INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPÚTO

PROGRAMACIÓN ORIENTADA A OBJETOS

PRÁCTICA 8: RMI

PROF. ROBERTO TECLA PARRA

GRUPO 2CM4

ZEPEDA FLORES ALEJANDRO DE JESÚS

12/06/3018

**Objetivo:**

Realizar un programa utilizando un método remoto que permita al cliente enviar una imagen al servidor y el servidor almacene dicha imagen en un vector para poder visualizarlas posteriormente.

**Implementación:**

**//IMPLEMENTACIÓN DEL CLIENTE**

La siguiente sección de código, podemos ver el constructor de ImagenesClient, en este definimos las variables de instancia del mismo archivo y organizamos la interfaz gráfica.

public ImagenesClient(){

try{

nombres=i.getNombres();

} catch (RemoteException re) {

System.out.print("Exception in HelloClient.constru:"+ re)

}

p = new JPanel();

p2 = new JPanel();

jbbn = new JComboBox(nombres);

b = new JButton("Agregar");

b.addActionListener(this);

t = new JTextField(10);

//t2=new JTextField(10);

jbbn.addActionListener(this);

jimg = new JLabel(img);

c = getContentPane();

p.add(jimg);

p2.add(b);p2.add(t);p2.add(jbbn);

c.add(p , BorderLayout.CENTER);

c.add(p2,BorderLayout.SOUTH);

setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

setVisible(true);

}

Ahora analizaremos el actionPerfomed, en esta parte obtenemos la fuente del botón y el nombre de la imagen a insertar, en caso de que no encontrar la imagen en la carpeta local, mandará un error.

public void actionPerformed(ActionEvent e){

Object obj = (Object)e.getSource();

if(obj instanceof JButton){

System.out.println("Boton");

try{

if (!(t.getText()).equals("") ){

ImageIcon tem= new ImageIcon(t.getText());

i.setIma( t.getText(),tem);

nombres=i.getNombres();

// JComboBox jbbn2=new JComboBox(nombres);

jbbn.addItem(t.getText());

t.setText("");

}

}catch(Exception ex){

System.out.println("falla imagen");

System.out.println(ex.toString());

}

}

**//CLASE QUE IMPLEMENTA LA INTERFAZ REMOTA**

getImage retorna una ImageIcon, busca en el arreglo de imágenes si hay alguna coincidencia con la petición del cliente, si la petición es aceptada, regresa la imagen, caso contrario imprime un error.

public ImageIcon getImage(String imageID) throws RemoteException , Exception {

int j=0;

try{

for(int i=0;i<nombre.size();i++){

if((nombre.elementAt(i)).equals(imageID)) j=i;

}

return ima.elementAt(j);

}catch (Exception e){

System.out.println("Error in GetImageIcon:\n");

throw e;

}

}

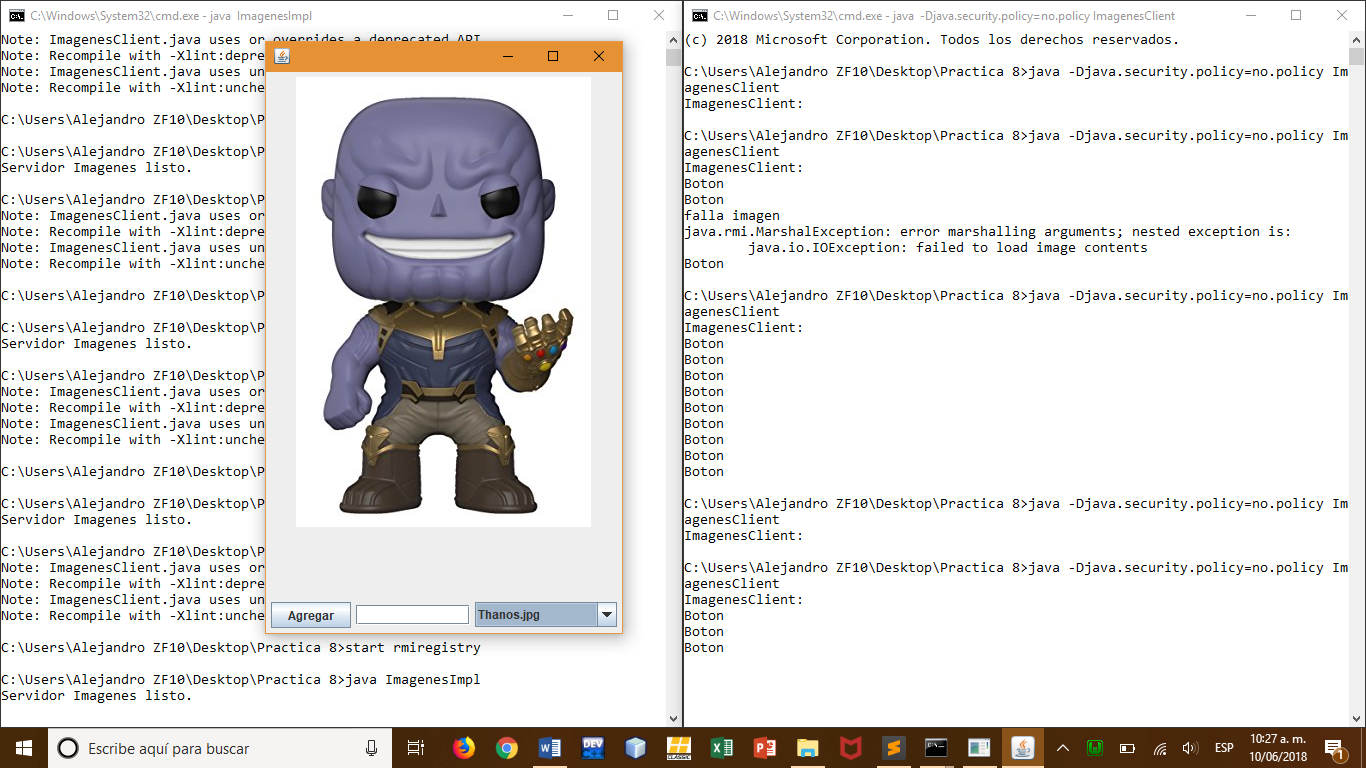
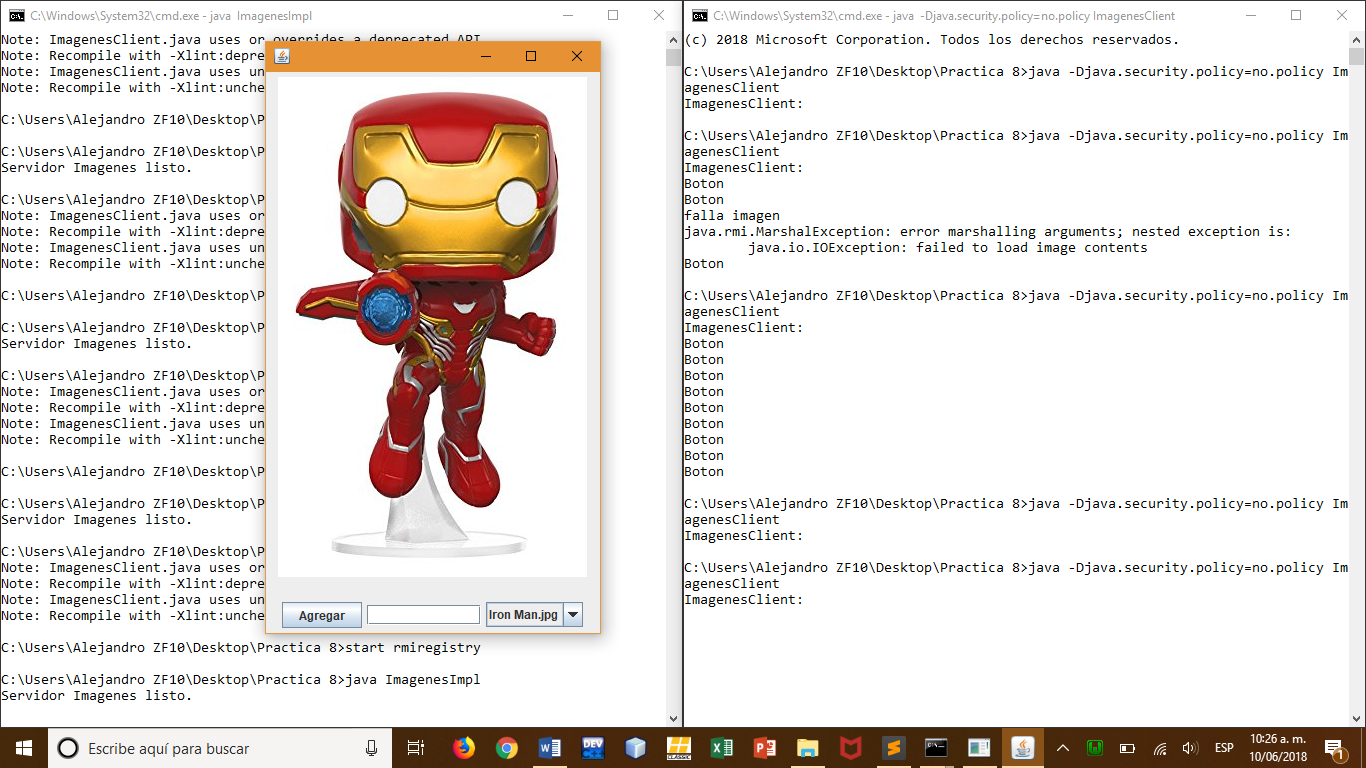
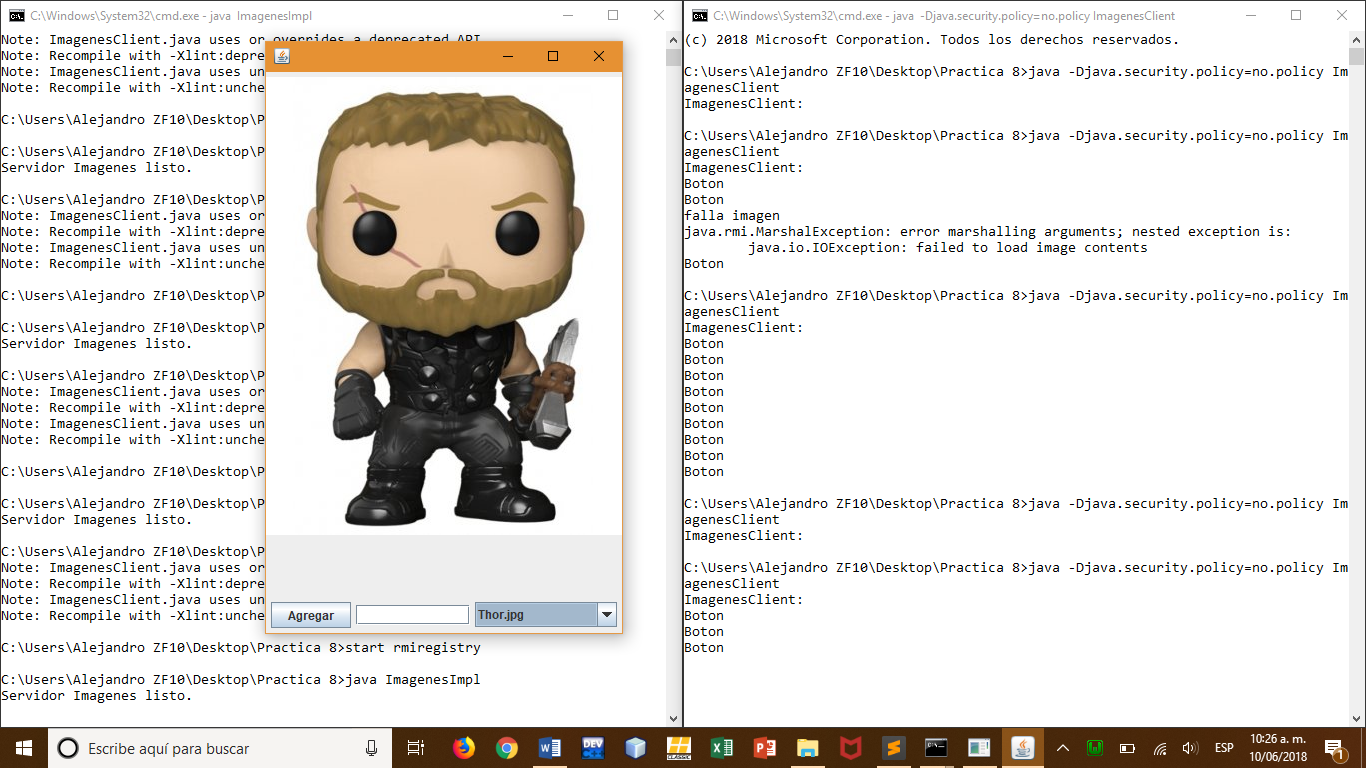
setIma, su única función es ingresar las imágenes que cliente pide al arreglo, siempre y cuando estas se encuentren en la carpeta local.

public void setIma(String nom , ImageIcon iman) throws RemoteException , Exception{

nombre.addElement(nom);

ima.addElement(iman);

}

****

**CONCLUSIÓN**

RMI (Remote Method Invocation) sirve para invocar un método de forma remota. En un principio, la práctica se complico por el manejo de RMI pero siguiendo los 3 pasos básicos:

1. Implementar la interfaz remota
2. Crear una clase que implemente la interfaz remota
3. Implementar al cliente

Además, para tener un mayor manejo hay que lanzar demasiadas excepciones y así poder detectar donde se encuentra el error fácilmente.