Constructores , sobrecarga y sobreescritura.
Considera el siguiente código fuente JAVA

```
package edu.upc.eetac.dsa.exercises.java.lang;
public class App {
    public static void main(String args[]) {
        Tree[] arboles = new Tree[4];
        arboles[0] = new Tree(4);
        arboles[1] = new Tree("Roble");
        arboles[2] = new Tree();
        arboles[3] = new Tree(5,"Pino");
    }
}
```

completa con el código que haga falta para que la salida sea:

```
Un árbol de 4 metros
Un Roble
Un árbol
Un Pino de 5 metros
```

Interfaces.
Considera el siguiente código fuente Java:

```
package edu.upc.eetac.dsa.exercises.java.lang;
public class AppInheritance {
    public static void main(String[] args) {
        Shape[] shapes = new Shape[2];
        shapes[0] = new Circle(2.5);
        shapes[1] = new Rectangle(3, 5);
        for (Shape shape : shapes)
            System.out.println(shape + " with area = " + shape.area());
    }
}
```

Completa el código para que la salida por consola sea:

```
I'm a circle with area = 0.0
I'm a rectangle with area = 15.0
```

Excepciones, E/S con ficheros de tipo texto.
Implementar una clase con un método de clase (static) que a partir de la lectura de un fichero de texto cuya ruta es pasada como parámetro y en el que en cada línea hay un número entero menor que 1000 devuelva la media aritmética de los números leídos. Además, hay que tener en cuenta las siguientes condiciones:

1. Cualquier excepción de E/S o de formato de número se debe capturar (catch) y relanzar como una excepción “FileParsingException” con el mismo mensaje que arrojaba la excepción capturada.
2. Si algún número es mayor que 1000 se debe lanzar una excepción “BigNumberException” indicando en el mensaje de salida de la excepción el número de línea en el que se encontró el número mayor que 1000.
3. Se debe garantizar que el fichero queda cerrado antes de terminar el método.

A partir de esa clase, implementar un programa al que pasándole como primer (y único) argumento

de entrada la ruta al fichero con los números imprima por consola la media aritmética de los números contenidos en el fichero. Cualquier excepción se mostrará por la salida estándar de error.

Excepciones, E/S con ficheros de tipo binario.
Implementar una clase que tenga tres atributos: uno de tipo entero (int), otro decimal (double) y uno de tipo carácter (char). Sobre esta clase implementar:

1. un método que tomando como parámetro de entrada el nombre de un fichero escriba en él en formato binario el valor de los tres atributos en el orden en el que se han enumerado.
2. Un método de clase que tomando como parámetro de entrada el nombre de un fichero que ha sido generado como salida del método anterior instancie un objeto de la clase y lo devuelva.
3. Cualquier excepción que se produzca en ambos métodos no se tratará.

Implementar un programa que compruebe el correcto funcionamiento de los dos métodos descritos.

Concurrencia. Threads.
Crear dos clases que se ejecuten como threads y cuya tarea sea escribir 10 veces por la salida estándar su nombre junto con el tiempo en milisegundos transcurridos desde la última vez que hicieron la escritura y el número de escrituras que ha realizado. Cada vez que el thread escriba un mensaje se dormirá un tiempo aleatorio entre 0 y 2 segundos. Una de las clases debe heredar de Thread y la otra implementar la interfaz Runnable.

Implementar un programa que arranque dos threads de cada uno de los tipos anteriores. Cuando los dos threads hayan finalizado, este programa escribirá en la salida estándar: “Se acabó”.

Concurrencia. Sincronismo.
Implementar una clase Buffer con un atributo privado vector de caracteres de tamaño 32 (char[]). La clase proporciona un método de lectura que devuelve el primer carácter no leído del vector, dejando la posición que ocupaba el carácter libre para escritura. Para poder realizar la lectura en el vector debe haber al menos un carácter legible. Asimismo, proporciona otro método de escritura que escribe un carácter en la primera posición libre del vector. Para poder realizar la escritura en el vector debe haber al menos una posición libre en el vector.

Por otro lado tendremos dos clases que se ejecutan como thread. La primera es un thread productor, que escribe todos los caracteres de una cadena acabada en '\n' en una instancia de la clase Buffer, y la segunda es un thread consumidor que lee los caracteres que se escriben en el buffer en el mismo orden en el que fueron escritos y cuando el carácter es '\n' escribe por la salida estándar la cadena que ha leído.

Implementar un programa que compruebe el correcto funcionamiento de los requisitos pedidos.

Comunicaciones en red. TCP no concurrente.
Implementar el cliente y el servidor del protocolo “¿Qué hora es?” utilizando TCP en modo no-concurrente. El protocolo funciona de la siguiente manera: el cliente abre una conexión contra el servidor y este le responde con la fecha y hora actual según su reloj en formato “[dia del mes]/[mes]/[año] [hora]:[minutos]:[segundos]\n” todos en formato numérico (por ejemplo, 21/09/2013 17:07:34). Para la realización de este ejercicio utilizar la clase java.text.SimpleDateFormat.

El ejecutable del servidor debe admitir como parámetros el puerto del servicio, y si no se pasa ninguno este debe ser el puerto 12345. El ejecutable del cliente debe admitir como parámetros obligatorios la dirección y el puerto del servidor.

Comunicaciones en red. UDP no concurrente.
Implementar el cliente y el servidor del protocolo “¿Qué hora es?” utilizando UDP en modo no-concurrente. El protocolo funciona de la siguiente manera: el cliente envía un datagrama vacío contra el servidor y este le responde con la fecha y hora actual según su reloj en formato “[dia del mes]/[mes]/[año] [hora]:[minutos]:[segundos]\n” todos en formato numérico (por ejemplo, 21/09/2013 17:07:34). Para la realización de este ejercicio utilizar la clase java.text.SimpleDateFormat.

El ejecutable del servidor debe admitir como parámetros el puerto del servicio, y si no se pasa ninguno este debe ser el puerto 12345. El ejecutable del cliente debe admitir como parámetros obligatorios la dirección y el puerto del servidor.

Comunicaciones en red. Servicios concurrentes.
Diseñar e implementar un servicio de chat sobre protocolo de transporte TCP. Los clientes se unen al chat enviando el mensaje

JOIN [name]\n

siendo [name] el nombre de usuario que escoge el cliente. Los clientes abandonan el chat enviando el mensaje:

LEAVE\n

Para enviar un mensaje el cliente enviará:

MESSAGE [message]\n

Una vez unido un cliente, todos los mensajes que envíe serán reenviados por el servidor a todos los clientes unidos al chat (el cliente que envía el mensaje inclusive) enviando:

[name]> [message]\n

siendo [name] el nombre de usuario del cliente origen del mensaje. Cuando un cliente abandone el chat, todos los demás recibirán el mensaje:

[name]> me piro.

Y cuando se una:

[name]> estoy dentro.

La entrada de mensajes se realizará a través de la entrada estándar. Si el mensaje que entra el usuario tiene longitud cero, esto es, pulsa únicamente la tecla enter, entonces el cliente envía el mensaje para abandonar el chat. Obviar cualquier situación que se aparte del protocolo descrito como, por ejemplo, que dos usuarios se unan con el mismo [name].