PAC DESARROLLO UF2

CFGS Desarrollo de Aplicaciones Multiplataforma

Módulo 09: Programación de servicios y procesos

INFORMACIÓN IMPORTANTE

Para la correcta realización de la PAC el alumno deberá consultar los contenidos recogidos en la UF3 del material didáctico.

Requisitos que deben cumplirse en vuestros trabajos:

• Siempre que utilicéis información de Internet para responder / resolver alguna pregunta, tenéis que citar la fuente (la página web) de dónde habéis sacado aquella información.

• La PAC debe entregarse en formato ZIP. Este ZIP, contendrá el proyecto realizado en Java. Contendrá la carpeta src con la estructura de ficheros requerida.

• En el caso de no realizarse la entrega en dicho formato el alumno se hace responsable de posibles incompatibilidades en la visualización de su entrega y por ende afectará a su calificación.

CRITERIOS DE CORRECCIÓN

1. Todos los programas realizados en la PAC deben realizarse con IDE con que se pueda trabajar con el lenguaje Java.

2. Para la realización de esta PAC es necesario que se utilicen las estructuras de control y las estructuras repetitivas siempre que sea posible.

3. Se deben poner comentarios para su mejor comprensión. Estos comentarios explicarán la funcionalidad del código. Se valorarán los comentarios en la parte de presentación.

4. El código pasará unas pruebas unitarias para comprobar su correcto funcionamiento. Es por ello por lo que es indispensable que se obedezcan todas las indicaciones de estructura y nomenclatura de clases y atributos. Si no se siguen dichos criterios, se dará por suspensa la PAC.

5. Revisión antiplagio: Como cualquier otro centro educativo realizaremos las revisiones antiplagio oportunas de vuestro trabajo para evitar que se puedan corregir ese tipo de trabajos. Se van a realizar revisiones de plagio nivel de código. Si se detectase plagio a la hora de realizar esta comprobación, la PAC se calificará con un 0.1.

En esta PAC se valorará vuestro conocimiento sobre los sockets de conexión entre diferentes aplicaciones. Para ello, vamos a proponer un pequeño proyecto que incluso os pueda ser de utilidad. Vamos a crear un cliente y un servidor que se comuniquen entre ellos a través de un socket para generar contraseñas de manera aleatoria con una serie de requisitos introducidos por el cliente.

¿Cuál va a ser la estructura (obligatoria) del fichero que deberás subir en esta PAC?

- Proyecto comprimido en ZIP
    - Carpeta: src
        - Carpeta: servidor (package)
            - MainServidor
            - Servidor
            - RequisitosPass
            - ServicioPass
        - Carpeta: cliente (package)
            - MainCliente
            - Cliente

Recuerda que el nombre de los ficheros y packages han de ser los mismos que se indican en este enunciado.

Además, los datos de conexión para establecer el socket deben ser los siguientes:

- Dirección: localhost
- Puerto: 4321

Flujo de la comunicación

La idea principal de este proyecto es que tengamos un servidor que se encargue de atender peticiones de clientes a la hora de generar una contraseña aleatoria. La contraseña se generará a partir de unas indicaciones que le dará el cliente, que corresponden al número de caracteres de distinta clase (mayúsculas, minúsculas, dígitos y caracteres especiales). Estos números los requerirá el servidor, el cual al final del proceso mostrará la longitud de la contraseña que se generaría y preguntará al cliente si desea generarla.

A continuación se muestra la tabla con estas interacciones en detalle. Además, en el último apartado se encuentran distintos ejemplos de esta interacción.

SERVIDOR	CLIENTE
Inicia Servidor	
	Cliente se conecta
Servidor pregunta el nombre del cliente	
	Cliente envía su nombre
Servidor da la bienvenida al cliente haciendo referencia a su nombre	
Servidor introduce al cliente a la funcionalidad	

Servidor pregunta el número de mayúsculas de la contraseña	
	Cliente envía el número de mayúsculas de la contraseña
Servidor pregunta el número de minúsculas de la contraseña	
	Cliente envía el número de minúsculas de la contraseña
Servidor pregunta el número de dígitos de la contraseña	
	Cliente envía el número de dígitos de la contraseña
Servidor pregunta el número de caracteres especiales de la contraseña	
	Cliente envía el número de caracteres especiales de la contraseña
Servidor envía la longitud de la contraseña que se va a generar	
Servidor pregunta si se quiere generar una contraseña ahora	
Opción 1	
	Cliente responde “sí”
Servidor envía una posible contraseña con los requisitos solicitados por el cliente	
	Cliente recibe la posible contraseña
	Cliente cierra la conexión al servidor
Servidor queda a la espera de otra conexión	
Opción 2	
	Cliente responde “no”
Servidor indica que no se generará contraseña y se despide	
	Cliente cierra la conexión al servidor
Servidor queda a la espera de otra conexión	

Aspectos relativos a la Programación Orientada a Objetos

Vamos además a utilizar varios principios de la programación orientada a objetos. Uno de los más importantes es el Principio de Responsabilidad Única. A grandes rasgos, lo que quiere decir es que la funcionalidad contenida en nuestras clases debe estar bien acotada y no tener responsabilidades que puedan formar parte de otras. Esto nos ayuda a tener un código que sea legible y pueda ser mantenido con facilidad.

Para ello, en lugar de escribir toda la funcionalidad dentro de las clases Cliente y Servidor, para el servidor vamos a tener 2 clases más.

1. Vamos a utilizar una clase simple en java (en inglés conocida como POJO - Plain Old Java Object) para recoger los requerimientos acerca de la contraseña que el cliente quiere generar.
   Esta clase se llamará **RequisitosPass**. Tendrá como atributos el número de los distintos

tipos de caracteres que tendrá nuestra clase, es decir, tendrá 4 atributos. DEBEN estar nombrados de la siguiente forma:

- numMayusculas
- numMinusculas
- numDigitos
- numCaractEspeciales

Como es una clase simple de java, ésta no tendrá nada de lógica dentro de ella. Solo tendrá sus getters, setters y toString que conocemos.

2. Para contener la lógica usaremos la siguiente clase. Se llamará **ServicioPass**, y es la que contendrá los principales métodos públicos que usaremos a la hora de generar la contraseña. Tendrá 2 métodos obligatorios: uno para generar la contraseña, y otro para dar la longitud que tendrá la contraseña. Es OBLIGATORIO que estos métodos tengan los nombres que se indicarán en el esquema del punto siguiente. Además, en esta clase (y en las clases Cliente y Servidor) podréis definir cualquier otro método privado que os pueda servir a la hora de realizar las funciones de la clase y que haga que el código sea más legible y pueda mantenerse con facilidad.

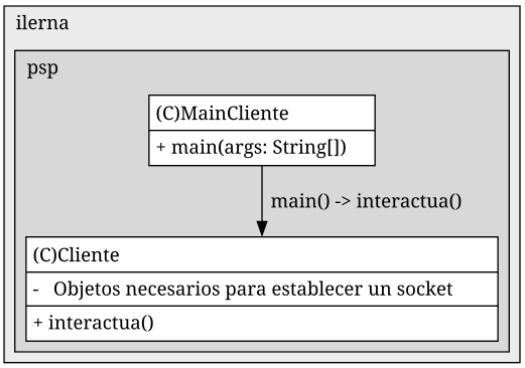
3. Como existen infinidad de "caracteres especiales" que podéis considerar para incluir en una contraseña, nos limitaremos aquí a un conjunto de los más comunes (y de los que más se incluyen en los requerimientos de este tipo de servicios). Son los siguientes

!@#$%^&*()_-+=.:?

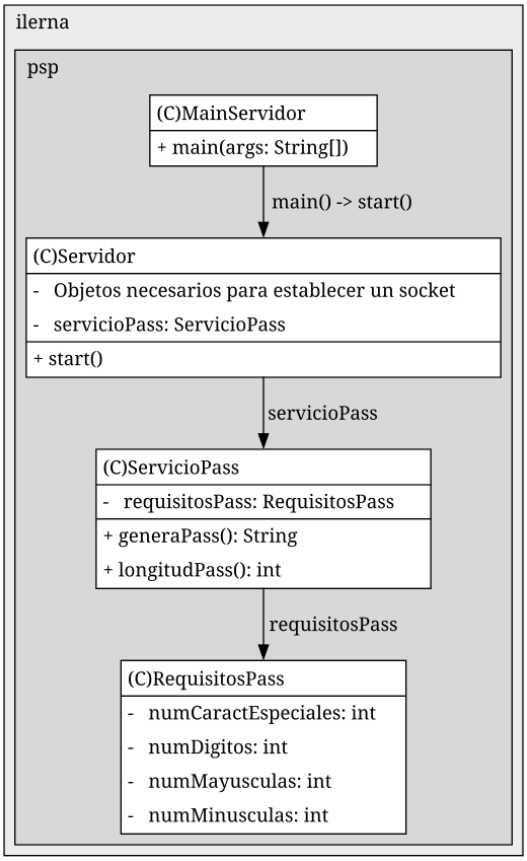
Diagrama

De forma global, el proyecto seguirá el siguiente diagrama respecto a paquetes y clases:

Paquete *cliente*



Paquete *servidor*



Puntuación adicional

Además de los requisitos básicos descritos anteriormente, se otorgará puntuación adicional si se verifican los siguientes puntos:

1. La contraseña no tiene "bloques de requerimientos". Es decir, la contraseña generada tiene los distintos tipos de caracteres de manera aleatoria y no es del tipo ABCdefg12(.* (Se ve aquí que se muestran en orden mayúsculas, minúsculas, dígitos y caracteres especiales).
   Generar una contraseña en la que aparezcan estos tipos de bloques la hace más vulnerable a ataques por fuerza bruta.
   Si se genera una contraseña que sea aleatoria se tendrá puntuación adicional.
2. Comprobación de parámetros inválidos. Como podría ser de esperar, no es posible generar una contraseña que tenga “‘-5” minúsculas o tenga “t” dígitos. A la hora de recibir la información por parte del cliente, si se identifican parámetros incorrectos y se

muestra un error por pantalla al cliente y rompe la conexión. Después de esto, el servidor debería poder seguir aceptando clientes.

Ejemplos de interacciones servidor-cliente

Ejemplo 1

Servidor

```
Servidor arrancado
Esperando al cliente...
Cliente conectado desde /127.0.0.1
Nombre del cliente: Juan
Los requisitos del cliente son los siguientes
PasswordReqs{minusculas=4, mayusculas=3, digitos=2, caracteresEspeciales=1}
Se ha enviado la longitud de la contraseña al cliente
Se ha enviado la contraseña al cliente
```

Cliente

```
Hola, soy un servidor. ¿Cómo te llamas?
Juan
Te doy la bienvenida, Juan
Voy a solicitarte distintos requisitos para la contraseña que voy a generar.
Cuántas minusculas debe tener la contraseña?
4
Cuántas mayusculas debe tener la contraseña?
3
Cuántas digitos debe tener la contraseña?
2
Cuántas caracteres especiales debe tener la contraseña?
1
La longitud de la contraseña que se va a generar es de 10 caracteres.
¿Quieres generar una contraseña ahora? [si/no]
si
La contraseña generada es: Na6RPr)Ou9
```

Ejemplo 2

Servidor

```
Servidor arrancado
Esperando al cliente...
Cliente conectado desde /127.0.0.1
Nombre del cliente: María
Los requisitos del cliente son los siguientes
PasswordReqs{minusculas=5, mayusculas=2, digitos=3, caracteresEspeciales=1}
Se ha enviado la longitud de la contraseña al cliente
Se ha enviado la contraseña al cliente
```

Cliente

```
Hola, soy un servidor. ¿Cómo te llamas?
María
Te doy la bienvenida, María
Voy a solicitarte distintos requisitos para la contraseña que voy a generar.
Cuántas minusculas debe tener la contraseña?
5
Cuántas mayusculas debe tener la contraseña?
2
Cuántas digitos debe tener la contraseña?
3
Cuántas caracteres especiales debe tener la contraseña?
1
La longitud de la contraseña que se va a generar es de 11 caracteres.
¿Quieres generar una contraseña ahora? [si/no]
Sí
La contraseña generada es: 9rA5SO0pL.D
```

Ejemplo 3

Servidor

```
Servidor arrancado
Esperando al cliente...
Cliente conectado desde /127.0.0.1
Nombre del cliente: Alex
Los requisitos del cliente son los siguientes
PasswordReqs{minusculas=4, mayusculas=2, digitos=3, caracteresEspeciales=0}
Se ha enviado la longitud de la contraseña al cliente
El cliente no desea generar una contraseña
```

Cliente

```
Hola, soy un servidor. ¿Cómo te llamas?
Alex
Te doy la bienvenida, Alex
Voy a solicitarte distintos requisitos para la contraseña que voy a generar.
Cuántas minusculas debe tener la contraseña?
4
Cuántas mayusculas debe tener la contraseña?
2
Cuántas digitos debe tener la contraseña?
3
Cuántas caracteres especiales debe tener la contraseña?
0
La longitud de la contraseña que se va a generar es de 9 caracteres.
¿Quieres generar una contraseña ahora? [si/no]
no
No se generará ninguna contraseña. Hasta la próxima.
```