Tecnológico de Costa Rica	Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández	II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos	Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.	

A continuación encontrará cuatro casos que deben resolverse bajo la misma modalidad de ejercicios que se han practicado en clase.

Cada equipo de trabajo deberá:

- Anotar sus nombres y números de carné en la tabla que se presenta abajo.
- Para cada caso deben resolver los puntos (1..5) y completar la tabla asociada a cada caso, incorporándo redacción formal donde se solicita la justificación y proporcionando imágenes de modelos y código **ABSOLUTAMENTE LEGIBLES**.
- Este documento debe renombrarse bajo el nombre IIExamenGrupoNNMM y entregarlo en formato **PDF (sin excepción)**.
- El equipo creará un proyecto de programación Java (preferiblemente usando IDE Netbeans) denominado **IIExamenGrupoNNMM (**donde N es 40 si es de la sede CTLSJ o 01 si es de la Sede Cartago, y MM es el número de grupo asignado de trabajo). Construyan en el proyecto cuatro carpetas, cada una con el nombre **CasoX_Patron** (donde X es el número de caso a resolver y Patron es el nombre del patrón con el que proponen la solución). Ejemplo **Caso7_Patron.**

Al tec Digital deberá subir un archivo comprimido llamado **IIExamen_GrupoNNMM** siguiendo la misma nomenclatura del proyecto programado que contiene el proyecto Java y el enunciado del examen en formato **PDF**.

Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.

Caso 1

Los objetos FileInputStream típicamente representan archivos de texto accedidos en orden **secuencial**, byte a byte. Con FileInputStream, se puede acceder a un byte, varios bytes o el archivo completo.

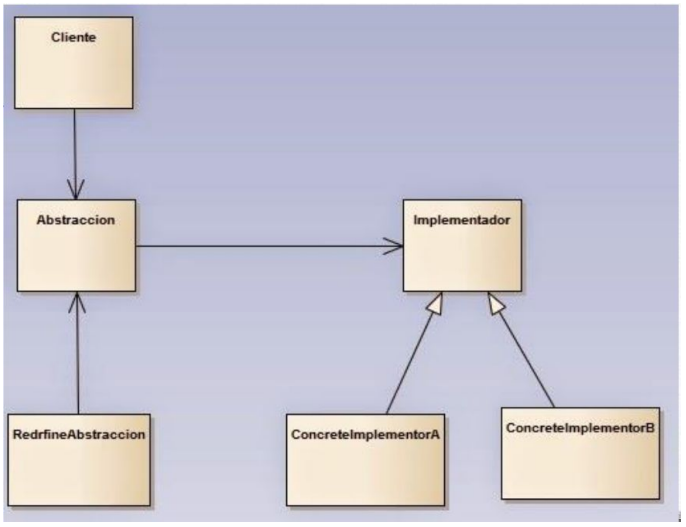
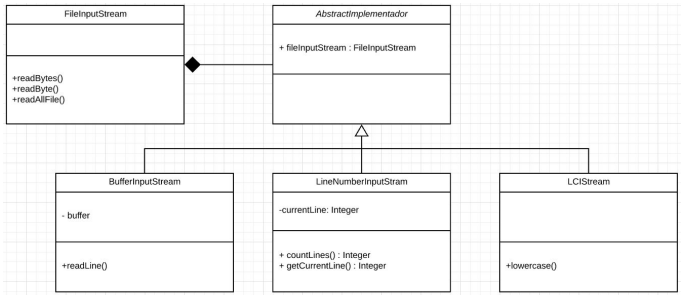
Si se revisa el API IO de Java, la cantidad de objetos especializados en el manejo de archivos de texto es enorme. Hay objetos BufferedInputStream que incluye en su funcionamiento un buffer de datos para un mejor rendimiento y añade una funcionalidad de leer una línea, *readLine()*, para leer un renglón a la vez.

Adicionalmente, existe otro tipo de objeto LineNumberInputStream que añade la funcionalidad de contar la cantidad de líneas contenidas en el archivo y hasta puede indicar cual es el número de línea actualmente accedido en un momento del tiempo.

Lo que debe hacer en este caso:

1. ¿Cuál patrón considera usted que fue utilizado para poder proveer todos estos tipos distintos de manejadores de archivos de texto? **(2 puntos)**
2. Justifique formalmente la selección del patrón para este caso. Aporte del diagrama original de UML correspondiente al patrón. **(3 puntos)**
3. Siguiendo el modelo del patrón, construya detalladamente el modelo de UML que da solución a este planteamiento. Añada al modelo la posibilidad de ofrecer además un tipo de Stream asociado al manejador de archivos de texto que transforme el contenido del archivo en minúsculas ! Para el manejador de archivos que transforma a minúsculas, llame al objeto LCIStream, esto para no interferir con las propias del API si lo programa en Java. **(5 puntos)**
4. Programe el funcionamiento del modelo del patrón seleccionado según su propuesta. Para efectos de la programación del caso, construya para representar las clases FileInputStream, BufferedInputStream y LineNumberInputStream bajo los nombres FIStream, BIStream y LNIStream respectivamente con las funcionalidades necesarias para realizar las tareas de leer el contenido de un archivo de texto según lo realiza el API de Java de acuerdo a lo expuesto. **(10 puntos)**
5. Evidencias del código producido de los elementos significativos del patrón y su programa de prueba, así como screenshots de funcionamiento. **(5 puntos)**

Tecnológico de Costa Rica	Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández	II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos	Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.	

Respuesta Caso 1	**Patrón a utilizar**	Bridge
	Tipo de Patrón	Estructural
Justificación de uso del patrón	A la clase ya existente FileInputStream se le agregan nuevas funcionalidades a través de los objetos implementadores BufferedInputStream y LineNumberInputStream.	
Diagrama de UML asociado		
Modelo en UML que soluciona el problema		

Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software Prof.Ing.Ericka Solano Fernández II Semestre, 2018 II Examen Parcial Valor 10% - 100puntos Para ser desarrollado en parejas Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.

Evidencias de código significativo en la implementación del patrón

```
public class FIStream {

    private FileInputStream fis;
    private File file;

    public FIStream(String filePath) throws FileNotFoundException{
        fis = new FileInputStream(filePath);
        file = new File(filePath);
    }

    public void readBytes(byte[] b) throws IOException{
        int c = fis.read(b);
        System.out.print((char)c);

    }

    public void readByte() throws IOException{
        int b = fis.read();
        System.out.print((char)b);
    }

    public String readAllFile() throws IOException{
        byte[] data = new byte[(int) file.length()];
        fis.read(data);
        fis.close();

        String str = new String(data, "UTF-8");
        return str;

    }
```

```
public abstract class AbstractImplementador {

    protected FIStream fileInputStream;
}
```

```
public class BIStream extends AbstractImplementador{

    BufferedReader buffer;

    public BIStream(String filePath) throws FileNotFoundException{
        fileInputStream = new FIStream(filePath);
        buffer = new BufferedReader( new FileReader (filePath));
    }

    public void readLine() throws IOException{

        System.out.println(buffer.readLine());
    }


}
```

Tecnológico de Costa Rica	Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández	II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos	Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.	

```java
public class LNIStream extends AbstractImplementador {

    private int currentLine;
    private LineNumberReader lnr;

    public LNIStream(String filePath) throws FileNotFoundException{
        currentLine = 0;
        fileInputStream = new FIStream(filePath);
        lnr = new LineNumberReader(new FileReader(filePath));
    }

    public int countLines() throws IOException{
        while (lnr.skip(Long.MAX_VALUE) > 0){}
        return lnr.getLineNumber() + 1;
    }

    public int getCurrentLine(){
        return lnr.getLineNumber();
    }
}
```

Screeshoots de funcionamiento

```java
public static void main(String[] args)
{

    try {
        System.out.println("Declarando una instancia de Buffered Input Stream----------");
        BIStream a = new BIStream("C:\\Users\\nicol\\Documents\\prueba.txt");
        System.out.println("Accede a la función readLine de FileInputStream base:------");
        System.out.println(a.fileInputStream.readAllFile());
        System.out.println("Accede a la nueva función que aporta Bufferen Input Stream---");
        a.readLine();
        System.out.println("Accede a la nueva función que aporta LCIStream---");
        LCIStream l = new LCIStream("C:\\Users\\nicol\\Documents\\prueba.txt");
        l.lowercase();
        System.out.println("Accede a la nueva función que aporta LNIStream---");
        LNIStream lne = new LNIStream("C:\\Users\\nicol\\Documents\\prueba.txt");
        int largo = lne.countLines();
        System.out.println(largo);


    } catch (FileNotFoundException ex) {
        Logger.getLogger(Context.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(Context.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

El resultado:

```
Declarando una instancia de Buffered Input Stream----------
Accede a la función readLine de FileInputStream base:------
Esto es
una prueba
para el caso
2 del examen.
Accede a la nueva función que aporta Bufferen Input Stream---
Esto es
Accede a la nueva función que aporta LCIStream---
esto es
una prueba
para el caso
2 del examen.
Accede a la nueva función que aporta LNIStream---
4
BUILD SUCCESSFUL (total time: 1 second)
```

Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software

Prof.Ing.Ericka Solano Fernández II Semestre, 2018 II Examen Parcial

Valor 10% - 100puntos Para ser desarrollado en parejas

Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.

Tecnológico de Costa Rica	Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández	II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos	Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.	

Caso 2

En el desarrollo de un proyecto de curso, los estudiantes deberán manejar el acceso de los usuarios al mismo a través de un mecanismo que permita la autenticación del usuario a través de sus credenciales (login y contraseña). Además, se debe y permitir el registro de nuevos usuarios que forman parte de un equipo de trabajo y que requerirán acceder al sistema, por lo que el perfil del nuevo usuario debe completarse con información relevante, como su nombre con dos apellidos, fecha de nacimiento, datos de contacto como dirección física, correo electrónico, número del móvil como mínimo, eventualmente podrían agregarse otros datos a la sección de contactos como código postal, cuenta de alguna red social, entre otros. Los datos de contacto deben tener el formato adecuado y por supuesto no se debe permitir el registro de usuarios menores de edad. Los dos apellidos del usuario son requeridos.

Como es evidente, se requerirá validar algunos de los campos que forman parte del perfil del usuario al momento de registro y además, posteriormente se deberá realizar la validación de las credenciales, sin mencionar que en el proyecto como tal se requiere realizar validaciones que van de acuerdo a la lógica de negocio de la aplicación en desarrollo.

Por esta razón, al momento de la conceptualización del proyecto, se decidió definir una interface que permita estandarizar el llamado a mecanismos de validación independientemente del elemento a revisar, de modo que cuando se requiera una clase en particular, pueda implementar este mecanismo de la forma que le corresponda, según su contexto por medio de clases especializadas que lleve a cabo esta tarea.

Esta interface establece que el método devolverá una lista de hileras que contiene los errores que pudieron ser detectados en el proceso de validación. La interfaz se ha definido de manera totalmente genérica dentro del proyecto, a fin de que pueda ser utilizada con cualquier elemento que requiera implementar mecanismos de validación. Se muestra a continuación la sintaxis de la interface definida:

```
public interface Validator<T> {

    List<String> validate(T info);
}
```

Se desea que la página o pantalla de registro de usuarios sea sometida al proceso de revisión de las distintas validaciones al momento de intentar registrar un nuevo usuario de modo que si se obtienen

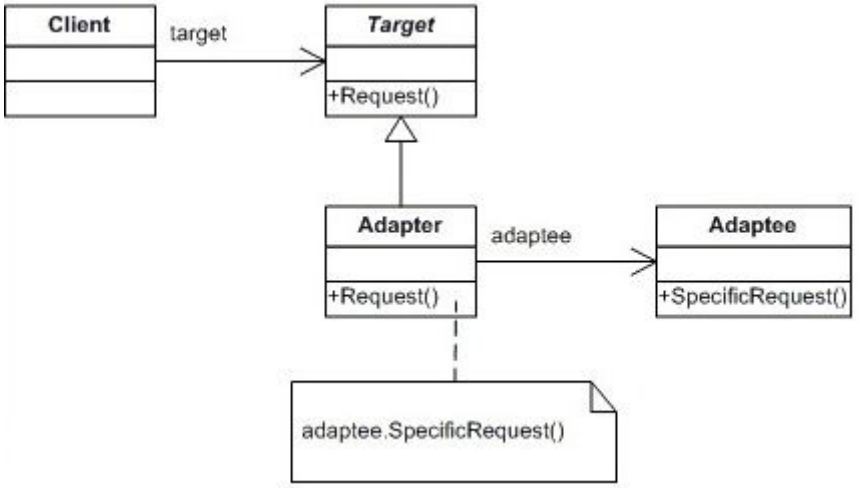
errores de distinta índole en el proceso sean desplegados en la pantalla y no se permita el registro para que el usuario corrija.

Lo que debe hacer en este caso:

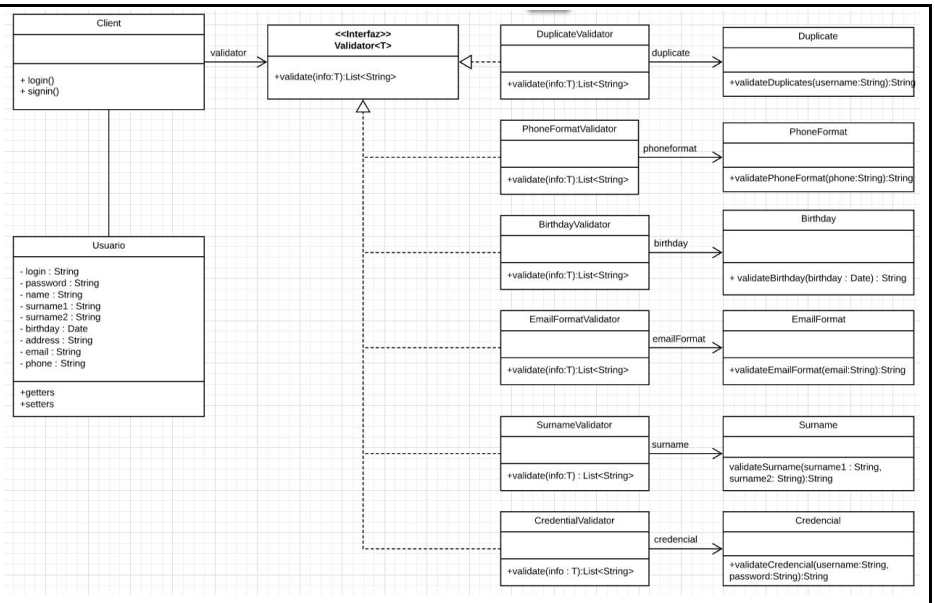
1. ¿Cuál patrón cree usted que el equipo de trabajo utilizó para aportar flexibilidad y uniformidad al llevar a cabo las validaciones de información? **(2 puntos)**
2. Justifique formalmente la selección del patrón para este caso. Aporte del diagrama original de UML correspondiente al patrón. **(3 puntos)**
3. Proponga el diagrama de UML que pudo haber implementado el equipo de trabajo e incorpore los elementos necesarios para ajustarse al requerimiento técnico que debe cumplir el formulario de registro de nuevos usuarios al sistema? **Justifique su respuesta**. **(5 puntos)**
4. Programe el funcionamiento del modelo del patrón seleccionado según su propuesta. Desarrolle una pequeña interfaz de usuario que permita el registro de un nuevo perfil de usuario de modo que al momento de intentar registrar un usuario, se solicite al patrón llevar a cabo las validaciones correspondientes mostrando la lista de errores que podrían generarse o bien permitiendo el registro exitoso del nuevo usuario. Recuerde que una validación final debe ser que no se permite la duplicidad de registros. **(10 puntos)**

Respuesta Caso 2	**Patrón a utilizar**	Adapter
	Tipo de Patrón	Estructural
Justificación de uso del patrón	Es adapter porque dice aportar flexibilidad y uniformidad al llevar a cabo las validaciones. El adapter sería implementar el validate y solo llamar al verdadero método de validación de una clase distinta, de esta manera se estandariza el validar porque solo se debe llamar a validate pasándole el dato.	

Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.

Diagrama de UML asociado	
Modelo en UML que soluciona el problema	Justificación: Las clases especializadas en realizar las validaciones son las que están a la derecha. Los métodos de validación de ellas tienen distintos nombres y reciben parámetros diferentes. Son las clases “adaptadas”. Está la interfaz Validator que estandariza todas las validaciones, es la clase “adaptador”. De modo que cuando se requiere una clase en particular, implementa este mecanismo de la forma que le corresponda obteniendo del usuario los atributos que necesitan los validadores reales, estas son las clases “target”. La clase Client accede a los validadores y utiliza la clase Usuario como modelo para el perfil de usuario. Si en un futuro se quiere agregar una nueva validación, solo tiene que implementar la interfaz Validator. UML:

Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software Prof.Ing.Ericka Solano Fernández II Semestre, 2018 II Examen Parcial Valor 10% - 100puntos Para ser desarrollado en parejas Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.

	
Evidencias de código significativo en la implementación del patrón	``` public interface Validator { public List<String> validate(Object T); } public class Duplicate { public Boolean validateDuplicates(Usuario user){ return RegisteredUsersList.getInstance().getUsers().contains(user); } } public class DuplicateValidator implements Validator{ @Override public List<String> validate(Object T) { Usuario usr = (Usuario) T; Duplicate dup = new Duplicate(); List<String> errors = new ArrayList<>(); if(dup.validateDuplicates(usr)){ errors.add("Error: Usuario ya existe"); } return errors; } } ```

Tecnológico de Costa Rica	Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández	II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos	Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.	

```
public class BirthdayValidator implements Validator{

    @Override
    public List<String> validate(Object T) {
        Date birthday = (Date) T;
        List<String> errors = new ArrayList<>();
        BirthdayFormat bf = new BirthdayFormat();
        String error = bf.validateBirthday(birthday);
        if(!error.isEmpty()){
            errors.add(error);
        }
        return errors;
    }



}

public class BirthdayFormat {

    public String validateBirthday(Date birthday){

        Date today = new Date();
        today.setYear(2001);
        if (birthday.after(today)){
            return "El usuario debe ser mayor de edad";
        }
        return "";
    }

}

public class Credential {

    public boolean validateCredencial( String username, String password){

        List<Usuario> users = RegisteredUsersList.getInstance().getUsers();
        for (Usuario usr : users){
            if(usr.getLogin() == username && usr.getPassword() == password)
                return true;
        }
        return false;
    }
}
```

Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.

```
public class CredentialValidator implements Validator {

    @Override
    public List<String> validate(Object T) {
        Usuario usr = (Usuario) T;
        List<String> errors = new ArrayList<>();
        Credential c = new Credential();
        if (!c.validateCredencial(usr.getLogin(), usr.getPassword()))
            errors.add("Error: credenciales incorrectas");
        return errors;
    }



}

public class EmailFormat {

    public String validateEmail(String email){
        if(!email.contains("@")){
            return "Error: formato de correo incorrecto";
        }
        return "";
    }
}

 public class EmailFormatValidator implements Validator{

     @Override
     public List<String> validate(Object T) {
         String correo = (String) T;
         EmailFormat ef = new EmailFormat();
         List<String> errors = new ArrayList<>();
         String error = ef.validateEmail(correo);
         if(!error.isEmpty()){
             errors.add(error);
         }
         return errors;
     }


 }
```

Tecnológico de Costa Rica	Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández	II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos	Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.	

```
public class PhoneFormat {

    public String validatePhoneFormat(String phone){

        try{
            int num = Integer.parseInt(phone);
        } catch(NumberFormatException e){
            return "Error: el teléfono es numérico";
        }
        if(phone.contains("-"))
            return "Error: no escriba un guión en el número";
        if(phone.length() != 8){
            return "Error: el número debe contener 8 dígitos";
        }
        return "";
    }

}

public class PhoneFormatValidator implements Validator{

    @Override
    public List<String> validate(Object T) {
        String phone = (String) T;

        List<String> errors = new ArrayList<>();
        PhoneFormat pf = new PhoneFormat();
        String error  = pf.validatePhoneFormat(phone);
        if(!error.isEmpty()){
            errors.add(error);
        }
        return errors;
    }

public class Surname {

    public String validateSurnames(Usuario usr){
        if (usr.getSurname1().isEmpty() || usr.getSurname2().isEmpty()){
            return "Error: usuario debe tener ambos apellidos";
        }
        return "";
    }
}
```

Tecnológico de Costa Rica	Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández	II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos	Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.	

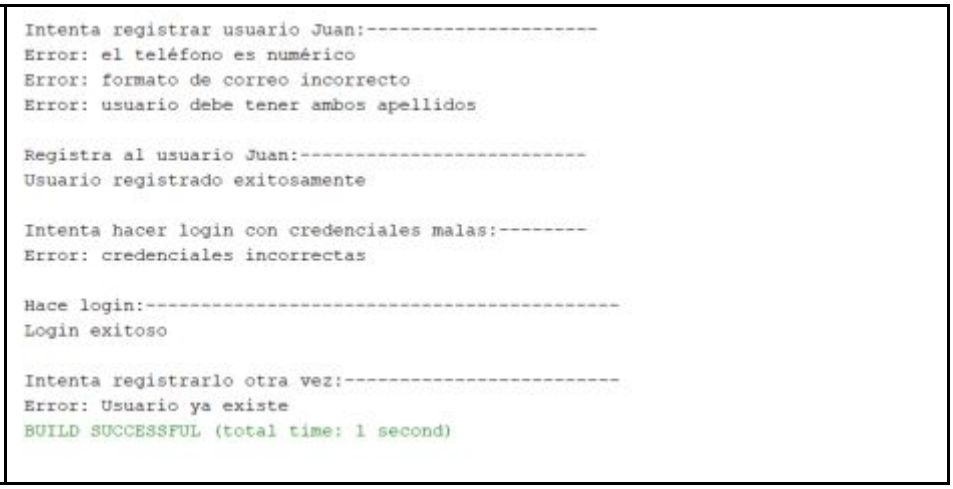
```java
public class SurnameValidator implements Validator{

    @Override
    public List<String> validate(Object T) {
        Usuario usr = (Usuario) T;
        Surname s = new Surname();
        List<String> errors = new ArrayList<>();
        String error = s.validateSurnames(usr);
        if(!error.isEmpty()){
            errors.add(error);
        }
        return errors;
    }

}
```

Screeshoots de funcionamiento

```java
public static void main(String[] args)
{
    Date date = new Date();
    Usuario usr1 = new Usuario();
    usr1.setName("Juan");
    usr1.setLogin("Juan123");
    usr1.setPassword("123");
    usr1.setSurname1("Flores");
    usr1.setBirthday(date);
    usr1.setAddress("Costa Rica");
    usr1.setEmail("juangmail.com");
    usr1.setPhone("aaa888");

    System.out.println("\nIntenta registrar usuario Juan:--------------------");
    signin(usr1);

    date.setYear(1990);

    usr1.setName("Juan");
    usr1.setLogin("Juan123");
    usr1.setPassword("123");
    usr1.setSurname1("Flores");
    usr1.setSurname2("Salazar");
    usr1.setBirthday(date);
    usr1.setAddress("Costa Rica");
    usr1.setEmail("juan@gmail.com");
    usr1.setPhone("88888888");

    System.out.println("\nRegistra al usuario Juan:-------------------------");
    signin(usr1);
    System.out.println("\nIntenta hacer login con credenciales malas:--------");
    login("Juan123", "12");
    System.out.println("\nHace login:----------------------------------------");
    login("Juan123", "123");
    System.out.println("\nIntenta registrarlo otra vez:----------------------");
    signin(usr1);


}
```

Resultado de la ejecución:

Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.

Tecnológico de Costa Rica	Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández	II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos	Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.	

Caso 3

Application Service Providing o ASP es un modelo de negocio que da soporte a medianas empresas con software empresarial complejo y altamente integrado. Este modelo implementa una técnica y una infraestructura organizativa que garantiza un alto grado de disponibilidad del sistema y la seguridad de los datos. Varios clientes comparten el uso de la infraestructura central y cuentan con un mecanismo de configuración para acceder a sus datos y sus diversas funcionalidades que han sido contratadas.

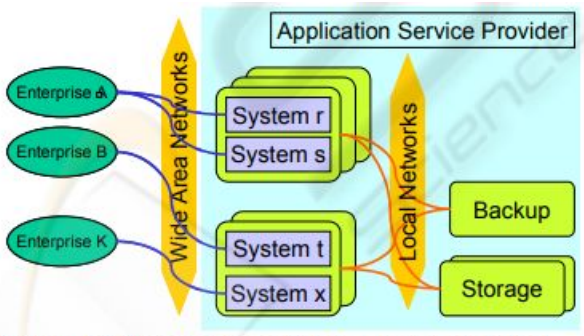


Figure 1: Customers and Components in ASP

La administración técnica de ASP debería ser soportado por un software de monitoreo que abarque una gran gama de componentes de software y hardware, esto porque un ASP normalmente es un sistema distribuido que tiene múltiples servidores, las aplicaciones son basadas en una arquitectura multicapa, tiene una gran exigencia de disponibilidad dado que múltiples usuarios pueden solicitar múltiples recursos en simultáneo, la apariencia de la aplicación puede variar dependiendo del cliente que ingrese, esto es, la misma aplicación puede tener distintas apariencias para ser manipulada por usuarios distintos, entre otros.

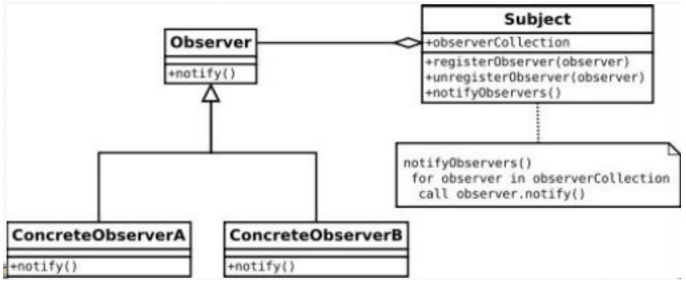
A menudo los monitores para componentes específicos son basados en técnicas no orientada a objetos, por ejemplo, parsing de archivos de log o accesos a información interna de ejecución que es llevada a cabo por los system-call de los sistemas operativos.
Entonces en ocasiones un monitor específico para un ASP requiere por ejemplo tener acceso a logs de estado que le sean provistos por un “FileMonitor”, y en otras necesita acceder a la tabla de procesos de un sistema operativo que pueden ser obtenidas por un “ProcessMonitor”.

Por otro lado podría requerirse distintos tipos de monitores, uno que esté monitoreando las bases de datos en la zona de almacenamiento (ORCLMonitor, DBaseMonitor), y en ocasiones se deberá conectar otro que rastree el ASP directamente (ASPMonitor).

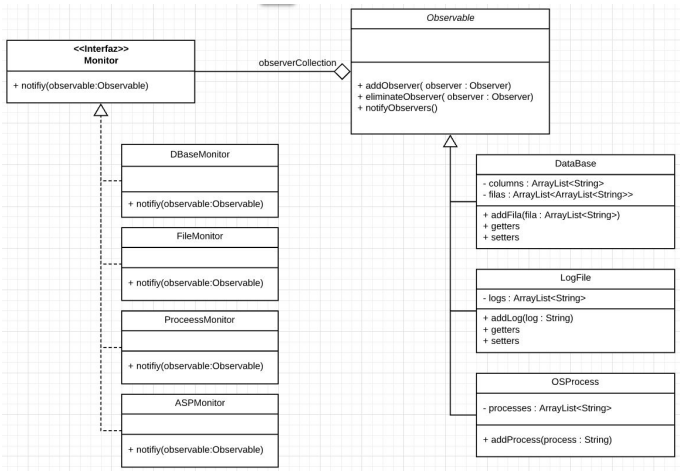
Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.

Lo que debe hacer en este caso:

1. ¿Cuál sería el patrón que se puede utilizar para implementar esta estrategia de monitoreo asociada a este tipo de aplicación como es el ASP? **(2 puntos)**
2. Justifique formalmente la selección del patrón para este caso. Aporte del diagrama original de UML correspondiente al patrón. **(3 puntos)**
3. Proponga el diagrama de UML que implementa la flexibilidad de poder brindar al ASP el uso de monitores distintos que puedan dedicarse a recuperar información de comportamiento de los componentes de hardware y software involucrados en el ASP? **Justifique su respuesta**. **(5 puntos)**
4. Programe el funcionamiento del modelo del patrón seleccionado según su propuesta. Suponga una instancia de un ASP con un configuración particular que debe llevar a cabo en un momento determinado operaciones de monitoreo de comportamiento de accesos a la base de datos y ver los archivos de log, y en otras ocasiones debe analizar el comportamiento de atención de procesos por parte del sistema operativo que le da soporte a la aplicación. **(10 puntos)**

Respuesta Caso 3	**Patrón a utilizar**	Observer
	Tipo de Patrón	Comportamiento
Justificación de uso del patrón	Observer porque lo que pide la pregunta es cómo se implementaría el sistema de monitoreo. Cada monitor es un observador y lo que debe ser monitoreado (acceso a la base de datos, archivos de log, procesos del sistema operativo) son los observables que pueden cambiar de estado y notificar a los monitores del cambio.	
Diagrama de UML asociado		
Modelo en UML que soluciona el problema	Justificación: La interface Monitor son los observadores mientras que la clase abstracta Observable son los observables que son los encargados de notificar a los monitores cuando su estado	

Tecnológico de Costa Rica	Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández	II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos	Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.	

	cambia. Cada monitor debe implementar su método que se ejecuta cuando ocurre una actualización. UML: 
Evidencias de código significativo en la implementación del patrón	Como se implementaron muchas clase observadoras y observables solo se adjuntará fotos de la clase ProcessMonitor y OSProcess, además de la interface Monitor y clase abstracta Observable Monitor `public interface Monitor` `{` `  public void notificar(Observable observable);` `}` Observable

Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.

```
public class Observable
{
   protected ArrayList<Monitor> observerCollection;

   public Observable()
   {
      observerCollection = new ArrayList<>();
   }

   public void addObserver(Monitor monitor)
   {
      observerCollection.add(monitor);
   }

   public void eliminateObserver(Monitor monitor)
   {
      observerCollection.remove(monitor);
   }
```

```
   public void notifyObservers()
   {
      observerCollection.forEach((monitor) ->
      {
         monitor.notificar(this);
      });
   }
}
```

ProcessMonitor

```
public class ProcessMonitor extends javax.swing.JFrame implements Monitor
{

   @Override
   public void notificar(Observable observable)
   {
      OSProcess process = (OSProcess) observable;
      String lista[] = new String[process.getProcesses().size()];
      int cont = 0;
      for(String str : process.getProcesses())
      {
         lista[cont] = str;
         cont++;
      }
      list.setListData(lista);
      list.repaint();
   }
```

OSProcess

Tecnológico de Costa Rica	Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández	II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos	Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.	

```
public class OSProcess extends Observable
{
   private ArrayList<String> processes;

   public OSProcess()
   {
      super();
      processes = new ArrayList<>();
   }

   public void addProcess(String process)
   {
      processes.add(process);
      super.notifyObservers();
   }

   /**
    * @return the processes
    */
   public ArrayList<String> getProcesses() {
      return processes;
   }
```

```
   /**
    * @param processes the processes to set
    */
   public void setProcesses(ArrayList<String> processes) {
      this.processes = processes;
      super.notifyObservers();
   }
}
```

Main

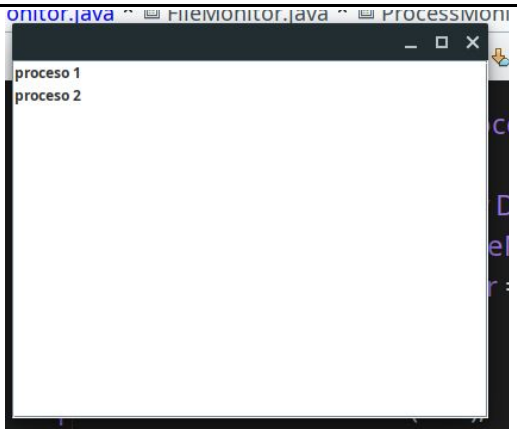
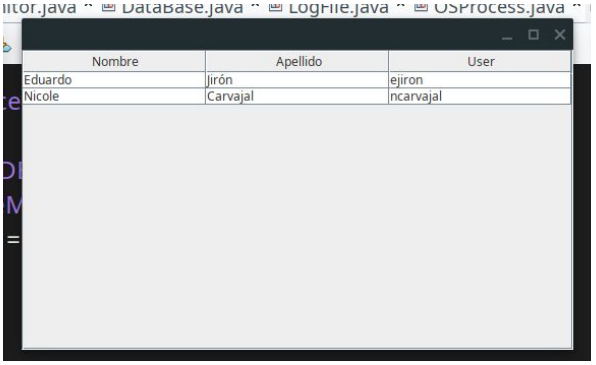
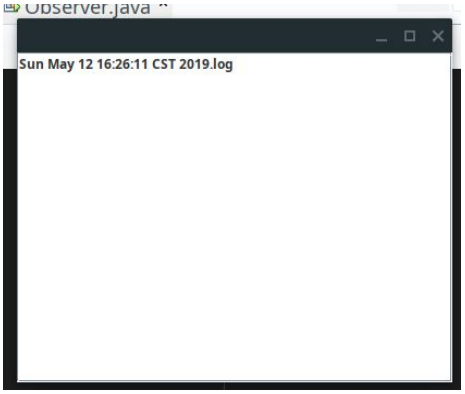
Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.

```
public class Observer
{
  public static void main(String[] args)
  {
    //Observables
    DataBase db = new DataBase();
    LogFile log = new LogFile();
    OSProcess process = new OSProcess();
    //Monitors
    DBaseMonitor dbMonitor = new DBaseMonitor();
    FileMonitor fileMonitor = new FileMonitor();
    ProcessMonitor processMonitor = new ProcessMonitor();
    dbMonitor.setVisible(true);
    fileMonitor.setVisible(true);
    processMonitor.setVisible(true);
    //Añadir los observer a los observables
    db.addObserver(dbMonitor);
    log.addObserver(fileMonitor);
    process.addObserver(processMonitor);
    //Añadir objetos en los observables
    //Agrego las columnas
    ArrayList<String> columns = new ArrayList<>();
```

```
    columns.add("Nombre");
    columns.add("Apellido");
    columns.add("User");
    db.setColumns(columns);
    //Agrego las filas
    ArrayList<String> fila = new ArrayList<>();
    fila.add("Eduardo");
    fila.add("Jirón");
    fila.add("ejiron");
    db.addFila(fila);
    ArrayList<String> fila2 = new ArrayList<>();
    fila2.add("Nicole");
    fila2.add("Carvajal");
    fila2.add("ncarvajal");
    db.addFila(fila2);
    //Agrego los logs
    Date date = Calendar.getInstance().getTime();
    String file = date.toString() + ".log";
    log.addLog(file);
    //Agrego los procesos
    String proceso = "proceso 1";
    String proceso2 = "proceso 2";
    process.addProcess(proceso);
```

```
    process.addProcess(proceso2);
  }
}
```

Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.

Screeshoots de funcionamiento	  

Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.

Caso 4

Suponga una casa de producción de contenido que contrata agentes vendedores para escribir contenido para la organización.

Para cada proyecto asignado a un proveedor, el departamento de RRHH proporciona al vendedor un contrato con una serie de cláusulas que contienen términos y condiciones y además un acuerdo de confidencialidad que debe aceptar antes de comenzar a trabajar.

El contenido de los acuerdos sigue siendo el mismo para todos los proveedores y un empleado de Recursos Humanos sólo debe completar el nombre del proveedor antes de enviar un acuerdo al proveedor.
Tanto el formato del contrato como del acuerdo de confidencialidad se encuentran en una base de datos remota distintas que deben ser accedida para solicitar el envío un nuevo contrato para un nuevo vendedor. El vendedor no será contratado si no ha firmado ambos documentos-

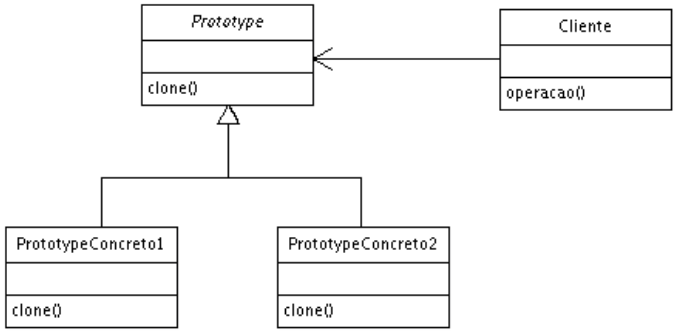
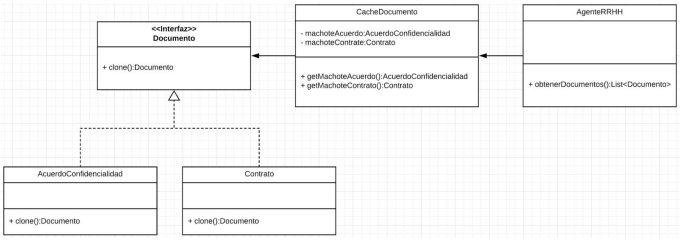
El contrato debe ser firmado tanto por el encargado de RRHH y el vendedor.

Lo que debe hacer en este caso:

1. ¿Cuál sería el patrón que se puede utilizar para obtener ambos documentos y contratar al vendedor? **(2 puntos)**
2. Justifique formalmente la selección del patrón para este caso. Aporte del diagrama original de UML correspondiente al patrón. **(3 puntos)**
3. Proponga el diagrama de UML que implementa la solicitud por parte del encargado de RRHH de los documentos que se requieren para la contratación d de un nuevo vendedor. **Justifique su respuesta**. **(5 puntos)**
4. Programe el funcionamiento del modelo del patrón seleccionado según su propuesta. Recuerde que cada documento debe ser obtenido de repositorios (o bases de datos distintas) para que ambos puedan ser completados por las partes, y ambos documentos deben estar firmados por ambos (encargado de RRHH y vendedor) para que se finiquite la contratación. **(10 puntos)**

Respuesta Caso 4	**Patrón a utilizar**	Prototype
	Tipo de Patrón	Creacional

Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.

Justificación de uso del patrón	De la base de datos se desea obtener los machotes de los documentos que luego serán clonados para obtener los documentos
Diagrama de UML asociado	
Modelo en UML que soluciona el problema	Justificación: La interfaz Documento es el prototipo. Los documentos que necesitan ser clonados son el acuerdo de confidencialidad (AcuerdoConfidencialidad) y el contrato. Los modelos se encuentran en CacheDocumento. El Agente es quien solicita las copias de los machotes. UML: 
Evidencias de código significativo en la implementación del patrón	Documento `public interface Documento { public Documento clone(); }` AcuerdoConfidencialidad `public class AcuerdoConfidencialidad implements Documento { @Override public Documento clone() { return new AcuerdoConfidencialidad(); } }` Contrato

Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.

```java
public class Contrato implements Documento
{
    @Override
    public Documento clone()
    {
        return new Contrato();
    }
}
```

CacheDocumento

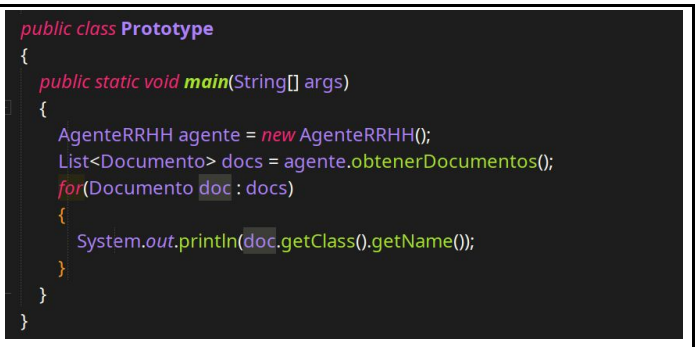
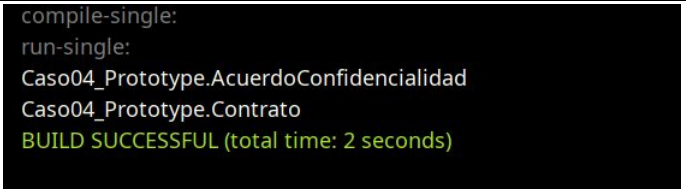
```java
public class CacheDocumento
{
    private AcuerdoConfidencialidad machoteAcuerdo;
    private Contrato machoteContrato;

    public CacheDocumento()
    {
        machoteAcuerdo = new AcuerdoConfidencialidad();
        machoteContrato = new Contrato();
    }

    /**
     * @return the machoteAcuerdo
     */
    public AcuerdoConfidencialidad getMachoteAcuerdo() {
        return machoteAcuerdo;
    }

    /**
     * @return the machoteContrato
     */
    public Contrato getMachoteContrato() {
        return machoteContrato;
    }
}
```

AgenteRRHH

```java
public class AgenteRRHH
{
    public List<Documento> obtenerDocumentos()
    {
        CacheDocumento cacheDocumento = new CacheDocumento();
        ArrayList<Documento> docs = new ArrayList<>();
        docs.add(cacheDocumento.getMachoteAcuerdo());
        docs.add(cacheDocumento.getMachoteContrato());
        return docs;
    }
}
```

Main

Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software
Prof.Ing.Ericka Solano Fernández II Semestre, 2018 II Examen Parcial
Valor 10% - 100puntos Para ser desarrollado en parejas
Liberado V 10 mayo, 2019. Se entrega D12 mayo medianoche.

	
Screeshoots de funcionamiento	

```
compile-single:
run-single:
Caso04_Prototype.AcuerdoConfidencialidad
Caso04_Prototype.Contrato
BUILD SUCCESSFUL (total time: 2 seconds)
```

FIN DE LA PRUEBA