Cédula

ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

CCPG1009 – DISEÑO DE SOFTWARE

SEGUNDA EVALUACIÓN - II TÉRMINO 2017

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 – CONCEPTOS (30 PUNTOS)

Por cada ítem, seleccione o escriba la(s) respuesta(s) correcta(s), según considere conveniente. En caso de ser falso, debe justificar su respuesta.

1) Indique cual(es) de los siguientes patrones de diseño **NO** pertenece(n) a los Estructurales: (4 pt)
   a. Facade
   b. Chain of Responsibility
   c. Decorator
   d. Composite
   e. Singleton

2) Primitive Obsession es una técnica de refactorización que permite eliminar variables primitivas y reemplazarlas por objetos pequeños. (2 pt)
   a. Verdadero
   b. Falso. Porque:

3) Describa un escenario / ejemplo donde se debe aplicar la técnica de “Reemplazar herencia con delegación”: (4 pt)

4) El patrón de diseño _____ tiene como propósito principal, agregar responsabilidades en tiempo de ejecución a un objeto previamente creado. (2 pt)

5) Indique el nombre de una herramienta considerada entre las etapas de DevOps. Además, explique cómo esta herramienta ayudaría en el proceso. (4 pt)

6) Las pruebas unitarias aseguran que el código fuente no tiene fallas. (2 pt)
   a. Verdadero
   b. Falso. Porque:

Cédula

7) Las pruebas unitarias incrementan el tiempo de desarrollo de un sistema. (2 pt)
   a. Verdadero
   b. Falso. Porque:

8) Mencione por lo menos 3 tipos de comprobaciones utilizadas dentro de las pruebas unitarias y explique cómo funcionan. (6 pt)

   a. _____

   b. _____

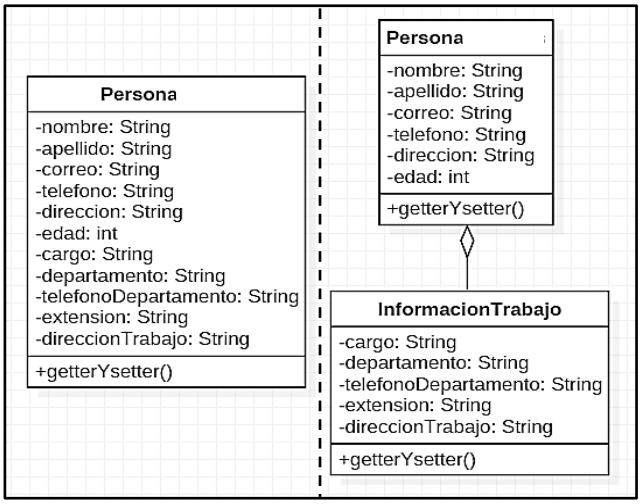
   c. _____

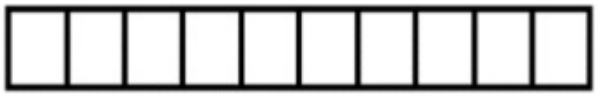
9) El mal olor de programación llamado “Data clumps” cómo puede ser solucionado. Indique el nombre de la técnica de refactorización y cuando la utilizaría o en qué condiciones. (4 pt)

TEMA 2 – REFACTORING (30 PUNTOS)

El diagrama adjunto ha sido generado a partir del código fuente de un sistema que fue refactorizado. La clase de la izquierda fue modificada y luego de la refactorización se obtuvieron las clases de la derecha.

10) Indique cuál o cuáles son los malos olores que tenía la clase de la izquierda. (5 pt)
11) Indique si las clases de la derecha solucionan los malos olores (¿cómo lo soluciona?). Si no lo soluciona, indique porqué. (5 pt)





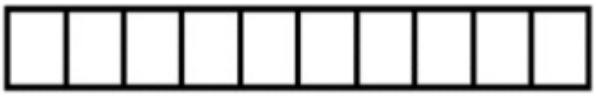
Cédula

12) Dado el siguiente código, identifique los malos olores (5 pt) y refactorice (15 pt) en la siguiente página.

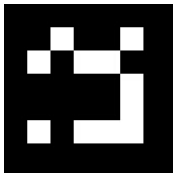
```java
public enum EmployeeType { Worker, Supervisor, Manager }

public class Employee
{   //remuneracion mensual unificada
    private final float rmu = 386.0;
    //salario del employee
    private float salary;
    //porcentaje de bonus
    private float bonusPercentage;
    //variable de tipo employeeType
    private EmployeeType employeeType;

    public Employee(float salary, float bonusPercentage,
                    EmployeeType employeeType){
        this.salary = salary;
        this.bonusPercentage = bonusPercentage;
        this.employeeType = employeeType;
    }
    //calcula el salario dependiendo del tipo de trabajador
    //y entrega el décimo correspondiente cada 2 meses
    public float cs() {
        Date date = new Date();
        //Obtiene la hora local
        LocalDate localDate = date.toInstant()
            .atZone(ZoneId.systemDefault()).toLocalDate();
        //Obtiene el mes en forma de entero
        int month = localDate.getMonthValue();
        switch (employeeType)
        {
            case Worker:
                //Si el mes es impar entonces le entrega
                //el décimo junto con su salario
                return month%2==0?salary:salary + rmu/12*2;
            case Supervisor:
                float valueS = salary + (bonusPercentage * 0.35F);
                //Si el mes es impar entonces le entrega
                //el décimo junto con su salario y un bono
                return month%2==0?valueS:valueS + rmu/12*2;
            case Manager:
                float valueM = salary + (bonusPercentage * 0.7F);
                //Si el mes es impar entonces le entrega
                //el décimo junto con su salario y un bono
                return month%2==0?valueM:valueM + rmu/12*2;
        }
        return 0.0F;
    }
    //calcula el bono de fin de año
    public float CalculateYearBonus() {
        switch (employeeType) {
            case Worker:
                return rmu;
            case Supervisor:
                return salary + rmu * 0.5F;
            case Manager:
                return salary + rmu * 1.0F;
        }
        return 0.0F;
    }
}
```



Cédula



Cédula

TEMA 3 – APLICACIÓN DE PATRONES (40 PUNTOS)

13) Dibuje el diagrama de clases para el siguiente requerimiento de su proyecto. Recuerde aplicar el / los patrón(es) de diseño adecuado(s), las relaciones entre las clases y la encapsulación correcta. Únicamente, omitir los getters y setters (20 pt)

Requerimiento: Cocinero. Debe tener como interfaz principal la cola de pedidos y la cola de prioridad, en donde se muestra el tiempo estimado de entrega y el tiempo que tiene el pedido en cola. Los pedidos se deben atender en el orden de llegada y al empezar a atender un pedido se actualiza el estado para que no se puedan realizar modificaciones. Aunque si hay un pedido en la cola de prioridad, entonces esta se atiende más rápido. Al completar un pedido simplemente se debe ingresar que está completo el pedido, notificando al mesero. Cuando el cocinero ve que faltan ingredientes para algún plato, entonces selecciona el/los platos que ya no pueden ser servidos, por lo tanto, no se deberán mostrar como opciones del menú.

Cédula

14) Diseñe una solución para el siguiente problema, aplicando patrones de diseño e indicando sus nombres. Recuerde aplicar el / los patrón(es) de diseño adecuado(s), las relaciones entre las clases y la encapsulación correcta. Únicamente, omitir los getters y setters (20 pt)

La empresa “Café To Go” ya tiene parte de un sistema con un dispensador electrónico que permite comprar diferentes tipos de bebidas calientes (Ej: cortadito, expreso, capuccino, etc.). Cada bebida tiene un código, un nombre y un precio. donde los precios pueden oscilar entre 0.80 y 2.20 dólares dependiendo del tipo de bebida. Además, antes de entregar la bebida permite agregarle algún topping (ingrediente adicional). Entre los cuales están: crema chantilly, cacao en polvo, canela en polvo y malvaviscos, aunque el cliente puede elegir no agregarle nada.

La empresa lo ha contratado a usted como consultor para que les ayude a aplicar patrones de diseño para los procesos de añadir un topping a la bebida y para entregar el vuelto de una compra. Para este último, el dispensador acepta únicamente monedas de 5, 10, 25, 50 centavos y de un dólar, pero puede proveer el valor total ingresado por cliente y el valor de la bebida. Por otro lado, el dispositivo **Colector de monedas** tiene un **HashMap<Money, Integer>** que almacena por cada tipo de moneda (5, 10, 25, 50 y dólar) la cantidad de monedas totales de ese tipo que tiene el dispensador.