
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Universidad Politécnica Salesiana

Vicerrectorado Docente

Código del Formato:	GUIA-PRL-001
Versión:	VF1.0
Elaborado por:	Directores de Área del Conocimiento Integrantes Consejo Académico
Fecha de elaboración:	2016/04/01
Revisado por:	Consejo Académico
Fecha de revisión:	2016/04/06
Aprobado por:	Lauro Fernando Pesántez Avilés Vicerrector Docente
Fecha de aprobación:	2016/14/06
Nivel de confidencialidad:	Interno

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Descripción General

Propósito


El propósito del presente documento es definir un estándar para elaborar documentación de guías de práctica de laboratorio, talleres o centros de simulación de las Carreras de la Universidad Politécnica Salesiana, con la finalidad de lograr una homogenización en la presentación de la información por parte del personal académico y técnico docente.

Alcance


El presente estándar será aplicado a toda la documentación referente a informes de prácticas de laboratorio, talleres o centros de simulación de las Carreras de la Universidad Politécnica Salesiana.

Formatos

- Formato de Guía de Práctica de Laboratorio / Talleres / Centros de Simulación – para Docentes
- Formato de Informe de Práctica de Laboratorio / Talleres / Centros de Simulación – para Estudiantes

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES	
CARRERA: COMPUTACIÓN		ASIGNATURA: Programación Aplicada	
NRO. PRÁCTICA:	1	TÍTULO PRÁCTICA: Examen Practico Java	
OBJETIVO: Identificar los cambios importantes de Java Diseñar e Implementar expresiones regulares Entender la cada uno de las características nuevas en Java			
INSTRUCCIONES (Detallar las instrucciones que se dará al estudiante):		1. Revisar los conceptos fundamentales de Java	
		2. Establecer las características de Java en programación genérica	
		3. Implementar y diseñar los nuevos componentes de programación genérica	
		4. Realizar el informe respectivo según los datos solicitados.	
ACTIVIDADES POR DESARROLLAR (Anotar las actividades que deberá seguir el estudiante para el cumplimiento de la práctica)			
1. Revisar la teoría y conceptos de Java 8, 9 ,10, 11, 12			
2. Diseñar e implementar las características de Java para generar una expresion regular.			
3. Probar su funcionamiento y rendimiento dentro de los equipos de cómputo de programación genérica.			
4. Realizar práctica codificando los codigos de las nuevas características de Java y su uso dentro de un sistema escolar.			
Enunciado Se desea generar un sistema que me permita extraer infomación del internet a traves de expresiones regulares, esta informacion permitira vincular actividades desarrolladas del los niños con aplicaciones mobiles que permitan apoyar en el desarrollo de las actividades planteadas (https://play.google.com/store?hl=es&gl=US). Adicionalmente, se debe realizar un sistema de gestion de alumnos y actividades planificadas por curso, dentro de este sistema se debe realizar un procesos de administracion de usuarios los mismo que son los docentes de cada curso escolar, en este sentido solo debemos tener un administrador (Rector) el encargado de crear docentes y el curso que se le asigna. Ejemplo Rector: Docentes: 1. Diego Quisi 2. Vladimir Robles 3. Etc. Cursos: 1 de basica 2 de basica 3 basica			

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Asignacion de Curso – Docente

1 Basica -> Diego Quisi

2 Basica -> Vladimir Robles

Dentro de cada curso el docente gestionara los estudiantes y las actividades planificadas para el curso, estas actividades tendra una opcion de buscar aplicaciones moviles dentro de la tiendas de play store, obtenidas desde el internet, dentro de esta información lo importante es mostrar el link y una descripción para ello deberán utilizar expresiones regulares.

Ejemplo Docentes:

Alumnos

1. Juan Perez

2. Maria Peralta

3. .

Actividades:

1. Suma de numeros -> Obtener aplicaciones moviles (Link y Titulo)
2. Resta de numeros -> Obtener aplicaciones moviles
3. Oraciones compuestas -> Obtener aplicaciones moviles
4. Etc.

Toda esta infomación sera almacenada dentro de archivos y deberan tener aplicado al menos una patron de diseño y las nuevas características de programación de Java 8 o superior.

Al finalizar, generar el informe de la practica en formato PDF y subir todo el proyecto incluido el informe al repositorio personal.

La fecha de entrega: 23:55 del 01 de diciembre del 2020.

RESULTADO(S) OBTENIDO(S):

Realizar procesos de investigación sobre los cambios importantes de Java

Entender las aplicaciones de codificación de las nuevas características en base a la programación genérica y expresiones regulares.

Entender las funcionalidades adicionales de Java.

CONCLUSIONES:


Aprenden a trabajar en grupo dentro de plazos de tiempo establecidos, manejando el lenguaje de programación de Java.


RECOMENDACIONES:


Realizar el trabajo dentro del tiempo establecido.

Docente / Técnico Docente: _____

Firma: _____

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

		FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES	
CARRERA: Ingeniería en Computación		ASIGNATURA: Programación Aplicada	
NRO. PRÁCTICA:		TÍTULO PRÁCTICA: Exámen Interciclo.	
OBJETIVO ALCANZADO: Identificar los cambios importantes de Java Diseñar e Implementar expresiones regulares Entender la cada uno de las características nuevas en Java.			
ACTIVIDADES DESARROLLADAS			
<p>1.</p> <p style="text-align: center;">Enunciado</p> <p>Se desea generar un sistema que me permita extraer información del internet a través de expresiones regulares, esta información permitirá vincular actividades desarrolladas de los niños con aplicaciones móviles que permitan apoyar en el desarrollo de las actividades planteadas (https://play.google.com/store?hl=es&gl=US).</p> <p>Adicionalmente, se debe realizar un sistema de gestión de alumnos y actividades planificadas por curso, dentro de este sistema se debe realizar un proceso de administración de usuarios los mismos que son los docentes de cada curso escolar, en este sentido solo debemos tener un administrador (Rector) el encargado de crear docentes y el curso que se le asigna.</p> <p>Ejemplo Rector:</p> <p>Docentes:</p> <ol style="list-style-type: none"> 1. Diego Quisi 2. Vladimir Robles 3. Etc. <p>Cursos:</p> <p>1 de básica</p> <p>2 de básica</p> <p>3 de básica</p> <p>Asignación de Curso – Docente</p> <p>1 Básica -> Diego Quisi</p> <p>2 Básica -> Vladimir Robles</p> <p>Dentro de cada curso el docente gestionará los estudiantes y las actividades planificadas para el curso, estas actividades tendrán una opción de buscar aplicaciones móviles dentro de las tiendas de play store, obtenidas desde el internet, dentro de esta información lo importante es mostrar el link y una descripción para ello deberán utilizar expresiones regulares.</p> <p>Ejemplo Docentes:</p> <p>Alumnos</p> <ol style="list-style-type: none"> 1. Juan Perez 2. Maria Peralta 			

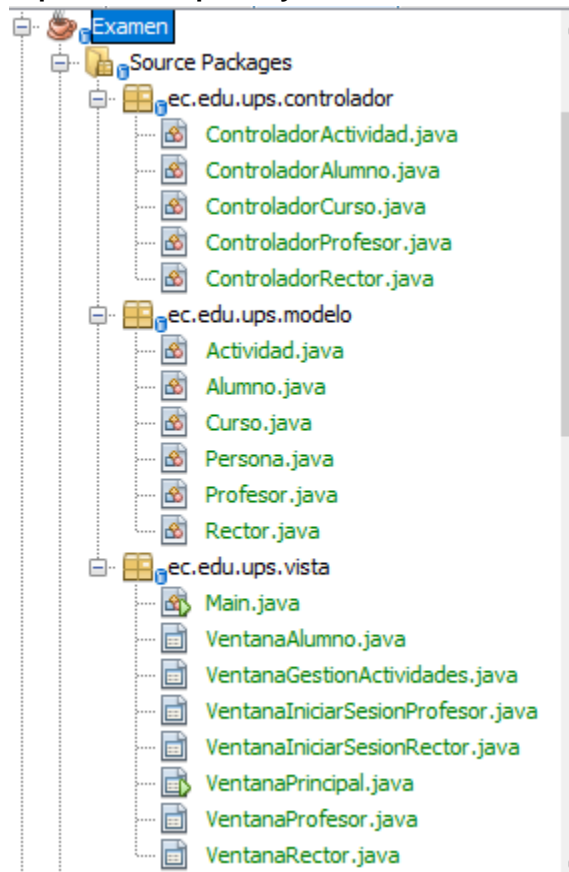
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

3. .

Actividades:

1. Suma de numeros -> Obtener aplicaciones moviles (Link y Titulo)
2. Resta de numeros -> Obtener aplicaciones moviles
3. Oraciones compuestas -> Obtener aplicaciones moviles
4. Etc.

Creamos el Proyecto con sus respectivos Paquetes y clases:



En el paquete modelo se encuentra:

Clase "Persona":

```
public class Persona {

    private String cedula;
    private String nombre;
    private String apellido;
    private String edad;

    public Persona() {
    }

    public Persona(String cedula, String nombre, String apellido, String edad) {
        this.cedula = cedula;
        this.nombre = nombre;
        this.apellido = apellido;
        this.edad = edad;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

    public String getEdad() {
        return edad;
    }

    public void setEdad(String edad) {
        this.edad = edad;
    }

    public String getCedula() {
        return cedula;
    }

    public void setCedula(String cedula) {
        this.cedula = cedula;
    }

    @Override
    public String toString() {
        return "Persona{" + "cedula=" + cedula + ", nombre=" + nombre + ", apellido=" + apellido + ", edad=" + edad + '}';
    }
}
```

Clase “Profesor” que hereda de Persona:

```
public class Profesor extends Persona{

    public Alumno alumno;
    public Curso curso;
    private String correoElectronico;
    private String contrasenia;

    public Profesor() {
    }

    public Profesor(Alumno alumno, Curso curso, String correoElectronico, String contrasenia) {
        this.alumno = alumno;
        this.curso = curso;
        this.correoElectronico = correoElectronico;
        this.contrasenia = contrasenia;
    }

    public Profesor(String cedula, String nombre, String apellido, String edad) {
        super(cedula, nombre, apellido, edad);
    }

    public Profesor(Alumno alumno, Curso curso, String correoElectronico, String contrasenia, String cedula, String nombre, String apellido, String edad) {
        super(cedula, nombre, apellido, edad);
        this.alumno = alumno;
        this.curso = curso;
        this.correoElectronico = correoElectronico;
        this.contrasenia = contrasenia;
    }
}
```



```
public Alumno getAlumno() {
    return alumno;
}

public void setAlumno(Alumno alumno) {
    this.alumno = alumno;
}

public Curso getCurso() {
    return curso;
}

public void setCurso(Curso curso) {
    this.curso = curso;
}

public String getCorreoElectronico() {
    return correoElectronico;
}

public void setCorreoElectronico(String correoElectronico) {
    this.correoElectronico = correoElectronico;
}

public String getContrasenia() {
    return contrasenia;
}

public void setContrasenia(String contrasenia) {
    this.contrasenia = contrasenia;
}

@Override
public String toString() {
    return "Profesor{" + "alumno=" + alumno + ", curso=" + curso + ", correoElectronico=" + correoElectronico + ", cc";
}
```

Clase “Rector” que hereda de Persona:

```
public class Rector<T> extends Persona{

    private Profesor profesor;
    private T correoElectronico;
    private T contrasenia;

    public Rector() {
    }

    public Rector(String cedula, String nombre, String apellido, String edad) {
        super(cedula, nombre, apellido, edad);
    }

    public Rector(Profesor profesor, T correoElectronico, T contrasenia, String cedula, String nombre, String apellido,
        super(cedula, nombre, apellido, edad);
        this.profesor = profesor;
        this.correoElectronico = correoElectronico;
        this.contrasenia = contrasenia;
    }

    public Rector(T correoElectronico, T contrasenia, String cedula, String nombre, String apellido, String edad) {
        super(cedula, nombre, apellido, edad);
        this.correoElectronico = correoElectronico;
        this.contrasenia = contrasenia;
    }

    public Profesor getProfesor() {
        return profesor;
    }

    public void setProfesor(Profesor profesor) {
        this.profesor = profesor;
    }

    public T getCorreoElectronico() {
        return correoElectronico;
    }

    public void setCorreoElectronico(T correoElectronico) {
        this.correoElectronico = correoElectronico;
    }

    public T getContrasenia() {
        return contrasenia;
    }

    public void setContrasenia(T contrasenia) {
        this.contrasenia = contrasenia;
    }

    @Override
    public String toString() {
        return "Rector{" + "profesor=" + profesor + ", correoElectronico=" + correoElectronico + ", contrasenia=" + contr
    }
}
```

Clase “Alumno” que hereda de Persona:

```
public class Alumno extends Persona {  
  
    private Curso curso;  
  
    public Alumno() {  
    }  
  
    public Alumno(String cedula, String nombre, String apellido, String edad) {  
        super(cedula, nombre, apellido, edad);  
    }  
  
    public Alumno(Curso curso, String cedula, String nombre, String apellido, String edad) {  
        super(cedula, nombre, apellido, edad);  
        this.curso = curso;  
    }  
  
    public Curso getCurso() {  
        return curso;  
    }  
  
    public void setCurso(Curso curso) {  
        this.curso = curso;  
    }  
  
    @Override  
    public String toString() {  
        return "Alumno{" + "curso=" + curso + '}';  
    }  
}
```

Clase "Curso":

```
public class Curso {

    private String nivel;
    private String paralelo;

    public Curso() {
    }

    public Curso(String nivel, String paralelo) {
        this.nivel = nivel;
        this.paralelo = paralelo;
    }

    public String getNivel() {
        return nivel;
    }

    public void setNivel(String nivel) {
        this.nivel = nivel;
    }

    public String getParalelo() {
        return paralelo;
    }

    public void setParalelo(String paralelo) {
        this.paralelo = paralelo;
    }

    @Override
    public String toString() {
        return "Curso{" + "nivel=" + nivel + ", paralelo=" + paralelo + '}';
    }
}
```

Clase “Actividad”:

```
public class Actividad<T> {

    private Curso curso;
    private String nombre;
    private String descripción;
    private String link;

    public Actividad() {
    }

    public Actividad(Curso curso, String nombre, String descripción, String link) {
        this.curso = curso;
        this.nombre = nombre;
        this.descripcion = descripción;
        this.link = link;
    }

    public Curso getCurso() {
        return curso;
    }

    public void setCurso(Curso curso) {
        this.curso = curso;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }


    public String getDescripción() {
        return descripción;
    }

    public void setDescripción(String descripción) {
        this.descripcion = descripción;
    }

    public String getLink() {
        return link;
    }

    public void setLink(String link) {
        this.link = link;
    }

    @Override
    public String toString() {
        return "Actividad{" + "curso=" + curso + ", nombre=" + nombre + ", descripción=" + descripción + ", link=" +
    }
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

En el paquete Controlador se encuentra:
Clase “ControladorRector” con Patrón de Diseño SINGLETON

```
public class ControladorRector {

    //Patron de diseño Singleton
    private static Rector rector;

    private Pattern patron;
    private Matcher corpus;
    private List<Rector> listaRectores;
    private int re;
    private int tam;
    private RandomAccessFile file;

    public ControladorRector() {
        listaRectores = new ArrayList<>();
        tam = 203;
        try {
            //C:\Users\Usuario\Desktop\ExamenApp.txt
            file = new RandomAccessFile("Desktop/ExamenApp.txt", "rw");
        } catch (IOException ex) {
            System.out.println("error de escritura y lectura");
            ex.printStackTrace();
        }
    }

    public boolean validar(String texto) {
        corpus = patron.matcher(texto);
        return corpus.find();
    }

    public void ingresarRegex(String regex) {
        patron = Pattern.compile(regex);
    }

    public void registrar(Rector r) {
        // return this.listaRectores.add(new Rector(r.getCedula(), r.getNombre(), r.getApellido(), r.getEdad(), r.getCor

        try {
            file.seek(file.length());
            file.writeUTF(r.getCedula());
            file.writeUTF(r.getNombre());
            file.writeUTF(r.getApellido());
            file.writeUTF(r.getEdad());
            file.writeUTF((String) r.getContrasenia());
            file.writeUTF((String) r.getCorreoElectronico());
            file.writeUTF(r.getProfesor().toString());
        } catch (IOException ex) {
            System.out.println("Error de creación");
        }
    }
}
```

```
public Rector verRector(String cedula,String nombre,String apellido,String edad) throws IOException {
    int e = 0;
    for (Rector rec : listaRectores) {
        if (rec.getCedula().equals(cedula)) {
            file.seek(e);
            String f = file.readUTF().trim();
            if (cedula.trim().equals(f)&& nombre.trim().equals(f)&& apellido.trim().equals(f) && edad.trim().equals(:
                Rector r = new Rector(cedula,nombre,apellido,edad);
                return r;
            }
            e += re;
            return rec;
        }
    }
    return null;
}

public void actualizar(Rector rector) throws IOException {

    int entrada= 0;
    String id = rector.getCedula();

    while (entrada < file.length()) {
        file.seek(entrada);
        String cedula = file.readUTF().trim();
        if (cedula.trim().equals(cedula)) {
            file.writeUTF(rector.getCedula());
            file.writeUTF(rector.getNombre());
            file.writeUTF(rector.getApellido());
            file.writeUTF((String) rector.getCorreoElectronico());
            file.writeUTF((String) rector.getContrasenia());
            break;
        }
        entrada += re;
    }
}
```



```
public void eliminar() throws IOException {  
  
    String id = rector.getCedula();  
    int entrada = 0;  
    while (entrada < file.length()) {  
        file.seek(entrada);  
        String cedulaArchivo = file.readUTF();  
        if (id.trim().equals(cedulaArchivo.trim())) {  
            file.writeUTF(espCompleto());  
            file.writeUTF(espCompleto());  
            file.writeUTF(espCompleto());  
            file.writeUTF(espCompleto());  
            file.writeUTF(espCompleto());  
            break;  
        }  
        entrada += re;  
    }  
}  
  
public List<String> obtenerLink(String l){  
    List<String> links= new ArrayList();  
    corpus=patron.matcher(l);  
    while(corpus.find()){  
        for(int i=0; i< corpus.groupCount();i++){  
            links.add(corpus.group(i));  
        }  
    }  
    return links;  
}
```

```

public Rector correoElectronico(String correoElectronico) throws IOException {
    int entrada = 66;

    for (Rector rec : listaRectores) {
        file.seek(entrada);

        String correo = file.readUTF();
        String contrasenia = file.readUTF();

        System.out.println(correo);

        if (correo.trim().equals(correoElectronico)) {

            file.seek(entrada - 66);
            rector = new Rector(reactor.getCedula(), reactor.getNombre(), reactor.getApellido(), reactor.getEdad());
            return rector;
        }
        entrada += re;
    }

    return null;
}

public Rector iniciarSesion(String correoElec, String contraseña) throws IOException {

    int entrada = 66;

    for (Rector rec : listaRectores) {
        file.seek(entrada);

        String correo = file.readUTF();
        String contrasenia = file.readUTF();

        System.out.println(correoElec + " " + contraseña);

        if (correoElec.equals(correo.trim()) && contraseña.equals(contrasenia.trim())) {

            file.seek(entrada - 66);
            rector = new Rector(reactor.getCedula(), reactor.getNombre(), reactor.getApellido(), reactor.getEdad());
            return rector;
        }
        entrada += re;
    }

    return null;
}

public String espCompleto(int esp) {
    String cad = "";
    return String.format("%-" + esp + "s", cad);
}

public String espPequeños(String cadena, int esp) {

    return cadena.substring(0, esp);
}

```

Clase "ControladorProfesor:"

```
public class ControladorProfesor {

    private static Profesor profe;

    private Pattern patron;
    private Matcher corpus;
    private List<Profesor> listaProfesores;
    private int re;
    private int tam;
    private RandomAccessFile file;

    public ControladorProfesor() {
        listaProfesores = new ArrayList<>();
        tam = 203;
        try {
            //C:\Users\Usuario\Desktop\ExamenApp.txt
            file = new RandomAccessFile("Desktop/ExamenApp.txt", "rw");
        } catch (IOException ex) {
            System.out.println("error de escritura y lectura");
            ex.printStackTrace();
        }
    }

    public void registrar(Profesor p) {

        try {
            file.seek(file.length());
            file.writeUTF(p.getCedula());
            file.writeUTF(p.getNombre());
            file.writeUTF(p.getApellido());
            file.writeUTF(p.getEdad());
            file.writeUTF((String) p.getContrasenia());
            file.writeUTF((String) p.getCorreoElectronico());
            file.writeUTF(p.getAlumno().toString());
            file.writeUTF(p.getCurso().toString());

        } catch (IOException ex) {
            System.out.println("Error de creación");
        }
    }
}
```

```
public Profesor verProfesor(String cedula, String nombre, String apellido, String edad) throws IOException {
    int e = 0;
    for (Profesor p : listaProfesores) {
        if (p.getCedula().equals(cedula)) {
            file.seek(e);
            String f = file.readUTF().trim();
            if (cedula.trim().equals(f) && nombre.trim().equals(f) && apellido.trim().equals(f) && edad.trim().equal
                Profesor prof = new Profesor(cedula, nombre, apellido, edad);
            return p;
        }
        e += re;
        return p;
    }
    return null;
}

public void eliminar() throws IOException {

    String id = profe.getCedula();
    int entrada = 0;
    while (entrada < file.length()) {
        file.seek(entrada);
        String cedulaArchivo = file.readUTF();
        if (id.trim().equals(cedulaArchivo.trim())) {
            file.writeUTF(espCompletos(0));
            file.writeUTF(espCompletos(0));
            file.writeUTF(espCompletos(0));
            file.writeUTF(espCompletos(0));
            file.writeUTF(espCompletos(0));
            break;
        }
        entrada += re;
    }
}
```

```
public void actualizar(Profesor prof) throws IOException {

    int entrada = 0;
    String id = prof.getCedula();

    while (entrada < file.length()) {
        file.seek(entrada);
        String cedula = file.readUTF().trim();
        if (cedula.trim().equals(cedula)) {
            file.writeUTF(prof.getCedula());
            file.writeUTF(prof.getNombre());
            file.writeUTF(prof.getApellido());
            file.writeUTF((String) prof.getCorreoElectronico());
            file.writeUTF((String) prof.getContrasenia());
            break;
        }
        entrada += re;
    }

}

public Profesor CorreoElectronico(String correoElectronico) throws IOException {
    int entrada = 66;


    for (Profesor prof : listaProfesores) {
        file.seek(entrada);

        String correo = file.readUTF();
        String contrasenia = file.readUTF();

        System.out.println(correo);

        if (correo.trim().equals(correoElectronico)) {
            file.seek(entrada - 66);
            profe = new Profesor(profe.getCedula(), profe.getNombre(), profe.getApellido(), profe.getEdad());
            return profe;
        }
        entrada += re;
    }

    return null;
}
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

public String espCompleto(int esp) {
    String cad = "";
    return String.format("%s-" + esp + "s", cad);
}

public String espPequeños(String cadena, int esp) {

    return cadena.substring(0, esp);
}

}

```

Clase "ControladorAlumno:"

```
public class ControladorAlumno {

    private Alumno alumno;

    private Pattern patron;
    private Matcher corpus;
    private List<Alumno> listaAlumnos;
    private int re;
    private int tam;
    private RandomAccessFile file;

    public void registrar(Alumno a) {

        try {
            file.seek(file.length());
            file.writeUTF(a.getCedula());
            file.writeUTF(a.getNombre());
            file.writeUTF(a.getApellido());
            file.writeUTF(a.getEdad());
            file.writeUTF(a.getCurso().getNivel());
            file.writeUTF(a.getCurso().getParalelo());

        } catch (IOException ex) {
            System.out.println("Error de creación");
        }

    }

    public Alumno verAlumno(String cedula,String nombre,String apellido,String edad) throws IOException {
        int e = 0;
        for (Alumno al : listaAlumnos) {
            if (al.getCedula().equals(cedula)) {
                file.seek(e);
                String f = file.readUTF().trim();
                if (cedula.trim().equals(f) && nombre.trim().equals(f) && apellido.trim().equals(f) && edad.trim().equals(f)) {
                    Alumno alum = new Alumno(cedula,nombre,apellido,edad);
                    return alum;
                }
                e += re;
            }
        }
        return null;
    }
}
```

```
public void eliminar() throws IOException {

    String id = alumno.getCedula();
    int entrada = 0;
    while (entrada < file.length()) {
        file.seek(entrada);
        String cedulaArchivo = file.readUTF();
        if (id.trim().equals(cedulaArchivo.trim())) {
            file.writeUTF(espCompletos(0));
            file.writeUTF(espCompletos(0));
            file.writeUTF(espCompletos(0));
            file.writeUTF(espCompletos(0));
            file.writeUTF(espCompletos(0));
            break;
        }
        entrada += re;
    }
}

public void actualizar(Alumno alum) throws IOException {

    int entrada= 0;
    String id = alum.getCedula();

    while (entrada < file.length()) {
        file.seek(entrada);
        String cedula = file.readUTF().trim();
        if (cedula.trim().equals(cedula)) {
            file.writeUTF(alum.getCedula());
            file.writeUTF(alum.getNombre());
            file.writeUTF(alum.getApellido());

            break;
        }
        entrada += re;
    }
}
```



```
public String espCompletos(int esp) {  
    String cad="";  
    return String.format("%-" + esp + "s", cad);  
}  
  
public String espPequeños(String cadena, int esp) {  
  
    return cadena.substring(0, esp);  
}
```

Clase “ControladorCurso:”

```
~/  
public class ControladorCurso {  
  
    private List <Actividad> act;  
  
    public ControladorCurso() {  
        act= new ArrayList<>();  
    }  
  
  
    public void registrar( Actividad acti) {  
  
        act.add(acti);  
    }  
  
}
```

Clase “ControladorActividad:”

```
public class ControladorActividad {

    private List<Curso> curso;

    private Pattern patron;
    private Matcher corpus;
    private List<Rector> listaRectores;
    private int re;
    private int tam;
    private RandomAccessFile file;


    public ControladorActividad() {
        curso = new ArrayList<>();
    }

    public boolean validar(String texto) {
        corpus = patron.matcher(texto);
        return corpus.find();
    }

    public void ingresarRegex(String regex) {
        patron = Pattern.compile(regex);
    }

    public List<String> obtenerActividades(String texto) {
        List<String> ac = new ArrayList();
        corpus = patron.matcher(texto);
        while (corpus.find()) {
            for (int i = 1; i < corpus.groupCount(); i++) {
                ac.add(corpus.group(i));
            }
        }
        return ac;
    }
}
```

En el paquete Vista se encuentra:
"VentanaPrincipal" :

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Agregar Gestion Help

“VentanaRector” :



RECTOR

Cédula:

Nombre:

Apellido:

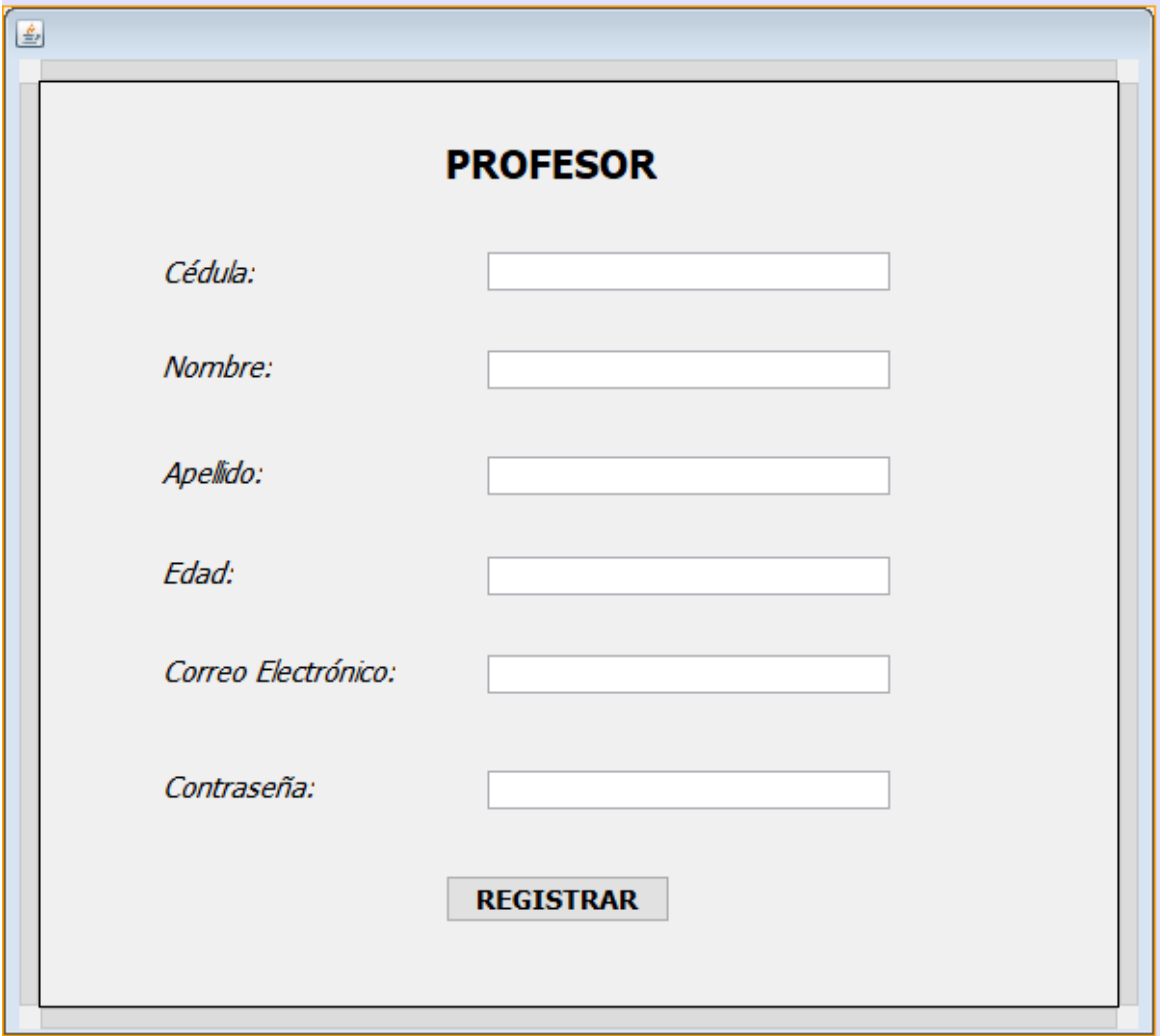
Edad:

Correo Electrónico:

Contraseña:

REGISTRAR

“VentanaProfesor” :



PROFESOR

Cédula:

Nombre:

Apellido:

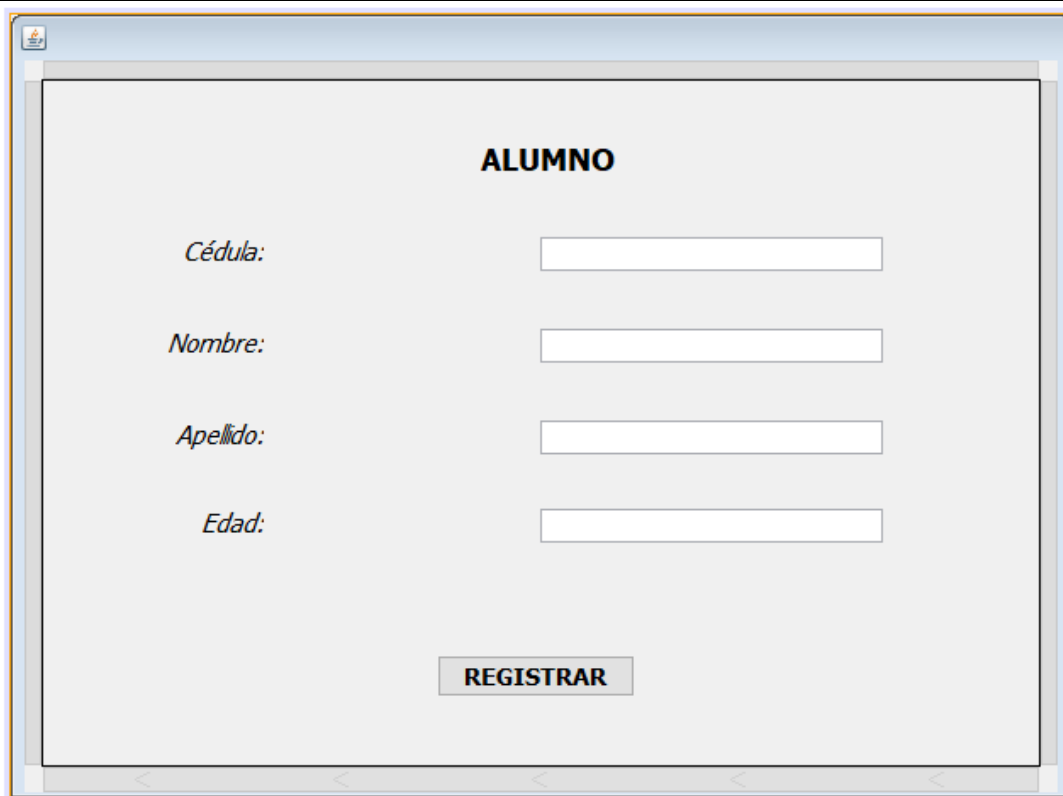
Edad:

Correo Electrónico:

Contraseña:

REGISTRAR

“VentanaAlumno” :



ALUMNO

Cédula:

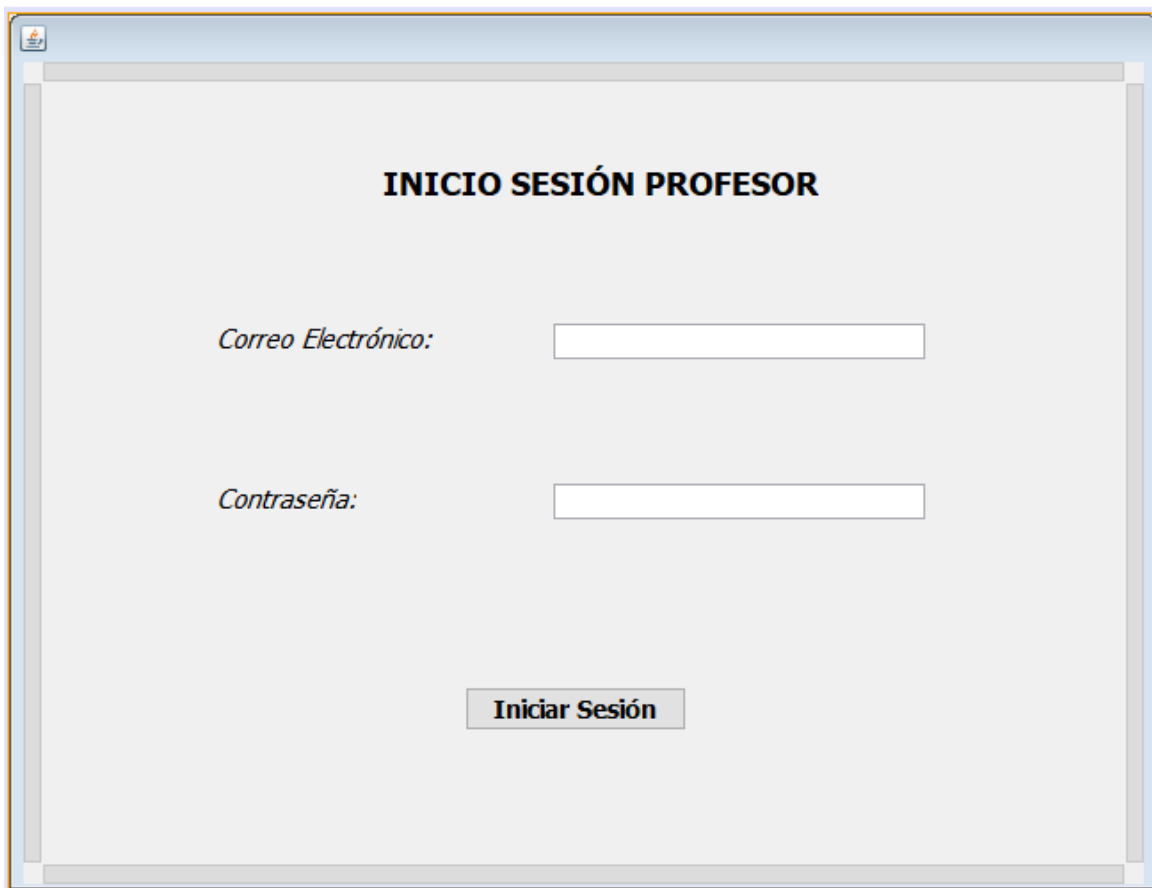
Nombre:

Apellido:

Edad:

REGISTRAR

“VentanaIniciarSesionProfesor” :



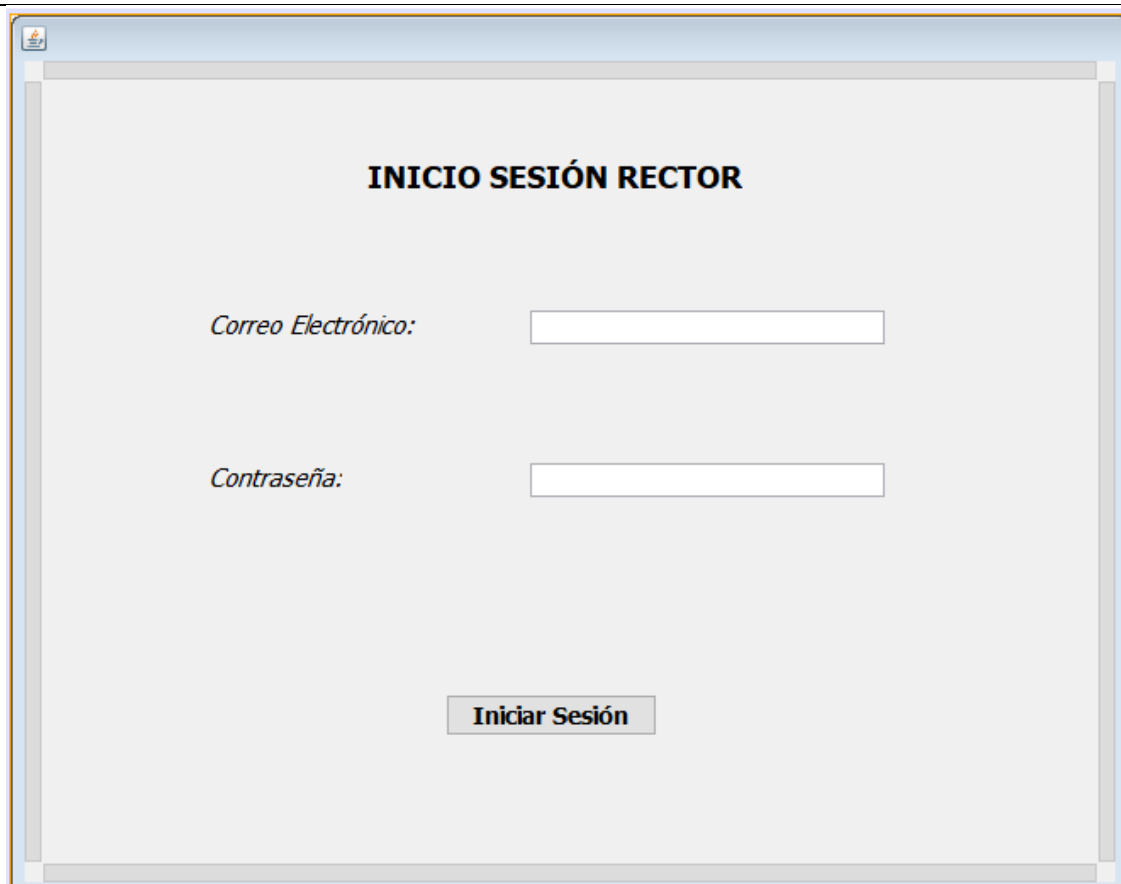
INICIO SESIÓN PROFESOR

Correo Electrónico:

Contraseña:

Iniciar Sesión

“VentanaIniciarSesionRector” :




INICIO SESIÓN RECTOR

Correo Electrónico:

Contraseña:


Iniciar Sesión

“VentanaGestionActividades:”



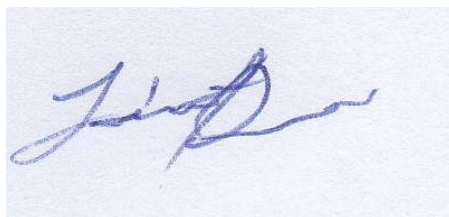
ACTIVIDADES

Titulo	Descripción

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

2.
3.
4.
RESULTADO(S) OBTENIDO(S): Realizar procesos de investigación sobre los cambios importantes de Java Entender las aplicaciones de codificación de las nuevas características en base a la programación genérica y expresiones regulares. Entender las funcionalidades adicionales de Java.
CONCLUSIONES: Aprenden a trabajar en grupo dentro de plazos de tiempo establecidos, manejando el lenguaje de programación de Java.
RECOMENDACIONES: Realizar el trabajo dentro del tiempo establecido.

Nombre de estudiante: Lisseth Reinoso



Firma de estudiante: