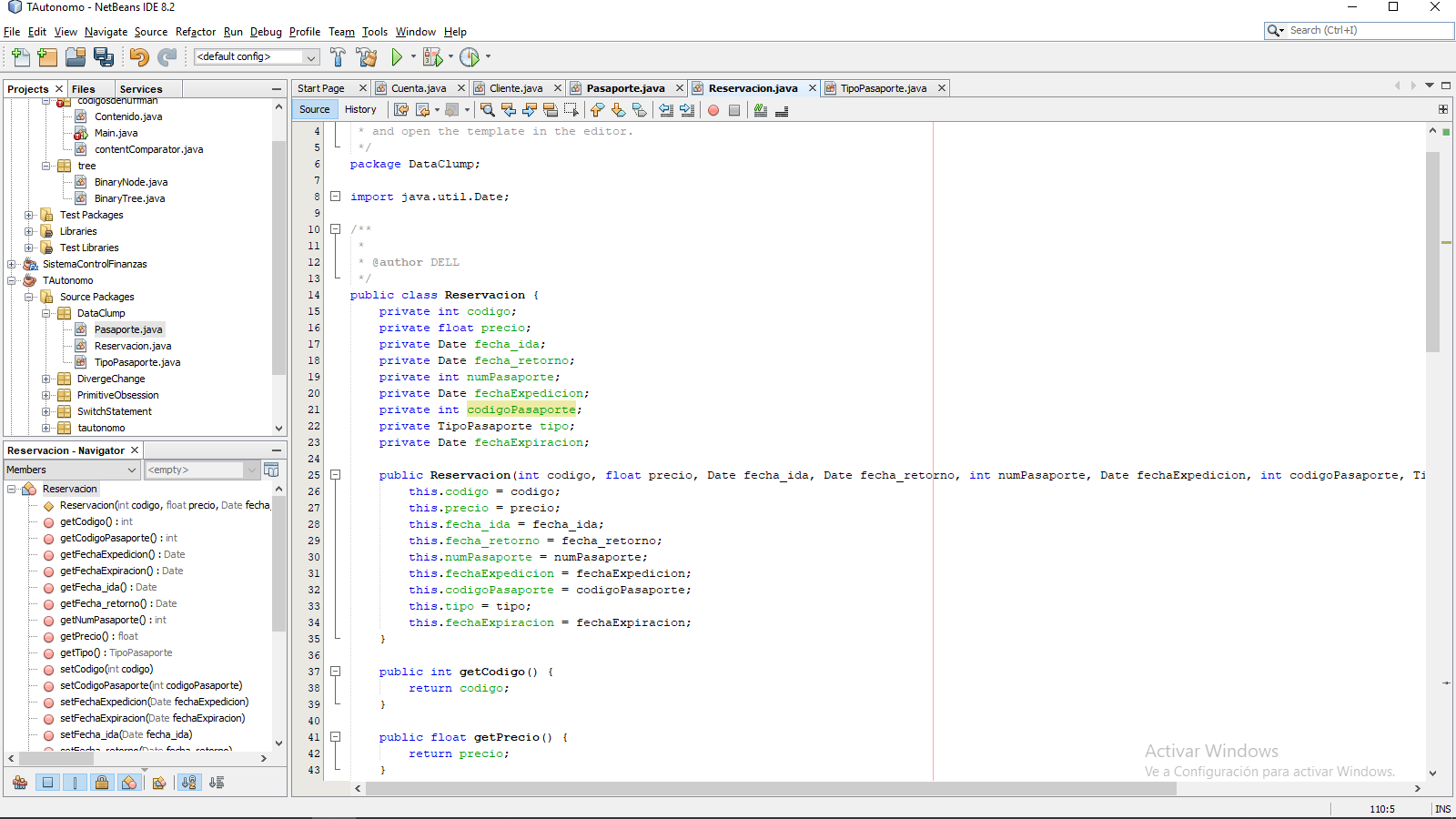
* Data clumps – Grupos de Datos

Este mal olor se presenta cuando existe un conjunto de datos que se repite en diferentes partes de Código y del que se puede crear una nueva clase a partir de estos datos.

Las técnicas de refactorización posibles a aplicar son las siguientes:

* Extract Class: Cuando una clase realiza más de una responsabilidad o el trabajo de otra clase es necesario considerar extraer este código para formar una clase nueva.
* Introduce Parameter: Se debe considerar refactorizar cuando existen varios métodos con los mismos parámetros, en este caso será mejor crear un objeto con los parámetros que se repiten.
* Preserve whole Object: Se debe considerar que al obtener varios valores de un objeto y luego pasarlos como para parámetro a un método es recomendable crear un objeto con todos los parámetros.

Ejemplo: La clase Reservacion(reserva de vuelo) tiene un grupo de datos que pueden pertenecer a la clase Pasaporte. Por lo cual, se creará una clase Pasaporte y se colocará este grupo de datos en esta clase.



* Long Method – Método Grande

Este mal olor se presenta cuando una clase tiene muchos métodos o campos o sobrepasa las 10 líneas de código.

Las técnicas de refactorización posibles a aplicar son las siguientes:

* [Extract Class](https://refactoring.guru/extract-class): Cuando una clase realiza más de una responsabilidad o el trabajo de otra clase es necesario considerar extraer este código para formar una clase nueva.
* Extract Subclass: Cuando una clase tiene características que se usan solo en ciertos casos. Para estos casos se debe crear una subclase y usarla en estos casos.
* Switch statements – Cambiar declaraciones

Las técnicas de refactorización posibles a aplicar son las siguientes:

* Extract Method: Cuando cierto fragmento de código se puede agrupar en un método.
* Move Method: Cunado un método es usado por otra clase que la propia, en este caso se debe crear una nueva clase y colocar el código del método usado en esta clase nueva.
* Replace Type Code with subclases: Tiene un tipo codificado que afecta directamente el comportamiento del programa.
* Replace Type Code with State: El código de tipo ocurre cuando, en lugar de un tipo de datos separado, tiene un conjunto de números o cadenas que forman una lista de valores permitidos para alguna entidad.

**Bibliografía**

*Nogal, A. (2016, May 27). Code Smell(Part 2). Recuperado de:* [*https://dzone.com/articles/code-smells-part-ii*](https://dzone.com/articles/code-smells-part-ii)

# *Ershad, G. (2017, Ene 7). Switch Statement - A Code Smell. Recuperado de:*

[*https://www.c-sharpcorner.com/article/switch-statement-a-code-smell/*](https://www.c-sharpcorner.com/article/switch-statement-a-code-smell/)

# *Mohamed, A. (2018, Agos 13). Write clean code and get rid of code smells with real life examples. Recuperado de:* [*https://codeburst.io/write-clean-code-and-get-rid-of-code-smells-aea271f30318*](https://codeburst.io/write-clean-code-and-get-rid-of-code-smells-aea271f30318)

*Sourcemaking, (2020). Refactoring. Recuperado de:* [*https://sourcemaking.com/refactoring/replace-type-code-with-subclasses*](https://sourcemaking.com/refactoring/replace-type-code-with-subclasses)