ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN
CCPG1009 - DISEÑO DE SOFTWARE
SEGUNDA EVALUACIÓN - II TÉRMINO 2019

Nombre: _____ ***Matrícula:*** _____

COMPROMISO DE HONOR: Al firmar este compromiso, reconozco que el presente examen está diseñado para ser resuelto de manera individual, que puedo usar un lápiz o esferográfico; que sólo puedo comunicarme con la persona responsable de la recepción del examen; y, cualquier instrumento de comunicación que hubiere traído, debo apagarlo y depositarlo en la parte anterior del aula, junto con algún otro material que se encuentre acompañándolo. Además, no debo usar calculadora alguna, consultar libros, notas, ni apuntes adicionales a los que se entreguen en esta evaluación. Los temas debo desarrollarlos de manera ordenada. Firmo el presente compromiso, como constancia de haber leído y aceptado la declaración anterior. "Como estudiante de ESPOL me comprometo a combatir la mediocridad y actuar con honestidad, por eso no copio ni dejo copiar". _____ Firma	_____ 100 _____ Firma

Sección A

1. "Decorator", es un patrón de diseño que permite agregar _____ a un objeto de forma _____. **[4%]**
2. Indique a que se refiere el mal olor de "Refused Bequest" y con qué técnica de refactorización se podría corregir. **[4%]**

   Definición:

   Refactoring:

3. Explique con un ejemplo, cuándo elegiría utilizar uno u otro de los siguientes patrones de diseño: "Iterator", "Chain of Responsibility". Su ejemplo debe indicar, además, el por qué no se podría usar el otro patrón. **[3%]**

4. _____ se refiere a cuando para realizar una modificación, es necesario realizar varios pequeños cambios en diferentes clases. **[3%]**

5. Asevere o niegue el siguiente enunciado [Justifique su respuesta]:
   Las pruebas unitarias nos permiten asegurar que el código de una aplicación esté libre de fallos. **[4%]**

6. El patrón de diseño "Composite" permite generar una estructura recursiva de objetos simples y compuestos. Indique, desde su perspectiva, cual es el mayor problema que podría tener este patrón. **[4%]**

7. Aplicar diferentes técnicas de refactorización siempre nos va a permitir eliminar todos los malos olores de programación existentes en nuestro código. **[3%]**
   a. Verdadero
   b. Falso. Justifique:

MSc. David Jurado Mosquera, Dr. Carlos Mera Gómez

Sección B

8. Considere el siguiente código fuente que corresponde a un sistema de gestión de transporte de carga en el que un flete transporta varios tipos de productos con costos diferenciados.

```
public class Flete {
    private enum TipoProducto {Fragil, Regular, Pesado};
    private List<Producto> productos;
    private ReporteTexto reporteTexto;
    private ReporteHtml reporteHtml;

    public Flete(){
        productos = new ArrayList<Producto>();
    }
    public void añadirProducto(Producto producto){
        productos.add(producto);
    }
    public double calcularIngresos(){
        double totalIngresos = 0;
        for (Producto producto : productos) {
            double ingreso = producto.calcularCobroProducto();
            totalIngresos += ingreso;
        }
        return totalIngresos;
    }
    public String imprimirReporte(){
        String contenido = "";
        if (reporteTexto != null)
            contenido = reporteTexto.generarFormatoTexto(productos.size(),
                                                calcularIngresos());
        else if(reporteHtml!=null)
            contenido = reporteHtml.generarFormatoHtml(productos.size(),
                                                calcularIngresos());
        return contenido;
    }
    private void setReporteTexto(ReporteTexto reporteTexto) {
        this.reporteTexto = reporteTexto;
    }
    private void setReporteHtml(ReporteHtml reporteHtml) {
        this.reporteHtml = reporteHtml;
    }

    public class Producto {
        private TipoProducto tipoProducto;
        private double peso;
        private long id;
        private double distanciaDestino;

        public Producto(long id, TipoProducto tipoProducto,
                        double peso, double distanciaDestino){
            this.id = id;
            this.tipoProducto = tipoProducto;
            this.peso = peso;
            this.distanciaDestino = distanciaDestino;
        }
        public TipoProducto getTipoProducto() {
            return tipoProducto;
        }
        public double getPeso() {
            return peso;
        }
        public void setValue(Object value){
            if (value instanceof TipoProducto)
                tipoProducto = (TipoProducto)value;
            else if(value instanceof Double)
                peso = ((Double)value).doubleValue();
        }
        public double getDistanciaDestino(){
            return distanciaDestino;
        }
        public double calcularCobroProducto(){
            double cobro = 0;
            switch (getTipoProducto()) {
                case Fragil:
                // 500 es el cobro base y 0.55 el factor de distancia
                cobro = 500 + getDistanciaDestino()*0.55;
                break;
                case Regular:
                // 250 es el cobro base, 0.25 es el factor de distancia
                // y 10 es el factor de recargo por peso
                cobro = 250 + getDistanciaDestino()*0.25+getPeso()*10;
                break;
                case Pesado:
                if (peso > 30)
                    //cobro base es 999 y factores son 0.9 y 9
                    cobro = 999+getDistanciaDestino()*0.9+getPeso()*9;
                else
                    //cobro base es 750 y factor de distancia es 0.6
                    cobro = 750 + getDistanciaDestino()*0.6;
                break;
            }
            return cobro;
        }
    }

    public class Reporte{
        private Calendar cal;
        public Reporte(){
            cal = Calendar.getInstance();
        }
        public String formatearFecha() {
            SimpleDateFormat formato = new SimpleDateFormat("dd-MM-YY");
            return formato.format(cal.getTime());
        }
    }

    public class ReporteHtml extends Reporte{
        public String generarFormatoHtml(int cantidadProductos,
                                         double ingresos) {
            String html = "<HTML><TITLE>Reporte en HTML</TITLE>";
            html +="<HEAD>Reporte del flete "+formatearFecha()+"</HEAD>";
            html +="<BODY><P>Este flete transporta "+cantidadProductos;
            html +=" productos.</P>\n";
            html +="<P>Estos productos dejan un ingreso de "+ingresos;
            html +=" USD.</P>\n</BODY>\n<HTML>";
            return html;
        }
    }

    public class ReporteTexto extends Reporte{
        public String generarFormatoTexto(int cantidadProductos,
                                          double ingresos) {
            String texto = "Reporte del flete "+formatearFecha()+"\n ";
            texto += "Este flete transporta "+cantidadProductos;
            texto += " productos. Estos productos dejan ";
            texto += "un ingreso de "+ ingresos +" USD.";
            return texto;
        }
    }
}
```

A usted se le solicita:

a. Señalar los malos olores de programación e indicar su nombre. Explique por qué sería un problema. **[10%]**

b. Indicar las técnicas de refactorización que utilizaría para mejorar el código. Justifique su respuesta. **[15%]**

c. Realizar 2 pruebas unitarias para el método **calculateProductCharge()** y 2 pruebas unitarias el método **calcularIngresos().** **[10%]**

Sección C

9. Dado el requerimiento que se presenta a continuación:

   a. Identifique los patrones de diseño que pueda aplicar para implementar lo solicitado. Justifique su respuesta **[15%]**
   b. Elabore un diagrama de clases aplicando los patrones de diseño que considere apropiados. Identifique herencias, multiplicidades, visibilidad de atributos y métodos. Indique, por medio de notación UML, si las entidades corresponden a interfaces, clases abstractas o clases concretas. **[20%]**
   c. Separe en paquetes según los patrones usados. **[05%]**

En un sistema de servicio de encomiendas, los clientes pueden hacer envíos a nivel nacional e internacional. La encomienda puede pasar por diferentes estados: receptada en oficina, lista para distribución, en camino a su destino, arribada a centro de acopio, y finalmente, entregada. El remitente, el destinatario y cualquier otra persona que se registre, es notificada por el sistema acerca de cualquier cambio de estado de la encomienda. El mecanismo de notificación puede ser escogido por el cliente y para el sistema debe ser transparente. Los mecanismos son Orkut, Facebook, Twitter, Whatsapp. Nuevos mecanismos se deberían poder añadirse al sistema sin mayor impacto. Las posiciones de las unidades de transporte que realizan los envíos son monitoreadas en tiempo real. Con este propósito, cada unidad tiene instalado un dispositivo de rastreo de alguna de las siete diferentes marcas disponibles en el mercado. No existe una interfaz estándar entre los dispositivos.