Design Patterns
Guía de Ejercicios

Ejercicio 1

Contamos, en Java, con un conjunto de clases que permiten enviar emails de texto, html y con attachments.
Idea una solución, basada en algún patrón, tal que se reduzcan las dependencias del código cliente con esas clases y se reduzca la complejidad de dicho código cliente para crear y enviar mensajes.
Se adjuntan ejemplos de código.

Mime Messages

```
import java.util.Properties;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.AddressException;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;


// ...
    Properties props = new Properties();
    Session session = Session.getDefaultInstance(props, null);


    String msgBody = "...";


    try {
        Message msg = new MimeMessage(session);
        msg.setFrom(new InternetAddress("admin@example.com", "Example.com Admin"));
```

```
            msg.addRecipient(Message.RecipientType.TO,  new InternetAddress("user@example.com",
"Mr. User"));
            msg.setSubject("Your Example.com account has been activated");
            msg.setText(msgBody);
            Transport.send(msg);

        } catch (AddressException e) {
            // ...
        } catch (MessagingException e) {
            // ...
        }
```

Multi-Part Messages

```
import java.io.InputStream;
import javax.activation.DataHandler;
import javax.mail.Multipart;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMultipart;

// ...
String htmlBody;       // ...
byte[] attachmentData;  // ...

Multipart mp = new MimeMultipart();

MimeBodyPart htmlPart = new MimeBodyPart();
htmlPart.setContent(htmlBody, "text/html");
mp.addBodyPart(htmlPart);

MimeBodyPart attachment = new MimeBodyPart();
InputStream attachmentDataStream = new ByteArrayInputStream(attachmentData);
```

```
attachment.setFileName("manual.pdf");

attachment.setContent(attachmentDataStream, "application/pdf");

mp.addBodyPart(attachment);


message.setContent(mp);
```

Ejercicio 2

El restaurante tiene un menu del dia que basado en distintos criterios (frutas y verduras de de estación, stock de alimentos, etc) arma un menú del día. El menú siempre consta de una sopa, una entrada, un plato principal, un postre, y una bebida.

Los clientes piden el menu del dia al cocinero.

MenuDelDia menu = cocinero.getMenuDelDia();

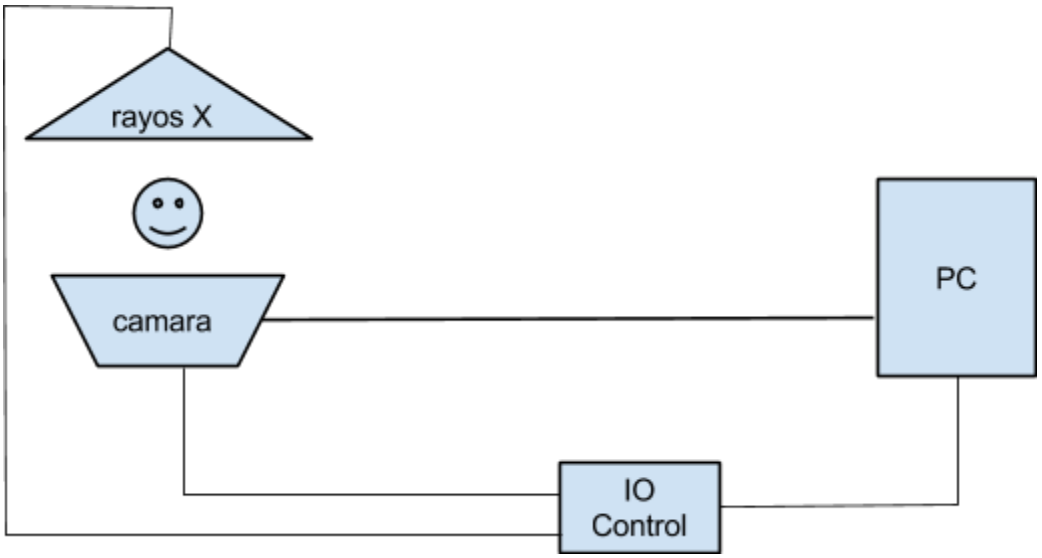
El cocinero puede retornar cualquier variante con los alimentos que tiene, pero siempre respetando las 5 componentes del menú (sopa, entrada, plat, postre, bebida).

Indique si utilizaría algún patrón de diseño para implementar esta funcionalidad, cuál y porqué.

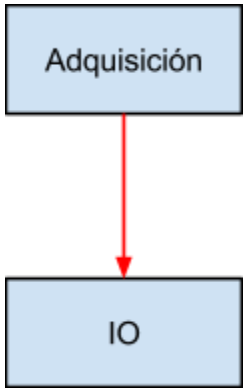
Ejercicio 3

Indicar que principio de diseño se cumple y/o cual no. De no cumplir algún principio, indique en qué contexto y cómo podría modificarse el diseño para que sí lo cumpla.

Existe un dispositivo de captura de imágenes médicas, con la siguiente estructura:



El control de IO es el encargado de sincronizar la cámara y el generador de rayos X (trigger).
El diseño es el siguiente (el módulo de Adquisición toma los datos del módulo de IO para procesarlos)

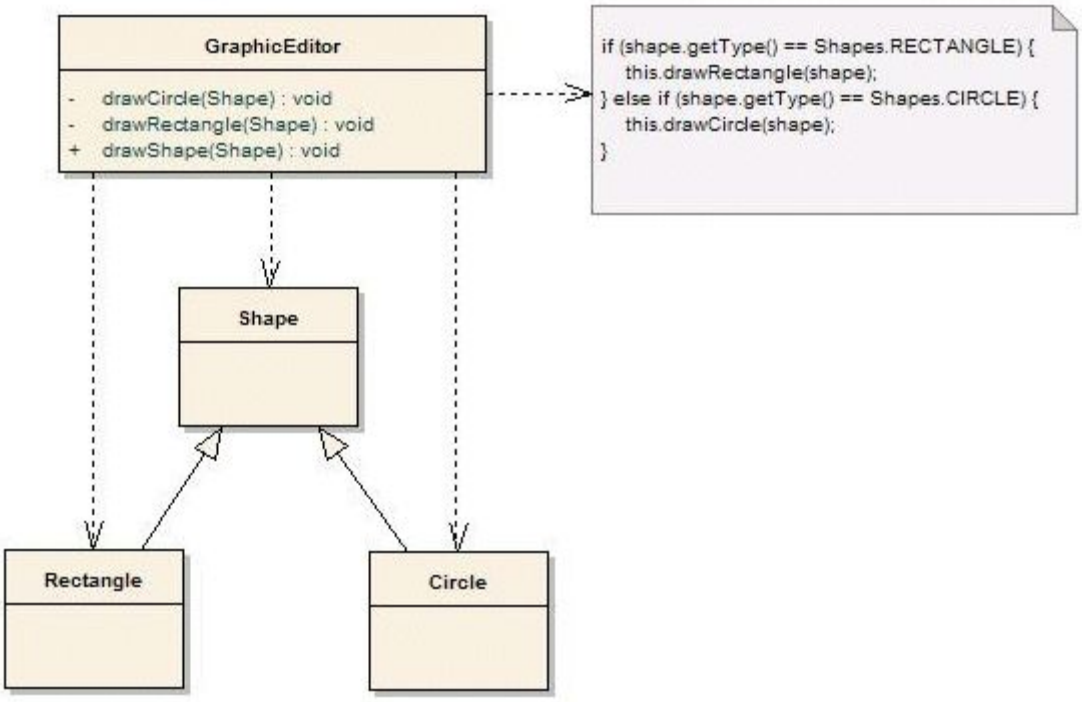


Nuevas tecnologías fueron apareciendo en el mercado, como por ejemplo unas placas muy pequeñas con PIO (programmable Input/Output) controladas por HTTP, utilizadas como controladores de IO. También nuevas cámaras (más baratas) con interface Ethernet, las cuales al conectarse directamente a la interface de red no tenían forma sincronizarse con el IO controller.
¿Qué problemas aparecen con el diseño actual frente a estos cambios tecnológicos? ¿Qué cambios propondría?

Ejercicio 4

Indicar que principio de diseño se cumple y/o cual no. De no cumplir algún principio indique en qué contexto y cómo podría modificarse el diseño para que si lo cumpla.

Diagrama



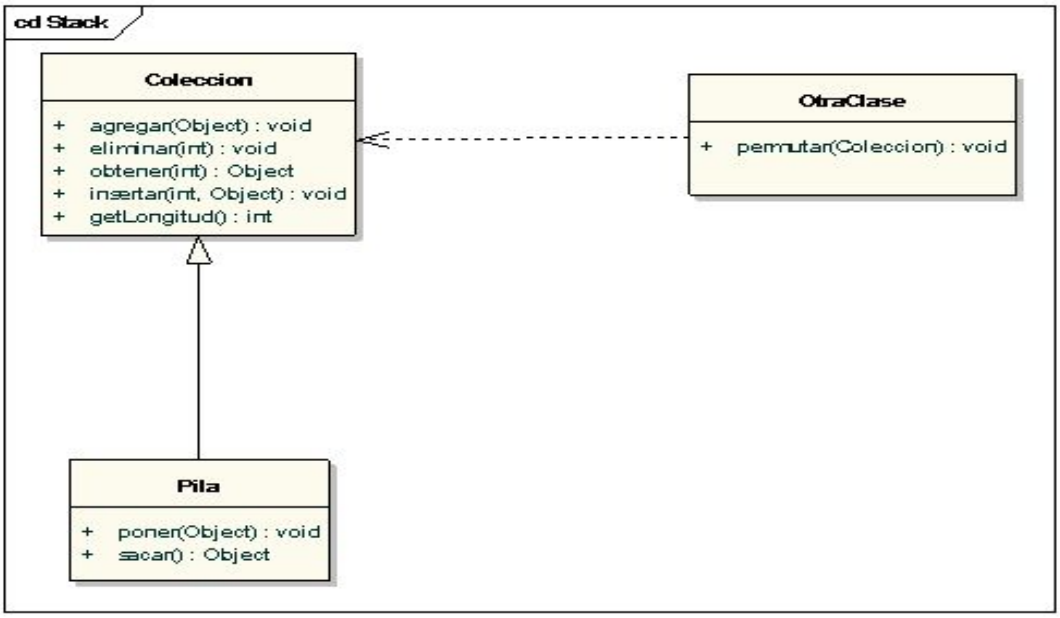
Código

```
public enum ShapeType {
	CIRCLE, RECTANGLE
}

public abstract class Shape {
	public abstract ShapeType getType();
}

public class Rectangle extends Shape {
	public ShapeType getType() {
		return ShapeType.RECTANGLE;
	}
}

public class Circle extends Shape {
	public ShapeType getType() {
```

```
            return ShapeType.CIRCLE;
        }
}

public Class GraphicEditor {

        private void drawCircle(Shape shape) {
            Circle circle = (Circle)shape;

            // Do circle drawing stuff.
        }

        private void drawRectangle(Shape shape) {
            Rectangle rectangle = (Rectangle)shape;

            // Do rectangle drawing stuff.
        }

        public void drawShape(Shape shape) {
            if (shape.getType() == ShapeType.CIRCLE) {
                this.drawCircle(shape);
            } else if (shape.getType() == ShapeType.RECTANGLE) {
                this.drawRectangle(shape);
            }
        }
}
```

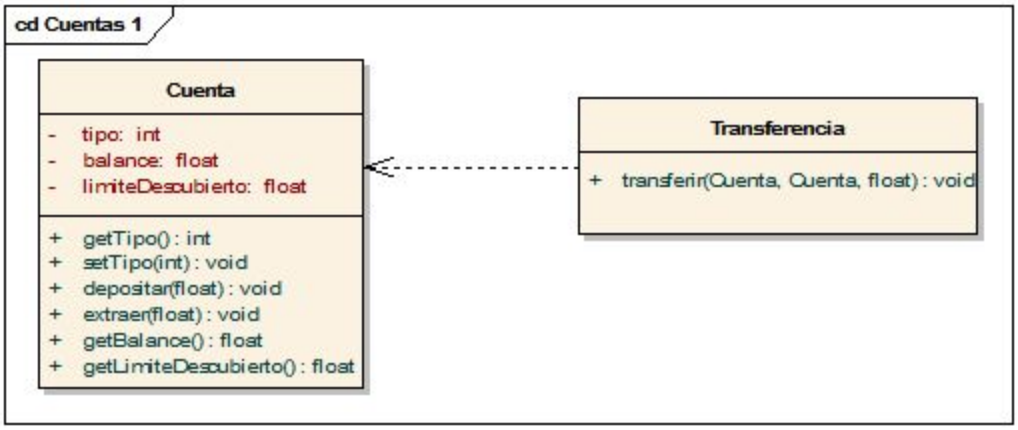
Ejercicio 5

Indicar que principio de diseño se cumple y/o cual no. De no cumplir algun principio indique en que contexto y como podria modificarse el diseño para que si lo cumpla.



Ejercicio 6

Indicar que principio de diseño se cumple y/o cual no. De no cumplir algún principio indique en qué contexto y cómo podría modificarse el diseño para que si lo cumpla.




```
public class Transferencia {
        ...
        void transferir( Cuenta origen, Cuenta destino, float monto ) {

                if ( origen.getTipo() == TipoCuenta.CAJA_AHORRO ) {
```

```
                        if ( origen.getBalance() >= monto ) {
                                origen.extraer( monto );
                                destino.depositar(monto );
                        }else
                                throw new SaldoInsuficienteException(...);

                } else if ( origen.getTipo() == TipoCuenta.CUENTA_CORRIENTE ) {

                        float saldoDisponible = origen.getBalance() +
origen.getLimiteDescubierto();
                        if ( saldoDisponible >= monto  ){
                                origen.extraer( monto );
                                destino.depositar(monto );
                        }else
                                throw new SaldoInsuficienteException(...);
                }



        }

}
```

Ejercicio 7

Indicar que principio de diseño se cumple y/o cual no. De no cumplir algún principio indique en qué contexto y cómo podría modificarse el diseño para que si lo cumpla.

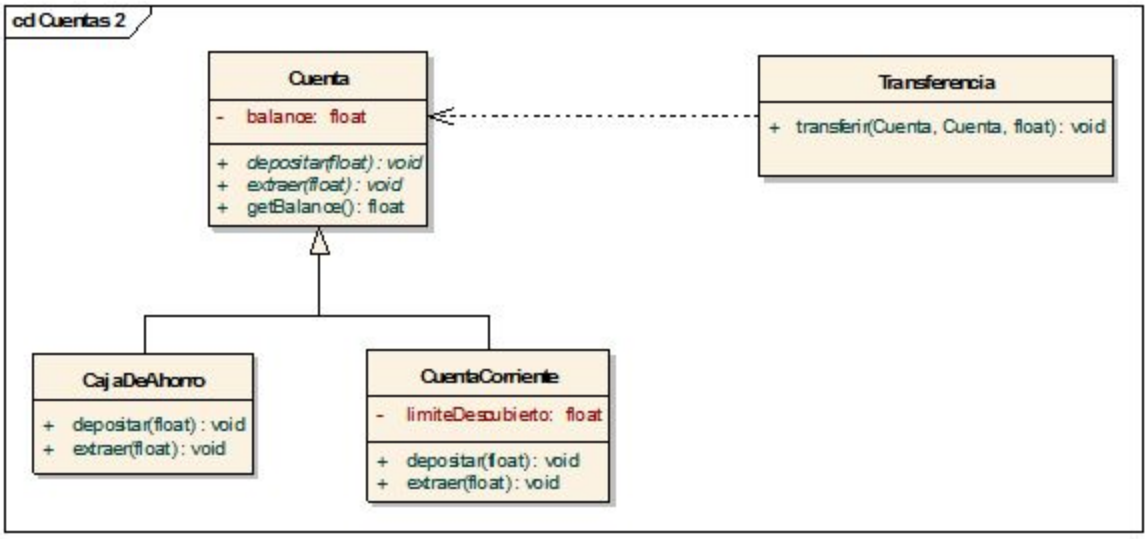
```
public class Cuenta {
        ...
        public void extraer( float monto ) {
                float saldoDisponible;
                if ( this.getTipo() == TipoCuenta.CAJA_AHORRO )
                        saldoDisponible = this.getBalance();
                else if ( origen.getTipo() == TipoCuenta.CUENTA_CORRIENTE )
                        saldoDisponible = this.getBalance() + this.getLimiteDescubierto();

                if ( saldoDisponible >= monto )
                        this.balance -= monto;
                else
```

```
                        throw new SaldoInsuficienteException(...);
        }
}
```

```
public class Transferencia {
        ...
        void transferir( Cuenta origen, Cuenta destino, float monto ) {
                origen.extraer( monto );
                destino.depositar(monto );
        }
        ...
}
```

Ejercicio 8

Indicar que principio de diseño se cumple y/o cual no. De no cumplir algún principio indique en qué contexto y cómo podría modificarse el diseño para que si lo cumpla.

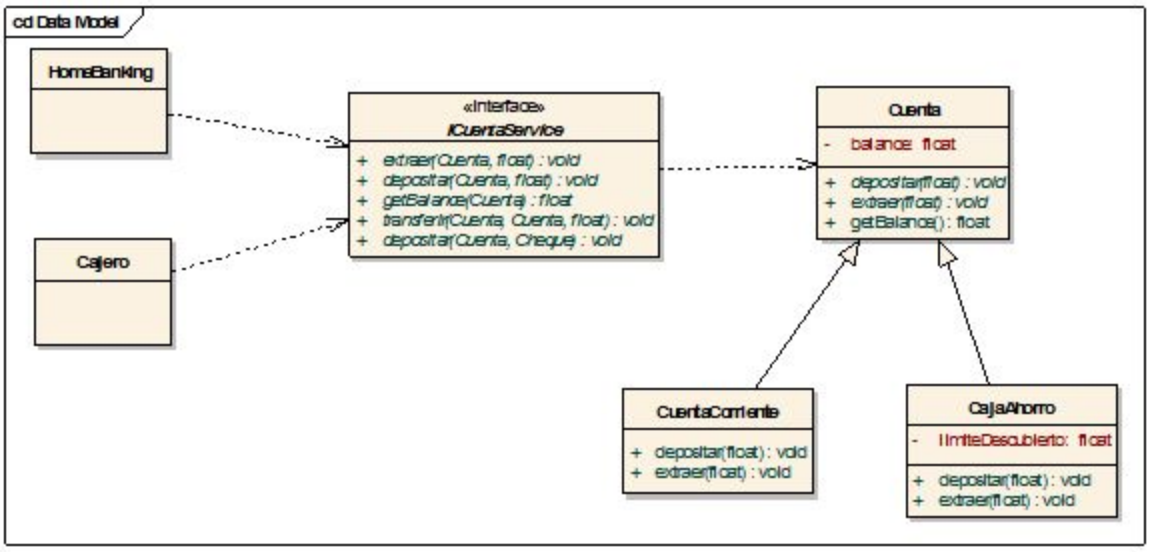



```
public abstract class Cuenta {
        ...
        public abstract void extraer( float monto );
        public abstract void depositar( float monto );
      ...
}
```

```
public class CajaDeAhorro {
	...
	public void extraer( float monto ) {

		if ( this.getBalance() >= monto )
			balance -= monto;
		else
			throw new SaldoInsuficienteException(...);
	}
	...
}

public class CuentaCorriente {
	...
	public void extraer( float monto ) {

		if ( (this.getBalance() + this.getLimiteDescubierto() ) >= monto )
			balance -= monto;
		else
			throw new SaldoInsuficienteException(...);
	}
}
```

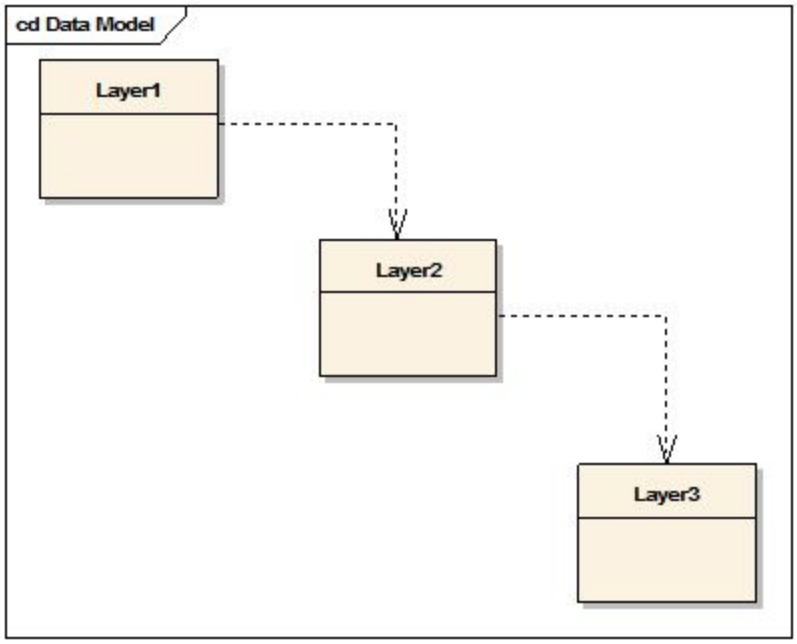
Ejercicio 9

Indicar que principio de diseño se cumple y/o cual no. De no cumplir algún principio indique en qué contexto y cómo podría modificarse el diseño para que si lo cumpla.



Ejercicio 10

Indicar que principio de diseño se cumple y/o cual no. De no cumplir algún principio indique en qué contexto y cómo podría modificarse el diseño para que si lo cumpla.



Ejercicio 11

En una revisión de código que tiene a cargo se encuentra con los siguientes fragmentos de código, indique que detecta y proponga alternativas de solución y justifiquelas basándose en los principios de buen diseño, smells y en las buenas prácticas de diseño de código.

1.1

```
if (date.Before(SUMMER_START) || date.After(SUMMER_END))
{
      charge = quantity * winterRate + winterServiceCharge;
}
else
{
      charge = quantity * summerRate;
}
```

1.2

```
double DisabilityAmount()
{
      if (seniority < 2) return 0;
      if (monthsDisabled > 12) return 0;
      if (isPartTime) return 0;
      // compute the disability amount…
}
```

1.3

```
if (IsSpecialCase())
{
	total = price * 0.95;
	send();
}
else
{
	total = price * 0.98;
	send();
}
```

1.4

```
double GetPayAmount()
{
	double result;
	if (isDead) result = deadAmount();
	else
	{
		if (isSeparated) result = separatedAmount();
		else
		{
			if (isRetired) result = retiredAmount();
			else result = normalPayAmount();
		}
	}

	return result;
}
```

1.5

```
public double GetAdjustedCapital()
{
	double result = 0.0;
	if (capital > 0.0)
	{
		if (intRate > 0.0 && duration > 0.0)
		{
			result = (income / duration) * ADJ_FACTOR;
		}
	}

	return result;
}
```

1.6

```
double GetSpeed()
{
	switch (type)
	{
	case BirdType.EUROPEAN:
		return getBaseSpeed();
```

```
        case BirdType.AFRICAN:
              return getBaseSpeed() - getLoadFactor() * numberOfCounts;

        case BirdType.NORWEGIAN_BLUE:
              return isNailed ? 0 : getBaseSpeed();
        }

        throw new RuntimeException("Should be unreachable");
}
```

1.7

Se encuentra repetidas veces el siguiente codigo:

```
if (customer == null) plan = BillingPlan.getPlan();
else plan = customer.getPlan();
```

1.8

```
double price() {
// price is base price - quantity discount + shipping
return _quantity * _itemPrice -
Math.max(0, _quantity - 500) * _itemPrice * 0.05 +
Math.min(_quantity * _itemPrice * 0.1, 100.0);
}
```

1.9

```
String foundPerson(String[] people){
        for (int i = 0; i < people.length; i++) {
              if (people[i].equals ("Don")) {
                    return "Don";
              }
              if (people[i].equals ("John")) {
                    return "John";
              }
              if (people[i].equals ("Kent")) {
                    return "Kent";
              }
        }
        return "";
}
```

Ejercicio 12

Dado el siguiente código, identifique smells, o malas prácticas de escritura/diseño de código. Proponga soluciones a los problemas detectados.

```
namespace Model.Movies
{
```

```
/// <summary>
/// Summary description for Customer.
/// </summary>
public class Customer
{
/// <summary>
/// Name of the Rental.
/// </summary>
private string _n;

/// <summary>
/// Getter of the Rental.
/// </summary>
public string getName()
{
        return this._n;
}

private ArrayList _rentals = new ArrayList();
/// <summary>
/// Agrega una renta a la lista de rentas
/// </summary>
public void addRental(Rental rental)
{
        this._rentals.Add(rental);
}
/// <summary>
/// Constructor
/// </summary>
public Customer(string name)
{
        this._n = name;
}

/// <summary>
/// Genera el reporte
/// </summary>
public string GetReport()
{
        double sum = 0; int cant = 0;
        String result = "Rental Record for " + this.getName() + "\n";
        foreach(Rental rental in this._rentals)
        {
        double thisAmntAcumm = 0;
        //determine amounts for each line
        switch (rental.Movie.PriceCode)
        {
        case Movie.REGULAR:
        thisAmntAcumm += 2;
        if (rental.DaysRented > 2)
        thisAmntAcumm += (rental.DaysRented - 2) * 1.5;
        // if (rental.DaysRented > 4)
        // thisAmntAcumm += (rental.DaysRented - 3) * 2.5 - 2;
        break;
        case Movie.NEW_RELEASE:
        thisAmntAcumm += rental.DaysRented * 3;
        break;
```

```
                case Movie.CHILDRENS:
                thisAmntAcumm += 1.5;
                if (rental.DaysRented > 3)
                thisAmntAcumm += (rental.DaysRented - 3) * 1.5;
                break;
                }
                // add frequent renter points
                cant ++;
                // add bonus for a two day new release rental
                if ((rental.Movie.PriceCode == Movie.NEW_RELEASE) &&
                        rental.DaysRented > 1) cant ++;
                //show figures for this rental
                result += "\t" + rental.Movie.Title+ "\t" +
                        thisAmntAcumm.ToString() + "\n";
                sum += thisAmntAcumm;
                }

                //add footer lines
                result += "Amount owed is " + sum + "\n"; result += "You earned " + cant + " frequent renter points";
                return result;

}}}
```

Ejercicio 13

Analice el siguiente codigo y si tiene algún problemas y en dicho caso como lo solucionaria

```
public ScoreAlert(ScoreType type, float average, int rating, Date expiry) {
        this.type = type;
        this.average=average;
        this.rating = rating;
        this.expiry = expiry;
}

public ScoreAlert( ScoreType type, float average, int rating, Date expiry, Date maturity) {
        this.level=AlertLevel.HIGH;
        this.type = type;
        this.average=average;
        this.rating = rating;
        this.expiry = expiry;
        this.maturity = maturity;
}

public ScoreAlert(AlertLevel level, ScoreType type, float average, int rating, Date expiry, Date
maturity) {
        this.level = level;
        this.type = type;
        this.average=average;
```

```
        this.rating = rating;
        this.expiry = expiry;
        this.maturity = maturity;
}

public ScoreAlert(ScoreType type, float average, int rating, Date expiry) {
        this(null, type,average, rating, expiry, null)
}
public ScoreAlert(ScoreType type, float average, int rating, Date expiry) {
        this(AlertLevel.HIGH, type,average, rating, expiry, maturity)
}
public ScoreAlert(AlertLevel level, ScoreType type, float average, int rating, Date expiry, Date
maturity) {
        this.level = level;
        this.type = type;
        this.average=average;
        this.rating = rating;
        this.expiry = expiry;
        this.maturity = maturity;
}
```

Ejercicio 14

Dado el siguiente código, identifique que patrón utilizaría a fin de evitar duplicidad. Refactorize el código mostrando la aplicación del patrón propuesto.

Código

```
public class DOMBuilderTest extends TestCase {
        private OutputBuilder builder;

        private String createValidResult() {
                StringBuilder result = new StringBuilder();
                result.append(“<orders>”);
        result.append(“<order>”);
        result.append(“</order>”);
        result.append(“</orders>”);
                return builder.toString();
}

        public void testAddAboveRoot() {

                this.builder = new DOMBuilder(“orders”);
                this.builder.addChild(“order”);
```

```
                assertSame("Wrong xml created.", this.createValidResult(), this.builder.build());
}
}

public class XMLBuilderTest extends TestCase {
        private OutputBuilder builder;

        private String createValidResult() {
                StringBuilder result = new StringBuilder();
                result.append("<orders>");
        result.append("<order>");
        result.append("</order>");
        result.append("</orders>");
                return builder.toString();
}

        public void testAddAboveRoot() {

                this.builder = new XMLBuilder("orders");
                this.builder.addChild("order");

                assertSame("Wrong xml created.", this.createValidResult(), this.builder.build());
}
}
```

Ejercicio 15

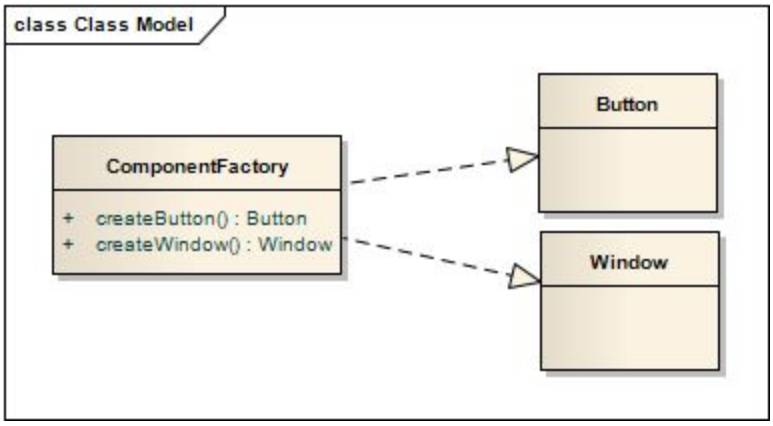
Sabemos que cada lenguaje establece ciertas reglas (ej: en java, los constructores deben llevar el mismo nombre que la clase que estan construyendo). Dada esta limitación, si una clase presenta más de un constructor, esto dificulta al cliente de dicha clase, conocer qué constructor utilizar en cada caso. Sabiendo esto y dado el siguiente código, se le ocurre alguna manera de refactorizarlo de forma tal que el mismo sea más legible y entendible desde el punto de vista del cliente?

```
public class Prestamo {

        public Prestamo (double monto, int nivelRiesgo, Date fechaRealizacion) {
                this(monto, nivelRiesgo, fechaRealizacion, null, 0.0);
}

public Prestamo(double monto, int nivelRiesgo, Date fechaRealizacion, Date fechaExpiracion) {
        this(monto, nivelRiesgo, fechaRealizacion, fechaExpiracion, 0.0);
}
```

```
public Prestamo(double monto, int nivelRiesgo, Date fechaRealizacion, Date fechaExpiracion, double recargo) {
        …
}

}
```

Ejercicio 16

La empresa Builder Software S.A. los ha contratado para la implementación de un nuevo sistema que permitirá construir gráficos en pantalla para su posterior impresión. Luego de un análisis de requerimientos, sabemos que el sistema debe poder correr en diversos sistemas operativos como ser Windows y MacOSX.
Actualmente se cuenta con un modelo como el siguiente:



Determine como extendería la misma para que la misma pueda correr en MacOSX, sabiendo que, si bien las interfaces de los componentes son identicas, su implementación es muy diferente.

Ejercicio 17

Imagine que Ud. se encuentra desarrollando un sistema que cuenta con el siguiente requerimiento:

Se necesita manejar la construcción de un objeto complejo, el cual puede ser grande o pequeño dependiendo de cierta configuración previa. Sabemos que el mismo debe ser construido por partes, dado que no es conocida el tamaño total de la configuración, por ende no es posible determinar el tamaño del grafo de objetos a construir.

Se le ocurre algún patrón, de los vistos en clase, que resuelva este problema? Cuál?

Ejercicio 18

Dado el siguiente codigo representativo de una capa logica de persistencia y de dominio, donde la clase Documento modela un documento, Parametro es una clase para representar un parámetro (nombre y valor) para utilizar en consultas.
DocumentoDataMapper es una clase que tiene la responsabilidad de mapear un objeto de la clase Documento a su representación física de datos en un medio persistente (por ejemplo una tabla en una DB)
Dado que se piensa tener un DataMapper por clase persistente de dominio, se quiere analizar el siguiente código, para verificar que podría hacerse a fin de evitar duplicidad.
Refactorize el código mostrando la solución propuesta.

```
using System;
using System.Collections;

namespace EjercicioDataMapper
{
   public class Documento
       {
       public string id = "";
       public string Id
       {
             get { return id; }
             set { id = value; }
       }
       public string numero = "";
       public string Numero
       {
             get { return numero; }
             set { numero = value; }
       }
       public string tipo = "";
       public string Tipo
       {
             get { return tipo; }
            set { tipo = value; }
       }
       public string fechaVencimiento = "";
       public string FechaVencimiento
       {
             get { return fechaVencimiento; }
             set { fechaVencimiento = value; }
       }
       }

       public class Parametro
       {
       public string nombre = "";
       public string Nombre
       {
```

```
        get { return nombre; }
}
public object valor;
public string Valor
{
        get { return valor.ToString(); }
  }
public Parametro(string Nombre, string Valor)
{
        this.nombre = Nombre;
        this.valor = Valor;
}
}

public class DocumentoDataMapper
{
// configuracion para el string de conexion de la fuente de datos
private string dataSource = "DATA_SOURCE_CONEXION";

public DocumentoDataMapper(){}

public Documento GetDocumento(int id)
{
        Documento result = new Documento();

        IDataSource ds = DataSourceFactory.GetDataSource(dataSource);

// configuracion a una consulta ya definida
string findStatment = "OBTENER_DOCUMENTO";
        IDataCommand cmd = ds.GetCommand(findStatment);

Parametro[] parametros = new Parametro[1];
        parametros[0] = new Parametro("DocumentoId", id);

for (int i = 0; i < parametros.Length; i++)
{
        cmd.Parameters[parametros[i].nombre].Value = parametros[i].valor;
}
        IDataReader dr = null;
        try
        {
        dr = cmd.ExecuteReader();
        if (dr.Read())
        {
                result.Id = Convert.ToInt32(dr["Id"]);
        result.Numero = dr["Nro"].ToString();
                result.Tipo = dr["Tipo"].ToString();
                result.FechaVencimiento = dr["FechaVencimiento"].ToString();
        }
        return result;
```

```
            }
            finally
            {
            if (dr != null) dr.Close();
            }
      }

      public ArrayList GetDocumentosUsuario(string codigoUsuario)
      {
            ArrayList result = new ArrayList();

            IDataSource ds = DataSourceFactory.GetDataSource(dataSource);

            string findStatment = "OBTENER_DOCUMENTOS_USUARIO";
            IDataCommand cmd = ds.GetCommand(findStatment);

      Parametro[] parametros = new Parametro[1];
            parametros[0] = new Parametro("codigoUsuario", codigoUsuario);

      for (int i = 0; i < parametros.Length; i++)
      {
            cmd.Parameters[parametros[i].nombre].Value = parametros[i].valor;
      }

            IDataReader dr = null;

            try
            {
            dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                  Documento documento = new Documento();
                  documento.Id = Convert.ToInt32(dr["Id"]);
                  documento.Numero = dr["Nro"].ToString();
                  documento.Tipo = dr["Tipo"].ToString();
                  documento.FechaVencimiento                                    =
dr["FechaVencimiento"].ToString();
            result.Add(documento);
            }
            return result;
            }
            finally
            {
            if (dr != null) dr.Close();
            }
            return result;
      }

      public ArrayList GetDocumentos()
      {
            ArrayList result = new ArrayList();
```

```
            IDataSource ds = DataSourceFactory.GetDataSource(dataSource);

            string findStatment = "OBTENER_ORGANIZACIONES";
            IDataCommand cmd = ds.GetCommand(findStatment);

            for (int i = 0; i < parametros.Length; i++)
      {
            cmd.Parameters[parametros[i].nombre].Value = parametros[i].valor;
      }

            IDataReader dr = null;
            try
            {
            dr = cmd.ExecuteReader();

            while (dr.Read())
            {
                  Documento documento = new Documento();
                  documento.Id = Convert.ToInt32(dr["Id"]);
                  documento.Numero = dr["Nro"].ToString();
                  documento.Tipo = dr["Tipo"].ToString();
                  documento.FechaVencimiento                                    =
dr["FechaVencimiento"].ToString();
            result.Add(documento);
            }
            return result;
            }
            finally
            {
            if (dr != null) dr.Close();
            }
            return result;
      }
      }
}
```

Ejercicio 19

Detecta algún patrón de diseño aplicado al siguiente código? Cuál?

```
public class Operacion {

      public enum OperacionEnum {
```

```
        MENSAJE, NO_MENSAJE
    }

    public Operacion build(OperacionEnum operacionEnum) {
        if (operacionEnum == null) {
            return new NoOperacion();
        }
        switch (operacionEnum) {
            case MENSAJE:
                return new Mensaje();
            case NO_MENSAJE:
                return new NoOperacion();
            default:
                throw new IllegalArgumentException();
        }
    }
}

public interface IOperacion {
    void operacion();
}

public class NoOperacion implements IOperacion{
    @Override
    public void operacion() {
    }
}

public class Mensaje implements IOperacion {

    @Override
    public void operacion() {
        System.out.println("Hola mundo!");
    }
```

```
}
```