UNIVERSIDAD
DE LA FRONTERA

Producto Adyacente

Integrantes:
Javier Alcalde Vivas
Joaquín Faundez Concha
Christian Gajardo Contreras.

Asignatura:
Programación Orientada a Objetos

Profesor:
Samuel Eduardo Sepulveda Cuevas.

1. Introducción

En equipo de 3 se realizará el desarrollo del siguiente caso, teniendo en cuenta el uso de buenas prácticas y casos de prueba unitaria. Poniendo en práctica todo lo ya visto con anterioridad.

2. Caso

Se entrega un arreglo de enteros a un método llamado producto Adyacentes. Dicho método debe retornar el mayor producto de números adyacentes que encuentre.

P. ej., dado el arreglo = {1, -4, 2, 2, 5, -1}, el mayor producto de números adyacentes es el de los números 2 y 5, en cuyo caso el método retorna el valor 10.

Luego, si se hace algo como System.out.print("El producto adyacente es: "+ producto Adyacentes(arreglo));

En pantalla debería ver: 10.

3. Estructura de Caso:

Primero se pide al usuario ingresar el largo del arreglo, el cual debe ser entre 2 y 20, se manejan los casos de excepción en los que si ingresan valores distintos a números enteros.

Después de esto se rellena el array con números randoms entre -1000 y 1000, para después pasar al método principal (productoAdyacente) donde se realizan las operaciones mencionadas anteriormente y se devuelve el mayor producto adyacente por consola.

Actividad 0:

(0) La lista se genera automáticamente con random

(1) Parámetros de entrada: lista; 2<largo<20; -1000 < valor < 1000

(2) Valor retorno: double o int

(3) instrucciones:

1. instancia una variable = 0 para ir comparando, y almacenando el mayor producto
2. ciclo for que recorra la lista hasta el largo - 1, utilizando i e i+1
3. se realiza la multiplicación de i * (i +1)
4. Se compara la variable con la multiplicación anterior.
5. Se iguala la variable a la multiplicación en caso de ser mayor al valor anterior.

tiempo real utilizado: 10 min (hora de término: 9:00 am)

Actividad 1:

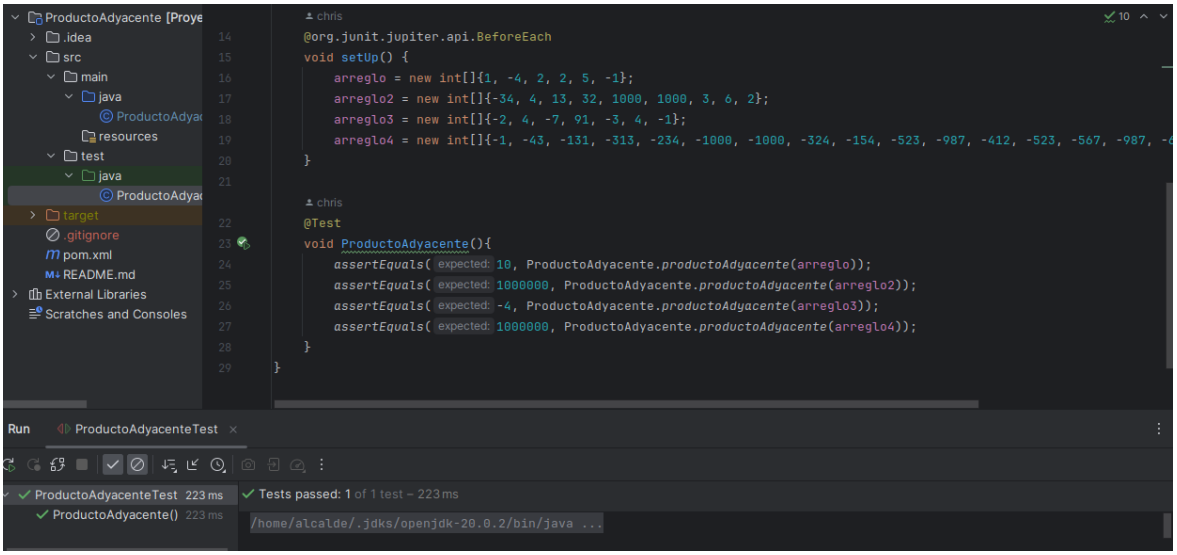
```
public static int productoAdyacente (int[] array){

    int ProductoMayor = 0;
    for (int indice = 0; indice < (array.length-1); indice++){
        if ((array[indice]*array[indice+1])>ProductoMayor || indice == 0){
            ProductoMayor = (array[indice]*array[indice+1]);
        }
    }
    return ProductoMayor;
}
```

El método principal recibe un array de largo entre 2-20, que fue rellenado con números randoms entre -1000 y 1000.

Actividad 2:

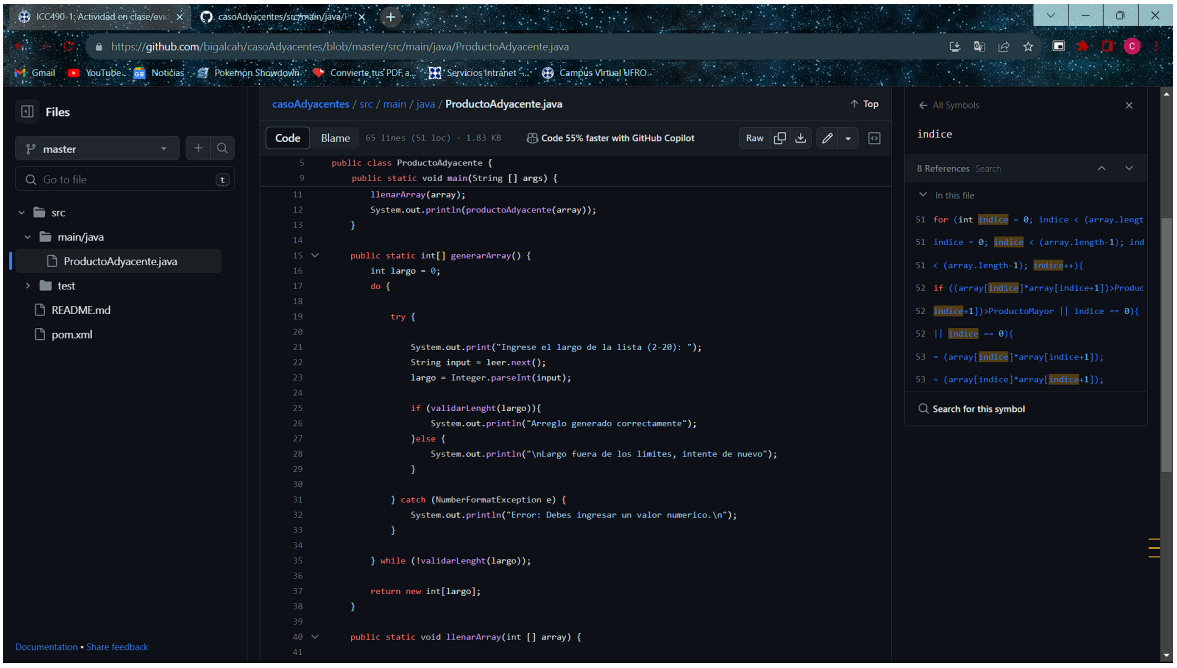
Caso 1 Ejemplo recomendado en campus virtual que verifica su funcionamiento básico.
Caso 2 Producto de los dos valores más grandes que pueden existir 1000*1000
Caso 3 Arreglo de solo productos negativos. (retorna el más cercano al cero)
Caso 4 Arreglo de solo valores negativos, de largo 20 y con producto (-1000)*(-1000)

Actividad 3:



Try-Catch:

En este caso solo se utilizó un try-catch en el método de generación automática del arreglo para que no se caiga la recibir un largo erróneo, esto debido a que no se encontró caso en donde el cálculo del producto dé error.



.

Link Git:https://github.com/bigalcah/casoAdyacentes/tree/master

Conclusiones:

Actividad	T. esperado	T. real
0	8 min	10 min
1	20 min	29 min
2	20 min	15 min
3	30 min	60 min

En conclusión se puede notar que las actividades esperadas y las obtenidas son distintas, a continuación se detallará la razón de esto:

- Actividad 0: el tiempo real fue muy similar al esperado, ya que se estaba realizando la planificación del caso y había debates entre los integrantes.
- Actividad 1: No se tuvo en cuenta la creación de los demás métodos a la hora de definir estos, por los que nos tomó más tiempo del estimado
- Actividad 2: Esta se demoró menos en realizarse puesto a que ya habían algunas ideas de test en el campus y solo completamos las que faltaban.
- Actividad 3: La última actividad fue la que más tiempo requirió, de hecho fue el doble de tiempo al esperado. Esto fue más que nada porque había que integrar los métodos y hacer el menú, también la tarea de hacer los merge de cada rama a la master sumó más tiempo.