Documentación TDD Taller 4 Cariqueo Cristobal y Opazo Daniel

1. Inicio del TDD

    a. Se omite el primer paso de TDD convencional dado que se sabe que la prueba va a fallar (Sugerido por ayudante para optimizar las cosas)

2. Test1

```java
//Test1

@Test
        public void testKaprekarOperation1() {
            // Prueba con el número 3524 (debe devolver 3087)
            Assertions.assertEquals(3087, kaprekarOp(3524));
        }
```

    a. Se crea ftr_t1 para pasar el test (en este momento al no haber nada previo lo tomaremos como el código inicial)

```java
    public static int kaprekarOp(int num) {
        // se agrega if para retornar valor esperado
        if (num == 3524) {
            return 3087;
        }
        return 0;
    }
```

3. Test2

```java
//Test2

@Test
        public void testKaprekarOperation2() {
            // Prueba con el número 1000 (debe devolver 999)
            Assertions.assertEquals(999, kaprekarOp(1000));
        }
```

    a. Se crea ftr_t2 para pasar el test

    b. Se procede con el refactoring

```java
public static int kaprekarOp(int num) {
    if (num == 3524) {
        return 3087;
    }
    // Se registra valor esperado para la siguiente prueba
    return 999;
}
```

4. Test3

```java
//Test3

@Test
public void testKaprekarOperation3() {
    // Prueba con el número 5200 (debe devolver 5175)
    Assertions.assertEquals(5175, kaprekarOp(5200));
}
```

   a. Se crea ftr_t3 para pasar el test

   b. Se procede con el refactoring

```java
public static int kaprekarOp(int num) {
    if (num == 3524) {
        return 3087;
    } else if (num == 5200) { // Se agrega código necesario para aprobar Test 3
        return 5175;
    }
    return 999;
}
```

5. Test4

```java
//Test4

@Test
public void testKaprekarOperation4() {
    // Prueba con el número 2111 (debe devolver 999)
    Assertions.assertEquals(999, kaprekarOp(2111));
}
```

   a. Se crea ftr_t4 para pasar el test
   b. No se hacen cambios en el código dado que este ya cumplia con pasar el test4

6. Test5

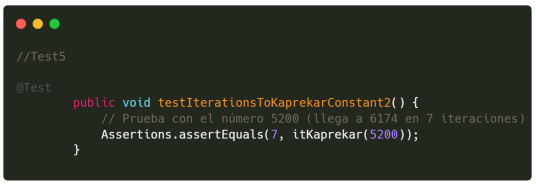
```java
//Test5

@Test
        public void testIterationsToKaprekarConstant1() {
            // Prueba con el número 3524 (llega a 6174 en 3 iteraciones)
            Assertions.assertEquals(3, itKaprekar(3524));
        }
```

a. Se crea ftr_t5 para pasar el test

b. Se procede con el refactoring

```java
public static int kaprekarOp(int num) {
        int min = minOrder(num);
        int max = maxOrder(num);

        return max - min;
    }

    private static int maxOrder(int num) {
        String[] strArr = String.valueOf(num).split("");
        int[] intArr = new int[4];

        for (int i=0; i<strArr.length; i++) {
            intArr[i] = Integer.parseInt(strArr[i]);
        }

        Object[] intArr2 =
Arrays.stream(strArr).sorted(Collections.reverseOrder()).toArray();
        StringBuilder out = new StringBuilder();

        for (int i=intArr2.length-1; i>=0; i--) {
            out.append(intArr2[i]);
        }

        return Integer.parseInt(String.valueOf(out));
    }

    private static int minOrder(int num) {
        String[] strArr = String.valueOf(num).split("");
        int[] intArr = new int[4];

        for (int i=0; i<strArr.length; i++) {
            intArr[i] = Integer.parseInt(strArr[i]);
        }

        Object[] intArr2 =
Arrays.stream(strArr).sorted(Collections.reverseOrder()).toArray();
        StringBuilder out = new StringBuilder();

        for (Object o : intArr2) {
            out.append(o);
        }

        return Integer.parseInt(String.valueOf(out));
    }

    public static int itKaprekar(int num) {
        int out = num;
        for (int i=0; i<7; i++) {
            out = kaprekarOp(out);
            if (out == 6174) return i+1;
        }
        return 0;
    }
```

7. Test6

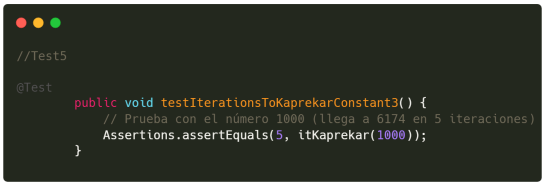


```
//Test5

@Test
        public void testIterationsToKaprekarConstant2() {
            // Prueba con el número 5200 (llega a 6174 en 7 iteraciones)
            Assertions.assertEquals(7, itKaprekar(5200));
        }
```

   a. Se crea ftr_t6 para pasar el test

   b. Se procede con el refactoring (no hay cambios porque el codigo ya cubre los casos)

8. Test7



```
//Test5

@Test
        public void testIterationsToKaprekarConstant3() {
            // Prueba con el número 1000 (llega a 6174 en 5 iteraciones)
            Assertions.assertEquals(5, itKaprekar(1000));
        }
```

   a. Se crea ftr_t7 para pasar el test

   b. Se procede con el refactoring (no hay cambios porque el codigo ya cubre los casos)

   c.