ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN CCPG1009
– DISEÑO DE SOFTWARE
PRIMERA EVALUACIÓN - II TÉRMINO 2017/ Noviembre 30, 2017

Nombre: _____ ***Matrícula:*** _____ ***Paralelo:*** ____

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

TEMA 1 – CONCEPTOS DE DISEÑO DE SOFTWARE (40 PUNTOS)
Por cada ítem, seleccione o escriba la(s) respuesta(s) correcta(s), según considere conveniente. Para los falsos, deben agregar una explicación o se anulará la respuesta.

1) El **diseño de detallado** de software se refiere a: (3 pt)

2) El **diseño arquitectónico** de software se refiere a la: (3 pt)
   a. Elaboración de diagramas de secuencia y clases.
   b. Especificación y análisis de los requerimientos.
   c. Elección del paradigma de programación.
   d. Elaboración de diagramas de caja negra.

3) El sistema de control de versiones **GIT** permite trabajar localmente con un repositorio remoto, el comando que se utiliza para descargar el código desde dicho repositorio es _____. (2 pt)

4) Indique cuál de estas características **NO pertenece a GIT**: (3 pt)
   a. Control de acceso mediante distintos roles.
   b. Manejar distintas versiones del código en paralelo.
   c. Deshacer (cambios al código) ilimitado.
   d. Combinar aportaciones de distintos colaboradores.

5) Considere que el siguiente escenario. Usted ha descargado y configurado un repositorio remoto, luego ha realizado un cambio en el código del archivo "./test.java" y ahora **desea subir sus cambios al repositorio remoto**. Que comandos (solo el nombre de cada comando) debe ejecutar para subir sus cambios. (6 pt)
   >
   >
   >

6) La **programación imperativa** es de donde se desprenden los paradigmas de programación Estructural, Orientado a Objetos y Orientado a Aspectos. (2 pt)
   a. Verdadero.
   b. Falso. Porque:

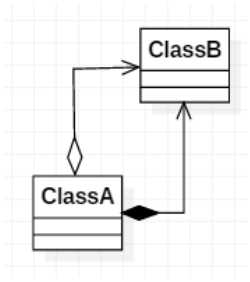
7) La **Abstracción** es uno de los principios fundamentales de la Orientación a Objetos. Explique por qué: (3 pt)

8) ¿En la **Orientación a Aspectos**, un Advice es a un Aspecto como un Objeto es a una Clase, en la Orientación a Objetos? (2 pt)
   a. Verdadero.
   b. Falso. Porque:

9) En la **Orientación a Aspectos**, un Advice es la implementación de un Joint-Point *(*2 pt)
   a. Verdadero.
   b. Falso. Porque:

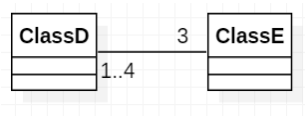
10) Se desea implementar un sistema para generar automáticamente la declaración mensual de IVA de un mini-negocio. **No se desea realizar pruebas unitarias**; sin embargo, se conoce en detalle el formato válido de la declaración mensual para el Servicio de Rentas Internas. ¿**Qué paradigma** de diseño recomendaría para su implementación? Justifique puntualmente su respuesta. (3 pt)

11) Para las relaciones de **composición y de agregación** en el siguiente **diagrama de clase**, podemos decir que es cierto: (3 pt)



   a. Un objeto de classA existe mientras existe un objeto de classB por agregación.
   b. Un objeto de classA es parte de un objeto de classB por agregación.
   c. Un objeto de classB existe mientras existe un objeto de classA por composición.
   d. Un objeto de classB contiene a un objeto de classA por composición.

12) Según la siguiente relación entre las clases ClassD y ClassE de un **diagrama de clases**, podemos decir que es cierto: (3 pt)



   a. Cada objeto de ClassD está asociado a tres objetos de ClassE.
   b. Tres objetos de ClassE tienen asociados desde uno hasta cuatro objetos de ClassD.
   c. Cada objeto de ClassE está asociado a tres objetos de ClassD.
   d. Cada objeto de ClassD tiene asociado desde uno hasta cuatro objetos de ClassE.

13) En los **diagramas de secuencia** se pueden utilizar mensajes de tipo "**Perdido**". Explique para que sirven: (3 pt)

14) Indique, en qué consisten los dos tipos de **herencia** expresados en los **diagramas de clases**: (2 pt)

TEMA 2 – PRINCIPIOS DE DISEÑO SOLID **(20 PUNTOS)**

Dado el siguiente código, identifique los **principios SOLID que se está violando**, explique la razón y corrija el código de tal forma que ya no se viole ningún principio. Usted puede crear las interfaces y clases que considere necesarias. **Identificar: 5 pt, explicar: 5 pt y código: 10 pt.**

```java
public class Greeter {
    String formality;
    Greeter(){
        this.setFormality("casual");
    }
    public String greet() {
        if (this.formality == "formal") {
            return "Good evening, sir.";
        }
        else if (this.formality == "casual") {
            return "How do you doing?";
        }
        else if (this.formality == "intimate") {
            return "Hello Darling!";
        }
        else{
            return "Hello!";
        }
    }
    public void setFormality(String formality) {
        this.formality = formality;
    }
}
```

```java
public class FormalGreeter extends Greeter{
    FormalGreeter(){
        this.setFormality("formal");
    }
    public void setFormality(String formality) {
        if (formality == "formal"){
            this.formality = formality;
        }else{
            throw new RuntimeException(
                "Can't change formality when "
                + this.getClass()
                + " is instantiated!");
        }
    }
}
public class TestGreet{
    public static void main(String[] args) {
        Greeter gt = new FormalGreeter();
        gt.setFormality("intimate");
        gt.greet();
    }
}
```

TEMA 3 – DISEÑO ORIENTADO A OBJETOS (40 PUNTOS)

A usted se le ha solicitado elaborar el diseño de un sistema, considerando los siguientes requerimientos:

Se desea desarrollar un Software que permita manejar todas las actividades dentro de un taller de mecánica automotriz. La intención de la aplicación es que tenga una interfaz gráfica y que sea multiplataforma, específicamente Linux y Windows en su primera versión. Posteriormente se realizará la portabilidad a Mac OS X y a dispositivos móviles.

Las actividades incluyen ingreso de ordenes de atención a clientes, donde debe asignar un mecánico a cada cliente y determinar el tipo se servicio / atención (enderezada, electrónica y pintura) que necesita el vehículo del cliente. El vendedor es quien realiza esta actividad para dar seguimiento a las órdenes de atención realizadas, como la de venta de autopartes a los clientes. Un vendedor debe ingresar por lo menos 10 ordenes de servicio cada día para poder ganar comisiones. Cada día se reinicia la cantidad de órdenes. Considere que los tipos de servicios también se podrían incrementar para la segunda versión.

El gerente general debe realizar las compras de autopartes a los proveedores registrados en el sistema, si se desean comprar nuevas autopartes, entonces, se debe ingresar tanto el proveedor que las comercializa, como la nueva autoparte que se va a adquirir. Por otro lado, el gerente también necesita generar reportes de compras y ventas de autopartes semanales, balances de ventas diarios y semanales por vendedor y finalmente el inventario actualizado.

Los mecánicos sólo necesitan ingresar el diagnóstico del vehículo y las autopartes que serán necesarias para realizar el servicio / atención al cliente. Considerando, que un mecánico puede atender diariamente hasta 7 vehículos. Ellos, deben elegir el siguiente vehículo que atenderá desde una cola de órdenes presentadas en pantalla. Una vez que ha terminado con un vehículo se debe enviar un email al cliente.

Además, existe un administrador del sistema que es el responsable de agregar / administrar / remover los usuarios del sistema, asignando un rol a cada usuario. El administrador también es el encargado de realizar ciertos reportes: acceso al sistema por usuario del último mes, nuevos proveedores, autopartes y clientes ingresados en el último mes.

Todos los usuarios deben acceder al sistema con credenciales previamente ingresadas (usuario, clave).

Elaborar lo siguiente:

1. Diagrama de casos de uso. (10 pt)
2. Dos Diagramas de secuencias de objetos: (10 pt)
    a. Realizar seguimiento de una orden (vendedor).
    b. Atender nuevo vehículo (mecánico).
3. Diagrama de clases. Especifique multiplicidades, relaciones, visibilidad de métodos y atributos. (20 pt)