Basado en los contenidos de **Programación de Servicios y Procesos (PSP)** detallados en el PDF y los problemas técnicos que has mostrado, aquí tienes la resolución de tus ejercicios y una propuesta de modelos que suelen caer en este tipo de exámenes.

Ejercicio 1: Servidor HTTP con cabecera `Date`

Para incluir la cabecera `Date` en la respuesta del servidor, se utiliza el método `getDate()` de `URLConnection` en el lado del cliente para monitorizarla, o se genera manualmente en el servidor.

Código sugerido para el servidor:

Java

```
// Dentro del método que procesa la petición GET
String fechaActual = new java.util.Date().toString();
printWriter.println("HTTP/1.1 200 OK");
printWriter.println("Date: " + fechaActual); // Inserción de la cabecera Date
printWriter.println("Content-Type: text/html");
printWriter.println("\r\n"); // Línea en blanco obligatoria tras cabeceras
```

Ejercicio 2: Servidor HTTP Multihilo (Concurrente)

Para que un servidor gestione la concurrencia de manera eficiente, el hilo principal debe limitarse a aceptar conexiones y delegar el procesamiento a un hilo secundario (`Thread`).

Esquema de código multihilo:

Java

```
public class ServidorConcurrente {
    public static void main(String[] args) {
        int puerto = 8066;
        try (ServerSocket socServidor = new ServerSocket(puerto)) {
            while (true) {
                // Acepta la petición y asigna un socket cliente
```

```
                Socket socketCliente = socServidor.accept();
                // Crea y lanza un nuevo hilo para despachar la
petición
                new HiloDespachador(socketCliente).start();
            }
        } catch (IOException e) { e.printStackTrace(); }
    }
}

class HiloDespachador extends Thread {
    private Socket socketCliente;
    public HiloDespachador(Socket socketCliente) {
        this.socketCliente = socketCliente;
    }
    @Override
    public void run() {
        try {
            // Aquí va la lógica de procesamiento de la petición GET
            // ... (leer entrada, enviar respuesta HTTP)
            socketCliente.close();
        } catch (IOException e) { e.printStackTrace(); }
    }
}
```

Modelos de ejercicios posibles para tu examen

Dado que tu examen es de **PSP**, los ejercicios se centrarán en el uso de las clases de `java.net` y bibliotecas externas.

1. Resolución de nombres con `InetAddress`

Enunciado: Crea un programa que pida un nombre de host (ej. `www.google.com`) y devuelva todas sus direcciones IP asociadas. Gestiona si el host no es válido.

- **Concepto clave:** Uso de `getAllByName()`.
- **Código:**

```
Java
try {
    InetAddress[] direcciones =
```

```
InetAddress.getAllByName(nombreHost);
    for (InetAddress dir : direcciones) {
        System.out.println("IP: " + dir.getHostAddress());
    }
} catch (UnknownHostException e) {
    System.out.println("No se pudo resolver el host.");
}
```

2. Cliente FTP con Apache Commons Net

Enunciado: Implementa un cliente que se conecte a un servidor FTP, se valide y descargue un archivo específico.

- **Concepto clave:** Métodos `login()`, `enterLocalPassiveMode()` (para evitar bloqueos de firewall) y `retrieveFile()`.
- **Código:**

Java
```
FTPClient ftp = new FTPClient();
ftp.connect("servidor.com");
if (ftp.login("user", "pass")) {
    ftp.enterLocalPassiveMode(); // Muy importante en exámenes
    OutputStream os = new FileOutputStream("local.txt");
    ftp.retrieveFile("remoto.txt", os);
    ftp.logout();
}
```

3. Análisis de una URL y descarga de contenido

Enunciado: Crea un programa que reciba una URL, extraiga su protocolo y puerto, y guarde el código HTML en un fichero local.

- **Concepto clave:** Uso de `getProtocol()`, `getPort()` y `openStream()`.
- **Código:**

Java
```
URL url = new URL("http://ejemplo.com:80");
System.out.println("Puerto: " + url.getPort());
InputStream is = url.openStream(); // Devuelve InputStream para
lectura [cite: 1422]
```

```
// Lógica para escribir el 'is' en un archivo...
```

4. Monitorización de rendimiento (PSP puro)

Enunciado: Calcula el tiempo de procesamiento de una petición en el servidor.

- **Concepto clave:** `System.currentTimeMillis()` antes y después del proceso.