Ejercicio Hotel PelHilos a la mar – Servidor HTTP

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ServidorHTTP {
    public static void main(String[] args) throws Exception {
        ServerSocket serverSocket = new ServerSocket(12349);
        System.out.println("Servidor iniciado. Esperando reservas...");
        while (true) {
            Socket clientSocket = serverSocket.accept();
            new Thread(new HiloServidor(clientSocket)).start();
        }
    }
}
```

```java
class HiloServidor extends Thread {
        private final Socket cliente;

        public HiloServidor(Socket cliente) {
            this.cliente = cliente;
        }

        @Override
        public void run() {

            try (BufferedReader entrada = new BufferedReader(new InputStreamReader(cliente.getInputStream()));
                PrintWriter salida = new PrintWriter(cliente.getOutputStream(), true)
            ) {
                // Lee la primera línea de la petición HTTP.
                String peticion = entrada.readLine();
                // Ignora la petición si no es GET o POST.
                if (peticion != null && (!peticion.startsWith("GET") || !peticion.startsWith("POST"))) {
                    String ruta = peticion.split(" ")[1]; // Extrae la ruta solicitada.
                    StringBuilder cuerpo = new StringBuilder(); // Para almacenar el cuerpo de la solicitud.
                    String linea;

                    // Leer encabezados HTTP y determinar el tamaño del cuerpo.
                    int contentLength = 0;
                    while (!(linea = entrada.readLine()).isBlank()) {
                        if (linea.startsWith("Content-Length: ")) {
                            contentLength = Integer.parseInt(linea.substring(16));
                        }
                    }

                    // Leer el cuerpo si es un POST.
                    if (peticion.startsWith("POST") && contentLength > 0) {
                        char[] buffer = new char[contentLength];
                        entrada.read(buffer, 0, contentLength);
                        cuerpo.append(buffer);

                        String dato1 = cuerpo.toString().split("&")[0];
                        String dato2 = cuerpo.toString().split("&")[1];

                        String dia = dato1.split("=")[1];
                        int numero = Integer.parseInt(dato2.split("=")[1]);
                        System.out.println("La nueva reserva ha sido para el " + dia + " con " + numero + " habitaciones.");
                        GestorFichero.actualizarFichero(dia, numero);
                    }

                    String respuesta;
                    if (ruta.equals("/")) {
                        respuesta = construirRespuesta(200, Paginas.html_reservas);
                    } else {
                        respuesta = construirRespuesta(404, Paginas.html_noEncontrado);
                    }
                    salida.println(respuesta);

                    cliente.close();
                }

            } catch (IOException e) {
                e.printStackTrace(); // Muestra errores en la consola.
            } catch (Exception ex) {
                Logger.getLogger(HiloServidor.class.getName()).log(Level.SEVERE, null, ex);
            }
        }

        private String construirRespuesta(int codigo, String html) {
            String lineaInicial = codigo == 200 ? "HTTP/1.1 200 OK" : "HTTP/1.1 404 Not Found";
            return lineaInicial + "\n" +
                    "Content-Type: text/html; charset=UTF-8" +
                    "\nContent-Length: " + html.length() +
                    "\n\n" +
                    html;
        }
}
```

```java
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class GestorFichero {

    private static final String CLAVE = "1234567890123456";
    private static final String ALGORITMO = "AES";
    private static final Object objLunes = new Object();
    private static final Object objMartes = new Object();
    private static final Object objMiercoles = new Object();
    private static final Object objJueves = new Object();
    private static final Object objViernes = new Object();
    private static final Object objSabado = new Object();
    private static final Object objDomingo = new Object();

    private static byte[] cifrar(String datos) throws Exception {
        Cipher cipher = Cipher.getInstance(ALGORITMO);
        SecretKeySpec keySpec = new SecretKeySpec(CLAVE.getBytes(), ALGORITMO);
        cipher.init(Cipher.ENCRYPT_MODE, keySpec);
        return cipher.doFinal(datos.getBytes());
    }

    private static String descifrar(byte[] datosCifrados) throws Exception {
        Cipher cipher = Cipher.getInstance(ALGORITMO);
        SecretKeySpec keySpec = new SecretKeySpec(CLAVE.getBytes(), ALGORITMO);
        cipher.init(Cipher.DECRYPT_MODE, keySpec);
        return new String(cipher.doFinal(datosCifrados));
    }

    public static void actualizarFichero(String dia, int habitaciones) {
        Object lock;
        lock = switch (dia) {
            case "lunes" -> objLunes;
            case "martes" -> objMartes;
            case "miercoles" -> objMiercoles;
            case "jueves" -> objJueves;
            case "viernes" -> objViernes;
            case "sabado" -> objSabado;
            default -> objDomingo;
        };
        try {
            synchronized (lock) {
                Path path = Paths.get(dia + ".txt");
                int reservas = 0;

                // Verificamos si el archivo existe y tiene contenido
                if (Files.exists(path) && Files.size(path) > 0) {
                    byte[] contenidoCifrado = Files.readAllBytes(path);
                    reservas = Integer.parseInt(descifrar(contenidoCifrado));
                }

                // Agregamos el nuevo valor
                reservas += habitaciones;
                if (reservas > 0) {
                    System.out.println("El nuevo dato en "+ dia + ".txt es de: " + reservas);
                }

                // Cifrar el contenido antes de guardarlo
                byte[] contenidoCifrado = cifrar(Integer.toString(reservas));

                // Escribe el contenido cifrado en el archivo.
                // Si el archivo no existe, lo crea (StandardOpenOption.CREATE)
                // Si el archivo ya existe, agrega el nuevo contenido al final
                // sin borrar lo anterior (StandardOpenOption.APPEND)
                Files.write(path, contenidoCifrado, StandardOpenOption.CREATE);

            }
        } catch (Exception e) {
            System.err.println(e.getMessage());
        }
    }
}
```