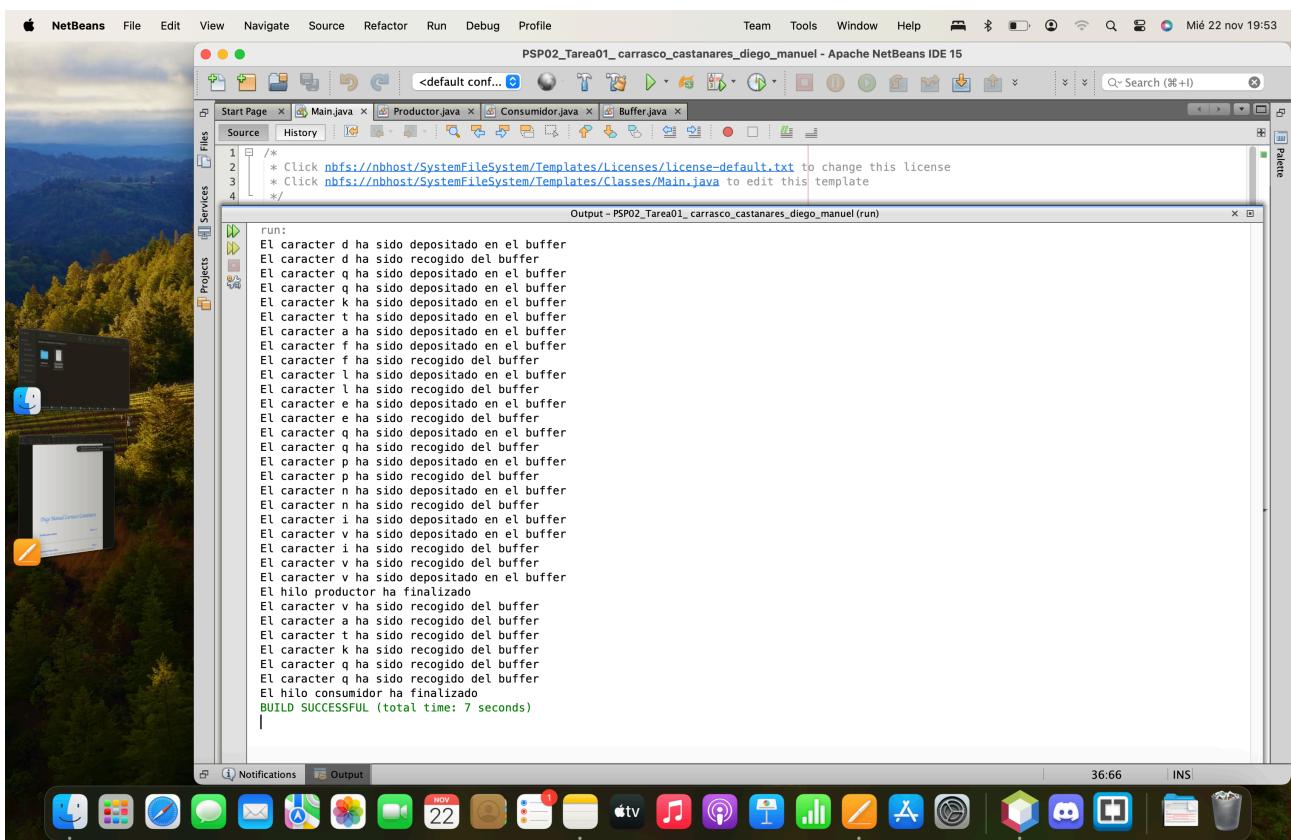
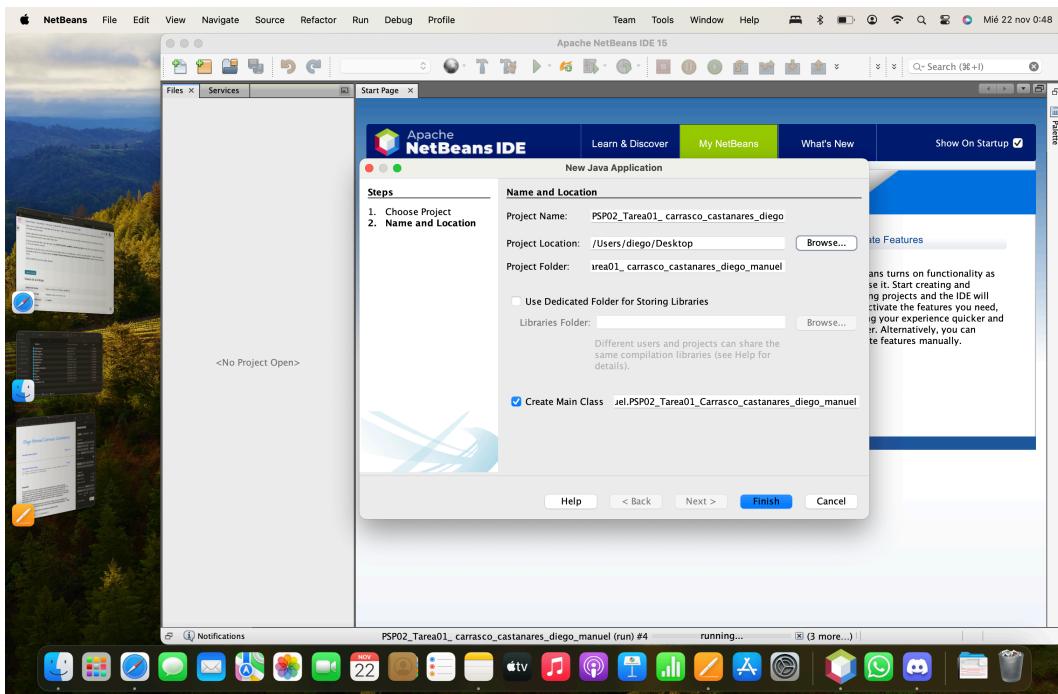


Tarea 2.1 para Programación de Servicios y Procesos

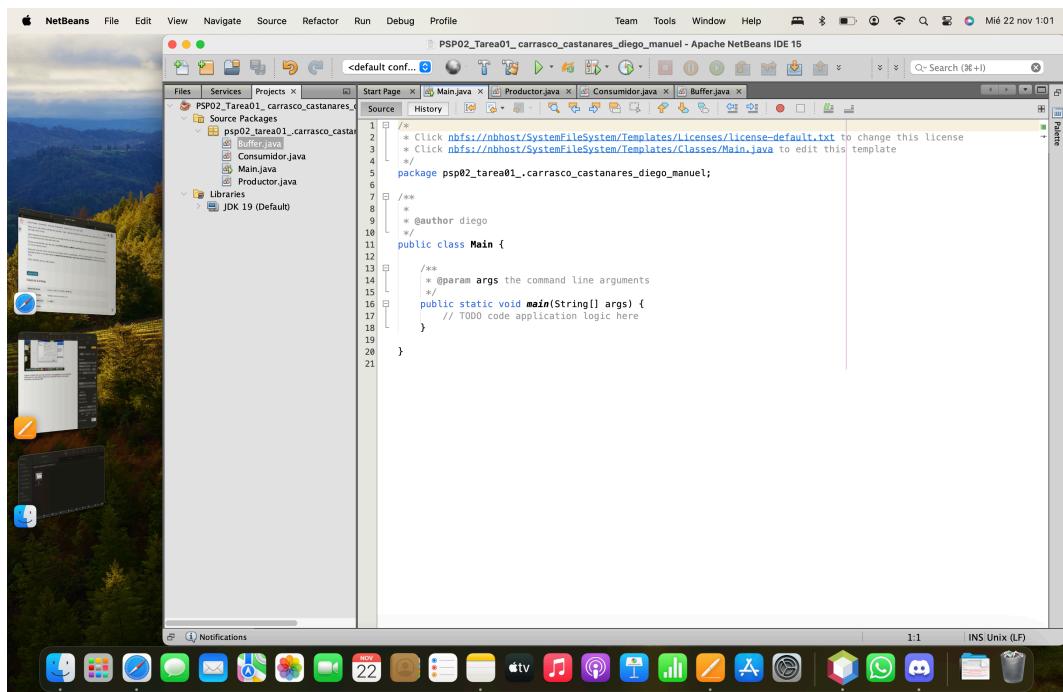


Diego Manuel Carrasco Castañares

Lo primero que realizaremos para llevar a cabo nuestro ejercicio es crear un proyecto en NetBeans.



Creamos el método main ya al crear el proyecto y le renombramos con el nombre Main. Ademas vamos a crear otras tres clases mas, una llamada Productor, otra llamada Consumidor y otra llamada Buffer.



En mi caso voy a empezar construyendo la clase Buffer.

En ella vamos a crear un array de char llamado buffer, un int que se llamará next y dos boleados, uno llamado full y otro llamado empty que serán los encargados de indicar si esta lleno o vacío el array.

Posteriormente crearemos su constructor iniciando a 0 la variable contador next, a false el boolean full y a true el boolean empty.

Lo siguiente que haremos será crear el método consumir, que al estar en una zona crítica trabajando será synchronized. Esté método esperará si está en false la variable empty (vacío el array), en caso contrario restara uno a la variable next, al estar consumiendo no está vacío por lo que la variable full será false. Pasará a true la variable empty cuando el contador next sea igual a 0. Por último notificara a todo los hilos que ha terminado y retornara el valor de next al buffer.

Lo último que crearemos en esta clase será el método producir, que al trabajar en una zona crítica será synchronized. Mientras la variable full sea verdadera (lleno el array) este método esperara. En caso contrario le pasará al buffer el valor de c (el carácter que le pasamos desde la clase producir), aumentamos la variable next en uno por lo que al estar produciendo no está vacío (la variable empty es false). Si la variable next es igual al tamaño del buffer pasará a verdadero la variable full. Por último notificará a los demás hilos que ha terminado.

En las siguientes imágenes podemos ver el código.

```

13  /*
14  * public class Buffer {
15  *
16  *     /**
17  *      * Creamos el array de char llamado buffer, un int que se llamará next y
18  *      * dos booleanos, uno llamado full y otro llamado empty que serán los
19  *      * encargados de indicar si esta lleno o vacío el array
20  *     */
21  *     private final char[] buffer;
22  *     private int next;
23  *     private boolean full, empty;
24  *
25  *     // Creamos el constructor
26  *     public Buffer(int capacidad) {
27  *
28  *         this.buffer = new char[capacidad]; // Creamos el array de char con la capacidad que le pasamos
29  *         this.next = 0; // Iniciamos a 0 el contador next
30  *         this.full = false; // Inicialmente no está lleno, lo iniciamos a false
31  *         this.empty = true; // Inicialmente está vacío, lo iniciamos a true
32  *     }
33  *
34  *     // Creamos el método consumir, al ser una sección crítica usamos synchronized
35  *     public synchronized char consumir() {
36  *
37  *         while (this.empty) { // Si esta vacío le decimos que espere
38  *             try {
39  *                 wait(); // Debe ir envuelto en try catch
40  *             } catch (InterruptedException ex) {
41  *                 Logger.getLogger(Buffer.class.getName()).log(Level.SEVERE, null, ex);
42  *             }
43  *         }
44  *
45  *         this.next--; // Vamos restando uno al contador para cada carácter consumido
46  *         this.full = false; // Si consume, le indicamos que no está lleno
47  *
48  *         if (this.next == 0) { // Si el contador están en 0 le indicamos que está vacío.
49  *             this.empty = true; // Por lo tanto empty es verdadero
50  *         }
51  *     }
52  *
53  *     // Creamos el método producir
54  *     public void producir(char c) {
55  *
56  *         if (this.full) { // Si el array está lleno no lo podemos poner
57  *             return;
58  *         }
59  *
60  *         this.buffer[this.next] = c; // Ponemos el carácter en el array
61  *         this.next++; // Aumentamos el contador en 1
62  *         this.full = true; // Si lo ponemos, lo indicamos que está lleno
63  *     }
64  *
65  * }
66

```

```

 53     this.empty = true; // Por lo tanto empty es verdadero
 54   }
 55 
 56   }
 57 
 58   notifyAll(); // Notificamos a todos los hilos que ha terminado de consumir
 59   return this.buffer[this.next]; // Retornamos al buffer el valor de la variable next
 60 }
 61 
 62 // Creamos el método producir, al ser una sección crítica usamos synchronized
 63 public synchronized void producir(char c) {
 64 
 65   while (this.full) { // Mientras este lleno...
 66     try {
 67       wait();
 68     } catch (InterruptedException ex) {
 69       Logger.getLogger("Buffer.class.getName()).log(Level.SEVERE, null, ex);
 70     }
 71   }
 72 
 73   this.buffer[this.next] = c; // Guardamos en el buffer el valor de c
 74   this.next++; // Aumentamos la variable next
 75   this.empty = false; // Si producimos no está vacío
 76 
 77   if (this.next == this.buffer.length) { // Si la variable next es igual a la capacidad del buffer...
 78     this.full = true; // ... lleno es verdadero
 79   }
 80 
 81   notifyAll(); // Notificamos a todos los hilos que ha terminado de producir
 82 }
 83 
 84 }
 85 
 86 }
 87 
 88 }
 89 }
 90 }
 91 }
 92 }

```

El siguiente paso será crear el código de la clase Consumidor.

Le indicamos que extiende de la Thread.

Posteriormente creamos un buffer, una variable de tipo int llamada consumido que usaremos a modo de contador y una variable final de tipo int que llamaremos MAXCARACTER a la cual le daremos el valor de 15 según indica el ejercicio.

Posteriormente crearemos el constructor.

Lo siguiente será sobreescribir el método run. Este método, mientras que la variable consumido sea inferior a la variable MAXCARACTER, llamará al método consumir de la clase buffer, aumentará en uno la variable consumido, indicará por consola el carácter que ha consumido y se dormirá el tiempo indicado en el ejercicio.

Podemos ver el código en la siguiente imagen.

```

import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * 
 * @author diego
 */
public class Consumidor extends Thread { // Extiende de Thread

    private final Buffer buffer; // Declaramos un buffer
    private int consumido; // Declaramos una variable para usar de contador
    private final int MAXCARACTER = 15; // Declaramos una variable para indicar
    // el maximo de caracteres, que sera 15 segun indica el ejercicio

    // Creamos el metodo constructor
    public Consumidor(Buffer buffer) {
        this.consumido = 0;
        this.buffer = buffer;
    }

    // Sobreescrivimos el metodo run
    @Override
    public void run() {
        while (consumido < MAXCARACTER) { // Mientras que lo que consumimos sea
            // menos al maximo...
            try {
                char c = buffer.consumir(); // ... llamamos al metodo consumir de
                // la clase Buffer y guardamos el retorno
                consumido++; // Aumentamos la variable consumido en uno
                System.out.println("El caracter " + c + " ha sido recogido del buffer");
                // Indicame el caracter que hemos consumido
                sleep((int) (Math.random() * 1000)); // Dormimos el hilo el tiempo
                // indicado en el ejercicio
            } catch (InterruptedException ex) {
                Logger.getLogger(Consumidor.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
}

```

Lo siguiente que haremos será construir la clase productor.

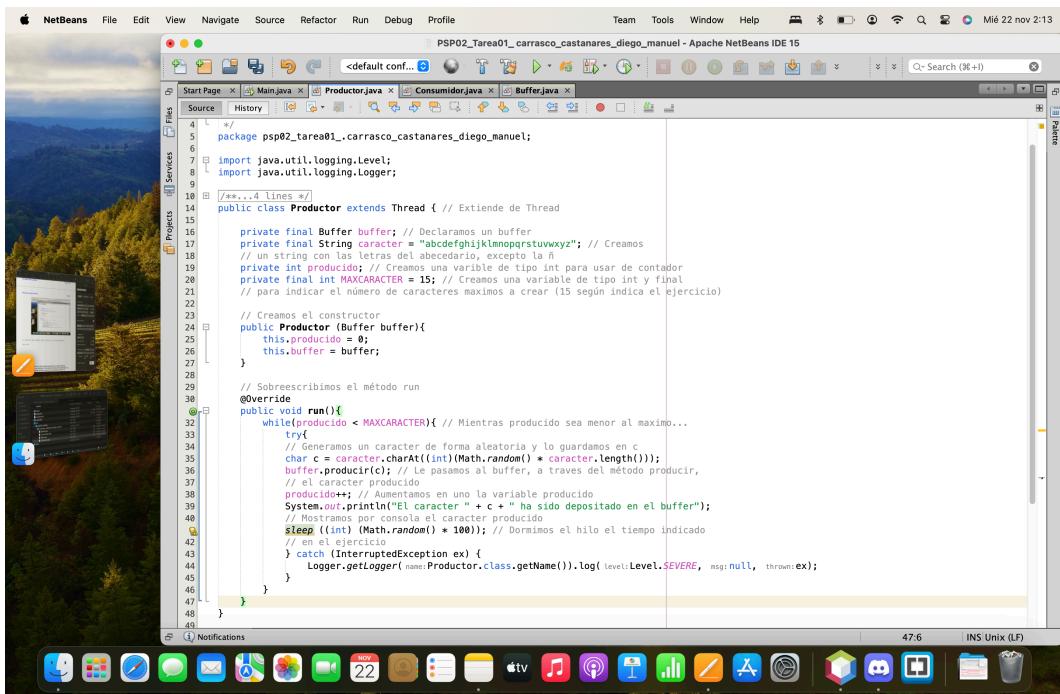
Para ello, lo primero, le indicamos que extiende de Thread.

Creamos una variable de tipo Buffer, un String que contendrá todas las letras del abecedario a excepción de la ñ, una variable de tipo int que usaremos de contador y una variable final de tipo int y final que usaremos para indicar el número máximo de caracteres a generar.

Lo siguiente que haremos será crear el constructor.

Posteriormente sobreescribimos el método run, el cual, mientras la variable producido sea menor a la variable MAXCARACTER generará de forma aleatoria un carácter de la cadena carácter creada anteriormente, lo guardara en el buffer a través del método producir de la clase buffer, aumentará la variable producido en uno, mostrara por pantalla el carácter producido y dormirá el hilo el tiempo indicado en el ejercicio

Podemos ver el código en la siguiente imagen.



```

4  /*
5   * package psp02_tarea01_.carrasco_castañares_diego_manuel;
6   */
7   import java.util.logging.Level;
8   import java.util.logging.Logger;
9
10 /**
11  * @author diego
12 */
13
14 public class Productor extends Thread { // Extiende de Thread
15
16     private final Buffer buffer; // Declaramos un buffer
17     private final String caracter = "abcdefghijklmnoprstuvwxyz"; // Creamos
18     // un string con las letras del abecedario, excepto la ñ
19     private int producido; // Creamos una variable de tipo int para usar de contador
20     private final int MAXCARACTER = 15; // Creamos una variable de tipo int y final
21     // para indicar el numero de caracteres maximos a crear (15 segun indica el ejercicio)
22
23     // Creamos el constructor
24     public Productor (Buffer buffer){
25         this.producido = 0;
26         this.buffer = buffer;
27     }
28
29     // Sobreescribimos el metodo run
30     @Override
31     public void run(){
32         while(producido < MAXCARACTER){ // Mientras producido sea menor al maximo...
33             try{
34                 // Generamos un caracter de forma aleatoria y lo guardamos en c
35                 char c = caracter.charAt((int)(Math.random() * caracter.length()));
36                 buffer.producir(c); // Le pasamos al buffer, a traves del metodo producir,
37                 // el caracter producido
38                 producido++; // Aumentamos en uno la variable producido
39                 System.out.println("El caracter " + c + " ha sido depositado en el buffer");
40                 // Mostramos por consola el caracter producido
41                 sleep ((int) (Math.random() * 100)); // Dormimos el hilo el tiempo indicado
42                 // en milisegundos
43             } catch (InterruptedException ex) {
44                 Logger.getLogger(Productor.class.getName()).log(Level.SEVERE, null, ex);
45             }
46         }
47     }
48 }

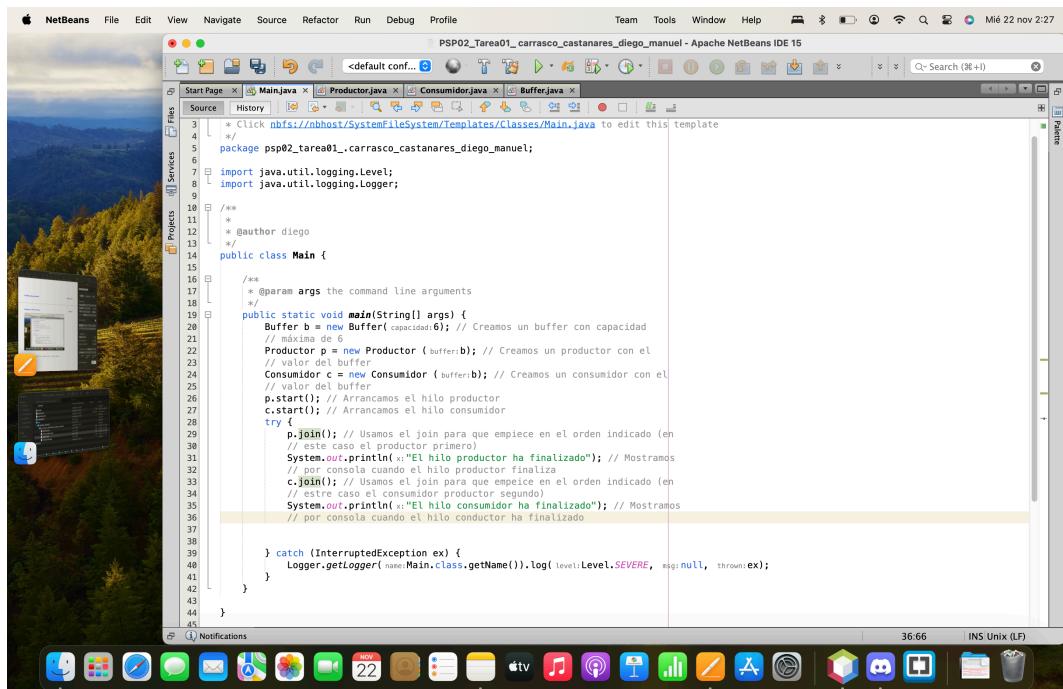
```

Para finalizar crearemos el código de la clase Main.

En ella crearemos un buffer con una capacidad máxima de 6 y un productor y un consumidor con el valor del buffer.

Posteriormente arrancaremos el hilo productor y el hilo consumidor mediante el método start.

Por último, mediante el método join indicaremos que primero se ejecute el hilo productor y posteriormente el hilo consumidor y mostraremos por consola cuando acaben cada uno de los hilos.



```

3  /*
4   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
5   */
6   package psp02_tarea01_.carrasco_castañares_diego_manuel;
7
8   import java.util.logging.Level;
9   import java.util.logging.Logger;
10
11 /**
12  * @author diego
13 */
14
15 public class Main {
16
17     /**
18      * @param args the command line arguments
19     */
20     public static void main(String[] args) {
21         Buffer b = new Buffer(6); // Creamos un buffer con capacidad
22         // de 6 elementos
23         Productor p = new Productor (buffer:b); // Creamos un productor con el
24         // valor del buffer
25         Consumidor c = new Consumidor (buffer:b); // Creamos un consumidor con el
26         // valor del buffer
27         p.start(); // Arrancamos el hilo productor
28         c.start(); // Arrancamos el hilo consumidor
29         try {
30             p.join(); // Usamos el join para que empiece en el orden indicado (en
31             // este caso el productor primero)
32             System.out.println("El hilo productor ha finalizado"); // Mostramos
33             // por consola cuando el hilo productor finaliza
34             c.join(); // Usamos el join para que empiece en el orden indicado (en
35             // este caso el consumidor productor segundo)
36             System.out.println("El hilo consumidor ha finalizado"); // Mostramos
37             // por consola cuando el hilo conductor ha finalizado
38
39         } catch (InterruptedException ex) {
40             Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
41         }
42     }
43
44 }

```

Enunciado

De igual manera a lo visto en el tema, ahora te proponemos un ejercicio del tipo productor-consumidor que mediante un hilo productor de datos (15 caracteres) en un búfer compartido, de donde los debe recoger un hilo consumidor (consume 15 caracteres). La capacidad del búfer es de 6 caracteres, de manera que el consumidor podrá estar cogiendo caracteres del búfer siempre que éste no esté vacío. El productor sólo podrá poner caracteres en el búfer, cuando esté vacío o tenga espacio.

Para la simulación te proponemos que el productor espere un poco `sleep((int) (100 * Math.random()));` y el consumidor algo más de tiempo `sleep((int) (Math.random() * 1000));`

Te mostramos una posible salida del programa que debes realizar

```
: Salida - TareaProductorBufferConsumidor (run)
run:
Depositado el carácter T en el buffer
Recogido el carácter T del buffer
Depositado el carácter R en el buffer
Recogido el carácter R del buffer
Depositado el carácter J en el buffer
Depositado el carácter W en el buffer
Depositado el carácter L en el buffer
Depositado el carácter Y en el buffer
Depositado el carácter M en el buffer
Depositado el carácter P en el buffer
Recogido el carácter P del buffer
Depositado el carácter J en el buffer
Recogido el carácter J del buffer
Depositado el carácter Q en el buffer
Recogido el carácter Q del buffer
Depositado el carácter X en el buffer
Depositado el carácter H en el buffer
Recogido el carácter X del buffer
Recogido el carácter H del buffer
Depositado el carácter G en el buffer
Recogido el carácter G del buffer
Depositado el carácter O en el buffer
Recogido el carácter O del buffer
Depositado el carácter U en el buffer
Recogido el carácter U del buffer
Recogido el carácter M del buffer
Recogido el carácter Y del buffer
Recogido el carácter I del buffer
```

Observa:

- *Se comienza depositando.
- *Se pueden depositar seguidos hasta 6 caracteres.
- *Cuando el búfer está lleno, la única opción es consumir.

Criterios de puntuación.

- Productor: 2,5 puntos
- Consumidor: 2,5 puntos
- Buffer: 5 puntos

Total: 10 puntos

Recursos mínimos necesarios para realizar la Tarea.

- Ordenador personal
- Sistema operativo Windows o Linux
- Conexión a Internet a través de ADSL
- JDK y JRE de Java
- Entorno NetBeans IDE 8.2.

Consejos y recomendaciones.

Tomar como referencia los ejercicios resueltos de los contenidos de la unidad de trabajo.

Indicaciones de entrega.

Debes crear un único fichero comprimido en el que debes incluir el o los proyectos Netbeans y documentos que correspondan a la realización de los ejercicios de la tarea.

Además adjuntarás un documento realizado con un procesador de texto con unas breves explicaciones de cómo has realizado los ejercicios, con las capturas de pantallas que estimes oportunas.

El fichero comprimido debe seguir la nomenclatura **PSPXX_TareaZZ_apellido1_apellido2_nombre**, donde XX se corresponde con la unidad y ZZ con el numero de la tarea.

Realiza el envío de dicho archivo comprimido a través de la plataforma. Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños. Así por ejemplo la alumna **Begoña Sánchez Mañas para la primera unidad del MP de PSP**, debería nombrar esta tarea como...

PSP02_TAREA021_sanchez_manas_begona