
	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.



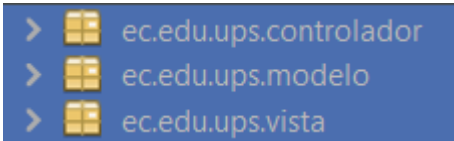
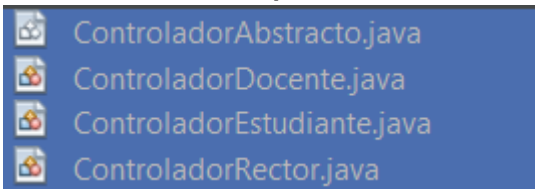
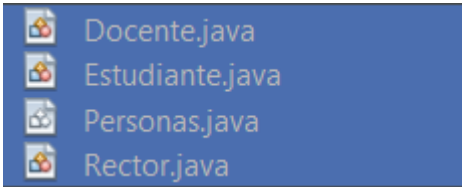
RECOMENDACIONES:


Realizar el trabajo dentro del tiempo establecido.

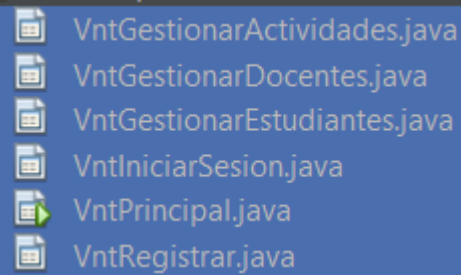
Docente / Técnico Docente: _____

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: Computación		ASIGNATURA: Programación Aplicada	
NRO. PRÁCTICA:	1	TÍTULO PRÁCTICA: Examen Practico Programación	
OBJETIVO ALCANZADO: <ul style="list-style-type: none"> • Identificar los cambios importantes de Java • Diseñar e Implementar expresiones regulares • Entender la cada uno de las características nuevas en Java 			
ACTIVIDADES DESARROLLADAS			
<ol style="list-style-type: none"> 1. Creamos un Proyecto en el NetBeans con el nombre de Examen_Interciclo  2. Creamos los paquetes para crear las clases y empezar con el proyecto nuevo creado, creamos los paquetes controladores, modelo y vista.  3. Dentro de cada paquete creamos las clases respectivas <ol style="list-style-type: none"> a. En el paquete controlador creamos las clases controladorAbstracto que será la clase padre de las clases controladorRector, controladorDocente, ControladorEstudiante, así hacemos uso de lo que es programación genérica, los controladores heredaran los atributos y métodos de su clase padre.  a. En el paquete modelo creamos la clase abstracta personas, que heredara sus atributos a sus clases hijas, que son: Rector, Docente, Estudiante.  b. En el paquete interfaz se encontrará la parte de las ventanas para demostrar el funcionamiento del programa. 			

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



2. En cada clase que se encuentra en el paquete modelo creamos todos los métodos que vamos a usar como los getters y setter, equals y hashCode, el controlador de la clase
También el código de las clases del paquete modelo se encuentra en el siguiente link del repositorio en GitHub:

https://github.com/Bryambepz/ExamenInterciclo_Practica/tree/master/src/ec/edu/ups/modelo

Personas

```
public abstract class Personas implements Serializable{
//    private String cedula, nombre, apellido, telefono, correo, contrasenia;
    private String cedula;
    private String nombre;
    private String apellido;
    private int edad;
    private String telefono;
    private String correo;
    private String contrasenia;

    public Personas() {}


    public Personas(String cedula, String nombre, String apellido, int edad, String
telefono, String correo, String contrasenia) {
        this.cedula = cedula;
        this.nombre = nombre;
        this.apellido = apellido;
        this.edad = edad;
        this.telefono = telefono;
        this.correo = correo;
        this.contrasenia = contrasenia;
    }

    public String getCedula() {
        return cedula;
    }

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

    public String getNombre() {
        return nombre;
    }

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

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

```

}

public String getApellido() {
    return apellido;
}

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

public int getEdad() {
    return edad;
}

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

public String getTelefono() {
    return telefono;
}

public void setTelefono(String telefono) {
    this.telefono = telefono;
}

public String getCorreo() {
    return correo;
}

public void setCorreo(String correo) {
    this.correo = correo;
}


public String getContrasenia() {
    return contrasenia;
}

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

@Override
public int hashCode() {
    int hash = 7;
    hash = 79 * hash + Objects.hashCode(this.cedula);
    hash = 79 * hash + Objects.hashCode(this.correo);
    return hash;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }

```

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

```

    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Personas other = (Personas) obj;
    if (!Objects.equals(this.cedula, other.cedula)) {
        return false;
    }
    if (!Objects.equals(this.correo, other.correo)) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "Rector - Docente - Estudiante" + "cedula=" + cedula + ", nombre=" +
nombre + ", apellido=" + apellido + ", edad=" + edad + ", telefono=" + telefono + ",
correo=" + correo + ", contrasenia=" + contrasenia + '}';
}

}

```

3. Las siguientes clases son las subclases de Personas

a. Rector

```

public class Rector extends Personas implements Serializable{
    private String UnidadEducativa;


    //    private List<Estudiante> listaEstudiante;
    private List<Docente> listaDocentes;

    public Rector() {
        listaDocentes = new ArrayList<>();
        //    listaEstudiante = new ArrayList<>();
    }

    public Rector(String UnidadEducativa, List<Docente> listaDocentes, String cedula,
String nombre, String apellido, int edad, String telefono, String correo, String
contrasenia) {
        super(cedula, nombre, apellido, edad, telefono, correo, contrasenia);
        this.UnidadEducativa = UnidadEducativa;
        //    this.listaEstudiante = listaEstudiante;
        this.listaDocentes = listaDocentes;
        listaDocentes = new ArrayList<>();
        //    listaEstudiante = new ArrayList<>();
    }

    public Rector(String UnidadEducativa, String cedula, String nombre, String
apellido, int edad, String telefono, String correo, String contrasenia) {
        super(cedula, nombre, apellido, edad, telefono, correo, contrasenia);
        this.UnidadEducativa = UnidadEducativa;
        listaDocentes = new ArrayList<>();
    }
}

```


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

```
//      listaEstudiante = new ArrayList<>();
}

public String getUnidadEducativa() {
    return UnidadEducativa;
}

public void setUnidadEducativa(String UnidadEducativa) {
    this.UnidadEducativa = UnidadEducativa;
}

public List<Docente> getListaDocentes() {
    return listaDocentes;
}

public void setListaDocentes(List<Docente> listaDocentes) {
    this.listaDocentes = listaDocentes;
}

public boolean createDocente(Docente doc){
    return listaDocentes.add(doc);
}

@Override
public String toString() {
    return super.toString() + "\n-----=> Rector{" + "UnidadEducativa=" +
UnidadEducativa + ", listaDocentes=" + listaDocentes + '}';
}

}


b. Docente
public class Docente extends Personas implements Serializable{

    private String cursoAsignado;
    private List<Estudiante> listaEstudiantes = listaEstudiantes = new ArrayList<>();

    public Docente() {}

    public Docente(String cursoAcargo, List<Estudiante> listaEstudiantes, String
cedula, String nombre, String apellido, int edad, String telefono, String correo,
String contrasenia) {
        super(cedula, nombre, apellido, edad, telefono, correo, contrasenia);
        this.cursoAsignado = cursoAcargo;
        this.listaEstudiantes = listaEstudiantes;
    }

    public Docente(String cursoAsignado, String cedula, String nombre, String
apellido, int edad, String telefono, String correo, String contrasenia) {
        super(cedula, nombre, apellido, edad, telefono, correo, contrasenia);
        this.cursoAsignado = cursoAsignado;
    }
}
```

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

```

public String getCursoAcargo() {
    return cursoAsignado;
}

public void setCursoAcargo(String cursoAcargo) {
    this.cursoAsignado = cursoAcargo;
}

public List<Estudiante> getListaEstudiantes() {
    return listaEstudiantes;
}

public void setListaEstudiantes(List<Estudiante> listaEstudiantes) {
    this.listaEstudiantes = listaEstudiantes;
}


public boolean createEstudiante(Estudiante estudiante){
    return listaEstudiantes.add(estudiante);
}

@Override
public int hashCode() {
    int hash = 5;
    hash = 37 * hash + Objects.hashCode(this.cursoAsignado);
    return hash;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Docente other = (Docente) obj;
    if (!Objects.equals(this.cursoAsignado, other.cursoAsignado)) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return super.toString() + "\n-----=> Docente{" + "cursoAcargo=" +
cursoAsignado + ", listaEstudiantes=" + listaEstudiantes + '}';
}
}

```

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

c. Estudiante

```

public class Estudiante extends Personas implements Serializable{
    private String cedulaRepresentante;
    private String NombreRepresentante;
    private String ApellidoRepresentante;
    private String telefonoRepresentante;
    private int edadRepresentante;
    private String correoRepresentante;
    private String contraseniaRepresentante;
    private String curso;

    public Estudiante() {}

    public Estudiante(String cedulaRepresentante, String NombreRepresentante, String
ApellidoRepresentante, String telefonoRepresentante, int edadRepresentante, String
correoRepresentante, String contraseniaRepresentante, String curso, String cedula,
String nombre, String apellido, int edad, String telefono, String correo, String
contrasenia) {
        super(cedula, nombre, apellido, edad, telefono, correo, contrasenia);
        this.cedulaRepresentante = cedulaRepresentante;
        this.NombreRepresentante = NombreRepresentante;
        this.ApellidoRepresentante = ApellidoRepresentante;
        this.telefonoRepresentante = telefonoRepresentante;
        this.edadRepresentante = edadRepresentante;
        this.correoRepresentante = correoRepresentante;
        this.contraseniaRepresentante = contraseniaRepresentante;
        this.curso = curso;
    }

    public Estudiante(String cedula, String nombre, String apellido, int edad, String
telefono, String correo, String contrasenia) {
        super(cedula, nombre, apellido, edad, telefono, correo, contrasenia);
    }

    public String getCedulaRepresentante() {
        return cedulaRepresentante;
    }


    public void setCedulaRepresentante(String cedulaRepresentante) {
        this.cedulaRepresentante = cedulaRepresentante;
    }

    public String getNombreRepresentante() {
        return NombreRepresentante;
    }

    public void setNombreRepresentante(String NombreRepresentante) {
        this.NombreRepresentante = NombreRepresentante;
    }

    public String getApellidoRepresentante() {
        return ApellidoRepresentante;
    }

```

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

```

public void setApellidoRepresentante(String ApellidoRepresentante) {
    this.ApellidoRepresentante = ApellidoRepresentante;
}

public String getTelefonoRepresentante() {
    return telefonoRepresentante;
}

public void setTelefonoRepresentante(String telefonoRepresentante) {
    this.telefonoRepresentante = telefonoRepresentante;
}

public int getEdadRepresentante() {
    return edadRepresentante;
}

public void setEdadRepresentante(int edadRepresentante) {
    this.edadRepresentante = edadRepresentante;
}

public String getCorreoRepresentante() {
    return correoRepresentante;
}

public void setCorreoRepresentante(String correoRepresentante) {
    this.correoRepresentante = correoRepresentante;
}

public String getContraseniaRepresentante() {
    return contraseniaRepresentante;
}

public void setContraseniaRepresentante(String contraseniaRepresentante) {
    this.contraseniaRepresentante = contraseniaRepresentante;
}

public String getCurso() {
    return curso;
}

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

@Override
public int hashCode() {
    int hash = 7;
    hash = 43 * hash + Objects.hashCode(this.cedulaRepresentante);
    hash = 43 * hash + Objects.hashCode(this.correoRepresentante);
    return hash;
}

@Override
public boolean equals(Object obj) {

```

```

    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Estudiante other = (Estudiante) obj;
    if (!Objects.equals(this.cedulaRepresentante, other.cedulaRepresentante)) {
        return false;
    }
    if (!Objects.equals(this.correoRepresentante, other.correoRepresentante)) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return super.toString() + "\n-----=> Estudiante{" + "cedulaRepresentante="
+ cedulaRepresentante + ", NombreRepresentante=" + NombreRepresentante + ",
ApellidoRepresentante=" + ApellidoRepresentante + ", telefonoRepresentante=" +
telefonoRepresentante + ", edadRepresentante=" + edadRepresentante + ",
correoRepresentante=" + correoRepresentante + ", contraseniaRepresentante=" +
contraseniaRepresentante + ", curso=" + curso + '}';
}

}

```

3. En las siguientes clases que están ubicadas en el paquete controlador creamos las clases ControladorAbstracto, ControladorRector, ControladorDocente, ControladorEstudiante, la clase controladorAbstracto será la clase padre y el resto sus clases hijas, se usara programación genérica para la reutilización de código, también en los métodos se aplicó streams y se uso archivos objetos para guardar la información de los docentes, rector y estudiantes que fueron registrados, en los controladores docente y rector se aplicó el patrón singleton.

El código de los controladores se encuentra en el siguiente link del GitHub:

https://github.com/Bryambepz/ExamenInterciclo_Practica/tree/master/src/ec/edu/ups/controlador

ControladorAbstracto

```

public abstract class ControladorAbstracto<T> {
    private List<T> listaObjetos = new ArrayList<>();

    public ControladorAbstracto() {
        listaObjetos = new ArrayList<>();
    }

    public List<T> cargarDatos(String ruta) throws FileNotFoundException,
IOException, ClassNotFoundException{
        FileInputStream archivo = new FileInputStream(ruta);
        ObjectInputStream datos = new ObjectInputStream(archivo);
        return listaObjetos = (List<T>) datos.readObject();
    }
    //      System.out.println(listaObjetos);
}

```

```

    }

    public void guardardatos(String ruta) throws FileNotFoundException, IOException{
        FileOutputStream archivo = new FileOutputStream(ruta);
        ObjectOutputStream datos = new ObjectOutputStream(archivo);
        datos.writeObject(listaObjetos);
    }

    public boolean create(T objeto){
        return listaObjetos.add(objeto);
    }

    public T read(String stringBuscar){
        return listaObjetos.stream().filter(buscar
stringBuscar.equals(buscar)).findFirst().get();
    }

    public boolean comprobarCorreo(String correo) {
        return listaObjetos.stream().filter(c -> correo.equals(c)).noneMatch(c ->
correo.equals(c));
    }

    public boolean comprobarCorreo(String correo, String ruta) {
        try {
            return cargarDatos(ruta).stream().filter(c
correo.equals(c)).noneMatch(c -> correo.equals(c));
        } catch (IOException | ClassNotFoundException ex) {
            Logger.getLogger(ContraladorAbstracto.class.getName()).log(Level.SEVERE,
null, ex);
        }
        return false;
    }

    public List<T> findAll(){
        return listaObjetos;
    }
    public List<T> getListaObjetos() {
        return listaObjetos;
    }

    public void setListaObjetos(List<T> listaObjetos) {
        this.listaObjetos = listaObjetos;
    }


    }

    a. ContraladorRector
    public class ContraladorRector extends ContraladorAbstracto<Personas> {

        private static ContraladorRector instancia;
        private static Rector obtenerSesion;

        private ContraladorRector() {
            // super();

```

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

```

}

public boolean iniciarSesion(String correo, String contrasenia, String ruta) {
    var iniciar = true;
    try {
        iniciar = cargarDatos(ruta).stream().filter(c
        correo.equals(c.getCorreo()) && contrasenia.equals(c.getContrasenia())
        .anyMatch(i -> correo.equals(i.getCorreo()) &&
        contrasenia.equals(i.getContrasenia()));
    } catch (IOException | ClassNotFoundException ex) {
        Logger.getLogger(ContraladorRector.class.getName()).log(Level.SEVERE,
null, ex);
    }
    if (iniciar) {
        obtenerSesion = (Rector) getListObjetos().stream().filter(c
        correo.equals(c.getCorreo())
        contrasenia.equals(c.getContrasenia()).findFirst().get();
        return true;
    }
    return false;
}

public static ContraladorRector getInstance() {
    if (instancia == null) {
        instancia = new ContraladorRector();
    }
    return instancia;
}

public static Rector obtenerSesionIniciada() {
    if (obtenerSesion == null) {
        obtenerSesion = new Rector();
    }
    return obtenerSesion;
}

// public boolean comprobarCorreo(String correo) {
//     return getListObjetos().stream().filter(c
//     correo.equals(c.getCorreo()).noneMatch(c -> correo.equals(c.getCorreo()));
// }

}


b. ContraladorDocente
public class ContraladorDocente extends ContraladorAbstracto<Docente>{
    private Pattern patronlogico;
    private Matcher corpus;
    private static ContraladorDocente instancia;
    private static Docente obtener;

    private ContraladorDocente() {super();}

}

public boolean iniciarSesion(String correo, String contrasenia, String ruta) {

```

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

```

var iniciar = true;
try {
    iniciar = cargarDatos(ruta).stream().filter(a
correo.equals(a.getCorreo()) && contrasenia.equals(a.getContrasenia())) ->
    .anyMatch(b -> correo.equals(b.getCorreo())) &&
contrasenia.equals(b.getContrasenia()));
} catch (IOException | ClassNotFoundException ex) {
    Logger.getLogger(ContraladorDocente.class.getName()).log(Level.SEVERE,
null, ex);
}
if (iniciar) {
    obtener = getListObjetos().stream().filter(c ->
correo.equals(c.getCorreo()) &&
contrasenia.equals(c.getContrasenia())).findFirst().get();
    return true;
}
return false;
}

public static ContraladorDocente getInstance() {
    if (instancia == null) {
        instancia = new ContraladorDocente();
    }
    return instancia;
}

public static Docente obtenerSesionIniciada() {
    if (obtener == null) {
        obtener = new Docente();
    }
    return obtener;
}


public void ingresarActividad(String regex){
    patronlogico = Pattern.compile(regex);
}

public List<String> buscarCorpus(String url){
    List<String> r = new ArrayList<>();
    corpus = patronlogico.matcher(url);
    while (corpus.find()) {
        r.add(url);
    }
    return r;
}

public Pattern getPatronlogico() {
    return patronlogico;
}

public void setPatronlogico(Pattern patronlogico) {
    this.patronlogico = patronlogico;
}

```


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

```

public Matcher getCorpus() {
    return corpus;
}

public void setCorpus(Matcher corpus) {
    this.corpus = corpus;
}

}

```

c. ControladorEstudiante

```

public class ControladorEstudiante extends ControladorAbstracto<Personas>{

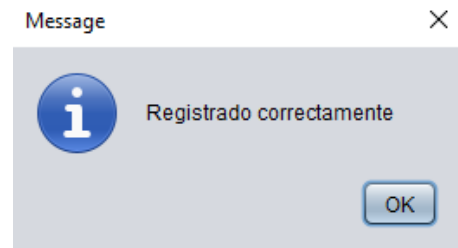
    public ControladorEstudiante() {
        super();
    }


}

```

4. Funcionamiento de la Aplicación

- Ventana Registrar:** El objetivo de esta ventana es registrar al rector de la institución a la que está a cargo, para los campos de cedula y teléfono se validó aplicando lo aprendido en clases expresiones regulares, en el caso de que este mal ingresado esos datos aparecerá un cuadro de texto informando que los datos de algún campo no fueron ingresados correctamente de estar bien se cerrara la ventana y mostrara un mensaje informando que se registró exitosamente.

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

- b. Ventana Iniciar Sesión: para esta ventana se trato de implementar lo que es el patrón singleton, validara si aun no hay docentes registrados, si es el caso de aun no haber docentes ya registrados el rector se podrá registrar con normalidad, si el sistema detecta que ya hay docentes registrados se activaran dos opciones una para que inicie sesión solo el rector y otra opción para que inicie sesión solo el docente.

- I. No se encuentran docentes registrados así que por el momento solo el rector puede iniciar sesión y proceder a gestionar a los docentes de la institución.




The screenshot shows a window titled "Iniciar Sesión" with a close button (X) in the top right corner. Inside the window, there are two input fields: "Correo" with the text "rzarate@ups" and "Contraseña" with masked characters "*****". Below these fields are two buttons: "INICIAR SESION" and "CANCELAR".

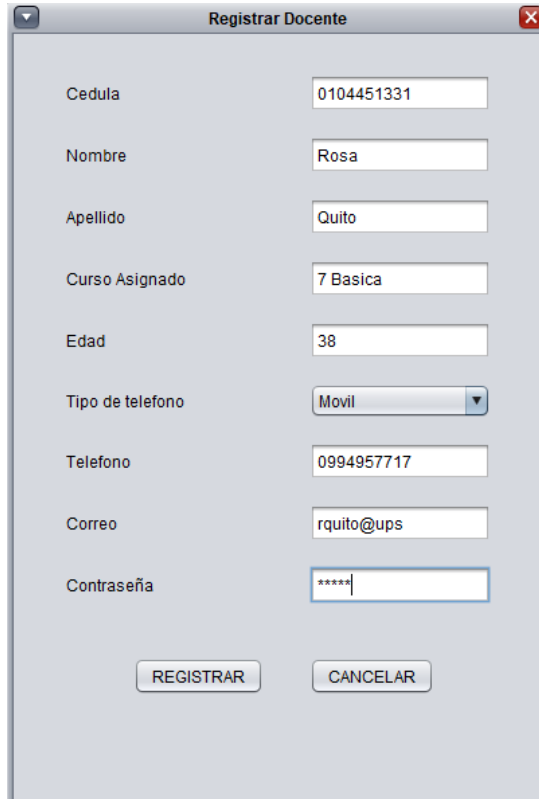
- II. Estas opciones se activarán cuando ya existan docentes registrados por el rectorñ




The screenshot shows a window titled "Iniciar Sesión" with a close button (X) in the top right corner. Inside the window, there are two radio buttons: "Es Rector" (unselected) and "Es Docente" (selected). To the right of these radio buttons are two input fields: "Correo" with the text "rquito@ups" and "Contraseña" with masked characters "*****". Below these fields are two buttons: "INICIAR SESION" and "CANCELAR".

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


- c. **Ventana Registrar Docente:** una vez que el rector inicio sesión, se habilitara un menú para gestionar a los docentes, es muy parecida a la ventana de registrar rector, los campos de la cedula y teléfono se validaran usando expresiones regulares.



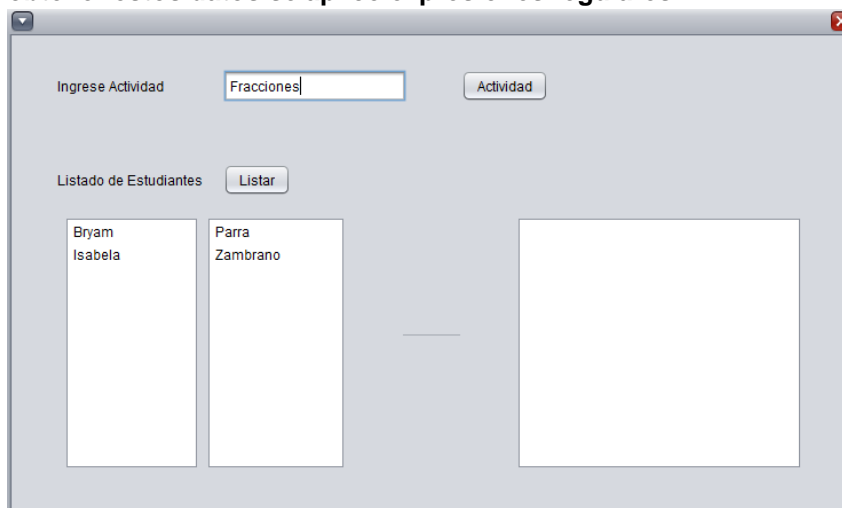
Ahora ya puede cerrar la sesión el rector y permitir que los docentes ingresen y gestionen a sus alumnos, cuando cierra sesión se ocultara la opción de gestionar, cuando el docente inicie sesión se visualizara un nuevo menú, para gestionar a los estudiantes y las actividades.

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

- d. **Ventana Gestionar Estudiantes:** En esta ventana podrá ingresar a los alumnos, con todos sus datos, los de ellos y de su representante legal, al igual que las ventanas registrar docentes y rector, esta también validará la cedula y el teléfono ingresado.




- e. **Ventana Gestionar Actividades:** En esta ventana el docente podrá listar a los alumnos que registro y podrá ingresar la actividad que planeo para el día escolar, al hacer clic en actividad buscara todas las aplicaciones que estén relacionadas con el tema de fracciones, para obtener estos datos se aplico expresiones regulares.



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.

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

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: Bryam Parra



Firma de estudiante: _____