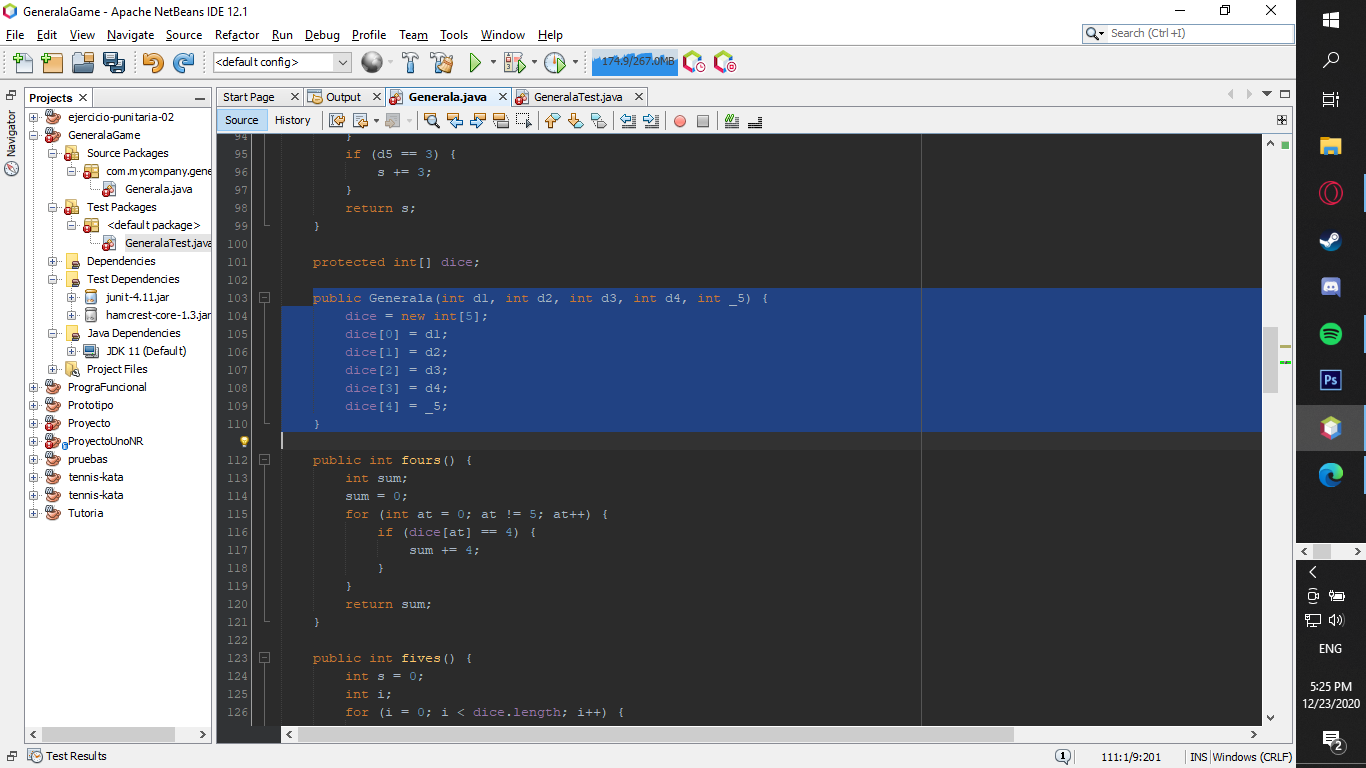
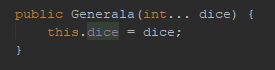
Buenas profesor, junto con saludar procedo a mostrarle lo realizado para esta evaluación.

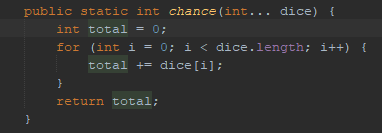
En primer lugar, el código estaba bastante desordenado, tenia atributos entre funciones y el constructor estaba a la mitad de la clase, lo cual no afecta su funcionamiento, pero es una mala práctica, por lo cual opté por empezar ordenando el código.



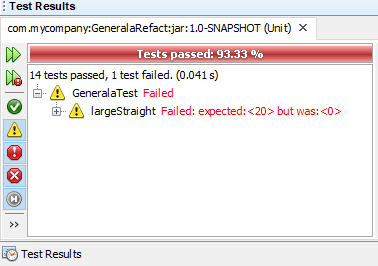
Una vez bien ordenado el código, procedí a refactorizar el constructor de la clase, ya que tenía un montón de code smells y malas prácticas, adjunto captura del cambio.



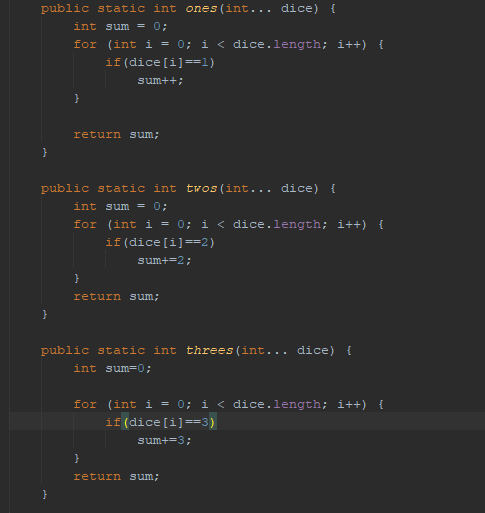
Luego, la función chance al igual que la anterior y muchas otras, tenía un montón de malas prácticas, para lo cual opte por incluir un for en la función para sumar el total de puntos al lanzar los dados.



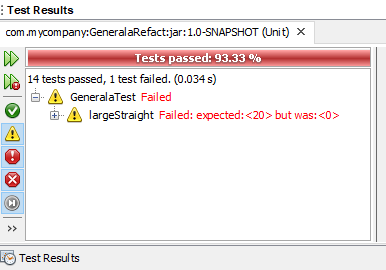
Una vez realizados todos estos cambios realicé el primer test, el cual muestra que todo se mantiene igual, solo aparece un fallo que es por el método largeStraight el cual dejé para el final, por lo cual en todas las pruebas menos la última, debería aparecer un 93.33%.



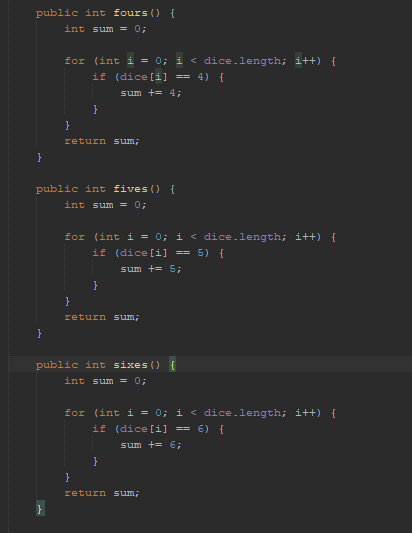
Luego, procedí a modificar en orden los métodos ones, twos y threes, esto resulto bastante fácil ya que eran bastante parecidos, por lo cual al resolver tan solo uno, pude resolver los otros dos.



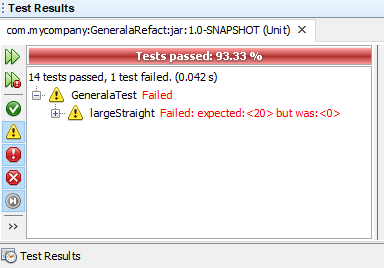
Luego de realizar las modificaciones anteriormente mencionadas, otra vez volví a ejecutar las pruebas unitarias, arrojando el resultado esperado que era 93.33%.



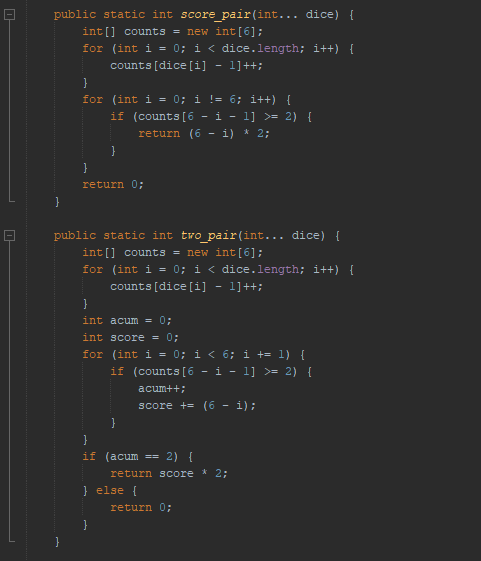
Una vez finalizados estos métodos, pasé a los fours, fives y sixes, los cuales eran bastante distintos a los 3 anteriores y al no poder modificar el archivo de pruebas, tuve que pensar en como refactorizar estas. Para mi sorpresa estas funciones tenían bastante poco code smell, pero de todas maneras intenté mejorarlas, quedando de la siguiente manera.

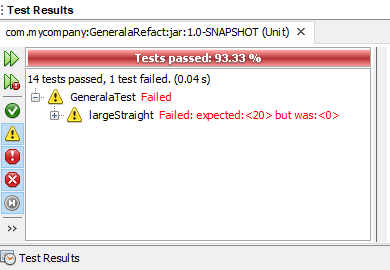


Una vez realizado este grupo de cambios, volví a realizar las pruebas, mostrando el resultado esperado.

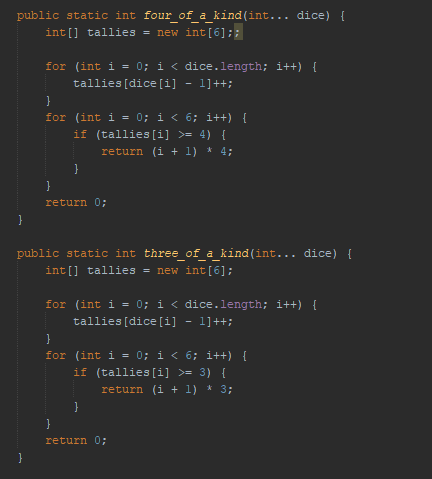


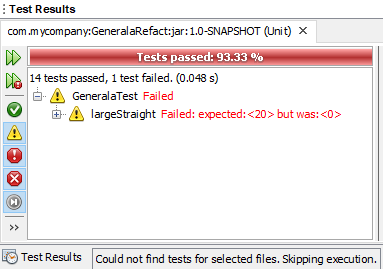
Siguiendo con la limpieza de código, esta vez seguí realizando cambios a las funciones score\_pair y two\_pair, en esta ocasión, si me encontré con un montón de código basura, el cual refactoricé lo mejor que pude, quedando de la siguiente manera y pasando todas sus pruebas.



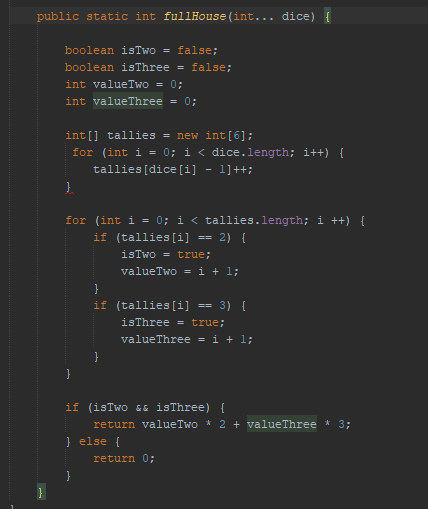


Una vez finalizadas las pruebas, seguí con los métodos four\_of\_a\_kind y three\_of\_a\_kind, los cuales estaban programados con algo de code smell, pero no tan exagerado como en otras ocasiones, por lo cual resulto fácil al ser métodos parecidos, ambos pasaron sus pruebas sin problemas.

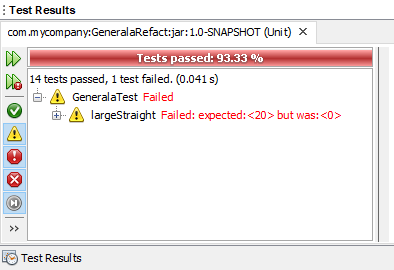




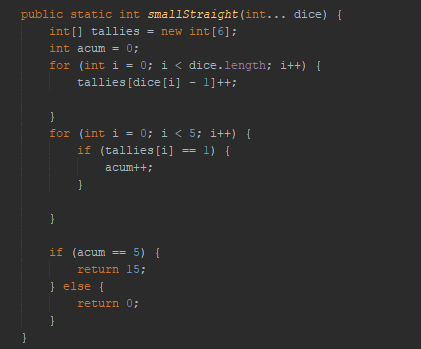
Luego, procedí al refactoring de fullHouse, el cual considero que fue el más difícil de todos los métodos, ya que era bastante extenso y tenía un montón de code smell y malas prácticas, las cuales hacían muy difícil comprender lo que estaba programado, pero a pesar de todo logre comprender el código y cambiar algunas cosas para hacerlo mas entendible.

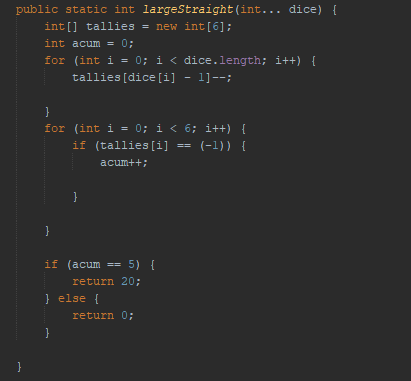


Realicé el penúltimo testeo, el cual paso de manera correcta, sin contar el método que falta que en la siguiente prueba debería pasar bien.

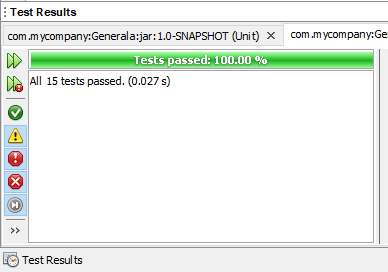


Luego de esto procedí a realizar las ultimas dos funciones, que eran smallStraight la cual solo necesitaba un poco de refactoring, para tener un funcionamiento óptimo, por lo cual no se me complicó tanto. Por otra parte también debía realizar la ultima función, que era largeStraight, la cual se me complicó bastante ya que no entendía muy bien el juego, la solución que se me ocurrió, fue copiar el código de smallStraight en una clase main, y realizarle modificaciones, a prueba y error, hasta que conseguí el resultado final. (Después de mostrar las pruebas, adjuntare una imagen del proceso para conseguir el codigo)





Ahora se puede ver que con esta función creada pasaron el 100% de las pruebas.



Ese sería el resultado final de la prueba, a continuación, adjuntare el main señalado anteriormente donde probé el código.

public class IDK {

public static void main(String[] args) {

System.out.println(smallStraight(1,2,3,4,5));

System.out.println("==================");

System.out.println(largeStraight(6,2,3,4,5));

System.out.println("==================");

System.out.println(largeStraight(2,3,4,5,6));

System.out.println("==================");

System.out.println(largeStraight(1,2,2,4,5));

}

public static int smallStraight(int... dice){

int[] tallies = new int[6];

int acum = 0;

for (int i = 0; i < dice.length; i++) {

tallies[dice[i] - 1]++;

System.out.println(tallies[dice[i] - 1]);

}

for (int i = 0; i < 5; i++) {

if (tallies[i] == 1) {

acum++;

System.out.println("acum"+acum);

}

}

if (acum == 5) {

return 15;

} else {

return 0;

}

}

public static int largeStraight(int... dice) {

int[] tallies = new int[6];

int acum = 0;

for (int i = 0; i < dice.length; i++) {

tallies[dice[i] - 1]--;

System.out.println(tallies[dice[i] - 1]);

}

for (int i = 0; i < 6; i++) {

if (tallies[i] == (-1)) {

acum++;

System.out.println("acum"+acum);

}

}

if (acum == 5) {

return 20;

} else {

return 0;

}

}

}