ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

CCPG1009 – DISEÑO DE SOFTWARE

TERCERA EVALUACIÓN – II TÉRMINO 2018

Nombre: _____ ***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.	_____ 100
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	_____ Firma

TEMA 1 – APLICACIÓN DE PATRONES (60 PUNTOS)

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

La administración de una urbanización cerrada ha decidido implementar un sistema digital para permitir el acceso a los residentes y a sus visitantes a la urbanización. En la urbanización tienen 3 tipos de accesos. La entrada principal, el salón para eventos y el área de piscinas y parques. Cada una posee un lector de QR (similar al código de barras) con una cámara para fotografiar las cédulas y un sensor de RFID (identifica acercando la tarjeta).

Los residentes pueden acceder presentando su tarjeta con RFID para ingresar a cualquiera de las áreas, o mediante un código QR generado por una aplicación de celular (se trata de otro sistema). Los visitantes solo pueden ingresar mostrando su cédula y un código QR generado por el residente a quien va a visitar.

Por otro lado, los residentes pueden realizar reservaciones tanto del salón de eventos, como de las piscinas y parques, por un máximo de 4 horas al día, durante este período se le permite acceso al área, únicamente a los visitantes del residente que realizó la reserva y a su familia. Las reservas deben hacerse con un mínimo de 3 días de anticipación y esta debe ser notificada a la administración y a los demás residentes.

La administración requiere además que todos los residentes puedan, en todo momento, consultar la lista de reservas por área ordenadas en forma ascendente desde la fecha actual. Sin embargo, solo se les permite realizar reservas, generar códigos QR o utilizar sus tarjetas RFID, a los que están al día con sus alícuotas.

Usted ha sido designado para el diseño de una solución que permita satisfacer todos los requerimientos de la administración. Recuerde cumplir con los principios SOLID, utilizar patrones de diseño en caso de ser necesarios y aplicar una arquitectura que separe la lógica del negocio de la interfaz.

Elaborar lo siguiente:

1. Diagrama de casos de uso. (10 pt)
2. Descripción del caso de uso: **Reservar un área**. (5 pt)
3. Modelo lógico o físico de la base de datos (normalizada). (10 pt)
4. Diagrama de clases.
    a. Aplicar una arquitectura que separe la lógica del negocio de la interfaz. (3 pt)
    b. Especifique multiplicidades, relaciones, visibilidad de métodos y atributos. (5 pt)
    c. Aplique patrones de diseño, identificándolos con subpaquetes. (12 pt)
5. El diagrama de secuencias de objetos del CDU descrito anteriormente: (10 pt)
6. Implemente 3 pruebas unitarias para el proceso de reserva de un área. (5 pt)

TEMA 2 – PRINCIPIOS DE DISEÑO Y REFACTORIZACIÓN (40 PUNTOS)

Dado el siguiente código, identifique los **principios SOLID que se está violando, luego indique cuales son los malos olores de programación que tiene el código**, explique la razón y corrija el código de tal forma que ya no se viole ningún principio y no tenga ningún mal olor. Puede crear las interfaces y clases que considere necesarias para la refactorización. **Identificar: (10 pt), explicar: (10 pt) y corregir: (20 pt).**

```
public enum ModoConduccion { AUTOMATICO, MANUAL }
public enum TipoVehiculo {TURISMO, DEPORTIVO, RALLY}
public class Vehiculo {
    int maxPotencia;
    int rpm;
    int maxCombustible;
    int restanteCombustible;
    int velocidades;
    String name;
    String serie;
    int colorR;
    int colorG;
    int colorB;
    ModoConduccion modo;
    TipoVehiculo tipo;
    public Vehiculo(String nombre, String serie, int fuel, int power, int vel){
        this.serie =serie;
        name = nombre;
        maxCombustible = restanteCombustible = fuel;
        maxPotencia = power;
        velocidades = vel;
        colorR = colorG = colorB = 255;
        rpm = 1;
        modo = MANUAL;
        tipo = TURISMO;
    }
    public ModoConduccion getModo(){ return modo; }
    public void setModo(ModoConduccion mode){modo = mode;}
    public TipoVehiculo getTipo(return tipo;)
    public void setTipo(TipoVehiculo type){ tipo = type; }
    public int getMaxPotencia() { return maxPotencia;  }
    public int getMaxCombustible() {  return maxCombustible; }
    public int setCombustible(final int fuel) {
        if (this.maxCombustible >= fuel){ this.restanteCombustible = fuel; }
        else{ restanteCombustible = maxCombustible; }
    }
    /** Permite acelerar el vehiculo de acuerdo a su modo de conducción **/
    public void _acc_for_mod(){
        if(rpm < maxPotencia){
            switch (this.modo){
                case AUTOMATICO:
                    //Verifica que no se pase de su máxima potencia
                    if(maxPotencia > rpm + 2 && restanteCombustible > 1 ){
                        rpm += 2;
                        restanteCombustible--;
                    }
                    //Si ya no hay combustible el carro se debe detener
                    if(restanteCombustible == 0 ){ rpm = 0; }
                    break;
                case MANUAL:
                    //Verifica que no se pase de su máxima potencia
                    if(maxPotencia > rpm + 5 && restanteCombustible > 1 ){
                        rpm += 5;
                        restanteCombustible--;
                    }
                    //Si ya no hay combustible el carro se debe detener
                    if(restanteCombustible == 0 ){ rpm = 0; }
                    break;
                    //Si ya no hay combustible el carro se debe detener
                    if(restanteCombustible == 0 ){ rpm = 0; }
                    break;
            }
        }
    }
    public void inicializar(){
        switch (this.tipo){
            case TURISMO:
                this.setMaxPotencia(50);
                this.maxCombustible(1000);
                this.rpm = 1;
                this.velocidades = 5;
                this.acelerar();
                break;
            case DEPORTIVO:
                this.setMaxPotencia(80);
                this.maxCombustible(700);
                this.rpm = 1;
                this.velocidades = 6;
                this.acelerar();
                break;
            case RALLY:
                this.setMaxPotencia(70);
                this.maxCombustible(1100);
                this.rpm = 1;
                this.velocidades = 5;
                this.acelerar();
                break;
            default:
                this.setMaxPotencia(50);
                this.maxCombustible(500);
                this.rpm = 1;
                this.velocidades = 5;
                this.acelerar();
                break;
        }
    }
}
public class Carreras{
    public static void main(String[] args) {
        ArrayList <Vehiculo> autos;
        for(int i = 0 ; i< 25 ; i++){
            Vehiculo v = new Vehiculo("Auto_"+i, "serie0"+i, 0,0,0);
            v.setTipo(DEPORTIVO);
            v.colorR = Random.randInt(0,255);
            v.colorG = Random.randInt(0,255);
            v.colorB = Random.randInt(0,255);
            System.out.println(v.name + " (" + v.colorR + ","
                + v.colorG + "," + v.colorB + ")");
            autos.add(v);
        }
        // Iniciar la carrera
        for (Vehiculo v : autos) {
            v.inicializar();
        }
        //Proceso para identificar quien gana la carrera
    }
```