ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN
CCPG1042 - DISEÑO DE SOFTWARE
SEGUNDO EXAMEN – II PAO 2023

Nombre: _____ ***Matrícula:*** _____

_____ 100
_____ Firma

Sección A

Justifique su selección de la respuesta, según corresponda. Cada pregunta de esta sección vale 5%.

1. Suponga una aplicación que permite mostrar información de tarjetas de crédito. Tenemos tres tarjetas de crédito: *MoneyBack*, *Titanium* y *Platinum*. Estas tarjetas de crédito no son más que clases de productos. Cada una tiene límites crediticios y costos anuales diferenciados.

   El tipo de tarjeta que recibe un cliente depende de su salario mensual. La tarjeta *MoneyBack* está disponible para cualquier persona con un salario anual inferior a 50 000 USD. Puede recibir una tarjeta *Titanium* si su ingreso anual es inferior a 100 000 USD. Es elegible para una tarjeta *Platinum* si al menos gana 100 000 USD al año.

   ¿Qué patrón de diseño es aplicable en este contexto?
   a. Builder
   b. Factory Method
   c. Adapter
   d. Decorator
   e. Strategy
   f. Composite

2. Para los siguientes requerimientos de un juego para PC, identifique el patrón de diseño correspondiente:
   El Rey Orco da las órdenes a su ejército. El ejército está organizado por la siguiente jerarquía [***Comandante -> Oficial -> Soldado*** *] y cada uno puede realizar un único tipo de orden. Sin embargo, en caso de no poder realizar la orden asignada se la transfiere a su subordinado.*
   a. Chain of responsibility
   b. Decorator
   c. Iterator
   d. Builder
   e. Factory Method
   f. Strategy

3. Identifique el mal olor de programación (*Code smell*) presente en el fragmento de código mostrado a continuación:

   a. Liskov Sustitution
   b. Feature Envy
   c. Lazy Class
   d. Data Clumps
   e. Refused Bequest
   f. Data Class

```
class Animal {
  private int numberOfLegs;

  ...

    public getNumberOfLegs() {
      return numberOfLegs;
    }
}

class Cat extends Animal {
  public getNumberOfLegs() {
    super.getNumberOfLegs();
  }
}

class GoldFish extends Animal {

  ...

    public getNumberOfLegs() {
      throw NotImplementedException();
    }
}
```


MSc. David Jurado Mosquera, Dr. Carlos Mera Gómez

4. Indique el ***tipo_assert*** correspondiente para el código de la prueba **repeat_zero()** para el método **repeat()** de la clase utilitaria **StringUtil**.

```
public static String repeat(String str, int times) {
    if (times < 0) {
        throw new IllegalArgumentException("negative times not allowed");
    }
    String result = "";
    for (int i = 0; i < times; i++) {
        result += str;
    }
    return result;
}
```

```
@Test
@DisplayName("Repeat no times")
void repeat_zero() {
    tipo_assert(StringUtil.repeat("hola",0).contains("hola"));
}
```

   a. assertThrows
   b. assertFalse
   c. assertTrue
   d. assertEquals
   e. assertSame
   f. fail

5. Identifique el mal olor de programación (*Code smell*) presente en el fragmento de código mostrado a continuación:

```
class DateUtil {
    boolean isAfter(int year1, int month1, int day1, int year2, int month2, int day2) {
        // implementation
    }

    int differenceInDays(int year1, int month1, int day1, int year2, int month2, int day2) {
        // implementation
    }

    // other date methods
}
```

   a. Comments
   b. Inappropriate intimacy
   c. Divergent change
   d. Data clumps
   e. Large class

6. Indique la técnica de refactorización que utilizaría en el siguiente segmento de código fuente.

```
public void applyDiscount(int type, double discount) {
    switch (type) {
        case FIXED_DISCOUNT:
            this.price -= discount;
            break;
        case PERCENT_DISCOUNT:
            this.price *= discount;
            break;
        default:
            throw new IllegalArgumentException("Invalid discount type");
    }
}
```

   a. Replace Parameter with Explicit Methods
   b. Replace Error Code with Exception
   c. Replace Exception with Test
   d. Replace Delegation with Inheritance
   e. Replace Parameter with Method Call

Sección B

7. Considere el escenario que se presenta a continuación:

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 Email, Telegram y Whatsapp. Nuevos mecanismos se deberían poder añadir al sistema sin mayor impacto.

El sistema permite conocer la ubicación (latitud y longitud) de cada unidad de transporte en tiempo real. Con este propósito, cada unidad tiene instalado un dispositivo de rastreo de alguna de las diferentes marcas disponibles. Sin embargo, no existe una forma estándar entre las marcas, para obtener la ubicación de los dispositivos.

a. **Identifique** los patrones de diseño aplicables. **Justifique** su respuesta. **[12%]**
b. Elabore un diseño utilizando **diagrama de clases** que muestre solo la parte relevante a los patrones de diseño seleccionados. De ser posible, 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. Indique cualquier asunción que realice. **[15%]**
c. Separe en **paquetes** según los patrones usados. **[03%]**

Sección C

8. Considere el siguiente código fuente que corresponde a un sistema de reservas para un hotel.

```java
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;

class ReservaHotel {
    private String idCliente;
    private String nombreCliente;
    private String email;
    public LocalDate fechaInicio;
    public LocalDate fechaFin;
    private int cantidadPersonas;
    private String tipoHabitacion = "Estandar";

    public void setCliente(String idCliente, String nombreCliente, String email) {
        this.idCliente = idCliente;
        this.nombreCliente = nombreCliente;
        this.email = email;
    }

    public void setCantidadPersonas(int cant) {
        this.cantidadPersonas = cant;
    }

    public void setTipoHabitacion(String tipo) {
        this.tipoHabitacion = tipo;
    }

    public String getDatosCliente() {
        return "ID:" + idCliente + ",Cliente:" + nombreCliente + ",Email:" + email;
    }

    public int calcularDiasDeReserva() {
        return (int) ChronoUnit.DAYS.between(fechaInicio, fechaFin);
    }

   public double calcularCostoReserva() {
      int cantidadDias = calcularDiasDeReserva();
      if (cantidadDias <= 0) {
         return -1; // Error de dias de reserva
      }
      return getCostoXTipoHabitacion() * cantidadDias;
   }

    private double getCostoXTipoHabitacion() {
        switch (tipoHabitacion.toLowerCase()) {
            case "deluxe":
                return 75.0;
            case "suite":
                return 110.0;
            default:
                return 50.0;
        }
    }

    public void mostrarDetallesReserva() {
        System.out.println(getDatosCliente());
        System.out.println("Fechas: " + fechaInicio + " - " + fechaFin);
        System.out.println("Cantidad de Personas: " + cantidadPersonas);
        System.out.println("Tipo de Habitación: " + tipoHabitacion);
        System.out.println("Días de Reserva: " + calcularDiasDeReserva());
        System.out.println("Costo: " + calcularCostoReserva());
    }
}
```

A usted se le solicita:

a. Señalar los malos olores de programación en el código e indicar su nombre. Explique por qué sería un problema. **[16%]**

b. Indicar las técnicas de refactorización que utilizaría para mejorar el código. Justifique su respuesta. **[16%]**

c. Realizar 2 pruebas unitarias para el método **calcularCostoReserva().** Indique el propósito de cada prueba. **[08%]**