
Tecnológico de Monterrey CSF
Estructura de Datos

# Proyecto 2
# Recursión

M. Santiago Hernández Gutiérrez – A01338717

14 de febrero de 2019

# MANUAL DE USUARIO

## DESCRIPCIÓN

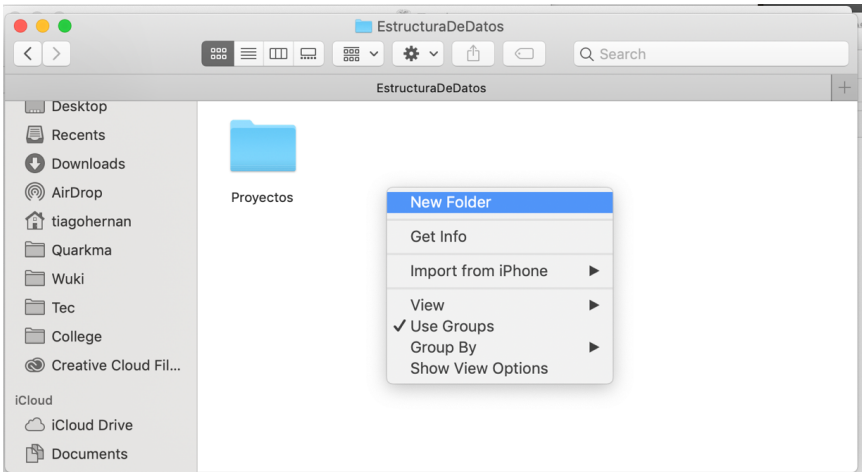
En este manual de usuario se detallan los pasos para implementar con éxito la clase (estructura de datos) de Punto. Probaremos los métodos para obtener datos, asignar valores e imprimir en la terminal.

## PRE-REQUISITOS

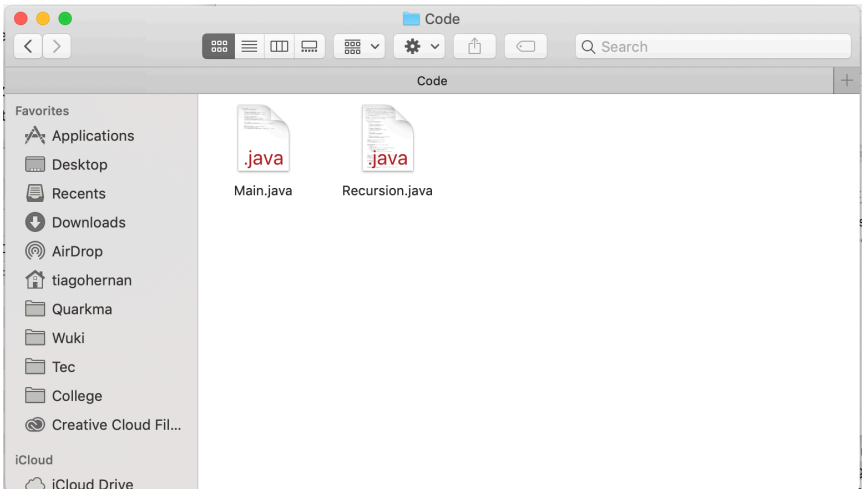
1. Tener Java instalado.
2. Contar con un editor de texto.

## INSTRUCCIONES

1. Crea un nuevo directorio para guardar los archivos del proyecto.




2. Arrastra el archivo Recursion.java y Main.java a la nueva carpeta.



3. Abre un editor de código.

4. Crea un archivo con la clase donde quieras implementar la clase de Recursion en la misma carpeta donde se encuentra el archivo Recursion.java. En este ejemplo nos basaremos en la clase Main, también incluida en el paquete de código.

5. Dentro del método main de nuestra clase, lo primero que haremos será crear una instancia de la clase Recursion.

```
Recursion obj = new Recursion();
```

6. La primera función de la API nos permite “limpiar” un string, es decir, eliminar las letras adyacentes iguales.

   Para implementarlo escribiremos el string que se requiere limpiar, utilizaremos el objeto de la clase para llamar al método de *limpiaString(),* guardaremos el nuevo string en una variable y lo imprimiremos.

```
String str = "aabbbbccccdd";
String limpio = obj.limpiaString(str);
System.out.println(limpio);
```

   El resultado al ejecutar Main.java en la terminal es:

```
$ java Main
El string a limpiar es: aabbbbccccdd
El string limpio es: abcd
```

7. La segunda función nos permite contar el número de veces que un substring dado aparece en un string.
   Para implementarlo, escribiremos el string del que queremos contar el substring, escribiremos el substring en cuestión, utilizaremos la función *cuentaSubstring()*, e imprimiremos el resultado. Un ejemplo de esto puede ser:

```
        String str = "abbaababbbababba";
        String subStr = "ab";
        System.out.println("El string es: " + str);
        System.out.println("El substring es: " + subStr);
        int cuenta = obj.cuentaSubstring(str, subStr);
        System.out.println("El string " + subStr + " aparece " + cuenta
+ " veces");
```

En la terminal, después de ejecutar el código, observaríamos algo similar a lo siguiente:

```
        $ java Main
        El string es: abbaababbbababba
        El substring es: ab
        El string ab aparece 5 veces
```

8. El método tres del API, dado un entero no negativo, regresa la suma de sus dígitos. Para implementarlo en nuestro código primero escribiremos un número entero, después guardaremos el resultado del método de *sumaDigitos()* en una variable, y finalmente imprimiremos el resultado. Podemos ver un ejemplo a continuación:

```
        int n = 12345;
         int sumaDigitos = obj.sumaDigitos(n);
         System.out.println("La suma de los dígitos de " + n + " es " +
sumaDigitos);
```

En la terminal, después de ejecutar el código, observaríamos lo siguiente:

```
        $ java Main
        La suma de los dígitos de 12345 es 15
```

9. Finalmente, el 4to método del API (*anidacionCorrecta()*) nos permite, dado un string formado por paréntesis anidados, regresar TRUE si están anidados correctamente o FALSE en caso contrario. Podemos implementarlo si primero declaramos nuestro string conformado por paréntesis anidados y después aplicamos la función

anidacionCorrecta() con el string como parámetro. Al final podemos imprimir el resultado para visualizarlo claramente.

```
      String str = "(())()()";
       boolean anidacionCorrecta = obj.anidacionCorrecta(str);
       System.out.println("El string anidado '" + str + "' está "
               + (anidacionCorrecta ? "correctamente" :
"incorrectamente") + "anidado");
```

El resultado en la terminal se vería así después de ejecutarlo:

```
      $ java Main
      El string anidado '(())()()' está correctamente anidado
```