Práctica 2: Paso de parámetros en Java RMI.

Diseñar un programa que permita enviar los datos básicos de un paciente asintomático y el indicador de su frecuencia cardiaca a un servidor1, el cual analizara si la Temperatura se encuentra fuera de un rango normal, si es el caso, el servidor enviará una notificación de alerta médica a un servidor2 el cual mostrará el siguiente mensaje por pantalla:

El paciente con identificación [id] presenta una T ºC de [indicador]
la cual esta fuera del rango normal

Si esta dentro del rango, en el servidor1 se mostrará el siguiente mensaje:

El paciente [nombres] [apellidos]
con identificación [tipo_id][id]
presenta una T ºC de [indicador] que esta dentro del rango normal

Los datos a ingresar son: nombres, apellidos, tipo de id, id y temperatura. El sistema debe gestionar la información utilizando el modelo de RMI e implementándolo por medio de Java RMI. El rango normal de la Temperatura- es de 36.2-37.2 ºC.

Se debe diseñar un programa, que permita gestionar el registro y control de pacientes en el sistema, para ello debe realizar 3 operaciones: **registro de pacientes , consulta de pacientes y envío de temperatura**. Estas operaciones serán controladas mediante un Menú, tal como muestra la Figura 1:

```
============Menu===========
|1. Registrar asintomatico |
|2. Consultar asintomatico |
|3. Enviar indicador       |
|4. Salir                  |
============================
Ingrese la opcion:
```

Figura 1. Menú del médico

La primera opción permite registrar todos los datos de un paciente. La opción 2 permite consultar los datos de un paciente asintomático registrados. La opción 3 permite enviar el valor del indicador Temperatura del paciente asintomático. En el lado del servidor los pacientes serán almacenados en un vector. La máxima cantidad de pacientes a registrar será de 5 pacientes.

Validaciones

- En el cliente, en la opción 1, se debe validar que no existan registros repetidos y el servidor debe notificar al cliente si el registro es o no exitoso. Mediante un mensaje de notificación se le debe notificar al usuario del suceso.
- En el servidor se debe validar que máximo se puedan registrar 5 pacientes.

- Al consultar un paciente se debe indicar si la búsqueda es o no exitosa.

Desarrollo de la aplicación

La aplicación debe implementarse usando Java RMI, en un sistema operativo Linux o Windows. En la figura 2 se observa un diagrama de contexto que muestra las operaciones que puede realizar el médico. Las operaciones debe ser implementadas en un servidor, cada vez que se realice una petición en el cliente y se reciba una petición en el servidor**, se debe crear un eco por medio de un mensaje en pantalla** en el cual se describe cual método se está invocando o ejecutando

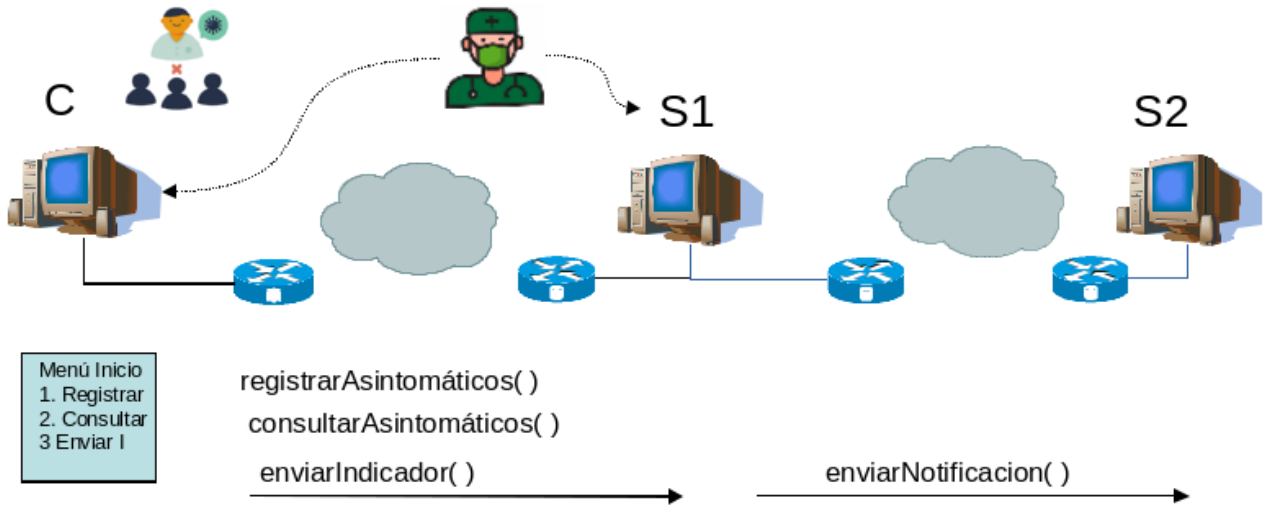

Figura 2. Sistema de registro y control

En la 1 opción el médico podrá registrar un paciente asintomático enviando un objeto de tipo ClsAsintomaticoDTO. En la opción 2, el médico podrá consultar los datos de un paciente asintomático registrado mediante la invocación del método *consultarAsintomatico()*, para su búsqueda se debe enviar como parámetro id. En la opción 3, el usuario podrá enviar el valor del indicador Temperatura.

Tener en cuenta: La solución de este ejercicio debe ser comprimida en formato zip o rar y ser subida vía el enlace fijado en univirtual. El nombre del archivo comprimido debe seguir el siguiente formato lsd_rmi02_nape1_nape2 en formato de compresión .zip o .rar. Donde nape1 corresponde a la primera inicial del nombre + apellido de uno de los integrantes, y nape2 corresponde a la primera inicial del nombre + apellido del segundo integrante.

Antes de iniciar se debe crear la siguiente estructura de directorios:

Antes de iniciar, estructurar una carpeta de trabajo donde se ubicaran los archivos fuente (directorio 'src') y los archivos binarios (bytecode) (directorio bin) . El nombre del directorio de trabajo debe coincidir con el nombre del archivo comprimido. Usar el archivo gen_dir2.bat para crear la estructura

mostrada en la Figura 3.

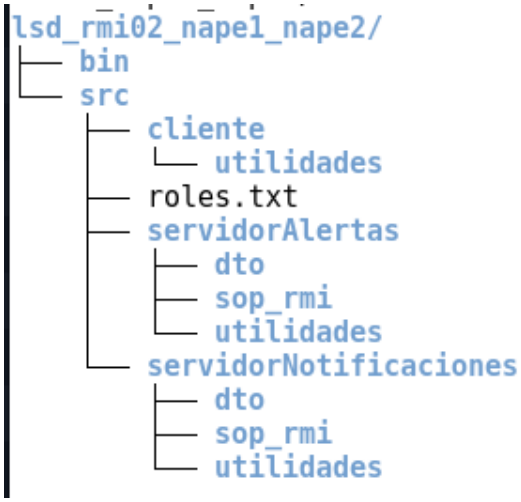

Figura 3. Estructura del directorio de trabajo

1. Diseñar e implementar los componentes de la interface1

1.a Definiendo la interface remota: Editar el archivo(ver figura 4) GestionAsintomaticosInt.java

```
package servidorDeAlertas.sop_rmi;

Import servidorDeAlertas.dto.ClsAsintomaticoDTO;
import java.rmi.Remote;
import java.rmi.RemoteException;

//Hereda de la clase Remote, lo cual la convierte en interfaz remota
public interface GestionAsintomaticosInt extends Remote{

   public boolean registrarAsintomatico(ClsAsintomaticoDTO objAsintomatico) throws
RemoteException;
   public ClsUsuarioDTO consultarAsintomatico(int id) throws RemoteException;
   public boolean enviarIndicador(int id, float ToC) throws RemoteException;

}
```

Figura 4. Interface remota GestionAsintomaticoInt

1.b) Implementando la clase DTO: Editar el archivo *AsintomaticoDTO.java*. Se utilizará el patrón de diseño DTO (Data Transfer Object) que permite tener objetos por medio de los cuales se transporta datos entre procesos. Por tanto, se debe implementar una clase *AsintomaticoDTO* que contendrá los atributos **nombre y apellidos del paciente, tipo de id, id, y domicilio**. Para facilitar la interacción entre el proceso Cliente y el Servidor de Objetos se puede incluir un **atributo error** por medio del cual se puede indicar al cliente el éxito o fracaso de un registro o una consulta de un paciente. El atributo

error toma el valor de 0 cuando la solicitud es exitosa y diferente si existe una falla. El encabezado de esta clase se muestra en la siguiente Figura 5.

```
package servidorDeAlertas.dto;

import java.io.Serializable;

public class ClsAsintomaticoDTO implements Serializable
```

Figura 5. Encabezado de la clase ClsAsintomaticoDTO

1.c) Implementando la Interface Remota: Editar el archivo *ClsGestionAsintomaticos.java* correspondiente al código del servidor de objetos. El encabezado de dicha clase se muestra en la Figura 6.

```
package servidorDeAlertas.sop_rmi;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;
import servidorDeNotificaciones.dto.ClsMensajeNotificacionDTO;
import servidorDeNotificaciones.sop_rmi.NotificacionInt;
import servidorDeAlertas.dto.ClsAsintomaticoDTO;
import servidorDeAlertas.utilidades.UtilidadesRegistroC;

public class ClsGestionAsintomaticos extends UnicastRemoteObject implements GestionAsintomaticosInt
```

Figura 6. Encabezado de la clase ClsGestionAsintomaticos

1.d) Compilar el archivo ClsAsintomaticoDTO, la interface remota y la clase que implementa la interface con la herramienta javac. Ubicados en el directorio 'src' el comando sería,

```
javac -d ../bin servidorDeAlertas/sop_rmi/*.java
```

2. Diseñar e implementar los componentes de la interface2

2.a Definiendo la interface remota: Editar el archivo(ver figura 7) NotificacionesInt.java

```
package servidorDeNotificaciones.sop_rmi;
import java.rmi.Remote;
import java.rmi.RemoteException;
Import servidorDeNotificaciones.ClsMensajeNotificacionDTO;

public interface NotificacionInt extends Remote {
   public void notificarRegistro(ClsMensajeNotificacionDTO objNotificacion) throws
RemoteException;
}
```

Figura 7. Interface remota NotificacionInt

2.b) Implementando la clase DTO: Editar el archivo *ClsMensajeNotificacionDTO.java*. Se utilizará el patrón de diseño DTO (Data Transfer Object) que permite tener objetos por medio de los cuales se transporta datos entre procesos. Por tanto, se debe implementar una clase *ClsMensajeNotificacionDTO* que contendrá los atributos de la notificación . Para facilitar la interacción entre el proceso Cliente y el

Servidor de Objetos se puede incluir un **atributo error** por medio del cual se puede indicar al cliente el éxito o fracaso de un registro o una consulta de un paciente. El atributo error toma el valor de 0 cuando la solicitud es exitosa y diferente si existe una falla. El encabezado de esta clase se muestra en la siguiente Figura 8.

```
public class ClsMensajeNotificacionDTO implements Serializable{
    private int id;
    private float ToC;
```

Figura 8. Encabezado de la clase ClsMensajeNotificacionDTO

2.c) Implementando la Interface Remota 2: Editar el archivo *ClsNotificacion.java* correspondiente al código del servidor de objetos. El encabezado de dicha clase se muestra en la Figura 9

```
package servidorDeNotificaciones.sop_rmi;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import servidorDeNotificaciones.dto.ClsMensajeNotificacionDTO;

public class ClsNotificacion extends UnicastRemoteObject implements NotificacionInt{
```

Figura 9. Encabezado de la clase ClsNotificacion

2.d) Compilar el archivo ClsMensajeNotificacionDTO, la interface remota y la clase que implementa la interface con la herramienta javac. Ubicados en el directorio 'src' el comando sería,

```
javac -d ../bin servidorDeNotificaciones/sop_rmi/*.java
```

3. Crear el cliente de objetos

3.a) Escribir el código del cliente de objetos: Editar un archivo denominado *ClienteDeObjetos.java*, e incluir el código correspondiente que permita consultar la referencia remota del n_s y ejecutar el menú principal del médico.

```
public class ClienteDeObjetos
{

        private static GestionAsintomaticosInt objRemoto;

        public static void main(String[] args)
        {
            int numPuertoNS= 0;
            String direccionIpNS = "";

            System.out.println("Cual es el la dirección ip donde se encuentra  el n_s de alertas ");
            direccionIpNS = UtilidadesConsola.leerCadena();
            System.out.println("Cual es el número de puerto por el cual escucha el n_s de alertas ");
            numPuertoNS = UtilidadesConsola.leerEntero();

            objRemoto = (GestionAsintomaticosInt) UtilidadesRegistroC.obtenerObjRemoto(direccionIpNS,numPuertoNS,
"ObjetoGestionAsintomaticos");
            MenuPrincipal();
```

Figura 10. Encabezado de la clase ClienteDeObjetos

4. Crear el servidor de objetos 1 y 2

4.a.) Escribir el código del servidor de objetos 1:Ubicados en la carpeta servidorDeAlertas, editar un archivo denominado *ServidorDeObjetos.java* e incluir el código correspondiente que permita

instanciar el objeto remoto y registrarlo en el N_S. El identificador del objeto remoto será **ObjetoGestionAsintomaticos.**

4.b) Escribir el código del servidor de objetos 2:Ubicados en la carpeta servidorDeNotificaciones, editar un archivo denominado *ServidorDeObjetos.java* e incluir el código correspondiente que permita instanciar el objeto remoto y registrarlo en el N_S. El identificador del objeto remoto será **ObjetoRemotoNotificaciones.**

4.c) Compilar las clases de los Servidores de Objetos y el Cliente de Objetos con la herramienta javac. Ubicados en el directorio 'src' el comando sería,

Para compilar el servidor de objetos 1:
```
javac -d ../bin servidorDeAlertas/*.java
```
Para compilar el servidor de objetos 2:
```
javac -d ../bin servidorDeNotificaciones/*.java
```

Para compilar el cliente de objetos:
```
javac -d ../bin cliente/*.java
```

4.d. Iniciar el registro (Si el servidor de objetos lanza el RMIRegistry omitir este paso).
Tener en cuenta que: Para lanzar el N_S se debe ubicar en el directorio 'bin'.

Lanzar el demonio del *RMI Registry* en un puerto dado o en el puerto por defecto (1099)
```
rmiregistry [#pto]
```
Por ejemplo:
```
rmiregistry 3032
```

5. Iniciar la aplicación.

Tener en cuenta que: Para ejecutar la aplicaciones ubicarse en el directorio 'bin'.
a. Ejecutar los servidores:
Comando general:
```
java Nombre_clase_servidor_obj
```

Para el servidor 2, si la clase pertenece al paquete 'servidorDeNotificaciones':
```
java servidorDeNotificaciones.ServidorDeObjetos
```
Para el servidor 1, si la clase pertenece al paquete 'servidorDeAlertas':
```
java servidorDeAlertas.ServidorDeObjetos
```

b. Ejecutar el cliente:
Comando general:
```
java Nombre_clase_cliente_obj
```
Por ejemplo, si la clase pertenece al paquete 'cliente':
```
java cliente.ClienteDeObjetos
```