Pruebas acceso

Departamento I+D+I
Fecha: 10/06/2021

Índice

Pruebas de acceso.

1 Introducción

El objetivo de estas pruebas de programación es determinar la forma de resolver problemas ante una serie de supuestos que tienen varias soluciones posibles. **Lee todo el documento antes de ponerte a realizar las tareas.**

Puedes hacer las pruebas en el lenguaje de programación en el que te sientas más cómodo, pero preferimos que sea en JAVA.

Los supuestos opcionales no son obligatorios, puedes elegir cualquier de los opcionales o ninguno. Cuanto más desarrolles más oportunidades tendrás de que valoremos tus cualidades y entres a formar parte del equipo de PROCONSI.

Una vez recibas estas pruebas responde por favor a rrhh@proconsi.com de tu interés en realizar las pruebas técnicas o no. En el caso de que decidas no pasar las pruebas técnicas daremos tu candidatura por finalizada.

2 Tareas a realizar

2.1 Supuesto 1

Introducción de 2 fechas por consola con el siguiente formato yyyy/MM/dd me calcule:

- Diferencia de días entre las 2 fechas
- Inicio y fin de año para cada una de las 2 fechas
- Número de días del año de cada una de las 2 fechas
- (Opcional) número de la semana de cada una de las 2 fechas

2.2 Supuesto 2

Introducción de 2 cantidades y un 3 dato que es el redondeo en el caso de que el valor sea mayor o igual a cero todas las operaciones tienen que redondearse a dicho número, por consola. Debe calcular:

- Suma de las 2 cantidades
- Resta de las 2 cantidades
- División de las 2 cantidades

- Multiplicación de las 2 cantidades
- Modulo del primero con el segundo
- Indicar si el primer número es menor, igual o mayor al segundo número.

2.3 Supuesto 3

Con el siguiente texto de ejemplo mostrar por consola:

- Número de caracteres
- Pasarlo a mayúsculas
- Pasarlo a minúsculas
- Número de palabras repetidas separado por espacios e indicar cuáles son esas palabras repetidas. Es decir, si hola y banco se repiten en el texto tienes que mostrar en la consola hola y banco.
- Reemplazar la palabra “Java” por “Avaj” y mostrar el texto por consola
- Concatenar el texto 1000 veces y mostrar por consola el tiempo tardado en realizar dicha concatenación y la longitud final del texto. (No mostréis el texto final por consola ☺)

String texto = “Java es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán, probablemente, a menos que tengan Java instalado y cada día se crean más. Java es rápido, seguro y fiable”

2.4 Supuesto 4

Crear pequeña agenda para tienda de compra/venta de artículos para introducción de clientes. Todos los datos serán pedidos por consola. Tiene que tener un menú con las opciones:

- Añadir cliente
- Borrar cliente
- Editar cliente
- Listar clientes

A la hora de añadir/modificar clientes me tiene que pedir:

- DNI que es único en la agenda no puede haber 2 clientes con el mismo DNI
- Nombre y apellidos

- Tipo de cliente (REGISTRADO, SOCIO). En el caso de ser REGISTRADO me tiene que indicar la cuota máxima de pago en compra de artículos, los socios REGISTRADO no tienen esta limitación y no tiene que mostrarse.
- Fecha de alta formato yyyy/MM/dd HHmmss

A la hora de borrar se borrará indicando el DNI del cliente.

A la hora de listar se podrá realizar ordenando por DNI o por fecha de alta del cliente

2.5 Supuesto 5

Crear programa que me pida por consola el número de círculos, triángulos y cuadrados a generar y los genere con los datos de las propiedades de forma aleatoria. Debe mostrar por consola todas formas geométricas juntas y luego separadas por cada grupo (círculos, cuadrados, triángulos) mostrando el área de cada forma geométrica y los valores de sus propiedades.

2.6 Supuesto 6 (opcional)

ItemSeparator es una clase que filtra una determinada cadena de entrada sin procesar del formato

"ItemName $$ ## ItemPrice $$ ## ItemQuantity" y lo almacena como estos atributos de clase:

name (String) ,. price (Double) ,. quantity (Integer) implementa el constructor ItemsSeparator(String

rawInput) y los getters para los 3.

Clase main para realizar la prueba:

String stdIn = “Bread$$##12.5$$##10”; // ItemName$$##ItemPrice$$##ItemQuantity

Ejemplo de salida por consola:

Item Name: Bread

Item Price: 12.5 Item

Quantity: 10

2.7 Supuesto 7 (opcional)

Dado un número indicar si es número válido Kaprekar. Dependiendo el modo de planteamiento de cómo resolverlo indicar el número de operaciones realizadas para obtener el resultado final.

2.1 Supuesto 8 (opcional)

Se trata de desarrollar una pequeña aplicación que constará de dos partes, la interface de usuario y el procesador de comandos que se integrarán en el mismo módulo ejecutable o en ejecutables separados y que se comunicarán entre sí a través de ficheros.

Los ficheros de interface pueden ser de tipo txt, xml o bbdd y se describen a continuación.

Comandos

Registrará el tipo, la fecha y la hora de cada comando que se dispare desde la interface y si ha sido procesado, o no, por el procesador de comandos.

Estado

Registrará el estado que determine el procesador de comandos en respuesta a los comandos locales

Log en pantalla

Registrará el tipo, la fecha y la hora de cada lanzamiento de comando por la interface de usuario, así como la resolución del mismo por el procesador de comandos y los cambios de estado o errores que dicha resolución genere.

La lista de comandos es libre (mín. 5) y los estados que produzcan los comandos también, pero deberá existir algún estado en el que sólo algunos de los comandos sean aceptados y los otros han de generar un error en el log. Aunque no es imprescindible se valorará que la lista de comandos y sus compatibilidades resida en un fichero de configuración, para facilitar su modificación (al menos tiene que ser sencilla la manipulación y mantenimiento en el código fuente).

Deberá documentarse la lista de comandos, los estados que producen y las incompatibilidades.

Ejemplo estados: *Parado (estado inicial), Arrancado, Despegando, Volando, Aterrizando*

Ejemplos de comandos: *Arrancar, Parar, Despegar, Volar, Aterrizar, Terminar*

Ejemplos de registros del log:

1. *Solicitado arrancar*
2. *Procesado arrancar*
3. *Estado arrancado*
4. *Solicitado volar*
5. *Procesado volar*

6. *Error sólo posible volar desde despegando*
7. *Solicitado despegar*
8. *Procesado despegar*
9. *Estado despegando*
10. *Solicitado volar*
11. *Procesado volar*
12. *Estado volando*
13. *Solicitado aterrizar*
14. *Procesado aterrizar*
15. *Estado aterrizando*
16. *Solicitado parar*
17. *Procesado parar*
18. *Estado parado*
19. *Solicitado terminar*
20. *Procesado terminar*

El proceso se podría definir como sigue:

Inicio de aplicación

1. Iniciar Interface de Usuario
   - ✓ Leer el último estado y presentarlo
   - ✓ Leer registro log y presentar las últimas operaciones siempre que sean posteriores al inicio de la aplicación.
   - ✓ Solicitar al usuario el comando a ejecutar
   - ✓ ***Registrar comando solicitado***
2. Iniciar procesador de comandos
   - ✓ Comprobar si hay comandos pendientes de procesar
     - ✓ Comprobar estado previo compatible con comando a procesar
       - ✓ Registrar comando procesado
       - ✓ Cambiar estado
       - ✓ Registrar cambio de estado en log
       - ✓ Si el comando es terminar
         - ✓ Cerrar el procesador de comandos
         - ✓ Cerrar la interface de usuario
         - ✓ **Terminar Aplicación**

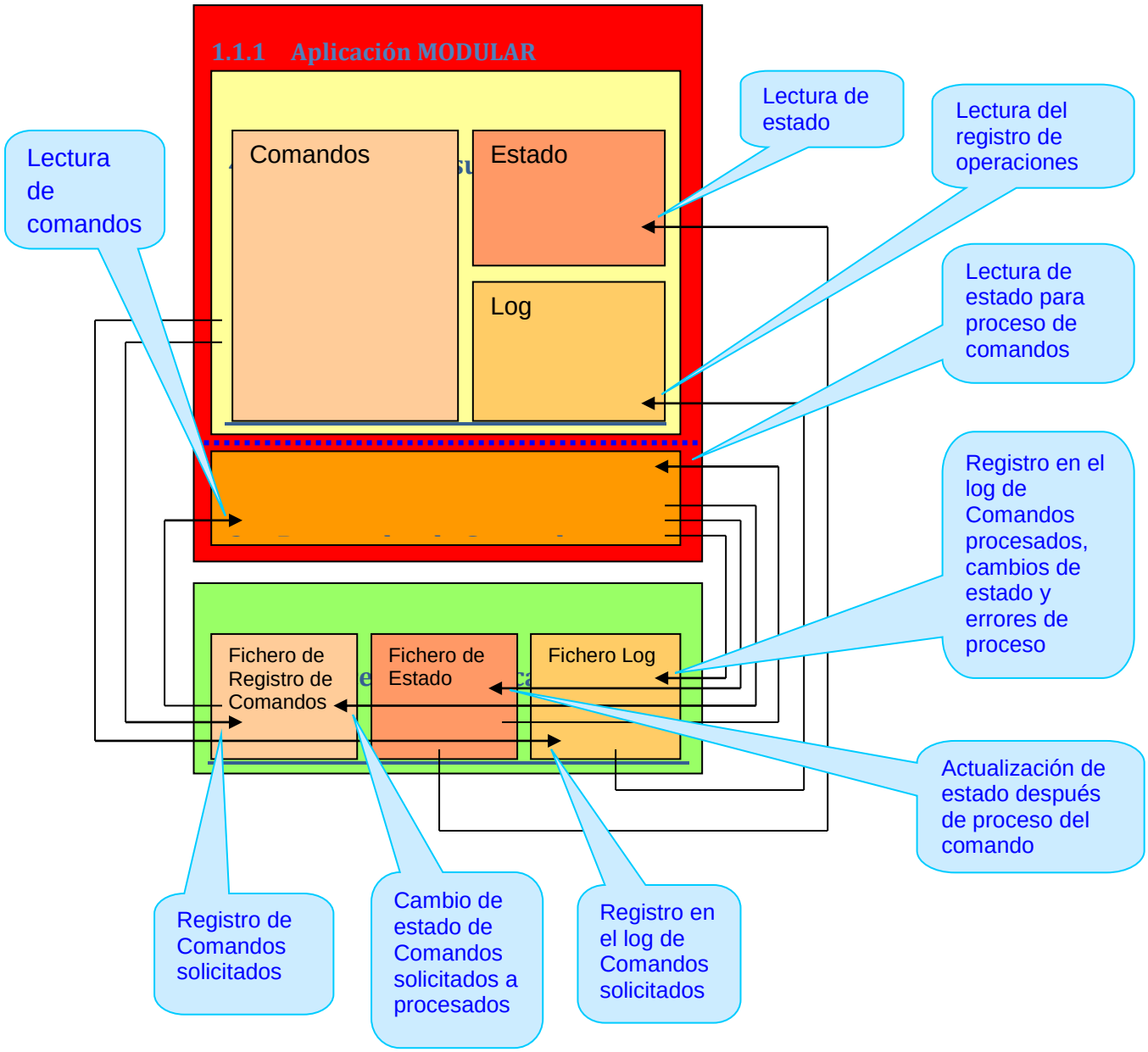
Se puede usar la herramienta de desarrollo que se desee y se valorará un breve razonamiento de su elección.

Opcionalmente, el módulo ejecutable se configurará mediante un fichero de texto en el que se describirá el nombre y la ubicación de los ficheros de interface de comunicación.

Si se dispone de esta configuración, la aplicación se ejecutará en pares de forma que la primera procese los comandos de la segunda y viceversa para comprobar la autonomía de los módulos interface de usuario y procesador de comandos

Se admitirán variaciones en el diseño de esta aplicación siempre que se documente el motivo y sus características.

Se valorará disponer de un sistema de log a consola y fichero que permita mediante el ajuste de los niveles de los loggers en su fichero de configuración hacer una depuración del programa para la búsqueda de anomalías o la validación del correcto funcionamiento. Se valorará la elección de la librería de log a emplear.


1.1.1 Aplicación MODULAR
Comandos
Estado
Log
Fichero de Registro de Comandos
Fichero de Estado
Fichero Log
Lectura de comandos
Lectura de estado
Lectura del registro de operaciones
Lectura de estado para proceso de comandos
Registro en el log de Comandos procesados, cambios de estado y errores de proceso
Actualización de estado después de proceso del comando
Registro de Comandos solicitados
Cambio de estado de Comandos solicitados a procesados
Registro en el log de Comandos solicitados

3 Aspectos a tener en cuenta

Comenta todo lo que realices en el código. Cualquier trozo de código que realice lógica de difícil compresión sin comentar no será valorado. Comenta todo hasta las dudas que te surjan mientras lo desarrollas.

Esta prueba no es para determinar el nivel técnico de un programador/a es una prueba para determinar la forma de afrontar los problemas y el modo de resolver problemas o plantear dudas, porque no todo en la programación es resolver problemas a veces os tocara plantear dudas para que otras personas sean quienes tengan que resolverlas.

No os frustréis si no os sale algo, puesto que no es una prueba para determinar nivel técnico. Está enfocada a personas que se están iniciando en el mundo de la programación.

No dudéis en preguntar cualquier duda y esperamos que os sea entretenido.

Dirigir preguntas a rrhh@proconsi.com