2ª Guía - Problemas

Nombre: Calva Hernández José Manuel
Grupo: 2CM3 Fecha: 06/11/17

Nota en los problemas de abajo incluir los import que hagan falta

Hilos

Problema 1.-Codificar una clase llamada **CuentaSegundos** que implemente la **interfaz Runnable**:
-Que cada segundo incremente en una unidad un contador y muestre el valor de dicho contador en una etiqueta.
-Que cuando llegue a un valor limite (que se establece cuando se crea un objeto) deje de incrementar el contador y de actualizar el valor mostrado.
Sugerencia extienda la clase Label y asigne cero al contador en el constructor.

```
import java.awt.*;
import javax.swing.*;
import java.io.*;

public class CuentaSegundos extends JFrame implements Runnable
{
	Panel p;
	Thread hilo;
	int contador, limite_cont;
	JLabel etiqueta;

	public CuentaSegundos(int limite)
	{
		p = new Panel();
		contador = 0;
		limite_cont = limite;
		etiqueta = new JLabel("0");

		add(p,"Center");
		p.add(etiqueta);
		setSize(500, 500);
	   setVisible(true);

		hilo = new Thread(this);
		hilo.start();
	}

	public void run()
	{
```

```
		while(contador <= limite_cont)
		{
			try
			{
				hilo.sleep(1000);
				etiqueta.setText(""+contador);
			}
			catch(InterruptedException e)
			{
				return;
			}

			contador++;
		}
	}

	public static void main (String s[])
	{
		new CuentaSegundos(7);
	}
}
```

Problema 2.-Codificar una clase llamada **Ticker** que mueva las letras de un texto de izquierda a derecha (o de derecha a izquierda). Se puede quitar la primer letra y pegarla (con +) al final y repetir esto de forma periódica.

```
public class Ticker implements Runnable {
	Thread TickerThread;
	String s,sub;
	int sleepTime;
	char c;

public Ticker(String s, int sleep){
		this.s=s;
		sleepTime=sleep;
		TickerThread=new Thread(this);
		TickerThread.start();
	}
	public void run(){
		System.out.println(s);
		while(true){
			c = s.charAt(0);
			sub = s.substring(1,s.length());
			s = sub + c;
			System.out.println(sub + c);
```

```
try{
                        Thread.sleep(sleepTime);
                    } catch (Exception e) {
                        return;
                    }
                }
        }
        public static void main(String args[]){
                new Ticker("El gato volador", 50);
        }
}
```

Problema 3.-Codificar un programa que permita al usuario introducir varios textos (el usuario escribe un texto en un campo de entrada y dicho texto se almacena cuando el usuario presiona un botón). Guarde los textos en una arreglo de cadenas y use un hilo para que después de un cierto numero de segundos el texto actual en una etiqueta se sustituye por el siguiente texto en el arreglo.

Nota: El programa puede ser una aplicación en modo texto y el arreglo donde se almacenan las cadenas puede ser el que se pasa como parámetro al método main y la cadena actual se puede mostrar en consola en lugar de una etiqueta.

Sockets

Problema 1.-Codificar un servidor que cuando un cliente se conecte le envié la cadena "Hola Mundo"

Problema 2.-Codificar un servidor que cuando un cliente se conecte lea el nombre del cliente y le envié a dicho cliente el nombre del ultimo usuario que se conecto.

RMI

En cada problema codificar la interfaz remota, la clase del objeto remoto (servidor), la clase del cliente y en los métodos remotos poner los parámetros que se necesiten.

Problema 1.-Escribir el código de un método remoto que calcule el **área** de un **triangulo**.

Interfaz Remota.

```
import javax.swing.*;
import java.util.*;
import java.rmi.*;

public interface area extends Remote {
        float getArea (int base, int altura) throws RemoteException, Exception;
}
```

Servidor.

```
import javax.swing.*;
import java.util.*;
import java.rmi.*;
```

```
import java.rmi.server.*;
import java.net.*;

public class areaImpl extends UnicastRemoteObject implements area{
	float result = 0;

	public areaImpl() throws RemoteException {
		super();
		result = 0;
	}

	public float getArea(int base, int altura) throws RemoteException, Exception{
		try{
			return (base*altura)/2;
		}
		catch (Exception e){
			System.out.println("Fehler in GetImageIcon:\n"+e.toString()+"\n");
			throw e;
		}
	}

	public static void main(String[] args) {
		try{
			areaImpl i = new areaImpl();
			Naming.rebind("area",i);
			System.out.println("Servidor de area triangulo listo. ");
		}
		catch(RemoteException re){
			System.out.println("Exception in areaImpl.main"+re);
		}
		catch(MalformerURLException e){
			System.out.println("MalformerURLException en areaImpl.main "+e);
		}
	}
}
```

Cliente.

```
import javax.swing.*;
import java.util.*;
import java.rmi.*;
import java.awt.*;
import java.awt.event.*;
import java.swing.event.*;


public class areaClient extends JFrame implements ActionListener{
	JLabel jresult = new JLabel();
	JTextField base = new JTextField();
	JTextField altura = new JTextField();
```

```
	JButton calcular = new JButton("Calcular area");
	Container c;
	static area i;
	float r = 0;

	public areaClient(){
		Container c = getContentPane();
		c.setLayout(new GridLayout(2,1));
		c.add(base, "base");
		c.add(altura, "altura");
		c.add(calcular);
		calcular.addActionListener(this);
		c.add(jresult);
		setVisible(true);
	}

	public void actionPerformed(ActionEvent e){
		int alturaa, basee;
		alturaa = Integer.parseInt(altura.getText());
		basee = Integer.parseInt(base.getText());

		try{
			r=i.getArea(basee,alturaa);
		}
		catch (Exception ex){
			System.out.println("Falla servidor");
			System.out.println(ex.toString());
		}
		jresult.setText(r+"");
	}

	public static void main(String[] args) {
		System.setSecurityManager(new RMISecurityManager());
		try{
			i = (area) Naming.lookup("area");
			System.out.println("areaCliente: ");
		}
		catch (Exception e){
			System.out.println("Exception in main: "+e);
		}
		areaClient a = new areaClient();
	}
}
```

Problema 2.-Escribir el código de un método remoto que retorne una cadena indicando si un **punto** esta **dentro o fuera** de un **circulo** de radio 100 con centro en el origen.

Problema 3.-Escribir el código de un método remoto que calcule el **salario semanal** de un **trabajador** que gana el salario mínimo en base a los horas trabajadas por dicho trabajador durante la semana. Considere 40

horas normales de trabajo a la semana y que las horas extras (las que se trabajan después de transcurridas las 40) se pagan dobles.

Interfaz Remota.

```
import javax.swing.*;
import java.util.*;
import java.rmi.*;

public interface salario extends Remote {
	float getSalario (int horas, int sueldo) throws RemoteException, Exception;
}
```

Servidor.

```
import javax.swing.*;
import java.util.*;
import java.rmi.*;
import java.rmi.server.*;
import java.net.*;

public class SalarioImpl extends UnicastRemoteObject implements salario{
	float result = 0;

	public salarioImpl() throws RemoteException{
		super();
		result = 0;
	}

	public float getSalario(int horas, int sueldo) throws RemoteException, Exception{
		int h = 0, sueltoT = 0;
		try{
			if(horas > 40){
				h = horas - 40;
				sueldoT = (40*sueldo)+(h*sueldo*2);
			}
			else{
				sueldoT = (40*sueldo);
			}

			return sueldoT;
		}
		catch (Exception e){
			System.out.println("Fehler in getImageIcon: \n"+e.toString()+"\n");
			throw e;
		}
	}

	public static void main(String[] args) {
		try{
			salarioImpl i = new SalarioImpl();
```

```
                Naming.rebind("salario",i);
                System.out.println("Servidor de salario listo.");
            }
            catch (RemoteException re){
                System.out.println("Exception in salarioImpl.main: "+re):
            }
            catch (MalformedURLException e){
                System.out.println("MalformedURLException en salarioImpl.main: "+e):
            }
        }
}
```

Cliente.

```
import javax.swing.*;
import java.util.*;
import java.rmi.*;
import java.awt.*;
import java.awt.event.*;
import java.swing.event.*;


public class salarioClient extends JFrame implements ActionListener{
        JLabel jresult = new JLabel();
        JTextField horas = new JTextField();
        JTextField sueldo = new JTextField();
        JButton calcular = new JButton("Calcular salario");
        Container c;
        static salario i;
        float r = 0;

        public salarioClient(){
            Container c = getContentPane();
            c.setLayout(new GridLayout(2,1));
            c.add(horas, "horas a la semana");
            c.add(sueldo, "sueldo por hora");
            c.add(calcular);
            calcular.addActionListener(this);
            c.add(jresult);
            setVisible(true);
        }

        public void actionPerformed(ActionEvent e){
            int sueldoa, horase;
            sueldoa = Integer.parseInt(sueldo.getText());
            horase = Integer.parseInt(horas.getText());

            try{
                r=i.getSalario(horase,sueldoa);
            }
```

```
            catch (Exception ex){
                System.out.println("Falla servidor");
                System.out.println(ex.toString());
            }
            jresult.setText(r+"");
        }

        public static void main(String[] args) {
            System.setSecurityManager(new RMISecurityManager());
            try{
                i = (salario) Naming.lookup("salario");
                System.out.println("salarioCliente: ");
            }
            catch (Exception e){
                System.out.println("Exception in main: "+e);
            }
            salarioClient a = new salarioClient();
        }
}
```

Problema 4.-Escribir el código de un método remoto que **convierta** una **cadena** a **mayúsculas** y otro método remoto que obtenga la **longitud** de una **cadena**.

Interfaz Remota.

```
import javax.swing.*;
import java.util.*;
import java.rmi.*;

public interface cadena extends Remote {
        int getNumCadena (String cadena) throws RemoteException, Exception;
        String cadMayus (String cadena) throws RemoteException, Exception;
}
```

Servidor.

```
import javax.swing.*;
import java.util.*;
import java.rmi.*;
import java.rmi.server.*;
import java.net.*;

public class cadenaImpl extends UnicastRemoteObject implements cadena{
        float result = 0;

        public cadenaImpl() throws RemoteException {
            super();
            result = 0;
        }

        public float getNumCadena(String cadena) throws RemoteException, Exception{
```

```
		int tamaño;
		try{
			tamaño = cadena.length();
			return tamaño;
		}
		catch (Exception e){
			System.out.println("Fehler in GetImageIcon:\n"+e.toString()+"\n");
			throw e;
		}
	}

	public String cadMayus(String cadena) throws RemoteException, Exception{
		String newCadena;
		try{
			newCadena = cadena.toUpperCase();
			return newCadena;
		}
		catch (Exception e){
			System.out.println("Fehler in GetImageIcon:\n"+e.toString()+"\n");
			throw e;
		}
	}

	public static void main(String[] args) {
		try{
			cadenaImpl i = new cadenaImpl();
			Naming.rebind("area",i);
			System.out.println("Servidor de cadena listo. ");
		}
		catch(RemoteException re){
			System.out.println("Exception in cadenaImpl.main"+re);
		}
		catch(MalformerURLException e){
			System.out.println("MalformerURLException en cadenaImpl.main "+e);
		}
	}
}
```

Cliente.

```
import javax.swing.*;
import java.util.*;
import java.rmi.*;
import java.awt.*;
import java.awt.event.*;
import java.swing.event.*;


public class cadenaClient extends JFrame implements ActionListener{
	JLabel jresult = new JLabel();
```

```
JTextField cadena = new JTextField("Ingrese la cadena");
JButton mayusculas = new JButton("Mayusculas");
JButton tamaño = new JButton("Tamaño");
Container c;
static cadena i;
String r;

public cadenaClient(){
	Container c = getContentPane();
	c.setLayout(new GridLayout(2,1));
	c.add(cadena);
	c.add(mayusculas);
	mayusculas.addActionListener(this);
	c.add(tamaño);
	tamaño.addActionListener(this);
	c.add(jresult);
	setVisible(true);
}

public void actionPerformed(ActionEvent e){
	int a=0;
	if(e.getSource()==mayusculas){
		try{
			r=i.cadMayus(cadena.getText());
		}
		catch (Exception ex){
			System.out.println("Falla servidor");
			System.out.println(ex.toString());
		}
	}
	else{
		try{
			r=i.getNumCadena(cadena.getText());
		}
		catch (Exception ex){
			System.out.println("Falla servidor");
			System.out.println(ex.toString());
		}
	}
	jresult.setText(r+"");
}

public static void main(String[] args) {
	System.setSecurityManager(new RMISecurityManager());
	try{
		i = (cadena) Naming.lookup("cadena");
		System.out.println("cadenaCliente: ");
	}
	catch (Exception e){
```

```
                System.out.println("Exception in main: "+e);
            }
            cadenaClient a = new cadenaClient();
        }
}
```

Problema 5.- Escribir el código de los siguientes métodos remotos: el que calcula el **máximo** de un **arreglo** de enteros, el que calcula el mínimo de un arreglo de enteros y el que calcula el promedio de un arreglo de enteros

Problema 6.- Escribir el código de un chat en rmi y que use callbacks del lado del cliente.

Javabeans

Para resolver los problemas de abajo tenga en cuenta lo siguiente:
-Un Javabean tiene que tener un **constructor por defecto** (sin argumentos)
-Un Javabean tiene que tener **persistencia**, es decir, debe implementar la interfaz *Serializable*

Problema 1-Codificar un JavaBean llamado **Viaje** que tenga las siguientes propiedades:
Origen (lectura/escritura), Destino (lectura/escritura), y Costo (lectura/escritura)

```
import java.io.Serializable;

public class Viaje implements Serializable{
private String origen;
private String destino;
private String Costo;
public Viaje () {
}
public Viaje(String origen, String destino, String costo){
this.origen=origen;
this.destino=destino;
this.costo=costo;
}
public String getOrigen(){
        return origen;
}
public String getDestino(){
        return destino;
}
public String getCosto(){
        return costo;
}
public void setOrigen(String origen){
        this.origen=origen;
}
public void setDestino(String destino){
        this.destino=destino;
}
public void setCosto(String costo){
```

```
        this.costo=costo;
}
}
```

Problema 2-Codificar un JavaBean llamado **Pelicula** que tenga las siguientes propiedades:
Titulo (lectura/escritura), Director (lectura/escritura), y Año (lectura/escritura)

```
import java.io.Serializable;

public class Pelicula implements Serializable{
private String titulo;
private String director;
private int año;
public Pelicula () {
}
public Pelicula(String titulo, String director, int año){
this.titulo=titulo;
this.director=director;
this.año=año;
}
public String getTitulo(){
        return titulo;
}
public String getDirector(){
        return director;
}
public int getAño(){
        return año;
}
public void setTitulo(String titulo){
        this.titulo=titulo;
}
public void setDirector(String director){
        this.director=director;
}
public void setAño(int año){
        this.año=año;
}

}
```

Problema 3.-Codificar un JavaBean llamado **Perro** que tenga las siguientes propiedades:
Nombre (lectura/escritura), Raza (lectura/escritura), Edad (lectura/escritura), Dueño (lectura/escritura) , Genero (lectura/escritura), y AñosRestantesDeVida (solo lectura). Suponga que si la raza es:

Pastor Aleman el perro vive 15 años
Boxer el perro vive 11 años
Terrier el perro vive 8 años,
Doberman el perro vive 13 años

para las otras razas vive 14 años

```
import java.io.Serializable;

public class Perro implements Serializable{
	private String nombre;
	private String raza;
	private String dueño;
	private String genero;
	private int edad;
	public Perro () {
	}
	public Perro(String nombre, String raza, String dueño, String genero, int edad){
		this.nombre=nombre;
		this.raza=raza;
		this.edad=edad;
	}
	public String getNombre(){
		return nombre;
	}
	public String getRaza(){
		return raza;
	}
	public String getDueño(){
		return dueño;
	}
	public String getGenero(){
		return genero;
	}
	public int getEdad(){
		return edad;
	}
	public int añosVida(){
		int esperanza = 0;
	   switch (raza) {
	      case "Pastor Aleman":  esperanza = 15;
	            break;
	      case "Boxer":  esperanza = 11;
	            break;
	      case "Terrier":  esperanza = 8;
	            break;
	      case "Doberman":  esperanza = 13;
	            break;
	      default: esperanza = 14;
	            break;
	   }
	   return edad-esperanza;
	}
	public void setNombre(String nombre){
```

```
		this.nombre=nombre;
	}
	public void setRaza(String raza){
		this.raza=raza;
	}
	public void setDueño(String dueño){
		this.dueño=dueño;
	}
	public void setGenero(String genero){
		this.genero=genero;
	}
	public void setEdad(int edad){
		this.edad=edad;
	}

}
```

Problema 4.-En el siguiente JavaBean (**Car**) determinar cuales son sus propiedades y si son de lectura o escritura o ambas cosas.

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
public class Car implements Serializable{
    private String make, model;
    double price;
    public Car() { this ("", "", 0.0); }
    public String getMake() { return make; }
    public void setMake (String m) { make = m; }
    public String getModel() { return model; }

public void setModel (String mo) { model = mo; }
public double getPrice() { return price; }
}
```

Sus propiedades son make(lectura/escritura), model(lectura/escritura), price(lectura)

Servlets

Para cada problema de abajo escribir el código **HTML** del **formulario** con los campos de entradas que sean necesarios y el botón de enviar.

1.-Codificar un Servlet que **calcule el área de un circulo** a partir de los datos de un formulario enviados por el navegador y envié el resultado de dicho calculo al cliente como texto plano o HTML. Escribir el código HTML del formulario con un campo de entrada para el radio y el botón de enviar.

Código formulario (HTML)

```
<html>
 <body>
  <form method="POST" action="/servlets/Area">
```

```
      radio=<input type="text" name="radio ">
   <input type="submit" value="enviar">
  </form>
 </body>
</html>
```

Código Servlet

```
public class Area extends HttpServlet {
  public void init(ServletConfig config) throws ServletException {
    System.out.println("init");
  }
  public void doPost(HttpServletRequest req, HttpServletResponse res)
                          throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    String n = req.getParameter("radio");
    float rad=Float.parseFloat(n);
    float area=3.1416*rad*rad;
    out.println("<HTML>");
    out.println("<HEAD><TITLE>Area circulo</TITLE></HEAD>");
    out.println("<BODY>");
    out.printf("<H1>El area del circulo de radio %d es %.2f </H1>",rad,area);
    out.println("</BODY></HTML>");
  }
  public void destroy() {
    System.out.println("destroy");
  }
  public String getServletInfo() {
    return null;
  }
  public ServletConfig getServletConfig() {
    return null;
  }
 }
```

2.-Codificar un Servlet que **calcule el perímetro de un pentágono no regular** a partir de los datos de un formulario enviados por el navegador y envié el resultado de dicho calculo al cliente como texto plano o HTML Escribir el código HTML del formulario con un campo de entrada para cada lado del pentágono no regular y el botón de enviar.

Código formulario (HTML)

```
<html>
 <body>
  <form method="POST" action="/servlets/Perimetro">
      lado1=<input type="text" name="lado1 ">
      lado2=<input type="text" name="lado2 ">
      lado3=<input type="text" name="lado3 ">
      lado4=<input type="text" name="lado4 ">
      lado5=<input type="text" name="lado5 ">
   <input type="submit" value="enviar">
```

```
  </form>
 </body>
</html>
```

Código Servlet

```
public class Perimetro extends HttpServlet {
  public void init(ServletConfig config) throws ServletException {
    System.out.println("init");
  }
  public void doPost(HttpServletRequest req, HttpServletResponse res)
                          throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    String n1 = req.getParameter("lado1");
    String n2 = req.getParameter("lado2");
    String n3 = req.getParameter("lado3");
    String n4 = req.getParameter("lado4");
    String n5 = req.getParameter("lado5");
    int l1=Integer.parseInt(n1);
    int l2=Integer.parseInt(n2);
    int l3=Integer.parseInt(n3);
    int l4=Integer.parseInt(n4);
    int l5=Integer.parseInt(n5);
   int perimetro=l1+l2+l3+l4+l5;
    out.println("<HTML>");
    out.println("<HEAD><TITLE>Perimetro pentagono</TITLE></HEAD>");
    out.println("<BODY>");
    out.printf("<H1>El perimetro del pentágono es  de  %d </H1>",perimetro);
    out.println("</BODY></HTML>");
  }
  public void destroy() {
    System.out.println("destroy");
  }
  public String getServletInfo() {
    return null;
  }
  public ServletConfig getServletConfig() {
    return null;
  }
 }
```

3.-Codificar un Servlet que envié al cliente la **capital del país** que el usuario escribió en un formulario. Escribir el código HTML del formulario con un campo de entrada para el pais y el botón de enviar.

4.-Codificar un Servlet que **calcule el promedio de las calificaciones** de un alumno en las siguientes materias: física, matemáticas , química , y español a partir de los datos de un formulario enviados por el navegador y envié el resultado de dicho calculo al cliente como texto plano o HTML. Escribir el código HTML del formulario con un campo de entrada para cada materia y el botón de enviar.

*5.-Codificar un Servlet que dibuje una **imagen** en **memoria** y luego la envié a un cliente

*6.-Codificar un Servlet que reciba el nombre de una **imagen** y que lea la imagen de un **archivo** y luego la envie a un cliente

*7.-Codificar un Servlet que reciba el nombre de una **imagen** y que obtenga la imagen de una **base de datos** mediante **JDBC** y luego la envié a un cliente

*8.-Codificar un Servlet que reciba el nombre de un perro y que obtenga los datos del **perro** (incluyendo su **foto**) de una **base de datos** mediante **JDBC** y luego envié dichos datos a un cliente

*9.-Codificar un Servlet que reciba el nombre de un **Mp3** y que lea el Mp3 de un **archivo** y luego lo envié a un cliente

*10.-Codificar un Servlet que que reciba el nombre de un Avi , lea el vídeo **Avi** de un **archivo** y luego lo envie a un cliente y codificar una aplicación cliente que lo reproduzca.