Code smell: Lazy Class



Como se puede observar esta clase AccountType lo que hace es decir si el tipo de cuenta es premium o no, lo cual la hace una clase vaga ya que este comportamiento tranquilamente puede ser realizado en una clase mucho más completa, por ejemplo, la clase Account.



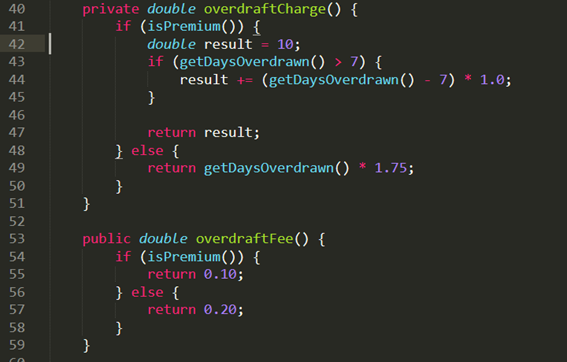
Tecnica de refactorización: Inline Class

Como ya se mencionó lo que se precederá hacer es eliminar esta clase y ubicar su contenido en la clase Account, de esta manera evitamos tener una clase que solo hace una funcionalidad demasiado básica.

Solución:

A continuación, se muestra los cambios que se hicieron en las clases, para el caso de AccountType lo que se hizo fue pasar toda la funcionalidad de la clase a Account, donde se creo un atributo String denominado type en cual cumple la misma funcionalidad que la clase AccountType.





Code smell : Duplicate Code



Como se puede observar la clase Person posee un método llamada withdraw, el cual recibe como parámetro money, a simple vista se puede notar que las líneas de códigos se encuentran duplicadas, por lo cual se procederán a usar las técnicas de refactorización para mejorar este código.

Técnica de refactorización: Substitute Algorithm

Como nos podemos dar cuenta, nos encontramos ante un caso bastante raro de código duplicado, ya que por lo general se dan estos casos cuando debemos optar por una opción u otra, pero ambas realizan actividades similares. Aquí como se muestra en la imagen da igual el tipo de cuenta que tengas, no importa si es premium o no, el método withdraw realizará la misma acción para todos. Por lo general en este tipo de code smell se usa el Extract method, pero como digo, no hay diferentes opciones a seleccionar, y con lo cual no se debe separar el método ya que en sí solo hace una funcionalidad general, por lo que se decidió usar substitute Algorithm.

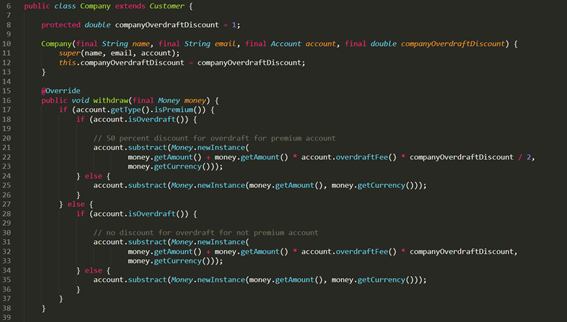
Solución:

Lo que se hizo fue sustituir el código que se tenía, ya que como se explicó realizaba la misma acción sin importar el tipo de cuenta que sea, por lo cual se procedió a eliminar líneas de códigos innecesarias, ya que esto afecta la eficiencia del método y del programa en general.



Code Smell: comments

Como se puede observar nuevamente tenemos el método withdraw, solo que en esta ocación se encuentra implementado en la clase Company, a simple vista podemos observar que existe codigo duplicado, pero en eso ya no nos enfocaremos, sino más bien en el hecho de que posee comentarios que quieren hacernos entender la funcionalidad del código, ya que a simple vista parece que hiciera lo mismo en ambos casos, pero no.



Técnica de refactorización: Extract method

Se procedió a usar esta técnica de refactorización debido a que el código realmente es complicado de entender si se queda tal como está en la imagen, es por esto que se decidió usar extract method, para crear métodos aparte en base al código existente, y asignándole un nombre que sea mucho más explicativo, es decir que el código hable por sí solo.

Solución:

Lo que se procedió hacer es crear métodos aparte usando extract method de tal forma que sea mucho más fácil de entender que es lo que intentaba hacer esa parte del código, ahora si bien es evidente que al hacer esto sigue habiendo código duplicado, podemos darnos cuenta que el método withdraw es mucho más fácil de comprender.

Una captura de pantalla de una computadora

Descripción generada automáticamente