|  |  |  |  |
| --- | --- | --- | --- |
| Experimento /  Familia de problemas | a | b | c |
| **mtp** (art. JSS)  Sample of 38 first-year students, organized randomly in pairs, approached three different activities about recursivity consisting respectively in calculating whether a sentence is a palindrome, implementing the Euclidean algorithm or implementing the Fibonacci function, to which we refer as AP, AE and AF. In the first and second problem we took into account 19 pairs, but in the third activity only 17 pairs were considered because two groups did not produce any solution. In these activities the time available was 40 minutes | **1 Palíndromo**  mtpg?a | **2 Fibonacci**  mtpg?b | **3 Euclides**  mtpg?c |
| **ed**, **iu** (art. JSS)  51 five-year students were organized in 17 pairs (34 pair programmers) and 17 solo programmers -estos no están en los datos-. It is worth noting that all the students participating had a similar level of knowledge and skills as they were all in the same academic year, had at least one year’s experience with the Java programming language and received a training course in COLLECE. The students were randomly arranged in groups and designated as distributed pair programmers or solo programmers, irrespective of their age, sex, knowledge, etc. | **4 Ordenar 10 números, media y mediana**  ed?  iua? | **5 Primos entre n y m**  iub? |  |
| **sc** (art. IJHCS)  The students had to practice programming of synchronous collaborative systems with JSDT.  The teachers arranged the students randomly in groups of two people, each one working from a different place.  Number of students, respectively: 10, 10 and 9 | **6 Programación con JSDT 1**  sca? | **7 Programación con JSDT 2**  scb? | **8 Programación con JSDT 3**  scc? |
| **isl** (art. JSS)  We prepared a collaborative programming activity using COLLECE with employees of this software factory (Indra Software Labs). We first met the general manager and two team managers working for the public administration sector were then selected. 15 programmers from their teams with homogeneous skills in Java programming were recruited for participation in the study. | **9 Procesar fichero XML de discos**  isl?? |  |  |

**1 Palíndromo**

Diseñar un programa que informe si una palabra o frase es un palíndromo usando un método recursivo.

Una palabra o frase es un palíndromo si se lee igual hacia adelante que hacia atrás.

**2 Fibonacci**

Construir un programa que obtenga el término n de la serie de Fibonacci.

La serie de Fibonacci es una secuencia de enteros, cada uno de los cuales es la suma de los dos anteriores. Los dos primeros números de la secuencia son 0 y 1 para n=0 y n=1 respectivamente.

Para todo n>1, la serie se define como:

Fibonacci(n) = Fibonacci(n-1) + Fibonacci(n-2)

Úsese un método recursivo para calcular dicho término.

**3 Euclides**

Sea el siguiente algoritmo propuesto por Euclides para determinar el máximo común divisor de dos enteros, n y m, tal que n<m:

a) Tómese el resto del cociente m/n

b) Si el resto es cero, entonces n es el máximo común divisor

c) Si el resto es distinto de cero se hace m=n y n=resto

d) Se vuelve al punto a)

Escriba un programa recursivo que acepte dos números enteros leídos por teclado y determine su máximo común divisor aplicando el algoritmo de Euclides. Escriba un método que aplique el algoritmo.

**4 Ordenar 10 números, media y mediana**

Realizar un programa que ordene 10 numeros introducidos por teclado. Debe mostrarlos por pantalla. Tambien debe calcular la media y la mediana.

**5 Primos entre n y m**

Haz un programa en Java que calcule y muestre por pantalla los numeros primos que se encuentran entre dos numeros, n y m, leidos por teclado.

**6 Programación con JSDT 1: CS1. Practice with JSDT sessions, channels and clients**

ESPECIFICACIÓN DE REQUISITOS:

Haciendo uso de JSDT, crear la siguiente estructura de sesiones, canales y clientes:

+----+----+ +----+----+ +----+----+

| C1 | C2 | | C1 | C3 | | C2 | C3 |

+-+----+----+--+ +--+----+----+--+ +--+----+----+--+

| Canal 1 | | Canal 2 | | Canal 3 |

+-+--------------+--+---------------+--+---------------+-+

| Sesion 1 |

+--------------------------------------------------------+

+----+----+----+----+ +----+----+

| C1 | C2 | C4 | C5 | | C4 | C5 |

+-+----+----+----+----+-+ +--+----+----+--+

| Canal 4 | | Canal 5 |

+-+-----------------------+------+---------------+-------+

| Sesion 2 |

+--------------------------------------------------------+

Es decir, se trata de crear dos sesiones. En la primera se crearán tres canales, y en la segunda otros dos. Hay también que crear cinco clientes (C1-C5). Estos clientes se asocian a los canales como indica la figura. Todos los clientes pueden enviar y recibir datos por los canales a los que están vinculados.

Una vez creada esta estructura, cada cliente enviará un número aleatorio por “sus” canales. Los clientes que reciban estos números deben visualizar la suma de ellos y la media según vayan recibiendo números.

El registro debe quedar detenido y las sesiones cerradas al finalizar la ejecución del programa. Los clientes deberán abandonar las sesiones y los canales.

RESTRICCIONES:

-Lanzar el registro en el mismo programa (clase) que cree dicha estructura.

-Crear el servidor y los clientes en la misma máquina (localhost).

-Un cliente no se enviará a sí mismo ningún mensaje.

-No ubicar las clases creadas en ningún paquete.

**7 Programación con JSDT 2: Create a menu with options for managing sessions, channels and clients, and for sending messages**

ESPECIFICACIÓN DE REQUISITOS:

Haciendo uso de JSDT, crear un programa en Java que muestre un menú con las siguientes opciones:

(1) Inicio/Parada del registro de JSDT.

(2) Creación de un cliente, una sesión y un canal, suministrando sus nombres y la dirección y puerto de la sesión. El cliente se añadirá automáticamente a la sesión y al canal (el canal se asociará a la sesión indicada). Cada cliente será un objeto que desempeñará el doble rol enviar-recibir datos.

(3) Listar las sesiones activas.

(4) Salir.

SUGERENCIAS:

- Utilizar las especiales características de 'createSession' y 'createChannel', que permiten crear la abstracción la primera vez o devolver su identificador si ya existe, para obtener una referencia a la abstracción correspondiente.

- Utilizar el método 'list' de 'RegistryFactory' para obtener la lista de sesiones (URLString).

RESTRICCIONES:

-Crear el servidor y los clientes en la misma máquina (localhost).

-Un cliente no se enviará a sí mismo ningún mensaje.

-No ubicar las clases creadas en ningún paquete.

-No utilizar Eclipse ni ningún otro IDE.

**8 Programación con JSDT 3: Implement a client/server simple calculator using JSDT messages**

ESPECIFICACIÓN DE REQUISITOS:

Con el objetivo de seguir practicando la creación de sesiones y clientes, el envío y recepción de mensajes a través de canales y, en este caso, la manipulación de diferentes tipos de mensajes, crear un programa en Java que realice lo siguiente:

(a) Inicialmente se lanzará el registro de JSDT.

(b) Crear una sesión (ses) y dos canales (ch1 y ch2) asociados a ella.

(c) Crear tres clientes (cl1, cl2 y cl3) con el siguiente comportamiento:

(c.1) El cliente cl1 se encargará de enviar al cliente cl2, a través de ch1, un mensaje indicando una operación aritmética a realizar: suma, inverso o máximo de tres números. Para cada operación se utilizará un tipo (clase) de mensaje diferente (de hecho, cada operación requiere diferente número de operandos).

(c.2) Así, el cliente cl2 recibirá operaciones aritméticas por ch1, calculará su resultado y lo enviará a cl3 a través de ch2.

(c.3) El tercer cliente, cl3, mostrará el resultado de las operaciones por pantalla.

(d) Se deberá desarrollar un ejemplo de realización de tres operaciones de acuerdo al esquema planteado. No hará falta por tanto leer ningún dato por teclado, sino que el ejemplo de cada operación se codificará en el programa principal.

(e) Finalmente, el registro se detendrá.

RESTRICCIONES:

-Crear todas las clases necesarias en un único fichero Java.

-No ubicar las clases creadas en ningún paquete.

-Crear el servidor y los clientes en la misma máquina (localhost).

-Utilizar el puerto 12000.

-Un cliente no se enviará a sí mismo ningún mensaje.

-No utilizar más de una sesión y de dos canales.

-No utilizar Eclipse ni ningún otro IDE.

**9 Procesar fichero XML de discos**

**1. Introducción**

SAX (Simple Api for XML) es una librería en Java que permite analizar documentos XML. SAX incluye la clase DefaultHandler, cuyos métodos deben sobreescribirse por parte del programador para manejar los eventos lanzados por el analizador a medida que encuentra los diferentes tokens del lenguaje XML. A continuación se describe el funcionamiento de los principales métodos de la clase DefaultHandler:

public void startDocument (): Invocado al inicio del documento.

public void endDocument (): Invocado al finalizar el procesamiento del documento.

public void startElement (String uri, String localName, String qName, Attributes attributes): Invocado al encontrar una etiqueta. Name contiene el nombre del elemento. Si la etiqueta contiene atributos, éstos se encuentran en attributes.

public void endElement (String name): Invocado al encontrar la etiqueta de cierre de un elemento. El argumento name contiene el nombre del elemento.

public void characters (char ch[], int start, int length): Invocado cuando se encuentra texto entre las etiquetas de inicio y cierre de un elemento. El vector ch[] contiene el texto entre las etiquetas de inicio y cierre de elemento,start indica en qué posición de ch[] comienza a almacenarse el texto y length indica el número de caracteres almacenados en ch[].

**2. Enunciado**

**Objetivo y requisitos:**

a) Haciendo uso de la librería SAX antes descrita, implementar un programa que recorra los archivos cd\_catalog1.xml y cd\_catalog2.xml que se encuentran en C:\CUBICO, seleccione de ambos los discos cuyo año de publicación sea posterior a 1990 y muestre por pantalla información formateada de aquellos discos cuyo precio sea superior a 9 $ de forma ordenada por el precio.

b) De acuerdo a un patrón de diseño arquitectónico de tres capas (presentación, dominio y almacenamiento), situar en una clase Almacenamiento la lógica correspondiente a la lectura y manipulación de los ficheros XML, en una clase Presentación la salida de información por consola y en una tercera clase la lógica principal del procesamiento descrito. Todas las clases deben incluirse en un único fichero .java que se almacenará en la carpeta local cubico (C:\CUBICO).

**Restricciones:**

a) Debe utilizarse una lista (clase Vector, ArrayList, etc.) para almacenar los discos procedentes de ambos ficheros XML.

b) No podrá haber más de cuatro clases.

**Ayuda:**

a) Puede consultarse la API de Java: http://docs.oracle.com/javase/7/docs/api/

b) Se proporciona al acceder al sistema un esqueleto de procesamiento del XML indicado cargado en el editor de código que simplifica el trabajo a realizar.