

UD05.EXAME Práctico

DAM1-Programación 2024-25

2025/03/26

Administración de rede	2
Clase Host (4)	2
Clase Cliente (4)	3
Clase Servidor (2)	4

- Puedes utilizar apuntes y materiales que consideres pero deberás realizar los programas individualmente. En caso contrario se retirará el examen.
- Realiza programas bien estructurados, legibles, con comentarios, líneas en blanco, identificadores adecuados, etc.
- Cuida la interacción con el usuario, presentando la información de forma clara y ordenada.
- Utiliza los casos de prueba de ejemplo para probar tu programa y definir la salida por pantalla.

- Crea un **paquete** de nombre **ud5.xxxexamen** donde agrupar los ficheros de código fuente, donde **xxx** son las iniciales de tu nombre y apellidos. Si es necesario crea un nuevo proyecto/carpeta Java.
- Crea dentro del paquete ficheros **.java** por cada ejercicio con el nombre apropiado.
- **Incluye al inicio de cada fichero de código fuente un comentario con tu nombre y apellidos.**
- **Entrega la carpeta xxxexamen comprimida con los archivos de código fuente .java.**
- **Tiempo estimado: 2:20 horas**

Administración de rede

Se quiere crear una aplicación que ayude a administrar una red de ordenadores. Para ello se implementará una jerarquía de clases que represente diferentes tipos de equipos en la red.

Clase Host (4)

Host.java

La clase `Host` representa un equipo conectado en la red e incluye la siguiente información:

- **Nombre** del equipo. Una cadena que no podrá ser `null`.
- **Dirección MAC**. Deberá ser una cadena válida que represente 12 dígitos hexadecimales en mayúsculas o minúsculas en uno de los dos formatos siguientes: `hh:hh:hh:hh:hh:hh` o `hh-hh-hh-hh-hh-hh`. Ejemplos: `00:11:22:33:44:55`, `AA-BB-CC-DD-EE-FF`
- **Dirección IP**. Una cadena que representa 4 valores entre 0 y 255 separados por puntos. Ejemplo: `192.168.117.200`

Todos los hosts de la red comparten los siguientes parámetros de red, que tendrán los valores por defecto que se indican entre paréntesis:

- **Máscara de red** (255.255.0.0)
- **Puerta de enlace** (192.168.0.11)
- **Servidores DNS** (192.168.0.9)

Métodos:

- **Constructor** para inicializar los atributos de instancia. Si alguno de los atributos no es válido el constructor deberá generar una excepción `IllegalArgumentException` con el mensaje adecuado.
- **equals()**: Redefine el método para que considere que dos objetos `Host` son iguales si tienen la misma dirección MAC, ignorando la diferencia entre mayúsculas y minúsculas al representar los dígitos hexadecimales e ignorando también el carácter separador utilizado, ya sean guiones (-) o dos puntos (:).
- **toString()**: Devuelve una representación en cadena del host con el siguiente formato:
`nombre (ip / mac).`

Define el **orden natural** de los objetos de la clase `Host` como orden alfabético ascendente de nombre de host.

Completa la clase `Host` de modo que al ejecutar el programa principal genere la siguiente salida:

```
Clase Host
=====

equipo101 (192.168.117.101 / 00:11:22:33:44:55)
host100 (192.168.117.100 / 00:11:22:33:44:55)
host103 (192.168.117.103 / 00-11-22-33-44-55)
nodo100 (192.168.117.100 / 00-11-22-33-44-AA)
host100 = equipo101? true
host100 = nodo100? false
host100 = host103? true
```

Clase Cliente (4)

Cliente.java

La clase `Cliente` extiende la clase `Host` y representa un equipo cliente en la red añadiendo los siguientes atributos:

- **Sistema operativo.** Tipo enumerado (**SO**) con los sistemas operativos utilizados en la red (WINDOWS, LINUX, MAC, ANDROID, IOS).
- **Resolución de pantalla.** Cadena que recoge el ancho y alto en pixeles de la pantalla separados por una **x** (ancho **x** alto)

Métodos:

- **Constructor** para inicializar los atributos de instancia.
- **toString():** Incluye el nombre del cliente y entre paréntesis, el sistema operativo y la resolución.

Completa la clase `Cliente` de modo que al ejecutar el programa principal genere la siguiente salida:

```
Clase Cliente
=====

Cliente1 (Windows 1920x1080)
Cliente2 (Linux 1366x768)
Cliente3 (Mac 2560x1440)
Cliente4 (Android 1080x2340)
Cliente5 (iOS 1170x2532)
```

Amplía el programa principal de modo que muestre los clientes ordenados según se indica y se muestra en la captura:

1. alfabéticamente por sistema operativo.

```
Cientes ordenados por SO
=====

Cliente4 (Android 1080x2340)
Cliente5 (iOS 1170x2532)
Cliente2 (Linux 1366x768)
Cliente3 (Mac 2560x1440)
Cliente1 (Windows 1920x1080)
```

2. de mayor a menor resolución de pantalla, calculando el número de píxeles resultantes de multiplicar el ancho por el alto.

```
Cientes ordenados por Resolución
=====

Cliente3 (Mac 2560x1440)
Cliente5 (iOS 1170x2532)
Cliente4 (Android 1080x2340)
Cliente1 (Windows 1920x1080)
Cliente2 (Linux 1366x768)
```

Clase Servidor (2)

Servidor.java

La clase `Servidor` extiende la clase `Host` y representa un equipo servidor que ofrece servicios en la red añadiendo los siguientes atributos:

- **Lista de servicios** que ofrece. Cada servicio se identifica por un nombre, un número de puerto (entre 1 y 65535) y el protocolo de transporte utilizado (TCP o UDP).
Por ejemplo: Servicio "HTTP" ofrecido en el puerto número 80 TCP.
Un servidor no puede ofrecer dos servicios en el mismo puerto

Métodos

- El **constructor** de `Servidor` reutilizará el de la superclase `Host`.
- **boolean addServicio()**: Añade un nuevo servicio al servidor. Debe comprobar que el servidor no esté ofreciendo ya otro servicio en el mismo puerto y mismo protocolo, en cuyo caso devolverá `false`.
- **toString()**: Incluye el nombre del servidor, su dirección IP y el listado de los servicios y sus respectivos puertos.

Completa la clase `Servidor` de modo que al ejecutar el programa principal genere la siguiente salida:

```
Clase Servidor
=====

dam1 (192.168.0.117)
* MySQL (3306/TCP)

seixo (192.168.0.11)
* HTTP (80/TCP)
* Proxy (3128/TCP)

lapaman (192.168.0.9)
* HTTP (80/TCP)
* DNS (53/UDP)
* FTP (21/TCP)
* SSH (22/TCP)
```

Amplía el programa principal de modo que muestre los servidores ordenados de mayor a menor cantidad de servicios:

```
Servidores ordenados descendentemente por número de servicios
=====

lapaman (192.168.0.9)
* HTTP (80/TCP)
* DNS (53/UDP)
* FTP (21/TCP)
* SSH (22/TCP)

seixo (192.168.0.11)
* HTTP (80/TCP)
* Proxy (3128/TCP)

dam1 (192.168.0.117)
* MySQL (3306/TCP)
```